Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем та програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Проектування та розробка бази даних поліклініки з використанням методології RUP та мови програмування C#

Виконав: студент 6 курсу, групи СПм-61
спеціальності 121 «Інженерія програмного забезпечення»
(шифр і назва спеціальності)

		Рудак В.М.
	(підпис)	(прізвище та ініціали)
Керівник		Стоянов Ю.М.
	(підпис)	(прізвище та ініціали)
Нормоконтроль		Стоянов Ю.М.
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		Петрик М.Р.
	(підпис)	(прізвище та ініціали)
Рецензент		
	(підпис)	(прізвище та ініціали)

Тернопіль
2023

РЕФЕРАТ

Кваліфікаційна робота магістра. Тернопільський національний технічний університет імені Івана Пулюя, кафедра програмної інженерії, спеціальність 121 «Інженерія програмного забезпечення». ТНТУ, 2023. Сторінок 79, таблиць 8, рисунків 40, презентація.

Тема: Проектування та розробка бази даних поліклініки з використанням методології RUP та мови програмування C#.

Об’єктами дослідження для подальшої розробки кваліфікаційної роботи магістра стали існуючі подібні програмні рішення, що зв’язані з базою даних.

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

Завданням проекту є розробка програмного додатку з інтерфейсом користувача для управління базою даних поліклініки.

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 Microsoft Visual Studio, що використовується для розробки різноманітних додатків та у тому ж числі десктопних на мові програмування C#. Для реалізації серверної частини використано СУБД MS SQL Server Management Studio.

Ключові слова: програмний додаток, SQL Server, T-SQL, C#, Windows Forms, RUP, клієнт-сервер, база даних, інтерфейс.

ANNOTATION

Master's certification work. Ternopil Ivan Puluj National Technical University, Department of Software Engineering, specialty 121 "Software Engineering". TNTU, 2023. Pages 79, tables 9, figures 40, presentation.

Topic: Design and development of the polyclinic database using RUP methodology and C# programming language.

The objects of research for the further development of the course project were existing similar software solutions that are connected to the database.

The purpose of the project is to develop a software application that will allow users to manage the database and view the necessary records using a convenient and intuitive interface.

The task of the project is to develop a software application with a user interface for managing the polyclinic database.

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

To develop the software application, we chose the Microsoft Visual Studio development environment, which is used to develop various applications, including desktop applications in the C# programming language. MS SQL Server Management Studio was used to implement the server side.

Keywords: software application, SQL Server, T-SQL, C#, Windows Forms, RUP, client-server, database, interface.

ЗМІСТ

ВСТУП

Сучасні поліклініки - це складні заклади, які обслуговують велику кількість пацієнтів. Для того, щоб поліклініка працювала ефективно, всі її процеси повинні управлятися автоматизованою системою. Одним з основних компонентів системи є база даних, яка містить дані про пацієнтів, співробітників, медичні картки та іншу інформацію, необхідну для роботи поліклініки.

У цій роботі методологія RUP та мова програмування C# використовуються для проектування та розробки бази даних та застосунку для поліклініки. Вимоги до системи, які були зібрані в ході дослідження предметної області, були проаналізовані для правильного проектування бази даних. Для інтегрування створеної бази даних за допомогою СУБД MS SQL Server Management Studio буде використано фреймворк Entity Framework Core та мову програмування C#.

Робота буде містити такі розідли. Вступ до методології RUP та мови програмування C#. Аналіз вимог та проектування логічної та фізичної структури бази даних. Етапи розробки бази даних, включаючи створення прикладних програм, DDL-скриптів та моделей даних. Перевірка роботи готового зібраного десктопного застосунку.

Результати роботи можуть бути використані для розробки автоматизованих систем для інших медичних закладів.

Актуальність теми. Використання автоматизованих систем управління охороною здоров'я набуває все більшої популярності. Вони сприяють економії коштів, кращому обслуговуванню пацієнтів та підвищенню ефективності роботи закладу. Одним з основних елементів є база даних, в якій зберігаються дані про пацієнтів, співробітників, медичну документацію та інша інформація, необхідна для роботи медичного закладу.

Створення та проектування бази даних для поліклініки є складним завданням, яке потребує спеціальних знань та досвіду. У цій роботі представлено досвід використання методології RUP та мови програмування C# для проектування

та розробки бази даних для поліклініки. Робота втілює в собі здобуті знання та досвід протягом всього навчання у вищому навчальному закладі.

Робота буде виконуватись на основі аналізу існуючих десктопних застосунків з користувацьким інтерфейсом для поліклінік, а також на основі вимог, що висуваються до подібних застосунків. У результаті роботи очікується десктопний застосунок з користувацьким інтерфейсом для поліклініки, який відповідає цим вимогам.

Мета роботи - розробити базу даних та десктопний застосунок на її основі для поліклініки, що відповідатиме вимогам до системи.

Для досягнення цієї мети необхідно виконати такі завдання:

– Ознайомитися з методологією RUP та мовою програмування C#.

– Проаналізувати вимоги до бази даних.

– Проектувати логічну та фізичну структуру бази даних.

– Розробити десктопний застосунок з інтерфейсом користувача за допомогою мови програмування C#.

У роботі також розглянуто такі питання, як вибір методології розробки, вибір інструментів розробки, тестування застосунку. Робота містить детальний опис процесу розробки застосунку, а також приклади коду.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПОЛІКЛІНІКИ

Областю застосування бази даних є Поліклініка. Поліклініка - це організація, яка призначена для контролю і діагностики стану здоров'я клієнтів. Отже, поліклініка працює з дуже великим обсягом інформації, як про співробітників, так і про пацієнтів. Лікарям необхідно завжди стежити за даними про своїх пацієнтів, про курс лікування хворих. А керівництву та бухгалтерії необхідно бути в курсі подій про своїх співробітників. Для цього потрібна спільна база даних, що включає всю необхідну інформацію. Програма є дуже актуальною на сьогоднішній день, вона автоматизує роботу з базою даних і надає користувачеві (оператору) зрозумілий і дружній інтерфейс.

