Галицький коледж імені В’ячеслава Чорновола
відділення комп’ютерних та видавничих технологій
циклова комісія інформатики та комп’ютерних дисциплін

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач відділенням
комп’ютерних та видавничих
технологій
Чубей О.О. /_____/
підпис
«___» _____ 2021 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту
освітньо-кваліфікаційного рівня «молодший спеціаліст»
зі спеціальності 122 «Комп’ютерні науки»

на тему: «Автоматизація інформаційної системи платної поліклініки»

Студент групи К-47 Романів О.В. _____
(підпис)

Керівник проєкту Посвятовська О.Б. _____
(підпис)

Консультанти:

з техніко-економічного
обґрунтування Меленчук Л.І. _____
(підпис)

нормоконтролер Гавришків Н.Г. _____
(підпис)

Тернопіль – 2021

Галицький коледж імені В’ячеслава Чорновола
відділення комп’ютерних та видавничих технологій
циклова комісія інформатики та комп’ютерних дисциплін

ЗАТВЕРДЖУЮ
Завідувач відділенням
комп’ютерних та видавничих
технологій
Чубей О.О. /_____/
підпис
«___» _____ 2020 р.

ЗАВДАННЯ

на дипломне проєктування
на здобуття освітньо-кваліфікаційного рівня «молодший спеціаліст»
студенту Романіву Олегу Володимировичу

(прізвище, ім’я та по-батькові студента)

1. Тема проєкту _____
_____
_____
затверджено наказом по коледжу від “___” _____ 202_ р., №___
2. Термін здачі студентом завершеного проєкту “___” _____ 202_ р.
3. Вихідні дані до проєкту _____
_____
_____
4. Перелік питань, які повинні бути розроблені в проєкті:
а) основна частина _____
_____
б) техніко-економічне обґрунтування _____
_____
_____
_____
5. Перелік графічного матеріалу _____
_____

6. Консультанти проєкту: _____

Розділ	Консультанти	Підпис, дата	
		Завдання видано	Завдання прийнято
з техніко-економічного обґрунтування	_____ (вчена ступінь, звання П.І.Б. _____ консультанта)		

КАЛЕНДАРНИЙ ПЛАН

дипломного проєктування

№ п/п	Найменування етапу	Терміни	
		початку	завершення
1.	Вибір теми, ознайомлення з вимогами до дипломного проєктування	16.11.20	30.11.20
2.	Огляд типових рішень та написання відповідного розділу ПЗ	01.12.20	26.01.21
3.	Дослідження технологій реалізації та написання відповідного розділу ПЗ	27.01.21	15.02.21
4.	Розробка функціональних вимог до проєкту та робота над структурою програмного продукту. Написання відповідного розділу ПЗ	15.02.21	02.03.21
5.	Встановлення та налаштування середовища реалізації та написання відповідного розділу ПЗ	02.03.21	16.03.21
6.	Проєктування програмного засобу (функціоналу, інтерфейсу, бази даних продукту) та написання відповідного розділу ПЗ	16.03.21	16.04.21
7.	Реалізація та налаштування програмного засобу та написання відповідного розділу ПЗ	17.04.21	03.05.21
8.	Доопрацювання модулів	03.05.21	17.05.21
9.	Опрацювання економічного розділу дипломного проєкту та оформлення спеціального розділу	17.05.21	18.06.21
10.	Тестування та налагодження програмного продукту та написання відповідного розділу ПЗ	18.05.21	04.06.21
11.	Робота над оформленням пояснювальної записки	04.06.21	11.06.21
12.	Попередній захист дипломного проєкту, доопрацювання	11.06.21	
13.	Підготовка до захисту дипломного проєкту	18.06.21	23.06.21
14.	Захист дипломного проєкту	24.06.21	

7. Дата видачі “___” _____ 2020р. Керівник _____/

Завдання прийняв до виконання _____/

Реферат

Дипломний проект. «Автоматизація інформаційної системи платної поліклініки». 73 сторінок, 22 рисунків, 17 таблиць, 12 джерел.

Об’єктами дослідження для подальшої розробки дипломного проекту стали існуючі подібні програмні рішення, що зв’язані з базую даних.

Мета проекту – розробити програмний додаток, який дасть можливість користувачам виконувати управління базою даних та переглядом потрібних записів.

Завданням проекту є розробка програмного додатку для управління базою даних.

Результат – програмний додаток, який повністю готовий до експлуатації. Присутність простого та зрозумілого користувацького інтерфейсу. Легка реєстрація для нових користувачів та можливість управляти інформацією таблиць.

Для розробки програмного додатку було обрано середовище розробки Visual Studio, що використовується для розробки різноманітних додатків та у тому ж числі десктопних на мові програмування C#. Для реалізації серверної частини використано СУБД SQL Server.

ПРОГРАМНИЙ ДОДАТОК, SQL SERVER, T-SQL, C#, WINDOWS FORMS

Abstract

Diploma project. "Information system of paid clinic automation". 73 pages, 22 pictures, 17 tables, 12 sources.

The objects of research for further development of the diploma project were the existing similar software solutions related to the database.

The aim of the project is to develop a software application that will allow users to manage the database and view the required records.

The task of the project is to develop a software application for database management.

The result is a software application that is completely ready for use. Presence of a simple and clear user interface. Easy registration for new users and the ability to manage table information.

Visual Studio development environment was chosen for the development of the software application, which is used for the development of various applications, including desktop applications in the C # programming language. SQL Server DBMS was used to implement the server part.

PROGRAM APPLICATION, SQL SERVER, T-SQL, C#, WINDOWS FORMS.

ЗМІСТ

					ДП. КН 21.438.17.000 ПЗ		
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розроб.		*Колачик Н.М.*			Автоматизована інформаційна система платної поліклініки	*Літ.*	*Арк.* / *Аркушів* 73
Перев.		*Павлюс В.П.*					
Рецензет.		*Чубей О.О.*				ГК. КВТ. К-47	
Н. Контр.		*Кульчинська Н.З.*					
Зав. від.		*Чубей О.О.*					

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

CLR – Common Language Runtime «загальномовне виконуюче середовище» це компонент пакету Microsoft .NET Framework, віртуальна машина, на якій виконуються всі мови платформи .NET Framework.

РСУБД – реляційна система управління базами даних.

SDK – набір із засобів розробки, утиліт і документації, який дозволяє програмістам створювати прикладні програми за визначеною технологією або для певної платформи (програмної або програмно-апаратної).

API – набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення.

GUI – графічний інтерфейс користувача тип інтерфейсу, який дає змогу користувачам взаємодіяти з електронними пристроями через графічні зображення та візуальні вказівки, на відміну від текстових інтерфейсів, заснованих на використанні тексту, текстовому наборі команд та текстовій навігації.

MFC – бібліотека, яка дає можливість розробляти GUI-застосунки для Microsoft Windows на мові C++ з використанням багатого набору бібліотечних класів. Велика частина MFC є відносно тонким об'єктно-орієнтованим шаром над Windows API.

ОС – операційна система.

ВСТУП

Програмні системи з кожним днем вдосконалюються, для можливості виконання більш глобальних завдань. Наприклад для створення системи управління підприємством.

Для розробки таких систем потрібно вміти уявляти сучасні методи і підходи , які існують такій галузі, і головні проблеми такого процесу. В складних програмних системах потреба зростає кожен день. Ціна на обчислювальну техніку з кожним днем падає при тому що збільшується продуктивність,з'являються все більше можливостей для виконання автоматизації складних процесів.

Щоб виконувати складніші математичні розрахунки головними на початку були комп'ютери (перш за все призначалися розрахунків, які були пов'язані із вдосконаленням ракетної техніки, а також створенням ядерної зброї), на сьогоднішній день головним напрямком є накопичення інформації також її обробка. Цілком зрозуміло чому в обчислювальній техніці трапився такий перерозподіл функцій. Зараз набагато поширеніший цивільний бізнес , в порівнянні з науковими і військовими обчисленнями, також для невеликих підприємств приватних осіб через зниження вартості комп'ютерів зробило їх більш доступними.

На сьогоднішній день важко уявити керування закладами без комп'ютерів неможливо. Вони уже давно є в таких галузях управління, як: бухгалтерський облік, керування складом і закупівлями також асортиментами. Але бізнес хоче більш широкого використання інформаційних технологій в управліні закладом. Сучасний бізнес дуже не пробачає помилок в управлінні цим і пояснюється життєздатність та розвиток сучасних інформаційних технологій. Щоб бути попереду інших інтуїції, особистого досвіду і розміру його капіталу уже не достатньо. Потрібно завжди тримати все під контролем для того щоб прийняти будь яке фінансове рішення в управлінні слід враховувати всі аспекти фінансово-господарської діяльності, такі як: надання будь-яких послуг,

виробництво і торгівля. Саме через це сучасний підхід для управління чекає великих вкладень ресурсів в інформаційні технології. Чим більше буде заклад, тим серйозніші будуть такі вкладення. Це є життєвою необхідністю - в сучасній конкурентній і перемогти зможе тільки ті, хто будуть ефективно організовані і краще оснащенні.

Комерційні заклади охорони здоров'я функціонують в будь якій фромі, передбачений чинним законодавством для підприємницької діяльності при дотримані вимог ліцензії,які стосуються надання медичних послуг.Основною метою комерційного закладу охорони здоров'я є отримання прибутку.

Одгим із видій комерційних закладів є поліклініка

Для ефективної роботи поліклініки потрібно щоб персонал мав постійний доступ до актуальної інформації про клієнтів, діагнози і курси лікування.В цьому і полягає важливість інформаційної системи для установ

Актуальність цієї тематики полягає в тому, що наш час інформаційних технологій, стало можливо всі документи перетворювати в електронний вигляд тому реєстратура швидко може знайти відомості про пацієнтів, виклики, кабінетах.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Про комерційні медичні заклади та автоматизацію діяльності.

Одним із можливих варіантів надання медичних послуг населенню на платній основі є комерційні поліклініки.

В загальному випадку поліклініка це частина лікарні призначена для лікування амбулаторних хворих або людей з проблемами здоров'я які відвідують лікарню для діагностики або лікування, але в цей час не потребують ліжка або прийому на нічний догляд.

Таким чином, клініка працює з великою кількістю різноманітної інформації як про персонал, так і про пацієнтів та їхні діагнози. Лікарі повинні завжди мати доступ до актуальної інформації про своїх пацієнтів та перебіг лікування стежити за даними про своїх пацієнтів, курс лікування пацієнтів. А адміністрація закладу та бухгалтерія повинні мати доступ до всієї інформації для здійснення керівництва закладу.

Діяльність комерційних поліклінік.

В основному комерційна поліклініка займається тим чим і державна а саме ставлять діагнози приймають аналізи і займаються не стаціонарним лікуванням великий плюс такого закладу це економія часу тому що у зв'язку з пандемією лікарі не мають багато часу на звичайних пацієнтів з звичайними хворобами.

Для цього потрібна загальна база даних, яка включає всю необхідну інформацію. Програма є найбільш актуальною на сьогоднішній день, вона автоматизує роботу з основними даними та забезпечує користувачеві (оператору) зрозумілий та зручний інтерфейс.

У базі даних Поліклініки використовується такий ввід:

- інформація споживача;
- інформація про лікарів;

- інформація про спеціалізацію лікарів;
- інформація про діагноз;
- інформація про системи користувачів;
- інформація про лікування.

Вихідна інформація - це результати записів та процедур збереження в базі даних баз даних програми та всіх даних, що відображаються у вікні додаткової інформації під час її роботи.

Вирішити проблему підвищення ефективності управління закладом у сучасних умовах неможливо без впровадження нових інформаційних технологій та сучасних методів управління. Постановка цілей. Аналіз існуючих методів та засобів автоматизації подібних об'єктів та формулювання на основі вимог користувача до досяжних цілей системи управління. Цілі повинні бути чіткими, чіткими та вимірюваними. Необхідно визначити: системи загального призначення, визначення різних груп користувачів та їх ролі, звіт про перелік функціональних систем, огляд необхідної документації, ефективність параметрів (продуктивність), сумісність з іншими продуктами та стандартами, конфігурацію обладнання, захист засоби, методи та засоби конфігурації служби, методи забезпечення надійності систем. Якщо один з них не повинен суперечити, їм слід керуватися тим, щоб створювати компроміси на наступних етапах проектування.

Розробка архітектури системи (декомпозиція функціональної структури та визначення зв’язків між її елементами). Вибір рівнів управління, підсистем, наборів завдань, завдань та функцій управління.

Розробка системної інфографічної моделі, що описує статику та динаміку об’єктів. Формалізація моделей стану потоків об’єктів, матеріалів, фінансів та інформації (управління) та їх взаємодії.

Розробка системи класифікації об’єктів бухгалтерського обліку та управління та виявлення їх параметрів. Словник описує основні поняття предметної області системи, необхідної для розробки стандартних алгоритмів обробки даних. Розробка системи класифікації цілей обліку та управління та

визначення їх параметрів. Словник описує основні поняття предметної області системи, необхідної для розробки стандартних алгоритмів обробки даних. Класифікатор описує структуру об’єкта (відділ, працівник, посада), зовнішнє середовище (замовник, площа, пункт завантаження / розвантаження), характеристики матеріального потоку (партія, фонд, одиниця виміру, показник якості, тип ціни, тип) І пояснить. оплата). Типова операція описує алгоритм управління (обробка інформації).

Розробка інформаційної моделі системи (проектування структури бази даних та її підключення). - Синтез структури програмного забезпечення (агрегація системи). Поєднуючи окремі функції управління в програмних модулях, потрібно прагнути до високої "міцності" та слабкої "адгезії" модулів. Міцність і зчеплення модуля є, відповідно, заходами його внутрішніх і зовнішніх зв'язків. Залежно від призначення модулів необхідно прагнути або до їх функціональної міцності (об'єднання взаємопов'язаних функцій управління), або до інформаційної міцності (об'єднання функцій, які виконуються на обмеженій підмножині інформаційного простору системи).

Вибір методу складання та випробування системи. Існує кілька методів побудови та випробування складних програмних систем: зростаюча, низхідна, модифікована низхідна, стрибок у довжину, сендвіч-метод, модифікований сендвіч-метод. Для тестування системи рекомендується використовувати модифікований сендвіч-метод, при якому нижні модулі управління випробовуються знизу вгору, а верхні модулі управління спочатку випробовуються автономно, а потім збираються в одиниці за спаданням [1]. Перевагами запропонованого методу є: висока паралельність у програмуванні модулів, мала кількість заглушок, мінімальний час роботи робочої версії системи. Зверніть увагу, що послідовність проектування та програмування окремих модулів сильно залежить від обраного способу складання та випробування. Отже, спосіб складання системи повинен бути обраний до етапу проектування модулів.