1.1 Огляд систем аналогів

Огляд систем аналогів є важливою частиною процесу розробки програмного забезпечення. Він дозволяє розробникам отримати уявлення про те, що вже існує на ринку, і визначити, як їх власний застосунок може відрізнятися від інших. Огляд систем аналогів може допомогти розробникам:

- Зрозуміти потреби користувачів. Огляд систем аналогів може допомогти розробникам зрозуміти, які функції та можливості цінують користувачі. Це може допомогти розробникам створити застосунок, який відповідає потребам користувачів.
- Визначити можливості ринку. Огляд систем аналогів може допомогти розробникам визначити, чи є попит на їхній застосунок. Якщо на ринку вже існують схожі застосунки, розробникам може знадобитися знайти спосіб виділитися з конкурентів.
- Уникнути повторення помилок. Огляд систем аналогів може допомогти розробникам уникнути повторення помилок, які вже були зроблені в інших

застосунках. Це може допомогти розробникам створити більш якісний і надійний застосунок.

Огляд систем аналогів можна провести різними способами. Найпростіший спосіб - це просто використовувати ці застосунки та спостерігати за тим, як вони працюють. Інший спосіб - це прочитати огляди та відгуки користувачів. Третій спосіб - це провести опитування користувачів. Огляд систем аналогів є важливим кроком у процесі розробки програмного забезпечення. Він може допомогти розробникам створити більш якісний і успішний застосунок. Це корисний інструмент, який може допомогти розробникам у будь-якому типі застосунку.

Для огляду взято максимально аналогічний програмний продукт, що знаходився у вільному доступі, для розробки якого використовувались схожі інструменти та технології. Назва застосунку Hospital Management System. При запуску можна побачити екран завантаження, що може значити низьку швидкість підключення та наповнення системи або про цілenаправлене збільшення часу очікування використання застосунку (рисунок 1.1).

Рисунок 1.1 – Екран завантаження системи Hospital Management System

Після завантаження в декілька секунд відкривається форма для входу в систему після введення логіна та паролю (рисунок 1.2). Форма входу передбачена лише для вже існуючих користувачів, що не є хорошою практикою і системи не може функціонувати децентралізовано та самостійно. Немає можливості реєстрації користувача.

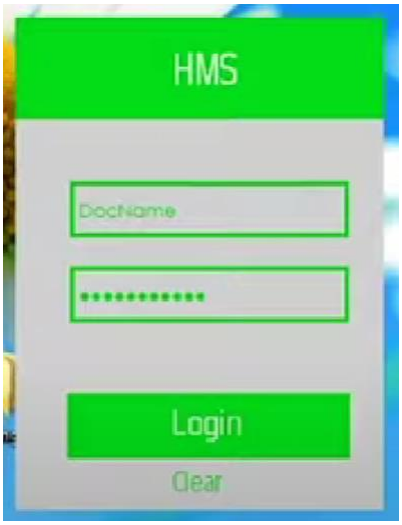


Рисунок 1.2 – Форма входу системи Hospital Management System

Після авторизації користувача система відкриває форму навігації, на якій розміщені основні пункти роботи з поданою інформаційною системою (рисунок 1.3). На формі присутній фон картинкою, що не є дуже привабливим та зрозумілим в сучасних тенденціях вимог інтерфейсу користувача. Також варто зазначити, що кольорова гамма шрифтів та деяких елементів застосунку також не відповідають вимогам. В цілому зовнішній вигляд не є ергономічним та привабливим для використання, но свою функцію елементи виконують повноцінно. На формі можна побачити три основні елементи, що призначенні для роботи з базою даних системи та кнопку виходу, закриття форми, що не є потрібним і дублює функцію стандартної кнопки закриття вікна.

Рисунок 1.3 – Форма навігації системи Hospital Management System

Основною формою для користувача є безпосередня взаємодія з інформацією бази даних сутності Пацієнт в даному випадку. При натисканні на іконку з відповідним підписом відкривається форма так званого CRUD формату, що дозволяє створювати, читати, оновлювати та видаляти інформацію (рисунок 1.4).

Рисунок 1.4 – CRUD форма пацієнтів системи Hospital Management System

Сама форма виглядає набагато краще та є більш читабельною з точки зору користувацького інтерфейсу. Більшість форми є однотонною та нейтральною, що добре поєднується з основним кольором функціональних елементів. На формі присутні текстові поля, з якими може взаємодіяти користувач разом з функціональними кнопками для виконання певних операцій над інформацією. Більшість форми займає відображення існуючої інформації бази даних сутності пацієнтів, що є правильною практикою в створенні подібного виду інтерфейсів.

В цілому аналог можна класифікувати трохи нижче середнього, через неоднозначні рішення у створені та оформлені важливою складової застосунку свого виду, а саме інтерфейс користувача. Також система досить проста та не надто багатофункціональна.

1.2 Опис та вибір технологій та інструментів розробки

При виборі технологій для створення програмного застосунку, в подальшому і для кваліфікаційної роботи магістра за темою «Проектування та розробка бази даних для поліклініки з використанням методології RUP та мови програмування C#», проведено опис для порівняння технологій створення веб- та десктопиних клієнт-серверних додатків.

Хоча веб-додатки не потребують конфігурації і можуть бути запущені відразу в браузері, десктопні клієнт-серверні програми завжди повинні бути налаштовані на комп'ютері клієнта. Клієнт-серверна програма використовує дворівневу архітектуру, тоді як веб-додаток використовує багаторівневу архітектуру. Середній рівень, сервер додатків і користувацький клієнт складають цю багаторівневу структуру.

На відміну від клієнт-серверних додатків, в яких працюють два користувачі - клієнт і сервер, веб-додатки використовують лише одну систему користувачів. Веб-додаток працює в середовищі, керованому браузером. Він також може бути написаний мовою, яка підтримується браузером. Оскільки на сервері виконуються програми, які спільно використовують ресурси з клієнтами, серверна машина слугує хостом для клієнт-серверних додатків. Клієнт ніколи не ділиться своїми ресурсами з сервером; натомість він завжди запитує у нього вміст або інформацію.

Клієнт-серверна модель за своєю суттю перевантажена через зростаючий обсяг запитів від сучасних клієнтів. Веб-браузер є єдиною необхідною умовою для роботи веб-додатків. До таких веб-додатків належать Yahoo Mail, Gmail та Google Apps. Недоліки безпеки менш значні, ніж у веб-додатках, оскільки архітектура клієнт-сервер є типовою для брандмауера компанії.

Для тестування клієнт-серверних додатків потрібні два різні компоненти. У той час як на кожному клієнтському комп'ютері завантажується виконавчий файл програми, на сервері завантажується сама програма. При клієнт-серверному тестуванні потрібно тестувати в загальних областях, таких як двосторонній

графічний інтерфейс, навантаження, функціональність, бекенд і взаємодія між клієнтом і сервером.

Під час тестування веб-додатків тестувальник не має великого контролю над цільовим додатком. Додатки завантажуються на сервері, і вони можуть бути або не бути відомими, а на клієнтській машині не встановлено жодного exe-файлу, і немає необхідності тестувати ці додатки в різних веб-браузерах. Веб-додатки слід тестувати на різних платформах ОС і браузерах [4]. Ці додатки, як правило, побудовані на HTML, Javascript, XML тощо.

Апаратне забезпечення, на якому працюють десктопні програми, визначає їхні обмеження. Коли програма для планшета працює в повноекранному режимі, програма для настільних комп'ютерів Windows відображає стандартний робочий стіл Windows. Користувач повинен застосовувати оновлення до додатків безпосередньо під час їхнього встановлення.

Ось чим саме веб-додаток відрізняється від десктопного:

– Веб-додатки повинні бути розгорнуті на одному наборі серверних машин як для розгортання, так і для оновлення. Однак розгортання та будь-які оновлення або виправлення виконуються на окремих клієнтських комп'ютерах.

– Веб-програми не обмежуються місцем розташування. Десктопні програми можуть бути доступні лише з того комп'ютера, на якому вони встановлені, тому що вони обмежені окремим комп'ютером.

– Адміністратори мають більше контролю та повноважень над десктопними програмами. Тому вони більш захищені. Однак, оскільки всі веб-програми мають бути більш доступними, існує більший ризик для їхньої безпеки.

– Для роботи веб-програм необхідне підключення до Інтернету. Десктопна програма працює без Інтернету. Однак для оновлення десктопних додатків потрібне підключення до Інтернету.

Створення та підтримка веб-додатків коштує дорожче і, як правило, передбачає повторювані дії. Десктопні програми можна придбати лише один раз, і вони не потребують постійних платежів. Однак за певних обставин можуть виникати витрати на обслуговування.

1.2.1 Порівняльний вибір бази даних

Різниця між Oracle та SQL Server.

Для внутрішнього використання Oracle пропонує реляційну систему управління даними, відому як сервер Oracle. Реляційні бази даних складають основу фреймворку. Oracle широко використовується в усьому світі і може бути масштабований відповідно до вимог. Microsoft створила SQL-сервер, який дозволяє користувачам створювати і виконувати запити відповідно до їхніх вимог. У корпоративному середовищі SQL використовується для всіх процесів, включаючи аналітику, бізнес-аналітику та транзакції. Окрім попереджувальних повідомлень про помилки, SQL пропонує онлайн-підтримку.

Для створення модульних серверів і сховищ, що досягається шляхом створення логічних і фізичних структур, функція бази даних Oracle, яка використовується для корпоративних грід-обчислень, допомагає в цьому процесі. Доступ до бази даних має лише клієнтське програмне забезпечення. Буфери даних, історія журналів, користувацькі дані та кешована інформація команд SQL зберігаються в так званій SGA (System Global Area), структурі пам'яті на стороні сервера. Масштабованість, можливість повторного запуску виробничих робочих навантажень для пакетних і онлайн користувачів в режимі реального часу, підтримка методів віртуалізації, сумісність з VMware, висока доступність і безперервна обробка роблять цю базу даних вишуканою і прекрасною для роботи.

SQL Server Database Engine керує обробкою, безпекою та зберіганням даних. Механізм зберігання, який є частиною реляційного механізму, працює з файлами, сторінками, таблицями, буферами даних, індексами та транзакціями бази даних. Сам реляційний механізм обробляє команди та запити. Рушій бази даних відповідає за керування тригерами, поданнями, збереженими процедурами та іншими об'єктами бази даних. Його можна використовувати для розробки, розгортання та управління хмарними або локальними додатками. Його вміст зазвичай містить зв'язані дані, що зменшує надмірність і підвищує цілісність даних. Остання версія

включає підтримку гібридної хмари, операційну аналітику в реальному часі, методи візуалізації даних і вбудовані метрики налаштування продуктивності. Це дозволяє адміністраторам баз даних запускати один і той самий додаток локально або в хмарі, щоб зменшити організаційні витрати.

Хоча Oracle і SQL Server є популярними продуктами на ринку, ось деякі ключові відмінності між ними:

Незважаючи на використання різних версій мови структурованих запитів, СУБД SQL Server і Oracle використовують різні мови. У той час як Oracle використовує мову структурованих запитів і процедурну мову PL/SQL, MS Server використовує Transact SQL. Крім вбудованих функцій, основні відмінності полягають у змінних, синтаксисі та обробці процедур. Однією з функцій, яку не пропонує MS SQL Server, є можливість пакувати процедури разом.

Ще однією суттєвою відмінністю між цими двома базами даних є їхня здатність реалізовувати контроль транзакцій. Набір дій і призначень, які повинні оброблятися як єдине ціле, називається транзакцією. За замовчуванням MS SQL виконує і фіксує кожну команду або завдання як єдине ціле, що ускладнює відкат. Команди BEGIN TRANSACTION, COMMIT, ROLLBACK, END TRANSACTION тощо можуть значно підвищити ефективність цього процесу. Кожне з'єднання з базою даних в Oracle вважається новим з'єднанням і обробляється як нова транзакція. Насправді нічого не робиться явно, якщо не використовується зовнішня команда COMMIT, всі зміни вносяться в пам'ять.

Організація об'єктів бази даних відрізняється в різних СУБД. Всі об'єкти бази даних в MS SQL, включаючи представлення, таблиці та процедури, розташовані відповідно до назв баз даних. Певні об'єкти та бази даних доступні користувачам за допомогою призначених їм логінів. База даних Oracle організована відповідно до схем і доступна відповідним користувачам, тоді як файл на сервері SQL має приватний, незаточений тип диска. Ролі та дозволи, призначені цій групі, контролюють кожну схему та доступ користувачів.