Дизайн модулів. Розробка зовнішніх специфікацій, що описують зв'язок (зв'язок) між модулями та проектування логіки (алгоритмів) модулів.

Програмування модулів на обраному програмному забезпеченні. Під час програмування пам’ятайте, що текст програми необхідний для спілкування з людьми, а не з машиною. Важливість цього твердження стане очевидною, коли настане етап технічного обслуговування системи. Для підвищення надійності програмного забезпечення необхідно використовувати при програмуванні метод взаємної недовіри до модулів, тобто до кожного модуля системи потрібно ставитися з певним ступенем недовіри, в розумних межах.

Інтеграція (збірка) системи відповідно до обраного методу та її тестування. Етапи тестування: автономне тестування - управління окремим програмним модулем, ізольованим від інших модулів, тестування з'єднань - контроль з'єднань між частинами системи, тестування функцій - контроль системних функцій автоматизованого контролю, комплексне тестування - тестування поведінки системи щодо початкові цілі, прийомне тестування - перевірка відповідності системи вимогам користувача. Тестування - це процес виконання програми з метою виявлення в ній помилок. Існує два підходи до проектування тестів - тестування за специфікаціями (не дбаючи про текст програми) та тестування за текстом програми (не дбаючи про специфікації). Розумний компроміс лежить десь посередині, зміщуючись в ту чи іншу сторону залежно від функцій, що виконуються певним модулем. Також зверніть увагу, що вартість етапу тестування становить до 25% від загальної вартості розробки системи.

Розробка методичного забезпечення. Посібники, інструкції з експлуатації, технологічні інструкції.

Впровадження системи на місці.

Технічне обслуговування системи: усунення помилок та коментарів користувачів, розробка додаткових режимів та функцій управління, функціональне розширення системи. Відповідно до спіральної моделі

життєвого циклу програмного забезпечення здійснюється перехід до 1 - 10 стадій проектування системи.

Слід зазначити, що етап технічного обслуговування є найдорожчим етапом, його вартість оцінюється експертами в 50% від загальної вартості розробки системи. Це можна пояснити тим, що насправді цей етап не є незалежним, а об'єднує групу вищевказаних етапів проектування на наступному етапі системної реалізації спіралі життєвого циклу програмного забезпечення.

Актуальність цієї тематики полягає в тому, що наш час інформаційних технологій, стало можливо всі документи перетворювати в електронний вигляд тому реєстратура швидко може знайти відомості про пацієнтів, виклики, кабінетах.

Мета роботи: зібрати матеріал і розробити інформаційну систему для роботи платної поліклініки.

1.2 Огляд та аналіз існуючих аналогів

Сучасні бази даних - це переважно програми Windows, оскільки це середовище дозволяє скористатися всіма перевагами вашого персонального комп’ютера, ніж середовище DOS. Зниження вартості високопродуктивних ПК не тільки призвело до широкого переходу до середовища Windows, де розробники програмного забезпечення можуть менше турбуватися про розподіл ресурсів, але і забезпечення для ПК загалом та СУБД, зокрема, менш критичними для апаратного забезпечення комп’ютера. .

Серед найвидатніших представників систем управління базами даних: Microsoft Access, Microsoft Visual, Microsoft Visual Basic, а також бази даних зробили програмне Microsoft SQL Server та Oracle, що використовуються в додатках, побудованих за технологією "клієнт". Насправді будь-яка сучасна база даних має аналог, вироблений іншою компанією зі схожими обсягами та можливостями, будь-яка програма здатна працювати з багатьма форматами

даних, експортувати та імпортувати дані завдяки великій кількості перетворювачів. Також поширеними є технології, що дозволяють використовувати інші програми, такі як текстові процесори, графічні пакети тощо, та вбудовані версії мов високого рівня (часто діалекти SQL та / або VBA) та засоби візуального програмування для інтерфейсів. Тому вже не має значення, якою мовою та на якій основі. Крім того, "де-факто" стало стандартним "швидким розвитком додатків" або RAD (від англійського Rapid Application Development), заснованим на широко декларованому в літературі "відкритому підході", тобто необхідності та можливості використання різноманітних додатків та технологій для розробки більш гнучких та потужних систем обробки даних із "класичною" базою даних, все частіше згадуються мови програмування Visual Basic 4.0 та Visual C.

- ODBC-драйвер для доступу до даних Paradox із додатків Windows.
- Інструменти для доступу до даних Paradox із програм Java.
- Часова версія Paradox для доставки з додатками.
- Інструменти для створення дистрибутивів.

Драйвери SQL Links для доступу до даних бази даних сервера:

- Доступ - у перекладі з англійської означає "доступ". MS Access - це функціонально повна реляційна база даних. Крім того, MS Access є однією з найпотужніших, гнучких і простих у використанні СУБД. У ньому ви можете створити більшість додатків, не написавши жодного рядка програми, але якщо ви хочете створити щось дуже складне, то в цьому випадку MS Access надає мову програмування - Visual Basic Application.

- Популярність бази даних Microsoft Access обумовлена такими причинами:

- Access - одна з найбільш легкодоступних і зрозумілих систем як для професіоналів, так і для початківців користувачів, що дозволяє швидко засвоїти основні принципи роботи з базами даних;

- система має повністю русифіковану версію;

− повна інтеграція з пакетами Microsoft Office: Word, Excel, Power Point, Mail;

− Visual Basic - це універсальна об'єктно-орієнтована мова програмування, діалекти якої вбудовані в Access, Visual FoxPro. Переваги: універсальність, можливість створення OLE-компонентів, низькі вимоги до апаратних ресурсів комп'ютера. Застосовується для створення додатків середньої потужності, які не пов'язані з високою інтенсивністю обробки даних, розробкою компонентів OLE, інтеграцією компонентів Microsoft Office. Ці програмні продукти мають можливість візуального дизайну інтерфейсу користувача, тобто розробник із готових фрагментів створює елементи інтерфейсу, програмує лише їх зміни у відповідь на будь-які події.

1.3 Постановка завдання

Для дипломного проекту потрібно розробити інформаційну систему для платної поліклініки, яка допоможе будь-якому користувачеві легко знайти потрібну інформацію про будь-якого лікаря або клієнта і не тільки.

Інформаційна система «Платна поліклініка» включає в себе дані про лікарів, клієнтів, діагнози, спеціальності лікарів, лікування, що будуть вміщати побічні ключі з більшості існуючих таблиць, щоб не звертатись до усіх таблиць окремо, так і дані про користувачів інформаційної системи, що використовуватимуть її. Система може виконувати пошук і видалення даних, додавати, змінювати і переглядати ці дані.

Для цього потрібна загальна база даних, яка включає всю необхідну інформацію. Програма є найбільш актуальною на сьогоднішній день, вона автоматизує роботу з основними даними та забезпечує користувачеві (оператору) зрозумілий та зручний інтерфейс.

У базі даних Поліклініки використовується такий ввід:

− Інформація споживача;

− інформація про лікарів;

- інформація про спеціалізацію лікарів;
- інформація про діагноз;
- інформація про системи користувачів;
- інформація про лікування.

Вихідна інформація - це результати записів та процедур збереження в базі даних баз даних програми та всіх даних, що відображаються у вікні додаткової інформації під час її роботи.

1.4 Визначення основних вимог системи

Після детального вивчення предметної області ми можемо перейти до визначення основних вимог до системи, яка має бути розроблена в дипломній роботі.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Розглянемо метод проектування автоматизованих інформаційних систем управління підприємством, який створюється з наступних етапів.Обстеження об'єкта -автоматизації (аналіз) і формулювання вимог користувачів до системи управління.

Реляційна модель даних включає такі компоненти:

− Структурний аспект (компонент) - дані в базі даних являють собою сукупність взаємозв’язків.

− Аспект (складова) цілісності - відношення (таблиці) відповідають певним умовам цілісності. Реляційна модель даних підтримує декларативні обмеження цілісності рівня домену (тип даних), рівня відносин та рівня бази даних.

Аспект (компонент) обробки (маніпуляції) - реляційна модель даних підтримує операторів, що маніпулюють відносинами (реляційна алгебра, реляційне числення).

У базі даних "Поліклініка" в таблицях "Діагностика", "Лікарі", "Клієнти", "Спеціалізації", "Лікування", "Користувачі" між атрибутами та первинним ключем існує функціональна взаємозв'язок, оскільки значення ключа однозначно визначити значення інших атрибутів у таблицях даних.

Функціональні взаємозв'язки між атрибутами "Лікарі" подано в таблиці 2.1.

Функціональні залежності між атрибутами «Клієнти» подано в таблиці 2.2.

Функціональні залежності між атрибутами «Спеціалізація» подано в таблиці 2.3.

Функціональні залежності між атрибутами «Діагнози» подано в таблиці 2.4.

Функціональні залежності між атрибутами «Лікування» подано в таблиці 2.5.

Функціональні залежності між атрибутами «Користувачі» подано в таблиці 2.6.

Таблиця 2.1 - Функціональні взаємозв'язки між атрибутами "Лікарі"

Найменування атрибутів	Функціональні залежності
Doc_id Doc_surname Doc_name Doc_middle_name Doc_birth_date Doc_gender Doc_category Sp_id	

Таблиця 2.2 – Функціональні залежності між атрибутами «Клієнти»

Найменування атрибутів	Функціональні залежності
Cl_id Cl_surname Cl_name Cl_middle_name Cl_birth_date Cl_gender Cl_blood_group	

Таблиця 2.3 – Функціональні залежності між атрибутами «Спеціалізація»

Найменування атрибутів	Функціональні залежності

Sp_id Sp_name	

Таблиця 2.4 – Функціональні залежності між атрибутами «Діагнози»

Найменування атрибутів	Функціональні залежності
Diag_id Cl_id Diag_text Doc_id Diag_date	

Таблиця 2.5 – Функціональні залежності між атрибутами «Лікування»

Найменування атрибутів	Функціональні залежності
Tr_id Doc_id Cl_id Diag_id Tr_cost Tr_begin_date Tr_end_date	

Таблиця 2.6 – Функціональні залежності між атрибутами «Користувачі»

Найменування атрибутів	Функціональні залежності
User_id User_username	

User_password User_permission	

Для кожної таблиці визначені свої ключі. Ключі таблиць знаходяться в таблиці 2.7.

Таблиця 2.7 – Ключі

Таблиця	Ключ
Лікарі	Doc_id Sp_id
Клієнти	Cl_id
Спеціалізації	Sp_id
Діагнози	Diag_id Cl_id Doc_id
Лікування	Tr_id Doc_id Cl_id Diag_id
Користувачі	User_id

Для виконання серверної функції було обрано Sql Server Management Studio 2018.

У базі даних "Поліклініка" нормалізація відносин:

Аналізуючи таблицю "Клієнти", можна сказати, що вона знаходиться в першій нормальній формі, оскільки має первинний ключ, кожне поле таблиці представляє унікальний тип інформації, всі поля є атомними. Подібним чином

ця таблиця є у 2NF, оскільки вона задовольняє умовам 1NF, і я також переконався, що кожне поле функціонально залежить від первинного ключа, який ідентифікує вихідний об'єкт таблиці. Таблиця "Клієнти" знаходиться в 3NF, як і в 2NF і не містить перехідних залежностей. Стовпці, які не є ключовими, залежать від первинного ключа таблиці і не залежать від усіх інших стовпців. Можна змінити значення будь-якого поля (не входить до первинного ключа), не впливаючи на дані інших полів.

Таблиця спеціалізації, подібна до таблиці лікарів, є у всіх трьох нормальних формах.

Таким чином, аналізуючи розроблену базу даних, можна зробити висновок, що вона нормалізована і відповідає трьом нормальним формам.

2.1 Інфологічна модель даних

Інфологічний дизайн - побудова формалізованої моделі предметної області. Така модель будується із застосуванням стандартних мовних засобів, як правило, графічних.

На етапі інфографічного проектування під час збору інформації про предметну область потрібно з’ясувати:

1. Основні об’єкти предметної області (об’єкти, про які слід зберігати інформацію в базі даних);
2. Атрибути об’єктів;
3. Зв’язок між об’єктами;
4. Основні запити до бази даних.

2.2 Опис суб’єктів

У проекті "Поліклініка" відповідно до предметної області були створені такі сутності:

- «Лікарі» - інформація про лікарів зберігається;
- "Клієнти" - зберігається інформація про клієнтів клініки;

– "Спеціалізації" - зберігається інформація про спеціалізації лікарів;

– "Діагностика" - зберігається інформація про діагностику клієнтів лікарями;

– «Лікування» - зберігається інформація про лікування клієнтів лікарями відповідно до відповідного діагнозу;

– "Користувачі" - зберігається інформація про користувачів системи.

– Кожен об’єкт має свої атрибути:

– код лікаря лікаря, ім'я лікаря, дата народження, стать, категорія, код спеціалізації;

– Клієнтський код клієнта, ім'я, дата народження, стать, група крові

– Код спеціалізації спеціалізації, назва;

– діагнози, код діагнозу, код клієнта, текст діагнозу, код лікаря, дата діагностики;

– Код лікування, код лікаря, код клієнта, код діагнозу, вартість лікування, дата початку лікування, дата закінчення лікування.

2.3 Опис зв’язків

У базі даних «Поліклініка» визначено такі відносини між таблицями:

Класифікація зв’язків подана в таблиці 2.8

Таблиця 2.8 “Класифікація зв’язків”

№	Батьківська таблиця	Дочірня таблиця	Ключі		Вид зв’язку
1	Лікарі	Спеціалізації	Sp_id	Sp_id	1:M
2	Діагнози	Лікарі	Doc_id	Doc_id	1:M
3	Діагнози	Клієнти	Cl_id	Cl_id	1:M
4	Лікування	Клієнти	Cl_id	Cl_id	1:M
5	Лікування	Лікарі	Doc_id	Doc_id	1:M

6	Лікування	Діагнози	Diag_id	Diag_id	1:M

Вибір таких зв'язків обумовлений тим що, в загальному обліку роботи поліклініки міститься інформація про всіх лікарів, клієнтів, діагнозів, лікувань та спеціалізацій.

2.4 ER-діаграма

На малюнку 2.1 представлена інфологічна модель бази даних, на якій відображені всі сутності БД, відношення між ними і атрибути.

ER-модель — це мета-модель даних, тобто засіб опису моделей даних. Існує ряд моделей для представлення знань, але одним з найзручніших інструментів уніфікованого представлення даних, незалежного від програмного забезпечення що його реалізує, є модель «сутність-зв'язок». Важливим є той факт, що з моделі «сутність-зв'язок» можуть бути породжені всі існуючі моделі даних (ієрархічна, мережева, реляційна, об'єктна), тому вона є найзагальнішою.