Порівнюються Oracle і SQL Server, дві потужні СУБД. Хоча між ними є багато відмінностей, які можуть допомогти визначити, яка з них найкраще

підходить для розробки клієнт-серверного десктопного застосунку, в більшості аспектів вони по суті однакові. Перш ніж прийняти базу даних, необхідно провести ретельний аналіз, оскільки вибір правильної бази даних має вирішальне значення. Серед найпоширеніших баз даних - SQL Server і Oracle, а також така популярна СУБД, як MySQL. Для повної картини та розуміння вибору потрібної технології бази даних, додатково виконано огляд СУБД MySQL.

Різниця між MySQL та SQL Server.

MySQL - реляційна система управління базами даних з відкритим вихідним кодом. Вона працює з усіма операційними системами, включаючи Linux, Free BSD, Mac OS, Windows і Solaris [2]. Розробники написали її переважно на мовах C та C++. Вона доступна лише англійською мовою і належить до типу СУБД. SQL Server працює з Windows, Linux та Microsoft Windows. SQL Server надає драйвери, щоб будь-який код можна було легко підключити.

MySQL містить деякі обмеження. Один тригер може бути використаний на таблиці, коли відбувається подія в MySQL, тому що кожен тригер може виконувати тільки одну дію за один раз. Крім того, MySQL не підтримує визначення тригерів на поданнях. Ще одним недоліком є те, що MySQL не дотримується всіх стандартів SQL.

Утиліта резервного копіювання "mysqldump", яка використовується MySQL, може підтримувати резервне копіювання даних з усіх механізмів зберігання. Інша програма для резервного копіювання MySQL з відкритим вихідним кодом називається "XtraBackup". MySQL може працювати в інфраструктурі хмарних обчислень, включаючи Amazon і Microsoft Azure.

Бази даних MySQL та SQL Server забезпечують миттєву узгодженість. Типи транзакцій ACID використовуються як серверами SQL, так і MySQL. Крім того, вони підтримують вторинні індекси, тригери, підтримку XML, схеми даних і типізацію.

Порівняно з SQL Server, MySQL є більш поширеною, оскільки вона є безкоштовною, з відкритим вихідним кодом і сумісною з широким спектром платформ. Робота з такими платформами, як Linux, є незручною для розробників

через дорогу платну ліцензію на SQL Server. MySQL є основним додатком, якщо платформою є Linux. Сервери SQL використовуються розробниками додатків для Windows лише для інтеграції таких мов, як C# .Net.

Додаток, що буде розроблено на основі досліджень практики у кваліфікаційній роботі магістра є єдиним фактором, який визначає, як вибрати базу даних. Отож для десктопного клієнт-серверного застосунку, що буде написаний на мові програмування C# обрано SQL Server, як технологію організації бази даних.

Для початку роботи в SQL Server Management Studio вибираємо тип сервера, назву сервера та режим автентифікації, щоб підключитися до сервера [1]. Будемо підключатися до локальної бази даних SQL Server, тому вибираємо Database Engine як тип сервера.

Далі вибираємо назву сервера, до якого будемо підключатися. Це може бути локальний або віддалений сервер БД.

Далі вибираємо режим автентифікації. Остання версія SSMS пропонує п’ять режимів автентифікації на вибір, як показано нижче. Найбільш поширеними є автентифікація Windows і автентифікація SQL Server (рисунок 1.5).

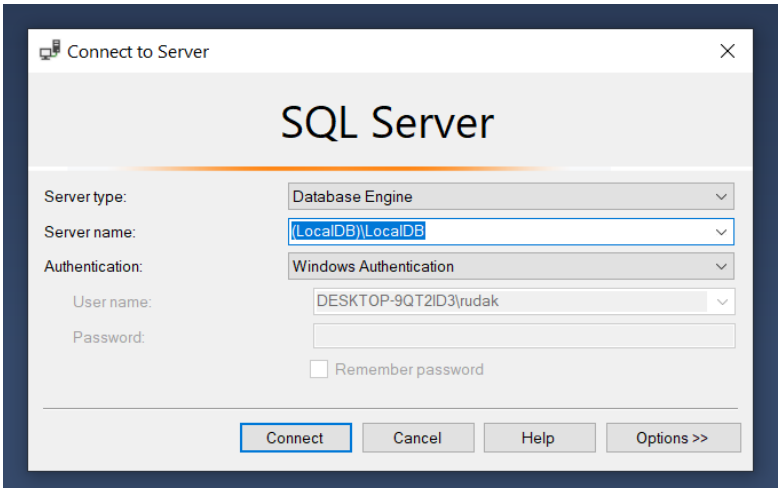


Рисунок 1.5 – З’єднання з локальним сервером для роботи з базою даних

SQL Server Management Studio містить такі компоненти:

– Провідник об'єктів

– Безпека

– Серверні об’єкти
– Редактор запитів і тексту
– Провідник шаблонів
– Провідник рішень
– Інструменти візуальної бази даних

Провідник об'єктів

Object Explorer містить різні компоненти одного або кількох екземплярів SQL Server в ієрархічній формі. Можна переглядати та керувати такими компонентами, як бази даних, безпека, серверні об’єкти, реплікація, керування тощо. Розгортаємо вузол компонентів, щоб побачити інші об’єкти.

Наприклад, розгортаємо папку Бази даних, щоб побачити всі бази даних, доступні в примірнику сервера. Будь-яка нова база даних, яку буде створено, буде доступна тут. Існує стандартна папка системних баз даних, яка містить чотири бази даних за замовчуванням: master, model, msdb і tempdb (рисунок 1.6).

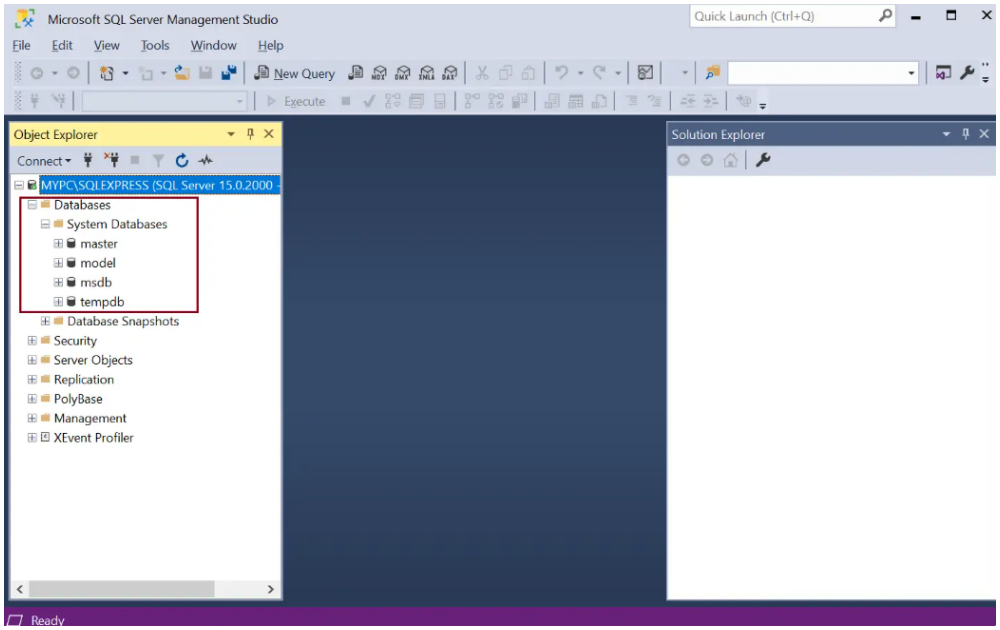


Рисунок 1.6 – Провідник об’єктів SQL Server Management Studio

Безпека

Управління безпекою сервера бази даних є надзвичайно важливим. Вузол «Безпека» знаходиться під вузлом «Бази даних» у провіднику об’єктів. Можна

можете створювати логіни та призначати ролі сервера для будь-якого екземпляра бази даних. Крім того, можна призначити захист на основі ролей для входу та користувачів. Ролі сервера, які тут будуть створені, охоплюють весь сервер (рисунок 1.7).

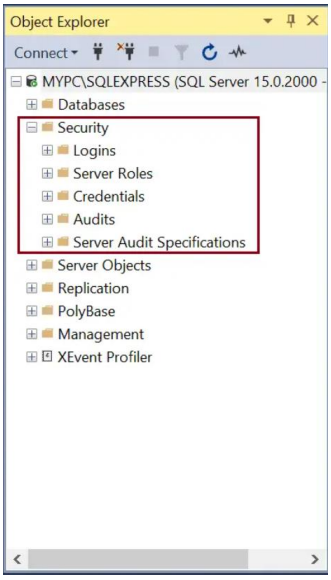


Рисунок 1.7 – Управління безпекою SQL Server Management Studio

Серверні об’єкти

Вузол «Об’єкти сервера» в SSMS має чотири підвузли: пристрої резервного копіювання, кінцеві точки, зв’язані сервери та тригери. Зв’язаний сервер — це метод, за допомогою якого SQL Server може спілкуватися з іншою базою даних ODBC за допомогою оператора T-SQL [5]. Кінцеві точки SQL Server є точкою входу в SQL Server. Це об’єкт бази даних, який визначає спосіб, у який SQL Server може спілкуватися через мережу. Усі об’єкти в розділі «Серверні об’єкти» охоплюють весь сервер (рисунок 1.8).

Рисунок 1.8 – Серверні об’єкти SQL Server Management Studio

Редактор запитів і тексту

Відкриваємо редактор запитів, натиснувши «Новий запит» на панелі інструментів. Редактор запитів дозволяє створювати, редагувати та виконувати оператори Transact SQL (T-SQL). Він оснащений підтримкою IntelliSense за допомогою автоматичного завершення сценарію, пропонуючи варіанти. Це робить написання та налагодження коду простішим і швидшим (рисунок 1.9).

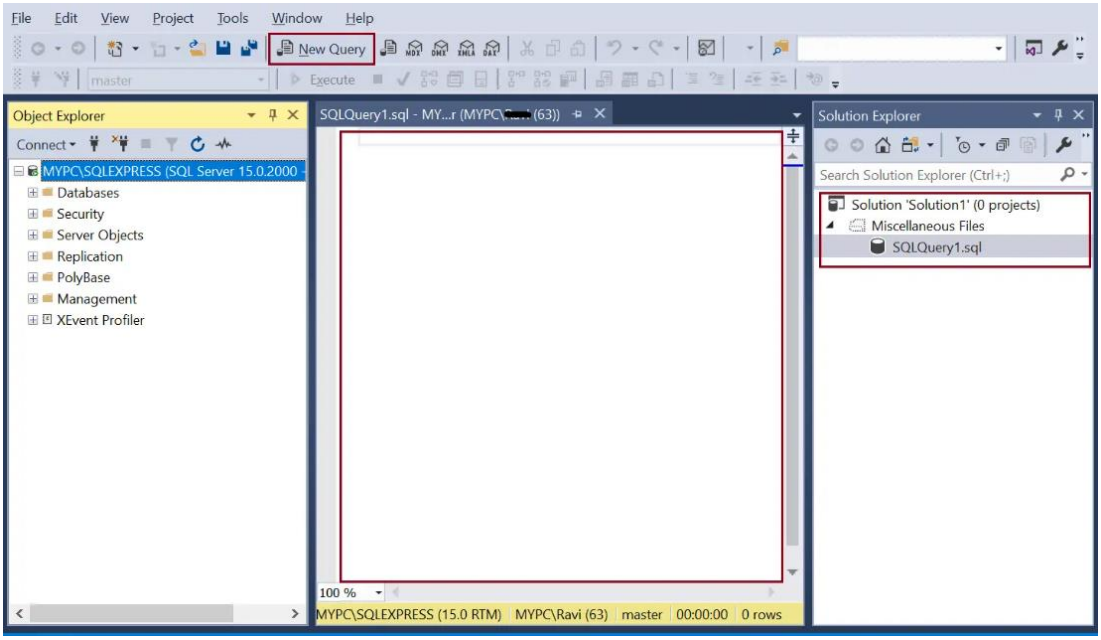


Рисунок 1.9 – Редактор запитів і тексту SQL Server Management Studio

Провідник шаблонів

Провідник шаблонів надає шаблони для створення різних об’єктів бази даних. Можна переглянути доступні шаблони в Template Explorer і відкрити їх у вікні редактора коду [3]. Також можна створювати власні шаблони.