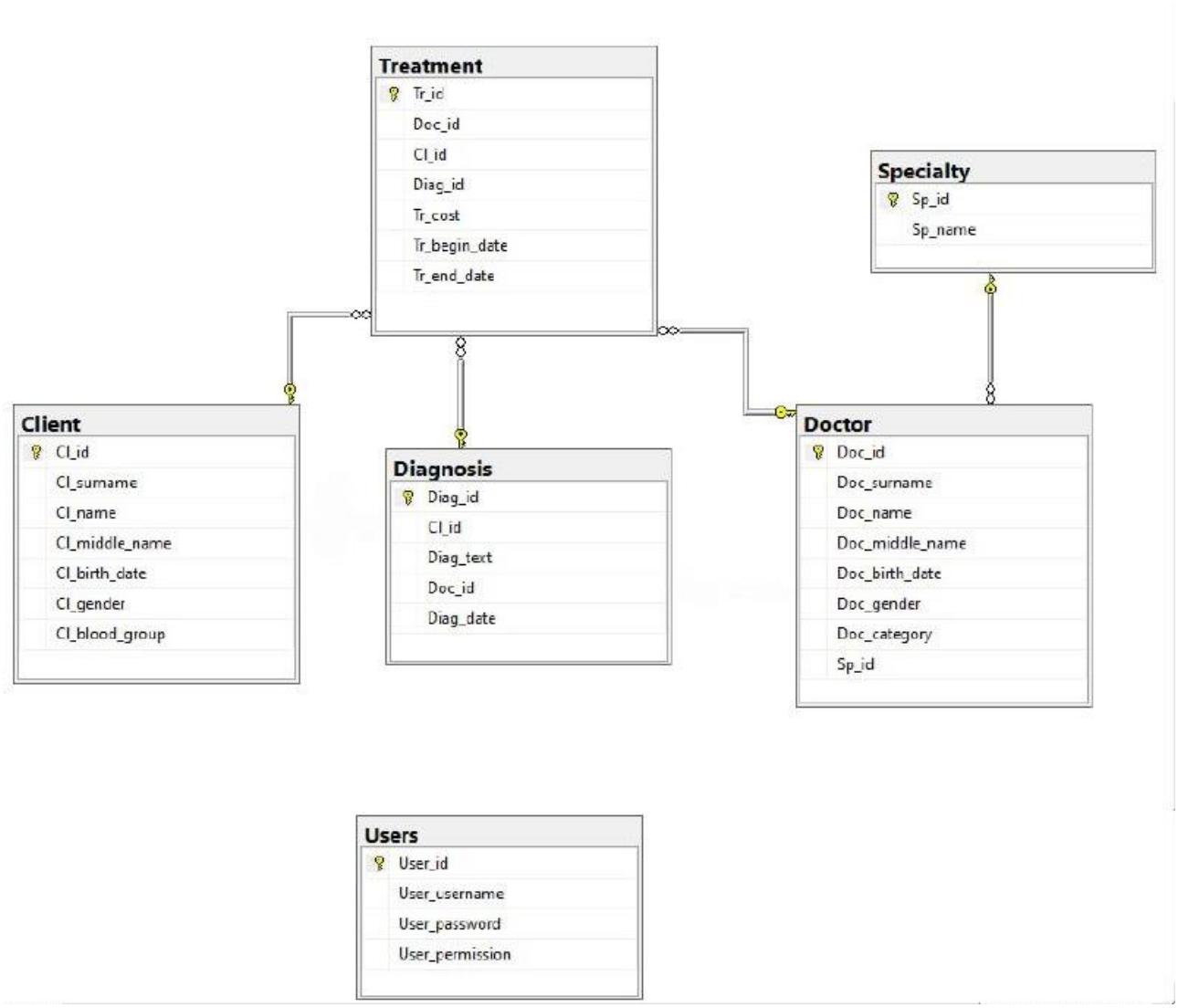

Рисунок 2.1 – Інфологічна модель бази даних

2.5 Даталогічна модель

У цьому розділі наводиться склад таблиць БД. Для кожного поля таблиці вказується розмір поля (кількість символів), тип. Для первинних ключів необхідно ввести заборону невизначених значень. Для інших полів можливість заборони невизначених значень визначається семантикою предметної області.

Склад таблиці «Спеціалізації» подано в таблиці 2.9.

Склад таблиці «Клієнти» подано в таблиці 2.10.

Склад таблиці «Лікарі» подано в таблиці 2.11.

Склад таблиці «Користувачі» подано в таблиці 2.12.

Склад таблиці «Діагнози» подано в таблиці 2.13.

Склад таблиці «Лікування» подано в таблиці 2.14.

Таблиця 2.9«Спеціалізації»

Найменування атрибута	Тип поля	NULL
Sp_id	int	Hi
Sp_name	nvarchar(50)	Hi

Таблиця 2.10«Клієнти»

Найменування атрибута	Тип поля	NULL
Cl_id	int	Hi
Cl_surname	nvarchar(50)	Hi
Cl_name	nvarchar(50)	Hi
Cl_middle_name	nvarchar(50)	Hi
Cl_birth_date	nvarchar(50)	Hi
Cl_gender	nvarchar(50)	Hi
Cl_blood_group	nvarchar(50)	Hi

Таблиця 2.11«Лікарі»

Найменування атрибута	Тип поля	NULL
Doc_id	int	Hi
Doc_surname	nvarchar(50)	Hi
Doc_name	nvarchar(50)	Hi
Doc_middle_name	nvarchar(50)	Hi
Doc_birth_date	nvarchar(50)	Hi
Doc_gender	nvarchar(50)	Hi

Doc_category	nvarchar(50)	Hi
Sp_id	int	Hi

Таблиця 2.12«Користувачі»

Найменування атрибута	Тип поля	NULL
User_id	int	Hi
User_username	nvarchar(50)	Hi
User_password	nvarchar(50)	Hi
User_permission	nvarchar(20)	Hi

Таблиця 2.13«Діагнози»

Найменування атрибута	Тип поля	NULL
Diag_id	int	Hi
Cl_id	int	Hi
Diag_text	nvarchar(500)	Hi
Doc_id	int	Hi
Diag_date	nvarchar(50)	Hi

Таблиця 2.14«Лікування»

Найменування атрибута	Тип поля	NULL
Tr_id	int	Hi
Doc_id	int	Hi
Cl_id	int	Hi
Diag_id	int	Hi

Tr_cost	nvarchar(50)	Hi
Tr_begin_date	nvarchar(50)	Hi
Tr_end_date	nvarchar(50)	Так

База даних інформаційної системи «Платна поліклініка» включає в себе дані про лікарів, клієнтів, діагнози, спеціальності лікарів, лікування, що будуть вміщати побічні ключі з більшості існуючих таблиць, щоб не звертатись до усіх таблиць окремо, так і дані про користувачів інформаційної системи, що використовуватимуть її.

2.6 Проектування клієнтського додатку

Визначення та реалізація основних класів та функцій системи

Основні функції системи відображені на діаграмі варіантів використання, яка зображена на рисунку 2.2.

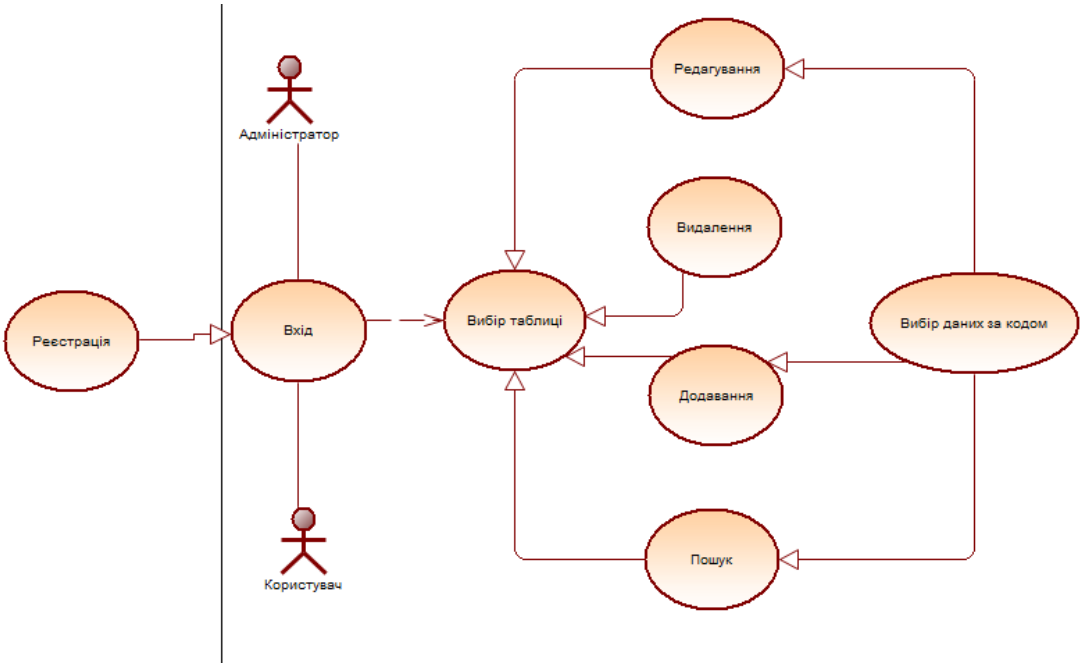

Рисунок 2.2 – Діаграма варіантів використання

Детальний опис основних функцій системи:

– Вхід в систему додатка відбуватиметься при введені унікального користувацького імені та відповідного паролю. Якщо користувач вперше в додатку то потрібно зареєструватись.

– Реєстрація відбувається за тим самим принципом, що і вхід, після реєстрації потрібно буде виконати вхід.

– Після входу у систему стає доступна функція вибору таблиці, над якою користувачу потрібну виконати певну операції, або переглянути певні записи.

– Коли таблиця вибрана буде доступно 4 функції: редагування – зміна обраного запису таблиці, та його оновлення; видалення – знищення вибраного запису з бази даних, попередньо підтвердивши дії у діалоговому вікні додатку; додавання – створення нового запису в базі даних з потрібною користувачеві інформацією; пошук – знаходження потрібного запису таблиці за будь-яким полем. Також в додатку присутні повноваження, адміністратор регулює повноваження. Повноваження дають доступ до функцій, що впливають за інформацію в базі даних, а саме: редагування, додавання та видалення. Пошук доступний усім користувачам [5].

– Вибір даних за кодом – функція, яка дозволяє відкривати нове вікно з таблицею, по якій на поточному основному вікні відображаються лише ключі. Також це зручно для простого перегляду потрібного запису у побічній таблиці.

Діаграма класів продемонстрована на рисунку 2.3.

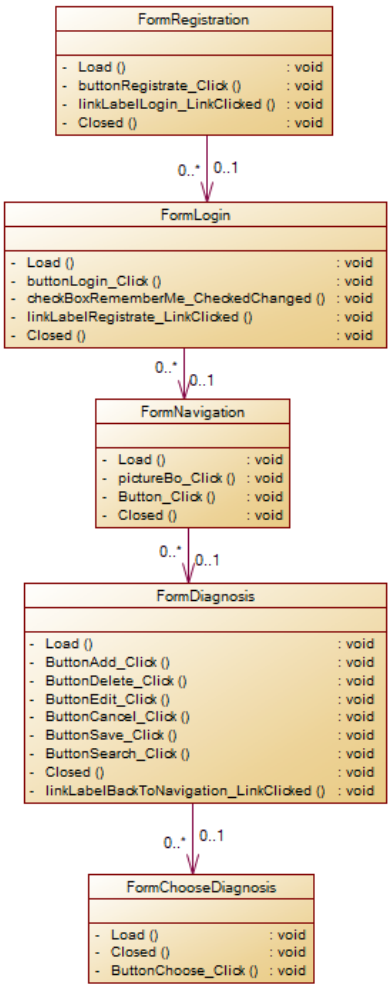

Рисунок 2.3 – Діаграма класів

В програмному рішені буде створено ряд класів для зберігання повного ряду інформації запису з бази даних, такі класи не містять логіки і лише зберігають та надають інформацію, яка було отримана з бази даних. Інший ряд класів це форми додатку, а саме вікна графічного інтерфейсу, з якими користувач взаємодіє напряму.

3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

Очевидно, що програма Windows Forms надає безліч цікавих рішень та переваг для вирішення проблеми пошуку роботи в Інтернеті. Однак слід зазначити, що вирішення деяких завдань на платформі є недоцільним через деякі обмеження. Отже, заявку слід розглядати як підсистему, інтегровану у продукти. Найпоширенішими та найбільш сумісними з такими програмами є веб-продукти. Насправді програма надаватиме майже всі основні функціональні можливості, пов'язані з певними класами користувачів, забезпечуючи більш зручні та гнучкі рішення та інструменти.

Аналіз існуючих продуктів допоміг зрозуміти потребу в певних технічних рішеннях, які зробили б систему занадто складною та невиправдано дорогою. Варто зазначити, що цей етап також показав основні проблеми систем, пов'язані з зовнішнім виглядом користувальницького інтерфейсу, і допоміг усунути деякі непотрібні функціональні можливості.

3.1 Організація вибірки інформації з бази даних

Інформація відбирається за допомогою запитів, представлених у цьому розділі. Переважно, вибірка даних використовується при використанні подібного висловлювання, що дозволяє виконувати зручний пошук за формами спеціального додатка [3].

Приклад зразка запиту, розподіленого в коді програми:

виберіть * із [таблиці], де [вікно пошуку], наприклад, '% [текстове поле форми]%'

Далі будуть продемонстровані потенційні запити щодо вибірки даних, що підкреслює ефективність курсу баз даних.

Запит 1. Вибірка даних однієї таблиці. Формулювання запиту: вибрати прізвище, ім’я та прізвище з таблиці «Doctor»(рисунок 3.1).

```
select Doc_surname, Doc_name, Doc_middle_name from Doctor
```

100 %

Results | Messages

	Doc_surname	Doc_name	Doc_middle_name
1	Рудак	Володимир	Мирославович
2	Коваль	Володимир	Вікторович
3	Мікула	Олег	Орестович
4	Гушпіт	Назар	Андрійович
5	Бойко	Ейнат	Петрівна
6	Разборський	Дмитро	Петрович
7	Дисевич	Денис	Олегович

Рисунок 3.1 – Запит 1

Запит 2. Вибірка з використанням оператора (природного) з'єднання. Формулювання запиту: вибрати прізвище, ім’я, прізвище лікарів та спеціалізацію з таблиці «Doctor» та «Specialty» шляхом з'єднання їх за кодом спеціалізації(рисунок 3.2).

```
select Doc_surname, Doc_name, Doc_middle_name, Sp_name As Specialty
from Doctor
inner join Specialty
on Doctor.Sp_id = Specialty.Sp_id
```

100 %

Results | Messages

	Doc_surname	Doc_name	Doc_middle_name	Specialty
1	Рудак	Володимир	Мирославович	Хірург
2	Коваль	Володимир	Вікторович	Проктолог
3	Мікула	Олег	Орестович	Кардіолог
4	Гушпіт	Назар	Андрійович	Ортопед
5	Бойко	Ейнат	Петрівна	Акушер - гінеколог
6	Разборський	Дмитро	Петрович	Офтальмолог
7	Дисевич	Денис	Олегович	Реабілітолог

Рисунок 3.2 – Запит 2

Запит 3. Вибірка з використанням шаблону. Формулювання запиту: вибрати всіх клієнтів з першою групою крові з таблиці «Client» (рисунок 3.3).

```
select * from Client where Cl_blood_group like 'Перша%'
```

Results | Messages

	Cl_id	Cl_surname	Cl_name	Cl_middle_name	Cl_birth_date	Cl_gender	Cl_blood_group
1	1	Верцімаха	Андрій	Віталійович	1998-12-06	Чоловіча	Перша позитивна
2	3	Зюбрецька	Інна	Володимирівна	2000-09-30	Жіноча	Перша негативна

Рисунок 3.3 – Запит 3

Запит 4. Вибірка інформації в заданому діапазоні. Формулювання запиту: вибрати прізвища клієнтів з вартістю лікування між 10000 і 20000 з таблиці «Treatment» та «Client» шляхом з'єднання їх за кодом клієнта(рисунок 3.4).

```
select Cl_surname, Tr_cost As 'Вартість'
from Client
inner join Treatment
on Client.Cl_id = Treatment.Cl_id
where Tr_cost between 10000 and 20000
```

Results | Messages

	Cl_surname	Вартість
1	Максимчук	10000
2	Коцур	12000
3	Лещишин	15000
4	Барнецький	20000
5	Халак	11000

Рисунок 3.4 – Запит 4

Запит 5. Вибірка з використанням механізму підзапитів. Формулювання запиту: вибрати прізвища клієнтів і вартістю лікування з таблиці «Client» та «Treatment», причому включаючи, тільки те лікування, у якого вартість більше середнього значення серед усіх лікувань(рисунок 3.5).

```
select Cl_surname, Tr_cost As 'Вартість'
from Client
inner join Treatment
on Client.Cl_id = Treatment.Cl_id
where Tr_cost >(select AVG (Tr_cost) from Treatment)
```

Results | Messages

	Cl_surname	Вартість
1	Лещишин	15000
2	Барнецький	20000
3	Мерц	25000

Рисунок 3.5 – Запит 5

Запит 6. Вибірка з використанням сортування. Формулювання запиту: вибрати все про лікарів та спеціалізацію з таблиці «Doctor» та «Specialty» шляхом з'єднання їх за кодом спеціалізації і відсортувати записи за прізвищем лікарів(рисунок 3.6).

```
select Doc_surname, Doc_name, Doc_middle_name, Doc_birth_date,
Doc_gender, Doc_category, Sp_name As Specialty
from Doctor
inner join Specialty
on Doctor.Sp_id = Specialty.Sp_id
order by Doc_surname
```

100 %

Results | Messages

	Doc_surname	Doc_name	Doc_middle_name	Doc_birth_date	Doc_gender	Doc_category	Specialty
1	Бойко	Ейнат	Петрівна	2001-04-25	Жіноча	Друга	Акушер - гінеколог
2	Гушпіт	Назар	Андрійович	2001-06-23	Чоловіча	Друга	Ортопед
3	Дисевич	Денис	Олегович	1977-09-10	Чоловіча	Перша	Реабілітолог
4	Коваль	Володимир	Вікторович	2000-10-17	Чоловіча	Вища	Проктолог
5	Мікула	Олег	Орестович	2000-10-14	Чоловіча	Вища	Кардіолог
6	Разборський	Дмитро	Петрович	1989-06-25	Чоловіча	Перша	Офтальмолог
7	Рудак	Володимир	Мирославович	2000-07-27	Чоловіча	Вища	Хірург

Рисунок 3.6 – Запит 6

3.2 Розробка представлень для відображень результатів вибірки

Представлення – це динамічна таблиця, що служить для відображення результатів вибірки інформації. Представлення є зручним інструментом для роботи з таблицями бази даних.

У базі даних розроблено представлення: «Діагнози»(рисунок 3.7).

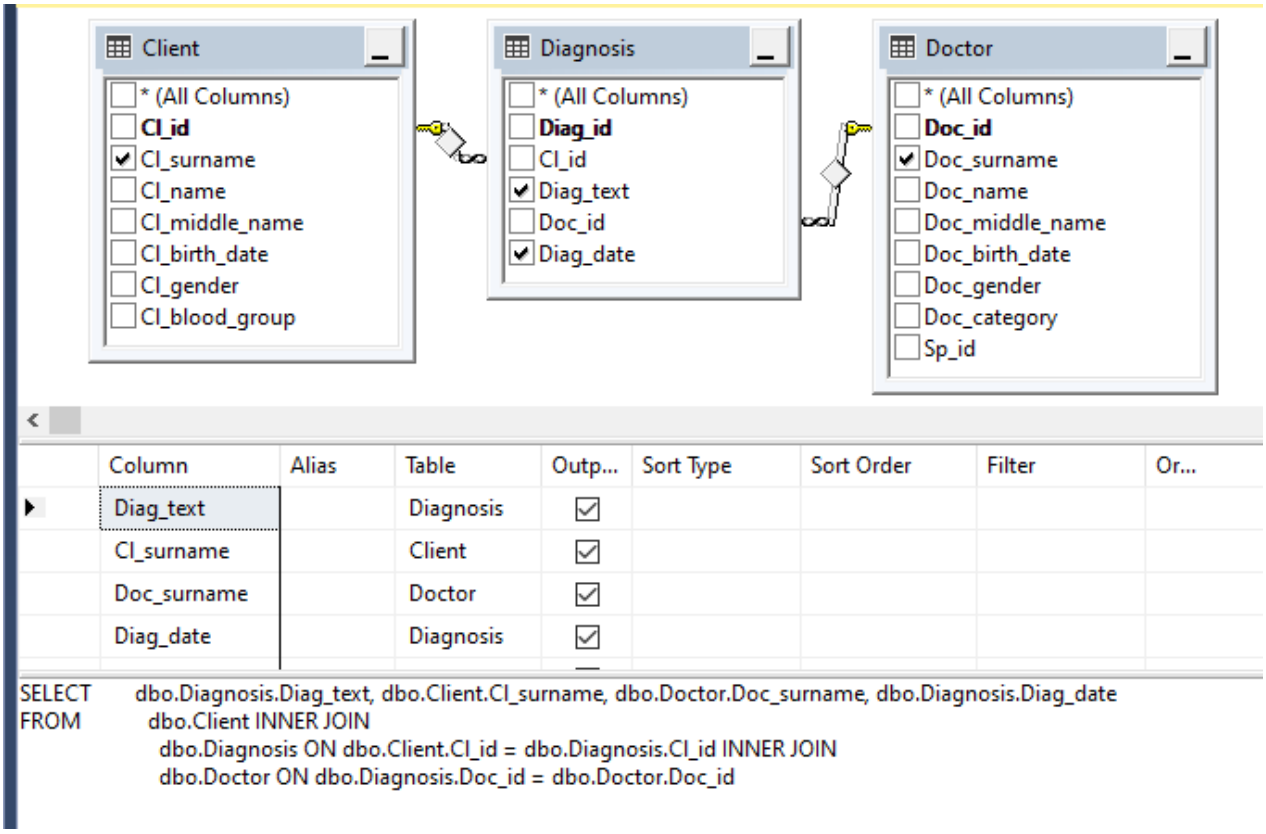

Рисунок 3.7 – Представлення «Діагнози»

Дане представлення містить в собі прізвище клієнта, прізвище лікаря, поставлений діагноз та дата встановлення діагнозу. Результат роботи предсталвення зображено на рисунку 3.8.

	Diag_text	Cl_surname	Doc_surname	Diag_date
1	Тахіокардія, папарам	Максимчук	Мікула	2020-03-12
2	ішемічна хвороба серця	Коцур	Мікула	2020-12-09
3	Видалення стороннього предмету	Зюбрецька	Рудак	2020-12-11
4	Акомодація мязу очей	Верцімаха	Разборський	2020-12-11
5	Помутніння хрусталика	Лещишин	Разборський	2020-12-11
6	Катаракта	Барнецький	Разборський	2020-12-11
7	Постороннiй предмет у оцi	Халак	Разборський	2020-12-11
8	Плоскостопість	Мерц	Гушпіт	2020-12-11

Рисунок 3.8 – Робота представлення «Діагнози»

3.3 Розробка збережувальних процедур

При розробці додатків, заснованих на принципі "клієнт - сервер", для полегшення реалізації будь-яких операцій з даними використовуються

механізми, за допомогою яких можна створювати підпрограми, що працюють на сервері, і контролювати обробку інформації. Ці механізми називаються процедурами збереження.

У дипломному проекті було розроблено ряд процедур зберігання, які використовуються клієнтською частиною системи для введення та оновлення даних певної таблиці [2].

Код процедур зберігання для введення та оновлення цих спеціалізацій наведено в лістингу 5.1.

Лістинг 5.1 – Збережувальні процедури для таблиці «Specialty»

```
ALTER procedure [dbo].[Specialty_Update]
(
    @Sp_id int output,
    @Sp_name nvarchar(50)
)
as
    update Specialty set Sp_name = @Sp_name
    where Sp_id = @Sp_id
ALTER procedure [dbo].[Specialty_Insert]
(
    @Sp_id int output,
    @Sp_name nvarchar(50)
)
as
   insert into Specialty(Sp_name)
   values(@Sp_name)
   set @Sp_id = SCOPE_IDENTITY()
```

Код збережувальних процедур для внесення та оновлення даних клієнтів знаходиться в лістингу 5.2.

Лістинг 5.2 – Збережувальні процедури для таблиці «Client»

```
ALTER procedure [dbo].[Client_Update]
(
    @Cl_id int output,
    @Cl_surname nvarchar(50),
    @Cl_name nvarchar(50),
    @Cl_middle_name nvarchar(50),
    @Cl_birth_date date,
    @Cl_gender nvarchar(50),
    @Cl_blood_group nvarchar(50)
)
as
    update Client set Cl_surname = @Cl_surname, Cl_name = @Cl_name,
Cl_middle_name = @Cl_middle_name, Cl_birth_date = @Cl_birth_date,
      Cl_gender = @Cl_gender, Cl_blood_group = @Cl_blood_group
        where Cl_id = @Cl_id
ALTER procedure [dbo].[Client_Insert]
(
```

```
    @Cl_id int output,
    @Cl_surname nvarchar(50),
    @Cl_name nvarchar(50),
    @Cl_middle_name nvarchar(50),
    @Cl_birth_date date,
    @Cl_gender nvarchar(50),
    @Cl_blood_group nvarchar(50)
)
as
    insert into Client(Cl_surname, Cl_name, Cl_middle_name, Cl_birth_date,
Cl_gender, Cl_blood_group)
    values(@Cl_surname, @Cl_name, @Cl_middle_name, @Cl_birth_date, @Cl_gender,
@Cl_blood_group)
          set @Cl_id = SCOPE_IDENTITY()
```

Код збережувальних процедур для внесення та оновлення даних клієнтів знаходиться в лістингу 5.3.

Лістинг 5.3 – Збережувальні процедури для таблиці «Doctor»

```
ALTER procedure [dbo].[Doctor_Update]
(
    @Doc_id int output,
    @Doc_surname nvarchar(50),
    @Doc_name nvarchar(50),
    @Doc_middle_name nvarchar(50),
    @Doc_birth_date nvarchar(500),
    @Doc_gender nvarchar(50),
    @Doc_category varchar(50),
      @Sp_id int
)
as
    update Doctor set Doc_surname = @Doc_surname, Doc_name = @Doc_name,
Doc_middle_name = @Doc_middle_name,
      Doc_birth_date = @Doc_birth_date, Doc_gender = @Doc_gender, Doc_category =
@Doc_category, Sp_id = @Sp_id
          where Doc_id = @Doc_id
ALTER procedure [dbo].[Doctor_Insert]
(
    @Doc_id int output,
    @Doc_surname nvarchar(50),
    @Doc_name nvarchar(50),
    @Doc_middle_name nvarchar(50),
    @Doc_birth_date nvarchar(500),
    @Doc_gender nvarchar(50),
    @Doc_category varchar(50),
      @Sp_id int
)
as
    insert into Doctor(Doc_surname, Doc_name, Doc_middle_name, Doc_birth_date,
Doc_gender, Doc_category, Sp_id)
    values(@Doc_surname, @Doc_name, @Doc_middle_name, @Doc_birth_date,
@Doc_gender, @Doc_category, @Sp_id)
          set @Doc_id = SCOPE_IDENTITY()
```

Код збережувальних процедур для внесення та оновлення даних клієнтів знаходиться в лістингу 5.4.

Лістинг 5.4 – Збережувальні процедури для таблиці «Diagnosis»

```
ALTER procedure [dbo].[Diagnosis_Update]
(
    @Diag_id int output,
    @Cl_id int,
    @Diag_text nvarchar(500),
    @Doc_id int,
    @Diag_date nvarchar(500)
)
as
    update Diagnosis set Cl_id = @Cl_id, Diag_text = @Diag_text, Doc_id =
@Doc_id, Diag_date = @Diag_date
          where Diag_id = @Diag_id
ALTER procedure [dbo].[Diagnosis_Insert]
(
    @Diag_id int output,
    @Cl_id int,
    @Diag_text nvarchar(500),
    @Doc_id int,
    @Diag_date nvarchar(500)
)
as
    insert into Diagnosis(Cl_id, Diag_text, Doc_id, Diag_date)
    values(@Cl_id, @Diag_text, @Doc_id, @Diag_date)
          set @Diag_id = SCOPE_IDENTITY()
```

Код збережувальних процедур для внесення та оновлення даних клієнтів знаходиться в лістингу 5.5.

Лістинг 5.5 – Збережувальні процедури для таблиці «Treatment»

```
ALTER procedure [dbo].[Treatment_Update]
(
    @Tr_id int output,
    @Doc_id int,
    @Cl_id int,
    @Diag_id int,
    @Tr_cost int,
    @Tr_begin_date nvarchar(500),
    @Tr_end_date nvarchar(500)
)
as
    update Treatment set Doc_id = @Doc_id, Cl_id = @Cl_id, Diag_id = @Diag_id,
Tr_cost = @Tr_cost,
      Tr_begin_date = @Tr_begin_date, Tr_end_date = @Tr_end_date
          where Tr_id = @Tr_id
ALTER procedure [dbo].[Treatment_Insert]
(
    @Tr_id int output,
    @Doc_id int,
    @Cl_id int,
    @Diag_id int,
    @Tr_cost int,
    @Tr_begin_date nvarchar(500),
```

```
    @Tr_end_date nvarchar(500)
)
as
    insert into Treatment(Doc_id, Cl_id, Diag_id, Tr_cost, Tr_begin_date,
Tr_end_date)
    values(@Doc_id, @Cl_id, @Diag_id, @Tr_cost, @Tr_begin_date, @Tr_end_date)
          set @Tr_id = SCOPE_IDENTITY()
```

Код збережувальних процедур для внесення та оновлення даних клієнтів знаходиться в лістингу 5.6.