Відкриваємо Провідник шаблонів у меню «Перегляд» -> Провідник шаблонів. Далі відображається шаблон створення бази даних (рисунок 1.10).

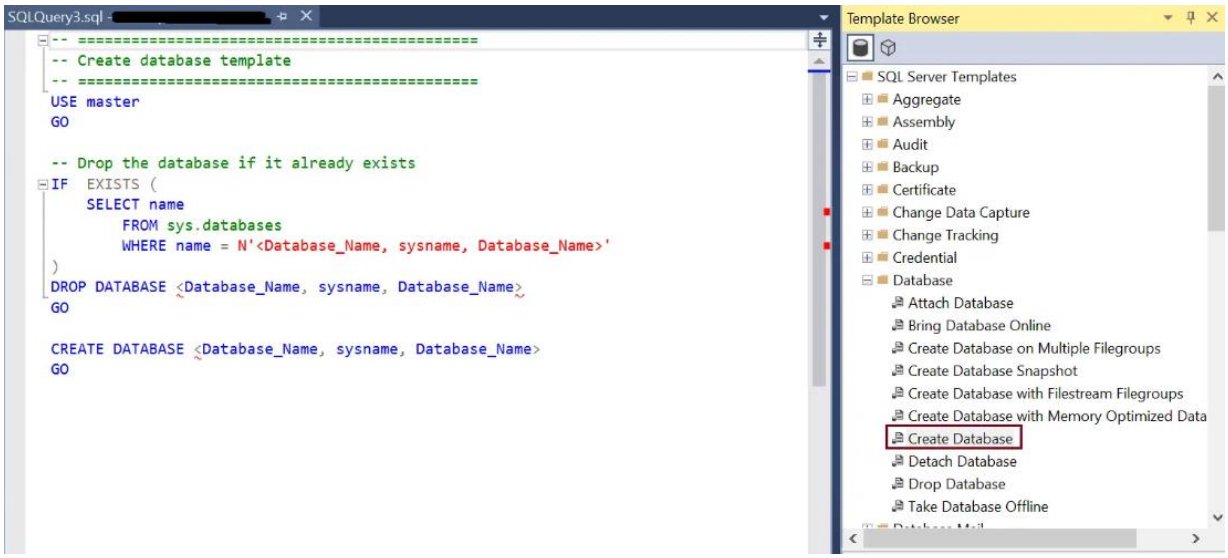


Рисунок 1.10 – Провідник шаблонів SQL Server Management Studio

Приклад розповсюдженного в програмному коді запиту на вибірку:

select * from [таблиця] where [поле для пошуку] like '%[текстове поле на формі]%'

1.2.2 Вибір середовища та технології написання програмного коду

Для створення коду додатка було обрано комплексне середовище Microsoft Visual Studio 2019. Інтегроване середовище розробки (IDE) – це багатофункціональна програма, яка підтримує багато аспектів розробки програмного забезпечення. Visual Studio – це креативна панель, яку можна

використовувати для редагування, налагодження та створення коду, а потім для публікації програми [6]. Крім стандартного редактора та налагоджувача, які надають більшість IDE, Visual Studio включає компілятори, інструменти завершення коду, графічні дизайнери та багато інших функцій для покращення процесу розробки програмного забезпечення.

Внутрішній вигляд та деякі поверхневі функції зображені на рисунку 1.11.

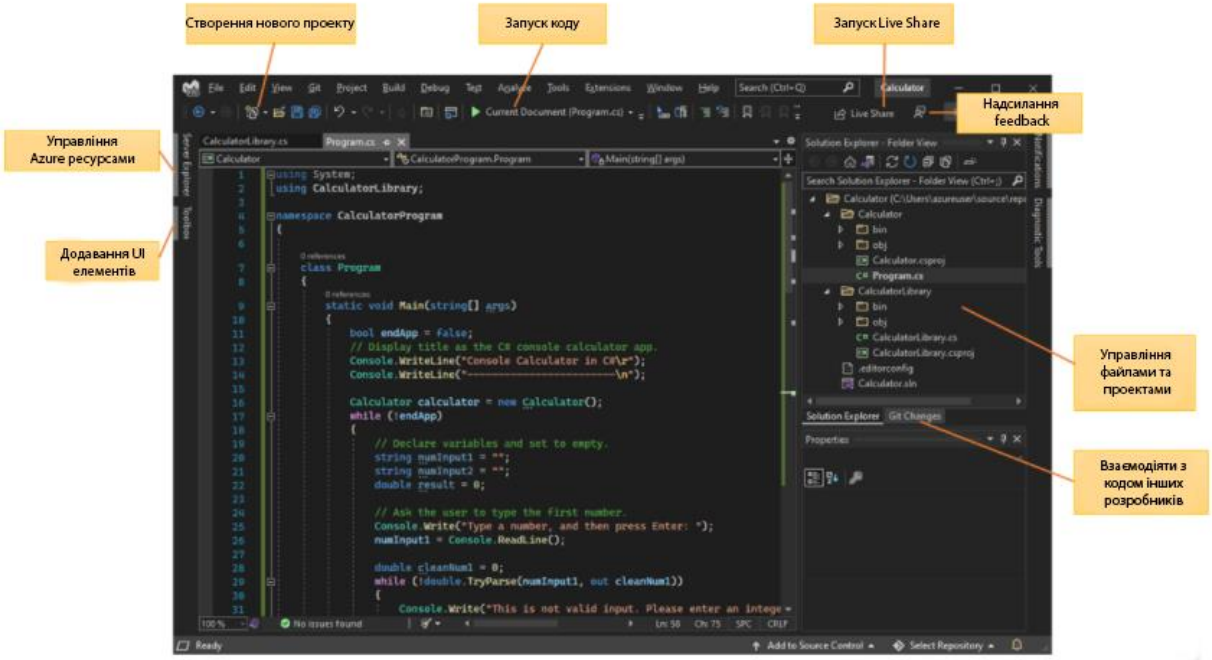


Рисунок 1.11 – Вигляд середовища MS Visual Studio 2019

Попереднє зображення показує Visual Studio з відкритим проектом, який показує ключові вікна та їх функціональність:

У провіднику рішень у верхньому правому куті можна переглядати, переміщатися та керувати файлами коду. Solution Explorer може допомогти впорядкувати ваш код, згрупувавши файли в рішення та проекти.