Лістинг 5.6 – Збережувальні процедури для таблиці «Users»

```
ALTER procedure [dbo].[users_Update]
(
    @User_id int output,
    @User_username nvarchar(50),
    @User_password nvarchar(50),
    @User_permission nvarchar(20)
)
as
    update Users set User_username = @User_username, User_password =
@User_password, User_permission = @User_permission
          where User_id = @User_id

ALTER Procedure[dbo].[users_Insert]
      @User_id int output,
    @User_username nvarchar(50),
    @User_password nvarchar(50),
    @User_permission nvarchar(20)
as
      Insert Into users( User_username, User_password, User_permission)
      Values( @User_username, @User_password, @User_permission)
          set @User_id = SCOPE_IDENTITY()
```

3.4 Розробка механізмів управління даними в базі за допомогою тригерів

Тригери - це особливий вид процедур збереження, які виконуються автоматично (запускаються) при зміні даних таблиці. Тригери знаходять різноманітні програми, від перевірки даних до складних бізнес-правил. Особливо корисною особливістю тригерів є те, що вони мають доступ до записів до та після модифікації; таким чином, ви можете порівняти два записи та прийняти відповідне рішення [4].

Приклад створення та роботи тригера. Для таблиці обробки розроблено тригер, який реагує на додавання нового або оновлення існуючого запису до таблиці, якщо значення менше нуля, тобто від'ємне, дія не виконується [5]. Код запуску наведено в лістингу 6.1.

Лістинг 6.1 – Код тригера 1

```
set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
GO
create TRIGGER [dbo].[tr] ON [dbo].[Treatment]
AFTER INSERT,UPDATE
AS
BEGIN
IF EXISTS (SELECT * FROM [dbo].[Treatment] WHERE Tr_cost<0)
ROLLBACK TRAN
PRINT 'Помилка, ціна не може бути менша нуля'
SET NOCOUNT ON;
      END
```

Результат роботи тригера зображено на рисунку 3.9.

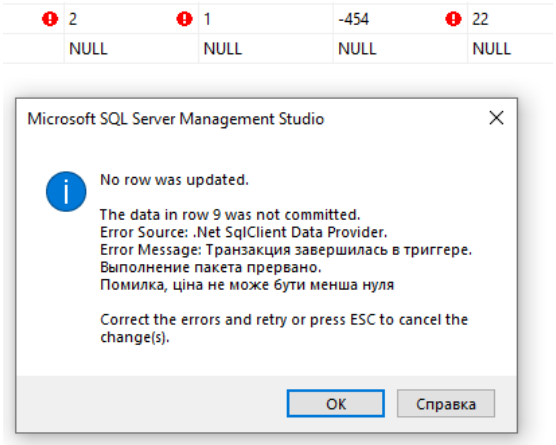

Рисунок 3.9 – Робота тригера 1

3.5 Класи для збереження та надання інформації:

Перелік класів для збереження та надання інформації виглядає наступним чином.

- Client – клас для зберігання та надання інформації про клієнтів;
- Specialty – клас для зберігання та надання інформації про спеціалізації;
- Doctor – клас для зберігання та надання інформації про лікарів;
- Users – клас для зберігання та надання інформації про користувачів;
- Treatment – клас для зберігання та надання інформації про лікування;
- Diagnosis – клас для зберігання та надання інформації про діагнози.

Перелік класів, що відповідають за форми(вікна програмного додатку):

- FormClient – клас для відображення вікна програми управління таблицею клієнтів;
- FormSpecialty – клас для відображення вікна програми управління таблицею спеціалізацій;
- FormDoctor – клас для відображення вікна програми управління таблицею лікарів;
- FormLogin – клас для відображення вікна програми авторизації користувача в системі додатка, також надає доступ до вікна реєстрації;
- FormTreatment – клас для відображення вікна програми управління таблицею лікувань;
- FormDiagnosis – клас для відображення вікна програми управління таблицею діагнозів;
- FormRegistrate – клас відображення вікна програми авторизації користувача в системі додатка, також надає доступ до вікна реєстрації;

– FormNavigation – клас відображення вікна програми навігації між таблицями та отримання інформації про вміст таблиць.

Усі вищеописані форми, крім форм реєстрації, авторизації та навігації, працюють по однаківій логіці та мають схожу структуру. Ці форми містять поле для відображення таблиці, яка отримується з бази даних. Це поле дозволяє обирати потрібний запис натисненням лівуї кнопки мишки, після чого інформація обраного запису буде загружена у відповідні поля, де по натисненню відповідної кнопки дії стає доступним їх обробка користувачем [8]. Окремо знаходиться понель з усіма текстовими полями для пошуку, яка активовуєця лише при натисненні функціональної кнопки пошуку, пріcля чого залишається лише вводити інформацію, по якій буде моментально проведено пошук у полі відображення даних з бази.

Окремий прелік форм призначений для вибору побічної інформації за відповідним кодом. Такі форми містять поле для відображення даних з бази та кнопку вибору, що обирає відповідний поточний вибраний елемент поля відображення даних та закриває відкриту форму для побічного вибору. Перелік таких форм:

– FormChooseSpecialty – клас для відображення вікна програми побічного вибору спеціалізацій;

– FormChooseDoctor – клас для відображення вікна програми побічного вибору лікарів;

– FormChooseDiagnosis – клас для відображення вікна програми побічного вибору діагнозів;

– FormChooseClient – клас для відображення вікна програми побічного вибору клієнтів.

Код відповідних описаних класів наведено в додатку А.

3.6 Опис технології реалізації та засобів створення додатку

Windows Forms - одна з найцікавіших функцій Microsoft .NET. Якщо ви знайомі з MFC (або Windows API), то Windows Forms - це хороший старт для роботи з бібліотекою класу .NET Framework, оскільки вона дозволяє писати традиційні графічні програми з вікнами, формами тощо. Після того, як ви почнете працювати з Windows Forms, ви зможете швидко зрозуміти .NET Framework.

Основна перевага написання програм Windows за допомогою Windows Forms полягає в тому, що Windows Forms гомогенізує (створює більш однорідну) модель моделі та усуває багато помилок та невідповідностей від використання Windows API. Наприклад, кожен досвідчений програміст Windows знає, що деякі стилі вікон можуть застосовуватися до вікна лише тоді, коли воно вже створене. Windows Forms значною мірою усуває це протиріччя. Якщо ви хочете вказати стиль для існуючого вікна, який можна призначити лише під час створення вікна, Windows Forms безпечно знищить вікно та відтворить його із зазначеним стилем. Крім того, бібліотека класу .NET Framework набагато багатша, ніж Windows API, і коли ви пишете програми за допомогою Windows Forms, ви отримуєте більше можливостей. Написання програми за допомогою Windows Forms вимагатиме менше коду, ніж програми, що використовують Windows API або MFC.

Ще однією перевагою Windows Forms є те, що ви використовуєте один і той же API, незалежно від вибраної мови програмування. У минулому вибір мови програмування керував вибором API. Якщо ви програмували в Visual Basic, ви використовували один API (реалізований у Visual Basic), тоді як програмісти C використовували Win32 API, а програмісти C ++ зазвичай використовували MFC. Програмісту MFC було важко перейти на Visual Basic і навпаки. Але зараз такого немає. Усі програми, які використовують Windows Forms, використовують один API із бібліотекою класу .NET Framework. Знання

одного API дозволить програмісту писати програми практично будь-якою мовою, яку він вибере.

Windows Forms - це не менше, ніж сучасна модель програмування для графічних інтерфейсів. На відміну від моделі програмування Win32, яка має багато спільного з Windows 1.0, нова модель була розроблена з урахуванням усіх сучасних вимог. Мета цієї статті - ознайомити читача з моделлю програмування Windows Forms. Для компіляції та запуску наведених нижче прикладів коду на комп'ютері потрібно встановити пакет SDK Microsoft .NET Framework.

У Windows Forms термін "форма" є синонімом вікна верхнього рівня. Основне вікно програми - форма. Будь-які інші вікна верхнього рівня, які має програма, також мають фігури. Діалогові вікна також вважаються формами. Незважаючи на назву, програма для використання Windows Forms не схожа на форми. Як і традиційні програми Windows, програми мають повний контроль над подіями у власних вікнах.

Програмісти бачать Microsoft .NET через бібліотечну лінзу класу .NET Framework. Уявіть, що MFC на порядок більше, і ви отримаєте точне зображення ширини та глибини бібліотеки класу .NET Framework. Щоб усунути суперечності в позначеннях та забезпечити організацію багатьох сотень класів, бібліотека класів .NET Framework розділена на ієрархічні розділи за іменами. Кореневий розділ System визначає основні типи даних, що використовуються усіма програмами .NET [9].

Програми, які використовують Windows Forms, використовують класи System.WinForms. Цей розділ включає класи, такі як Form, що імітує поведінку вікон або форм; Меню, що представляє меню; Буфер обміну, який дозволяє програмам Windows Forms використовувати буфер обміну. Він також містить численні класи, що забезпечують елементи керування, такі як: Button, TextBox, ListView, MonthCalendar тощо. Ці класи можуть бути включені в програму, використовуючи лише ім'я класу, або використовуючи повне ім'я, наприклад: System.WinForms. Кнопка.

Майже кожна програма, написана в Windows Forms, базується на системному класі, похідному від System.WinForms.Form. Приклад цього класу представляє головне вікно програми. System.WinForms.Form має безліч властивостей і методів, які мають багатий інтерфейс програмування форм. Хочете знати розміри області форми клієнта? У Windows ви б викликали функцію GetClientRect API. У Windows Forms потрібно використовувати властивості ClientRectangle або ClientSize.

Додатки на базі Windows Forms, які використовують кнопки, списки та інші типи компонентів Windows, використовують класи управління System.WinForms, значно спрощуючи програмування управління. Хочете створити стилізовану кнопку з фоновим зображенням? Нема проблем. Включіть бажане зображення в об'єкт System.Drawing.Bitmap і призначте його властивості кнопці BackgroundImage. Як щодо управління кольором? Ви коли-небудь пробували регулювати колір тла текстового поля? У Windows Forms це просто: вам просто потрібно призначити властивості кольору BackColor, а вся інша система зробить сама.

Іншим важливим "будівельним блоком" програми, яка використовує Windows Forms, є клас System.WinForms, який називається Application. Цей клас містить статичний метод запуску, який завантажує програму та відображає вікно.

Ви кажете: якщо програми, які є формами Windows, не обробляють повідомлення, як вони реагують на введення користувачами або знають, коли малювати? У багатьох класах є віртуальні методи, які можна замінити ... Наприклад, System.WinForms.Form містить віртуальний метод OnPaint, який викликається, коли потрібно оновити клієнтську область форми. OnPaint - це один із багатьох віртуальних методів, який можна замінити у похідному класі для формування інтерактивних форм.

Іншим важливим аспектом моделі програмування Windows Forms є механізм, який форми використовують для реагування на введення в меню, елементи керування та інші програми графічного інтерфейсу. Традиційні

програми Windows обробляють повідомлення WM_COMMAND та WM_NOTIFY, використовуючи події процесу

Форми Windows. У C # та інших мовах, які підтримують .NET Common Language Runtime (CLR), події є членами першого типу класу разом із методами, полями та властивостями [7]. Практично всі контрольні класи Windows Forms (а також багато некерованих класів) створюють події. Наприклад, кнопка (екземпляр System.WinForms.Button) створює подію Click при натисканні. Форма, яка повинна відповідати натисканню кнопки, може використовувати код, щоб зв’язати кнопку з обробником події Click.

EventHandler - це спеціальний обробник подій, який виконує метод OnButtonClicked, коли MyButton створює подію Click. Перший параметр OnButtonClicked визначає об'єкт, що спричинив подію. Другий параметр в основному має сенс для події Click, але використовується деякими іншими типами подій для передачі додаткової інформації.

3.7 Тестування та оцінка якості пс

В цьому розділі буде описано роботу програмного додакту, етапи його роботи, опис взаємодії з усіма типами форм програмного рішення. Усі кроки роботи додатку буде супроводжено знімками екрану для наглядності та демонстрування правильності роботи форм та їх елементів .

При зупуску програми відкривається форма входу, вигляд форми входу в систему програми за логіном та паролем зображено на рисунку 3.10.

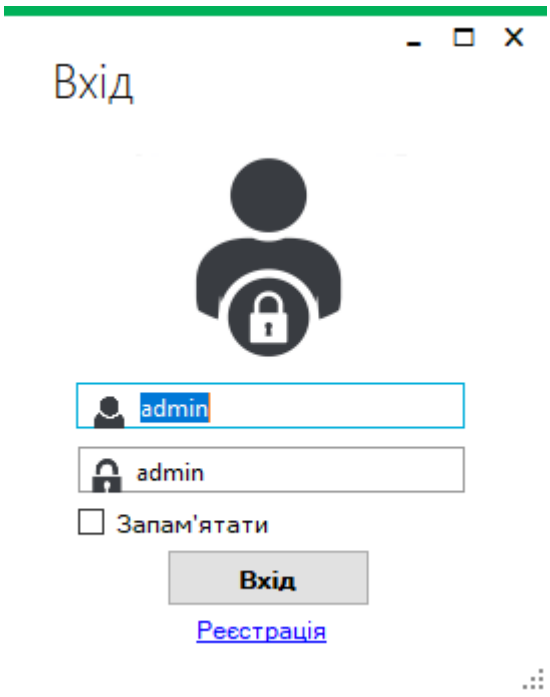

Рисунок 3.10 – Форма входу

Якщо не введено пароль або ім’я користувача та натиснуто кнопку входу в систему буде продемонстровано діалогове вікно з відповідним повідомленням. Повідомлення про відсутність імені користувача продемонстровано на рисунку 3.11.

Також на цій формі присутня функція Запам’ятати, що дозволяє зберігати дані входу після закриття та запуску додатку.

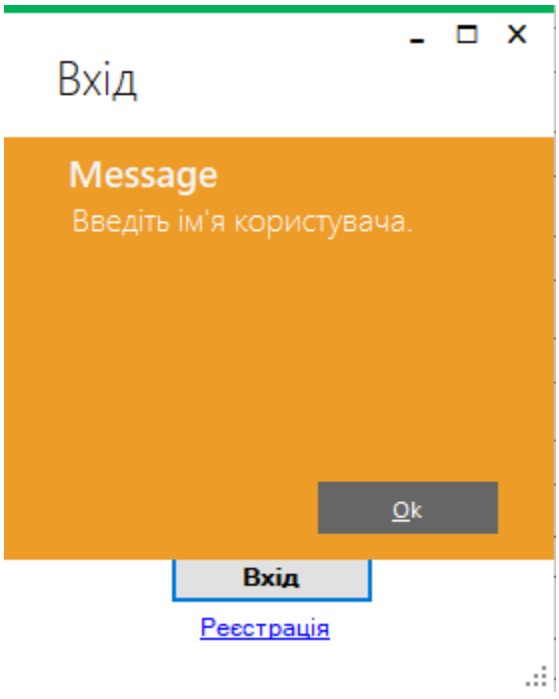

Рисунок 3.11 – Повідомлення про відсутність імені користувача

При введені невірних даних та натиснені на кнопку входу також буде виведено відповідне повідомлення, яке продемонстровано на рисунку 3.12.

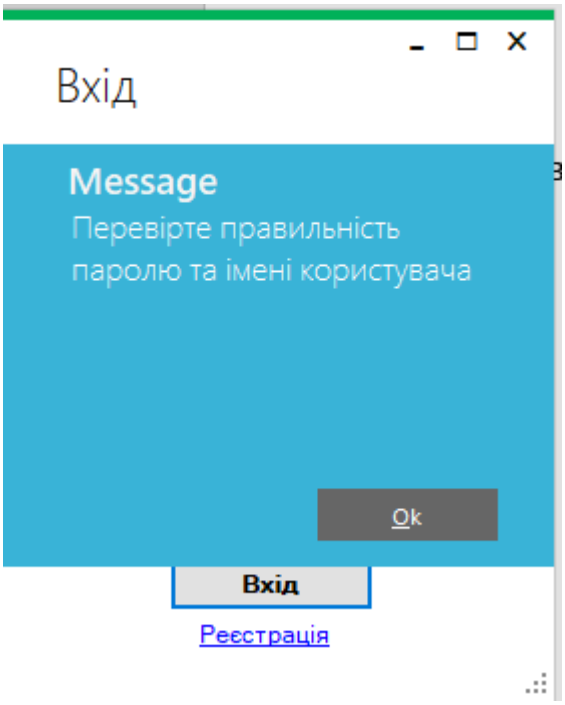

Рисунок 3.12 – Повідомлення про неправильність даних входу

З цієї форми мона перейти на форму реєстрації, що дозволить зареєструватись в системі новому користувачу. Для переходу на форму реєстрації потрібно натиснути по текстовому посиолані Реєстрація. Вигляд форми реєстрації продемонстровано на рисунку 3.13.

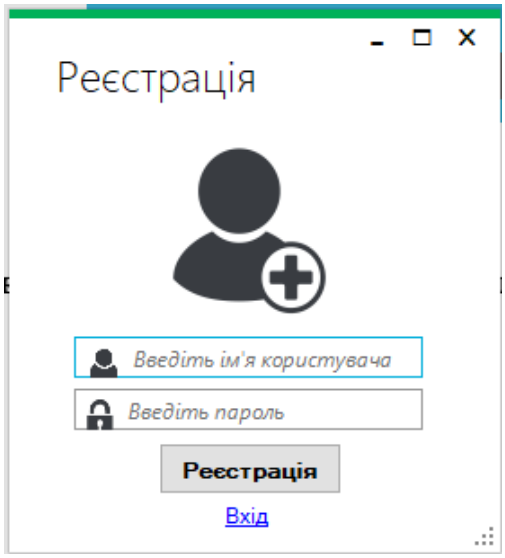

Рисунок 3.13 – Форма реєстрації

З форми реєстрації також можна повернутись на форму входу по натиснені на текстове посилання Вхід або виконати успішну реєстрацію. На даній формі користувач додатку придумує власне унікальне ім’я користувача та пароль. При введені імені користувача, що вже існує в системі буде виведено відповідне повідомлення, яке продемонстроване на рисунку 3.14.

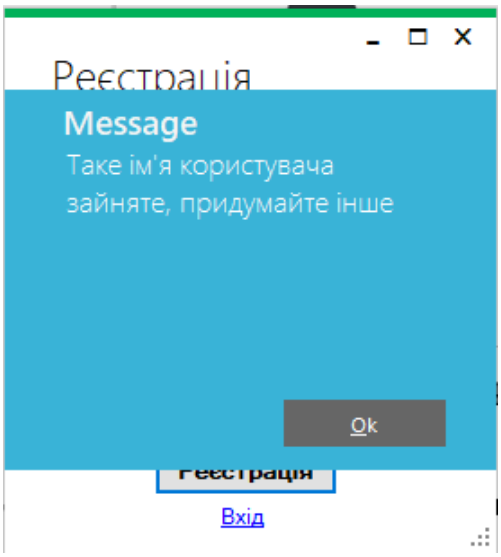

Рисунок 3.14 – Повідомлення про зайнятість введеного імені користувача

Після успішного входу в систему відкривається форма навігації, яка демонструє користувачу усі доступні таблиці системи та надає можливість

отримати інформацію про таблицю. Вигляд форми навігації продемонстровано на рисунку 3.15.

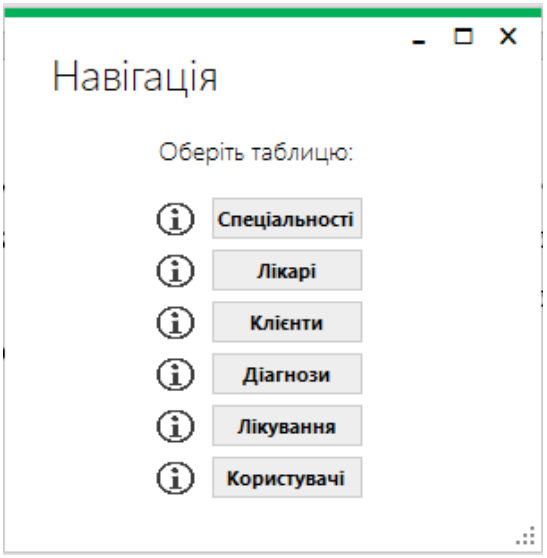

Рисунок 3.15. – Форма навігації

На даній формі присутні кнопки, які виконують відкриття форми відповідно обраної користувачем таблиці. Та інформаційні іконки, які надають коротку інформацію про таблицю, навпроти якої ця іконка знаходиться. Вигляд інформування користувача про таблицю лікарі продемонстровано на рисунку 3.16.

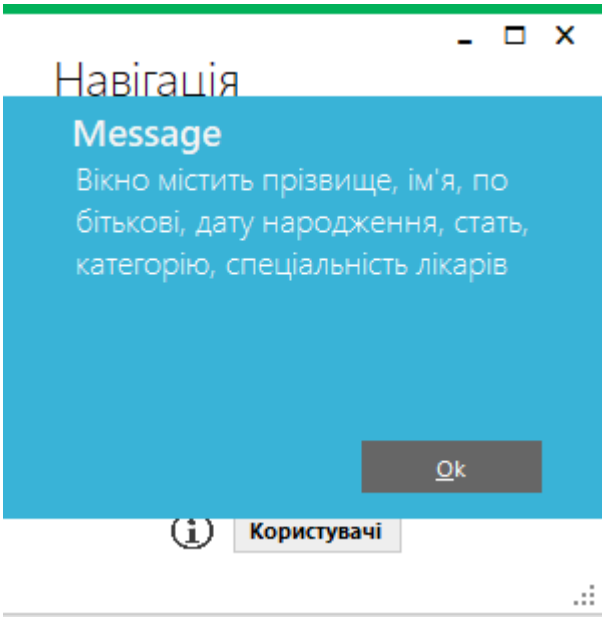

Рисунок 3.16 – Вигляд інформування про таблицю

При натисненні кнопки Лікарі на формі навігації буде відкрито вікно для управління відповідною таблицею. Вигляд відкритої форми Лікарі продемонстровано на рисунку 3.17.

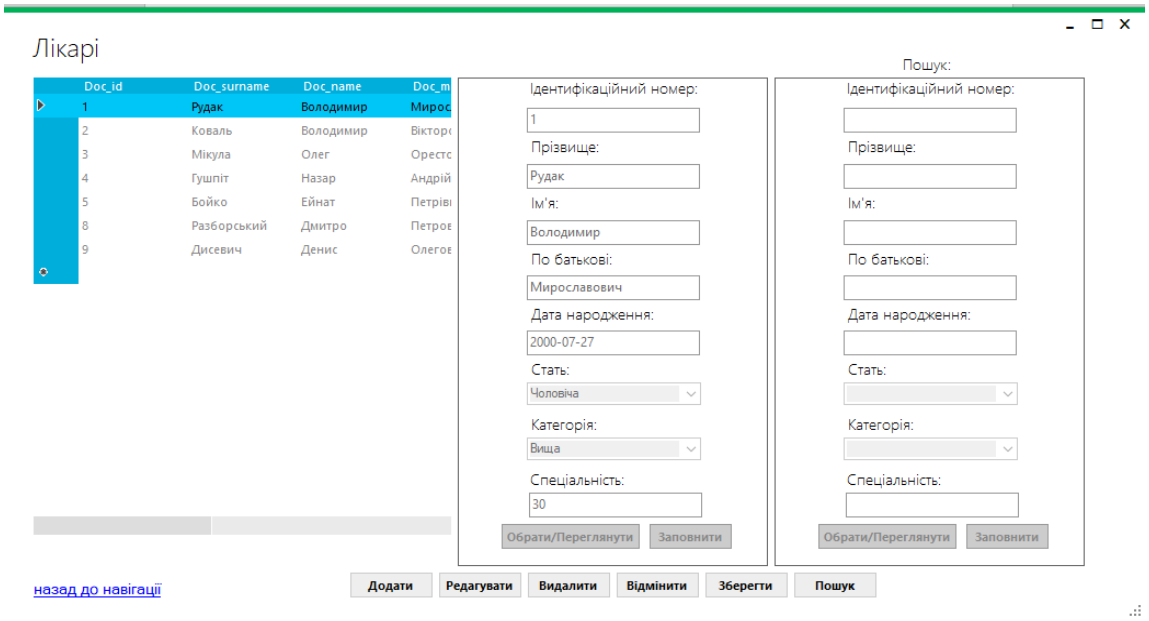

Рисунок 3.17 – Вигляд форми управління таблицею Лікарі

Усі інші форми, призначені для управління таблицями баз даних, виглядають подібними, відрізняючись лише кількістю та змістом текстових полів. Форма має такі елементи: поле для відображення інформації з бази даних, панель, що містить текстові поля для редагування, додавання та зміни обраного запису, панель з полями для пошуку відповідного запису в полі бази даних. Кнопки виконують функції відповідно до їх назви.

Панелі змінюють властивість своєї активності, коли ви натискаєте певні кнопки, що дають доступ до поля відповідної панелі. Після натискання кнопки додавання, редагування панель цих операцій активується, а панель пошуку активується. Кнопка пошуку відповідно активує рядок пошуку. Кнопка "Скасувати" деактивує панелі та скасовує раніше введені критерії пошуку, що відображаються в полі даних усі наявні в базі даних.

Також є текстове посилання для повернення до навігаційної форми та вибору наступної таблиці, необхідної користувачеві. Наступний вид форми це форми побічного вибору, які відкриваються при натисненні кнопки

Обрати/Переглянути. На даній формі присутня одна така кнопка, яка відкриває форму для обрання або перегляду інформації про спеціалізації лікарів. Кнопка Заповнити слугує оновленням даних в відповідному текстовому полі після обрання потрібного запису на формі побічного вибору спеціалізацій. Вигляд побічної форми вибору спеціалізації продемонстровано на рисунку 3.18.

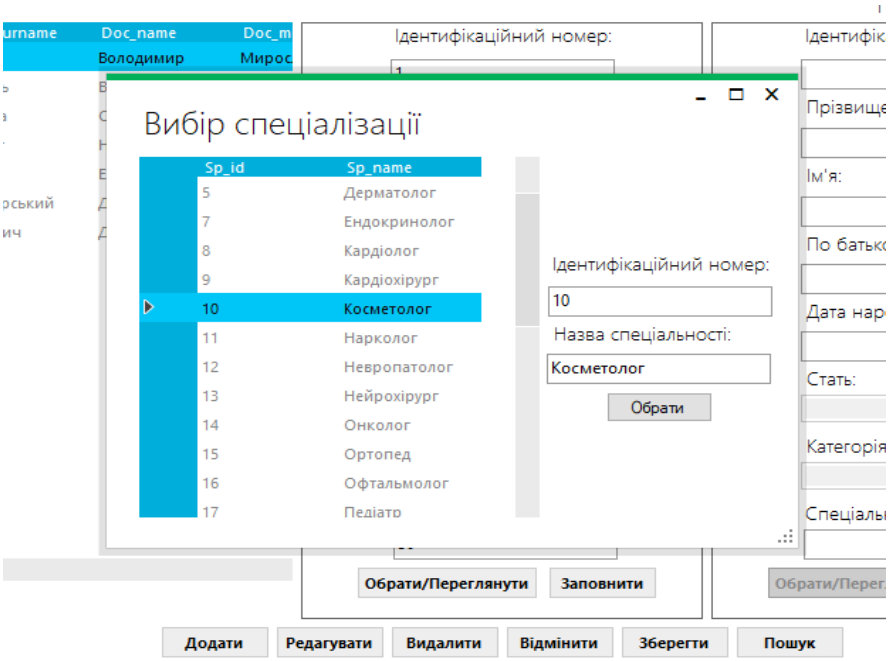

Рисунок 3.18 – Вигляд форми побічного вибору спеціалізацій

Форма містить знайоме поле для відображення даних з бази даних, текстові поля, що відповідають таблиці, та кнопку вибору, яка зберігає виділення в глобальній змінній і буде доступна в поточній формі таблиці після натискання кнопки вторинного форма відбору закривається [6].

Наступним клацанням на формі таблиці кнопки Заповнення буде введено вибраний код запису у відповідне текстове поле.

Деякі текстові поля недоступні для редагування за певних обставин програми, щоб запобігти введенню інформації неправильного формату або неправильного значення, що може пошкодити правильність та точність програми як її клієнтської частини, так і сервера.

З будь-якої форми ви можете перейти до потрібної іншої бажаної форми. Додаток також закривається з будь-якої точки програми, натискаючи кнопку, щоб закрити вікно.

Після тестування програми не було виявлено помилок, які заважали б запланованій та належній роботі системи.

Тестування показало, що система може виконувати всі передбачені функції, а саме: пошук і видалення даних, додавати, змінювати і переглядати ці дані

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ

4.1 Аналіз ринку

Метою техніко - економічного розділу дипломної роботи є проведення економічних розрахунків, спрямованих на визначення економічної ефективності програмного забезпечення для оптимізації роботи куратора групи та прийняття рішення про його подальший розвиток та впровадження або недоцільність відповідного розвитку. Для проведення цього дослідження необхідна низка розрахунків.