Центральне вікно редактора, де проводиться більшу частину часу роботи з середовищем, відображає вміст файлу [7]. У вікні редактора можна редагувати код або створювати інтерфейс користувача, наприклад вікно з кнопками та текстовими полями, що потрібно для даного проекту.

Популярні функції, що найбільше впливають на продуктивність написання коду чи роботи з програмними файлами:

Завитки та швидкі дії. Завитки – це хвилясті підкреслення, які сповіщають про помилки або потенційні проблеми у коді під час введення. Ці візуальні підказки допоможуть негайно виправити проблеми, не чекаючи виявлення помилок під час збірки або виконання. Якщо навести курсор на завитку, ви побачите більше інформації про помилку. На лівому полі також може з’явитися лампочка, яка показує швидкі дії, які можна виконати, щоб виправити помилку(рисунок 1.12).

Рисунок 1.12 – Приклад завитків

Очищення коду. Натиснувши кнопку, можна відформатувати свій код та застосувати будь-які виправлення коду, запропоновані налаштуваннями стилю коду, умовами .editorconfig та аналізаторами Roslyn [8]. Очищення коду, наразі доступне лише для коду C#, що допомагає вирішити проблеми у коді, перш ніж він буде перевірений компілятором(рисунок 1.13).

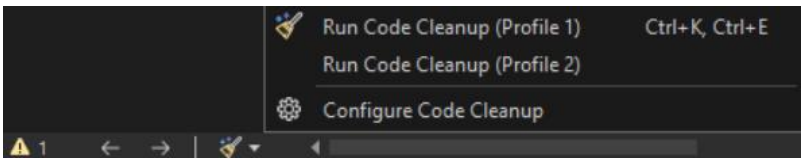


Рисунок 1.13 – Параметр Очищення коду

Рефакторинг включає такі операції, як інтелектуальне перейменування змінних, вилучення одного або кількох рядків коду в новий метод і зміна порядку параметрів методу. Фактично виконує багато роботи самостійно, якщо ти новачок у справі написання якісного та зрозумілого коду(рисунок 1.14).

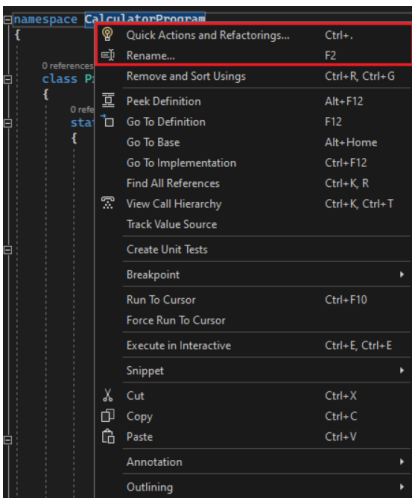


Рисунок 1.14 – Параметр Рефакторингу

IntelliSense – це набір функцій, які відображають інформацію про код безпосередньо в редакторі і, в деяких випадках, записують невеликі фрагменти коду для розробника [9]. Це як мати базову документацію в редакторі, тож не доведеться шукати інформацію про типи в іншому місці. На рисунку 1.15 показано, як IntelliSense відображає список можливих функцій для типу DateTime:

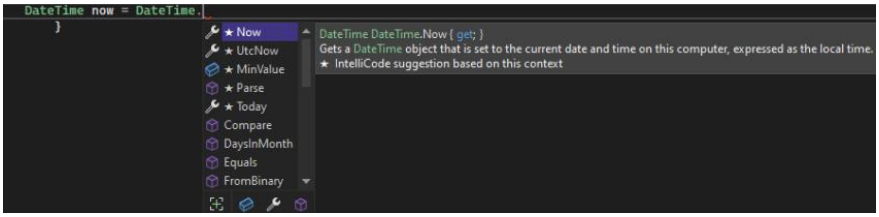


Рисунок 1.15 – IntelliSense на практиці

Пошук у Visual Studio. Меню, параметри та властивості Visual Studio інколи можуть здаватися заплутаними. Пошук у Visual Studio або Ctrl + Q – це чудовий спосіб швидко знайти функції та код IDE в одному місці(рисунок 1.16).

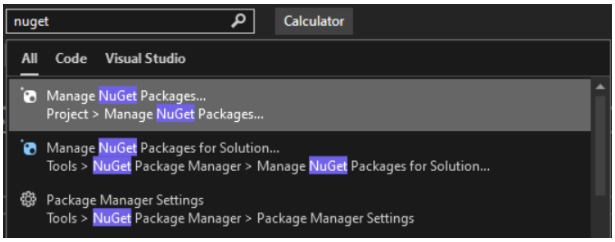


Рисунок 1.16 – Функція пошуку

Live Share. Можливість спільного редагування та налагоджувати код з іншими в режимі реального часу, незалежно від типу програми чи мови програмування. Можеливість миттєво та безпечно поділитися своїм проектом [10]. Також можна ділитися сеансами налагодження, екземплярами терміналів, веб-програмами localhost, голосовими дзвінками тощо

Ієрархія викликів. У вікні «Ієрархія викликів» показано методи, які викликають вибраний метод. Ця інформація може бути корисною, коли з'являється потреба про зміну чи видалення методу, або коли потрібно відстежити помилку(рисунок 1.17).