Було досліджено і проаналізовано інформаційні системи на основі 1С та bas бухгалтерії.

Характеристика програмного продукту:

Програмний продукт розроблений на досить популярній об’єктно-орієнтованій мові програмування C#. Додаток розроблено під операційну систему Windows, яка є найпопулярнішою в світі серед десктопних операційних систем.

Ринок збуту такого програмного продукту є цілеспрямованим та перспективним. Кожен користувач комп’ютера, який працює на Windows, зможе скористатись розробленим програмним продуктом. Ринок збуту зменшується та складається з тих користувачів, які зацікавлені у подібному додатку. Це не обов’язково користувач, який зацікавлений сферою інформаційних технологій. Після впровадження продукту на ринок планується його подальша підтримка та оновлення.

Через відкритість та масштабність ринку десктопних додатків для платформи Windows конкуренція є відповідною.

Далі перераховано характеристики експлуатації програмного продукту:

– Для користування продуктом потрібний мінімум: комп’ютер мінімальної продуктивності, операційна система Windows, встановлений додаток, щ обув розроблений в ході виконання дипломного проекту.

– Мінімальні можливості для клієнта продукту є можливість знайти користування функціональністю вибірки даних системи.

– Потенційним замовником продукту вважається платна поліклініка.

– Основним ринком реалізації продукту будуть різні призначенні для цього платформи в інтернеті. В якості додаткових ринків реалізації продукту можуть виступати різноманітні тематичні форуми та веб-сайти.

– На даний продукт очікується великий попит.

– Продукт реалізовано на некомерційній основі і буде знаходитись у вільному доступі абсолютно безкоштовно.

– Сервісне обслуговування буде організовано для початку єдиним адміністратором інформаційного простору продукту, який буде мати усі доступні права доступу.

– Головними конкурентами на ринку є усі аналогічні додатки.

– Продукція найбільших конкурентів являє собою краще оформлений дизайн додатку та більш інтерактивний користувацький інтерфейс. За кожним популярним продуктом закріплена ціла команда і навіть організація, яка займається його підтримкою.

4.2 Розрахунок витрат на проектування

Зведені розрахунки витрат наведено у таблиці 9.1.

Таблиця 9.1 – Зведені розрахунки матеріальних витрат

№ п/п	Найменування матеріальних ресурсів	Факт. витрачено матеріалів	Ціна за одиницю, грн.	Сума, грн	Транспортні витрати (10% від суми)	Загальна сума, грн
1.	Допоміжна література	1	600	600	60	660
2.	Папір (формат А4)	2	80	160	16	176

3.	Ручка кулькова	2	10	20	2	22
4.	Олівець простий	2	10	20	2	22
5.	Диски CD-R	2	15	30	3	33
6.	Зошит, 96 арк.	1	50	50	5	55
7.	Тонер для принтера	1	90	90	9	99
8.	Канцелярські маркери	2	20	40	4	44
Разом						1111,00

З кожним роком сучасні технології розвиваються, розробка десктопних додатків також не стоїть на місці. Їх реалізація сприяє покращенню комунікабельності, якості спілкування, вирішення різноманітних складних завдань. Хорошим професіоналам це приносить досить високий прибуток. Заробітна плата учасників проектування проекту наведено у таблиці 9.2.

Таблиця 9.2 – Розрахунок заробітної плати проектувальників

№ п/п	Посада виконавця	Оклад, грн/міс	Відрахування грн/міс	Кількість		Сума з/п, грн.
				чол.	місяців	
1	Технік-програміст	5200	950,5	2	6	4249,5
2	Тестувальник	4600	910	1	6	3690
Усього зарплати:						175122

Загальний кошторис витрат протягом усього проектування та по його завершенні наведено у таблиці 9.3.

Таблиця 9.3 – Кошторис витрат

Найменування статей витрат	Сума, грн	Обґрунтування
1. Зарплата проектувальників.	175122	При розробці мобільного додатку потрібно розробити великий функціонал і відповідно протестувати

2. Відрахування на соціальні потреби.	38526	22% від суми зарплат
3. Контрагентські роботи і послуги.	480	Прибирання приміщення
4. Витрати на відрядження.	870	Похід усієї команди в кінотеатр
5. Інші прямі витрати.	1111	Придбання канцелярських товарів, паперу, маркерів.
6. Усього прямих витрат.	216109	175122+38526+480+870+1111= 216109
7. Накладні витрати.	4400	25% - витрати на опалення, електроенергію, купівлю комплектуючих для ремонту
8. Планові накопичення.	3670	Преміювання працівників за хорошу роботу та якісне виконання поставлених задач (20%).
9. Усього, кошторисна вартість проекту.	224179	216109+4400+3670= 226768
10. Податок на додану вартість.	45353,6	20% від усього кошторису
11. Загалом, договірна ціна розробки З/П.	269532,6	224179+45353,6= 269532,6

4.3 Обґрунтування необхідності розробки

Попит на настільні пристрої неухильно зростає. І це дає підстави сперечатися щодо актуальності та доцільності процесу розробки таких настільних додатків. Основним завданням творців продуктів є оцінка цільової аудиторії, для якої він створений, оскільки лише корисна розробка зможе отримати реальне визнання з боку користувачів.

Розробка програми, яка насправді існуватиме як інформаційна система, особливо важлива для просування не тільки ринку настільних додатків, а й інших галузей інформаційних технологій тощо.

Таких програмних програм для задоволення описаних потреб в українському регіоні дуже мало, що дає шанс успіху розробленому додатку та надає певного значення.

Windows зазвичай називають найкращою ОС для початківців користувачів. За даними NetMarketShare на 2020 рік, глобальна частка Windows на настільних ПК становить 87%, тоді як MacOS займає 9% ринку, а Linux - лише 2%. Візуальний графік для кращого розуміння показаний на малюнку 9.1.

Частково мізерна частка Linux пов'язана з тим, що для більшості звичайних користувачів працювати набагато складніше, ніж з Windows, і тому привабливість Linux для домашнього використання дуже обмежена.

Основним сектором додатків Linux є серверне програмне забезпечення.

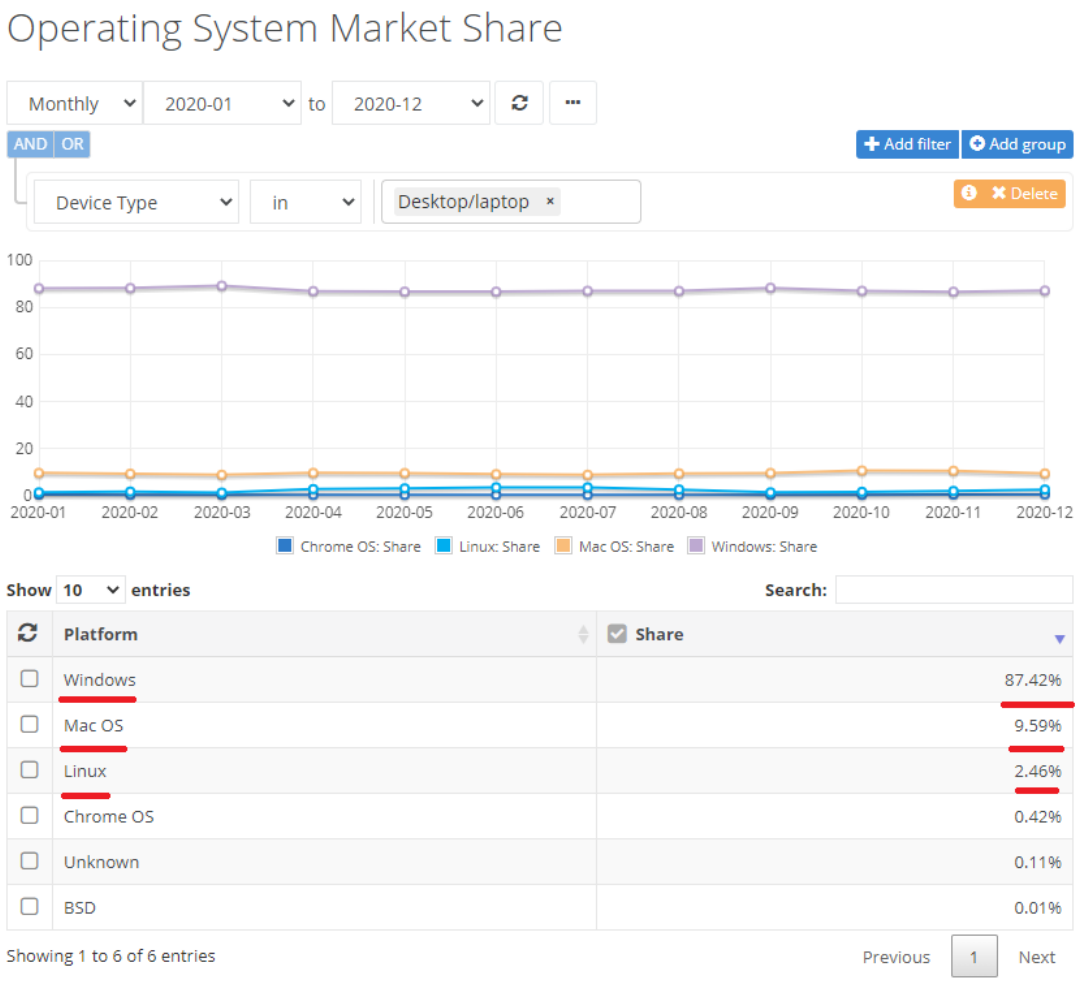

Рисунок 9.1 – Частка ринку різних ОС

Оскільки Windows є широко використовуваною операційною системою, кожен користувач час від часу стикався з проблемами безпеки та стабільності. Спочатку Windows була розроблена для націлювання на однокористувацьких ПК без підключення до мережі та не мала вбудованих функцій безпеки. У Windows шкідливі програми та віруси легко отримують доступ до системних файлів і можуть завдати великої шкоди. Крім того, максимальна кількість вірусів створюється під Windows (враховуючи величезну частку ринку). Варто зазначити, що Linux також не захищений від атак на систему, але якщо ви будете дотримуватися найпростіших правил і не надавати права суперкористувача усьому, що ви запускаєте, ви, мабуть, будете в безпеці, ніж Windows.

Також варто зазначити, що Microsoft зараз регулярно випускає виправлення безпеки через свою службу оновлення Windows. Вони випускаються раз на місяць, хоча критичні оновлення також доступні через коротші проміжки часу.

Якщо ви розглядаєте Linux, то підтримка належного рівня безпеки та конфіденційності персональних даних є наріжними каменями цієї ОС. За замовчуванням звичайні користувачі не мають доступу до кореневого каталогу або адміністративних привілеїв. І оскільки ядро Linux є відкритим, а сама система підтримується спільнотою та регулярно контролюється розробниками з усього світу, будь-яку проблему можна вирішити протягом декількох годин, отримавши необхідний патч із виправленнями. Ось чому Linux дуже популярний серед IT-фахівців.

За останні роки Linux досягла значних результатів з точки зору покращення якості та зручності застосування. Такі дистрибутиви, як Linux Mint та Ubuntu, навіть спростили їх встановлення та налаштування для нетехнологічних користувачів, щоб вони могли робити повсякденну роботу з максимальною легкістю.

Завдяки своєму розповсюдженню Windows є стандартною операційною системою на багатьох пристроях. Користувачі настільки звикли натискати

«Пуск» і відкривати свої улюблені програми, що їм дуже важко перейти на щось інше.

Незважаючи на нещодавні вдосконалення програмного забезпечення, перенесеного з інших платформ або розробленого на Linux, Windows все ще є "королем сумісності".

Користувачі Windows можуть бути впевнені, що майже будь-яке програмне забезпечення (навіть найвідоміше і застаріле) буде працювати, навіть якщо воно більше не розробляється самими розробниками. Windows має чудову підтримку застарілого програмного забезпечення.

C # належить до сімейства мов із C-подібним синтаксисом, синтаксис якого найближчий до C ++ та Java. Мова має сувору статичну типізацію, підтримує поліморфізм, перевантаження оператора, вказівники на функції члена класу, атрибути, події, властивості, винятки, коментарі у форматі

XML. Прийняття багатьох своїх попередників - C ++, Java, Delphi, Module та Smalltalk - C #, виходячи з практики їх використання, виключає деякі моделі, які виявились проблематичними при розробці програмного забезпечення: так, C # не підтримує декілька класи успадкування (на відміну від C ++).

C # був розроблений як мова програмування рівня додатків для CLR і, як такий, в першу чергу залежить від можливостей самого CLR. Це стосується насамперед системи типу C #, яка відображає FCL. Наявність чи відсутність певних виразних особливостей мови диктується тим, чи можна певну мовну ознаку перекласти у відповідні конструкції CLR. Таким чином, з розвитком CLR з версії 1.1 до 2.0, сам C # став значно багатшим; подібної взаємодії слід очікувати в майбутньому. (Однак цей шаблон було порушено з випуском C # 3.0, який є розширенням мови, яке не покладається на розширення платформи .NET.) CLR надає C #, як і всі інші мови, орієнтовані на .NET, багато функцій, які позбавлений "класики" »Мови програмування. Наприклад, збір сміття не реалізований у самому C #, а виконується CLR для програм, написаних на C #, так само, як це робиться для програм на VB.NET, J # тощо.

ВИСНОВКИ

Розроблена інформаційна система “Платна Поліклініка”є актуальною тому що дозволяє автоматизувати документообіг закладу, підвищити продуктивність праці і покращити якість обслуговування клієнтів.

Під час проектування інформаційної системи було створено інфологічну модель даних, ER-діаграму, даталогічну модель, діаграму класів.

Після проведеного аналізу актуального інструментарію для реалізації функцій автоматизації діяльності інформаційної системи була використана мова програмування C# 7.0/Windows Forms/MetroFrameword/Dapper [3]. Для бази даних було використано Sql Server Management Studio 2018/SQL/Transact-SQL.

В базі даних знаходиться інформація клієнтів, діагнози, лікарів, лікування, а також спеціалізації які потрібні для функціонування поліклініки також дані про користувачів, що необхідно для роботи програмного додатку, як клієнт-серверного. Програма дозволяє здійснювати пошук, зміну, додавання, видалення даних, і переглядати дані. В силу специфіки функціонування даної інформаційної системи створення мобільної версії не виглядає доцільним.
Було реалізовано захист бази даних за допомогою авторизації входу у систему.

В перспективі можна передбачити додатковий, окрім вбудованого, захист інформації, що зберігається в базі даних. Виглядає доцільним оптимізувати процедуру запуску програмного продукту. Проведене тестування дає підстави стверджувати що в загальному система функціонує відповідно до поставлених вимог та може бути використана за основним призначенням.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Документація Git. *Git:* веб-сайт. URL: https://git-scm.com/book/uk/v2 (дата звернення: 03.03. 2021)
2. Документація. Dagger *Dagger:* веб-сайт. URL: https://habr.com/ru/post/279125/ (дата звернення: 07.03. 2021).
3. Структура приложения для Windows Forms *Habrahabr:* веб-сайт. URL: https://habr.com/ru/company/microsoft/blog/218441/ (дата звернення: 25.03. 2021).
4. Васильев А.Н. Программирование на C# в примерах и задачах : навчальний посібник – М. : Эксмо-пресс, 2021. – 368 С.
5. Бєлов А. Вступ до програмування мовою C#. Організація обчислень : навч. посіб. / Ю. А. Бєлов, Т. О. Карнаух, Ю. В. Коваль, А. Б. Ставровський. – К. : видавничо-поліграфічний центр "Київський університет", 2021. – 175 С. С.: іл. Isbn (укр.)
6. Страуструп Бьярне. Программирование. Прицины и практика с использованием C# – М. : Вильямс, 2021. – 1328 С.
7. Довбуш Г. Visual C# на примерах/ Галина Довбуш, Анатолий Хомоненко. – М .: бхв-петербург, 2019. – 528 С.
8. Роберт С. Безопасное программирование на C и C# / Роберт С. Сікорд. – М. : рдгу, 2019. – 496 С.
9. SFML. Вікіпедія – вільна енциклопедія: веб-сайт. URL: https://uk.wikipedia.org/wiki/SFML.

ДОДАТКИ

Додаток А

Програмний код

Лістинг А1 – Код форми входу

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Dapper;

namespace PaidClinicDatabaseApp
{
    public partial class FormLogin :
MetroFramework.Forms.MetroForm
    {
        public FormLogin()
        {
            InitializeComponent();
        }

        private void FormLogin_Load(object sender, EventArgs e)
        {
            //Встановлення теми, стилю
            this.StyleManager = metroStyleManager1;
            metroStyleManager1.Theme =
MetroFramework.MetroThemeStyle.Light;
            metroStyleManager1.Style =
MetroFramework.MetroColorStyle.Green;
            if (Properties.Settings.Default.RememberMe)
            {
                //Заповнення полів входу, якщо попередньо
користувач використав функцію запамятовування
                TextBoxLoginUsername.Text =
Properties.Settings.Default.Username;
                TextBoxLoginPassword.Text =
Properties.Settings.Default.Password;
            }
        }

        private void buttonLogin_Click(object sender, EventArgs e)
```

```
        {

 //Перевірка, чи не пусте поле імені користувача та поле паролю
одночасно
            if (string.IsNullOrEmpty(TextBoxLoginUsername.Text) &&
string.IsNullOrEmpty(TextBoxLoginPassword.Text))
            {
                MetroFramework.MetroMessageBox.Show(this, "Введіть
ім'я користувача.", "Message", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                TextBoxLoginUsername.Focus();
                return;
            }
            //Перевірка, чи не пусте поле імені користувача
            if (string.IsNullOrEmpty(TextBoxLoginUsername.Text))
            {
                MetroFramework.MetroMessageBox.Show(this, "Введіть
ім'я користувача", "Message", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                TextBoxLoginUsername.Focus();
                return;
            }
            //Перевірка, чи не пусте поле імені паролю
            if (string.IsNullOrEmpty(TextBoxLoginPassword.Text))
            {
                MetroFramework.MetroMessageBox.Show(this, "Введіть
пароль", "Message", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                TextBoxLoginPassword.Focus();
                return;
            }
            try
            {
                using (IDbConnection db = new
SqlConnection(ConfigurationManager.ConnectionStrings["cn"].Connect
ionString))
                {
                    if (db.State == ConnectionState.Closed)
                        db.Open();
                    //Виконання sql запиту та заповнення
отриманими даними до класу User
                    Users obj = db.Query<Users>($"select * from
Users where User_username = '{TextBoxLoginUsername.Text}'",
commandType: CommandType.Text).SingleOrDefault();
                    if (obj != null)
                    {
                        if (obj.User_password ==
TextBoxLoginPassword.Text)//True
                        {
                            using (FormNavigation frm = new
FormNavigation())//Відкриття форми навігації
                            {
```

```
Properties.Settings.Default.CurrentUsername =
TextBoxLoginUsername.Text;


Properties.Settings.Default.CurrentPassword =
TextBoxLoginPassword.Text;

Properties.Settings.Default.CurrentPermission =
obj.User_permission;
                                    this.Hide();
                                    frm.ShowDialog();
                                }
                            }
                            else

MetroFramework.MetroMessageBox.Show(this, "Перевірте правильність
паролю та імені користувача", "Message", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                        }
                        else
                            MetroFramework.MetroMessageBox.Show(this,
"Перевірте правильність паролю та імені користувача", "Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                }
                catch (Exception ex)
                {
                    MetroFramework.MetroMessageBox.Show(this,
ex.Message, "Message", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                }
            }

            private void checkBoxRememberMe_CheckedChanged(object
sender, EventArgs e)
            {
                if (checkBoxRememberMe.Checked)//Встановлення значень
до налаштувань користувача при використанні функції
запам'ятовування даних входу
                {
                    Properties.Settings.Default.Username =
TextBoxLoginUsername.Text;
                    Properties.Settings.Default.Password =
TextBoxLoginPassword.Text;
                }
                else
                {
                    Properties.Settings.Default.Username = null;
                    Properties.Settings.Default.Password = null;
                }
                Properties.Settings.Default.RememberMe =
checkBoxRememberMe.Checked;
```

```
            Properties.Settings.Default.Save();//Збереження даних
користуваць ких налаштувань
        }



     private void linkLabelRegistrate_LinkClicked(object sender,
     LinkLabelLinkClickedEventArgs e)
        {
            using (FormRegistration frm = new
FormRegistration())//Відкриття форми реєстрації
            {
                this.Hide();
                frm.ShowDialog();
            }
        }

        private void FormLogin_FormClosed(object sender,
FormClosedEventArgs e)
        {
            Application.Exit();
        }
    }
}
```

Лістинг А2 – Код форми реєстрації

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Dapper;

namespace PaidClinicDatabaseApp
{
    public partial class FormRegistration :
MetroFramework.Forms.MetroForm
    {
        public FormRegistration()
        {
            InitializeComponent();
        }

        private void FormRegistration_Load(object sender,
EventArgs e)
        {
```

```
            //Встановлення теми, стилю
            this.StyleManager = metroStyleManager1;
            metroStyleManager1.Theme =
MetroFramework.MetroThemeStyle.Light;
            metroStyleManager1.Style =
MetroFramework.MetroColorStyle.Green;
        }

        private async void buttonRegistrate_Click(object sender,
EventArgs e)
        {
            //Перевірка, чи не пусте поле імені користувача та
поле паролю одночасно
            if
(string.IsNullOrEmpty(TextBoxRegistrationUsername.Text) &&
string.IsNullOrEmpty(TextBoxRegistrationPassword.Text))
            {
                MetroFramework.MetroMessageBox.Show(this, "Введіть
ім'я користувача", "Message", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                TextBoxRegistrationUsername.Focus();
                return;
            }
            //Перевірка, чи не пусте поле імені користувача
            if
(string.IsNullOrEmpty(TextBoxRegistrationUsername.Text))
            {
                MetroFramework.MetroMessageBox.Show(this, "Введіть
ім'я користувача", "Message", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                TextBoxRegistrationUsername.Focus();
                return;
            }
            //Перевірка, чи не пусте поле імені паролю
            if
(string.IsNullOrEmpty(TextBoxRegistrationPassword.Text))
            {
                MetroFramework.MetroMessageBox.Show(this, "Введіть
пароль", "Message", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                TextBoxRegistrationPassword.Focus();
                return;
            }
            try
            {
                using (IDbConnection db = new
SqlConnection(ConfigurationManager.ConnectionStrings["cn"].Connect
ionString))
                {
                    if (db.State == ConnectionState.Closed)
                        db.Open();
                    //Виконання sql запиту та заповнення
отриманими даними до класу User
```

```
                    Users obj = db.Query<Users>($"select * from
Users where User_username = '{TextBoxRegistrationUsername.Text}'",
commandType: CommandType.Text).SingleOrDefault();
                    if (obj == null)
                    {
                        IUserRepository repository = new
UserRepository();


 bool result = await repository.Insert(new Users() { User_username
= TextBoxRegistrationUsername.Text, User_password =
TextBoxRegistrationPassword.Text, User_permission = "Ні" });
                        if (result)
                        {
                            MessageBox.Show("Реєстрація пройшла
успішно", "Message", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                            using (FormLogin frm = new
FormLogin())//Відкриття форми входу
                            {
                                this.Hide();
                                frm.ShowDialog();
                            }
                        }
                        else
                            MessageBox.Show("Помилка реєстрації",
"Message", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                    else
                        MetroFramework.MetroMessageBox.Show(this,
"Таке ім'я користувача зайняте, придумайте інше", "Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
            }
            catch (Exception ex)
            {
                MetroFramework.MetroMessageBox.Show(this,
ex.Message, "Message", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        private void linkLabelLogin_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
        {
            using (FormLogin frm = new FormLogin())//Відкриття
форми входу
            {
                this.Hide();
                frm.ShowDialog();
            }
        }
```

```
        private void FormRegistration_FormClosed(object sender,
FormClosedEventArgs e)
        {
            Application.Exit();
        }
    }
}
```

Лістинг А3 – Код форми реєстрації

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace PaidClinicDatabaseApp
{
    public partial class FormNavigation :
MetroFramework.Forms.MetroForm
    {
        public FormNavigation()
        {
            InitializeComponent();
        }

        private void FormNavigation_Load(object sender, EventArgs
e)
        {
            if(Properties.Settings.Default.CurrentUsername ==
"admin")
            {
                pictureBoxUsers.Visible = true;
                ButtonUsers.Visible = true;
            }
            else
            {
                pictureBoxUsers.Visible = false;
                ButtonUsers.Visible = false;
            }
            //Встановлення теми, стилю
            this.StyleManager = metroStyleManager1;
            metroStyleManager1.Theme =
MetroFramework.MetroThemeStyle.Light;
            metroStyleManager1.Style =
MetroFramework.MetroColorStyle.Green;
        }

        private void FormNavigation_FormClosed(object sender,
FormClosedEventArgs e)
        {
            Application.Exit();
        }

        private void ButtonSpecialty_Click(object sender,
EventArgs e)
```

```
            {
                using (FormSpecialty frm = new
FormSpecialty())//Відкриття форми спеціальностей
                {
                    this.Hide();
                    frm.ShowDialog();
                }
            }

            private void pictureBoxSpecialty_Click(object sender,
EventArgs e)
            {
                MetroFramework.MetroMessageBox.Show(this, "Вікно
містить назви спеціальностей для лікарів", "Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            }

            private void ButtonDoctor_Click(object sender, EventArgs
e)
            {
                using (FormDoctor frm = new FormDoctor())//Відкриття
форми лікарів
                {
                    this.Hide();
                    frm.ShowDialog();
                }
            }

            private void pictureBoxDoctor_Click(object sender,
EventArgs e)
            {
                MetroFramework.MetroMessageBox.Show(this, "Вікно
містить прізвище, ім'я, по бітькові, дату народження, стать,
категорію, спеціальність лікарів", "Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            }

            private void ButtonClient_Click(object sender, EventArgs
e)
            {
                using (FormClient frm = new FormClient())//Відкриття
форми клієнтів
                {
                    this.Hide();
                    frm.ShowDialog();
                }
            }

            private void pictureBoxClient_Click(object sender,
EventArgs e)
```

```
        {


  MetroFramework.MetroMessageBox.Show(this, "Вікно містить
прізвище, ім'я, по бітькові, дату народження, стать, групу крові
клієнтів", "Message", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        }

 private void ButtonDiagnosis_Click(object sender, EventArgs e)
        {
            using (FormDiagnosis frm = new
FormDiagnosis())//Відкриття форми діагнозів
            {
                this.Hide();
                frm.ShowDialog();
            }
        }

        private void pictureBoxDiagnosis_Click(object sender,
EventArgs e)
        {
            MetroFramework.MetroMessageBox.Show(this, "Вікно
містить клієнта, текст діагнозу, лікаря, дату діагнозів",
"Message", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        private void ButtonTreatment_Click(object sender,
EventArgs e)
        {
            using (FormTreatment frm = new
FormTreatment())//Відкриття форми списку лікувань
            {
                this.Hide();
                frm.ShowDialog();
            }
        }

        private void pictureBoxTreatment_Click(object sender,
EventArgs e)
        {
            MetroFramework.MetroMessageBox.Show(this, "Вікно
містить лікаря, клієнта, діагноз, вартість, дата початку та
закінчення", "Message", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        }

        private void pictureBoxUsers_Click(object sender,
EventArgs e)
        {
```

```
            MetroFramework.MetroMessageBox.Show(this, "Вікно
містить ім'я користувача, пароль та права на редагування",
"Message", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        private void ButtonUsers_Click(object sender, EventArgs e)
        {
            using (FormUsers frm = new FormUsers())//Відкриття
форми користувачів
            {
                this.Hide();
                frm.ShowDialog();
            }

      }
    }
}
```

Лістинг А3 – Код форми реєстрації

```
using Dapper;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using System.Windows.Forms;

namespace PaidClinicDatabaseApp
{
    public partial class FormChooseDoctor :
MetroFramework.Forms.MetroForm
    {
        public FormChooseDoctor()
        {
            InitializeComponent();
        }

        private void FormChooseDoctor_Load(object sender,
EventArgs e)
        {
            //Встановлення теми, стилю
            this.StyleManager = metroStyleManager1;
            metroStyleManager1.Theme =
MetroFramework.MetroThemeStyle.Light;
            metroStyleManager1.Style =
MetroFramework.MetroColorStyle.Green;
            try
```

```
            {
                using (IDbConnection db = new
SqlConnection(ConfigurationManager.ConnectionStrings["cn"].Connect
ionString))
                {
                    if (db.State == ConnectionState.Closed)
                        db.Open();
                    //Отримання інформації з бази даних
                    doctorBindingSource.DataSource =
db.Query<Doctor>("select * from Doctor", commandType:
CommandType.Text);
                    if (TextBoxBirth_date.Text != "")
                        TextBoxBirth_date.Text =
TextBoxBirth_date.Text.Substring(0, 10);
                }

            }
            catch (Exception ex)
            {
                MetroFramework.MetroMessageBox.Show(this,
ex.Message, "Message", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        private void ButtonChoose_Click(object sender, EventArgs
e)
        {
            Properties.Settings.Default.ChosenDoctor =
TextBoxId.Text;
            this.Hide();
        }
    }
```