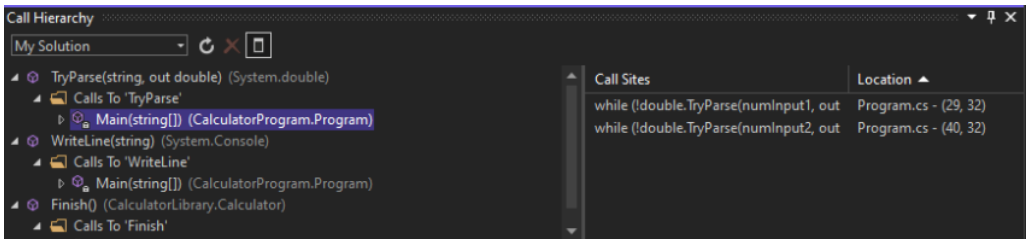


Рисунок 1.17 – Перегляд Ієрархії викликів методу

CodeLens допомагає знайти посилання на код, зміни коду, пов'язані помилки, робочі елементи, огляди коду та модульні тести, не виходячи з редактора(рисунок 1.18).

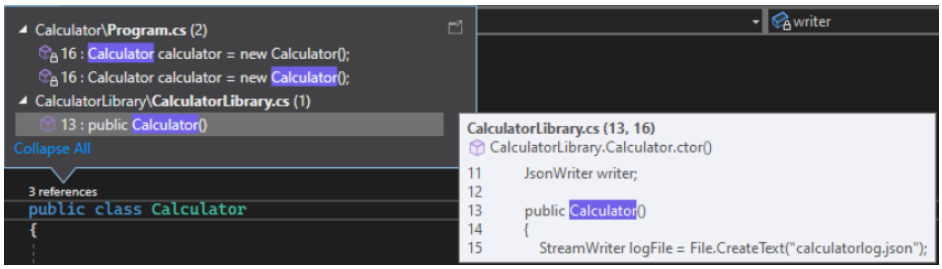


Рисунок 1.18 – Функція CodeLens

Функція Перейти до визначення переносить безпосередньо до розташування визначення функції або типу(рисунок 1.19).

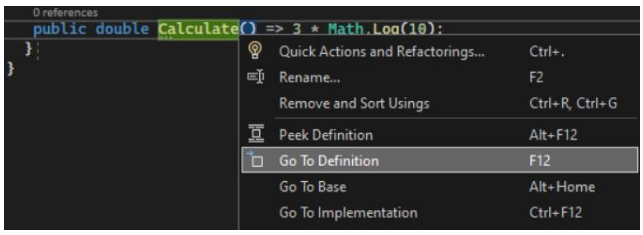


Рисунок 1.19 – Функція Перейти до визначення

У вікні Peek Definition показано визначення методу або типу без відкриття окремого файлу(рисунок 1.20).

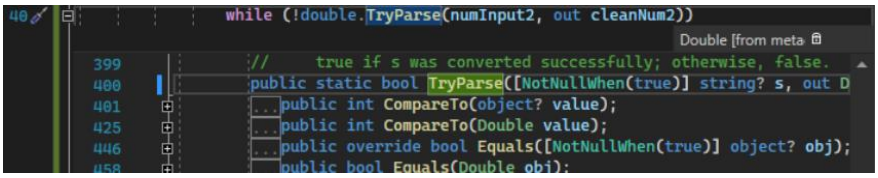


Рисунок 1.20 – Функція Peek Definition

В якості мови програмування обрано C# для реалізації технології Windows Form. Це бібліотека класів графічного інтерфейсу користувача (GUI), яка входить у .Net Framework [11]. Його головна мета – забезпечити спрощений інтерфейс розробки додатків для настільних комп'ютерів, планшетів, ПК. Його також називають WinForms. Програми, розроблені за допомогою Windows Forms або WinForms, відомі як програми, які працюють на настільному комп'ютері. WinForms можна використовувати лише для розробки додатків Windows Forms, а не веб-додатків. Програми WinForms можуть містити різні типи елементів керування, наприклад мітки, списки, підказку тощо.

1.3 Вибір та опис методології RUP

RUP використовує ітеративну модель розробки. Наприкінці кожної ітерації (в ідеалі, що триває від 2 до 6 тижнів) проектна команда повинна досягти

запланованих на дану ітерацію цілей, створити або доопрацювати проектні артефакти та отримати проміжну, але функціональну версію кінцевого продукту(рисунок 1.21).

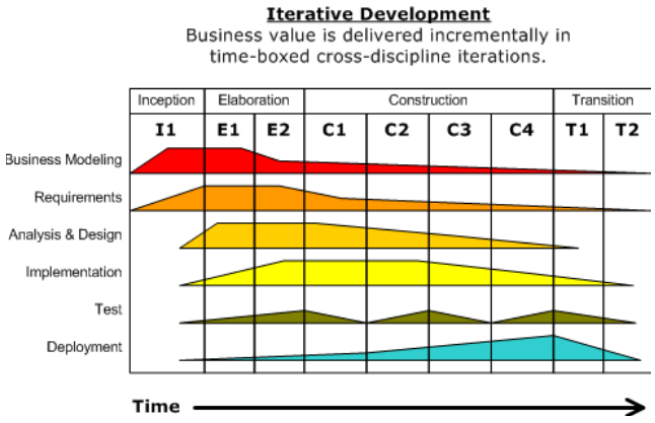


Рисунок 1.21 – Схема ітерацій RUP

Повний життєвий цикл розробки програмного забезпечення складається з 4 етапів:

1. Початковий етап (Inception)

На етапі розробки концепції проекту визначаються цілі та межі проекту.

Створюється бізнес-кейс.

Визначаються первинні специфікації, обмеження та основні характеристики продукту [12].

Розробляється рудиментарна форма прецедентної моделі.

(Прецедент описує типовий користувацький інтерфейс системи, прив'язується до певної функціональності системи та визначає один з варіантів використання системи. Варіанти використання, як правило, використовуються разом з оцінкою ризиків для визначення зовнішніх вимог до системи.

Згода зацікавлених сторін на продовження проекту передбачає оцінку досягнення цілей життєвого циклу наприкінці першої фази.

2. Уточнення (Elaboration)

Архітектура будується, а предметна область вивчається на етапі розробки. Детальні описи більшості прецедентів включені в документацію вимог [13].

Створено, реалізовано та протестовано виконувану архітектуру.

Оновлене бізнес-обґрунтування, а також більш точні прогнози бюджету та розкладу, зменшено основні ризики.

Після досягнення стадії життєвого циклу архітектури, фаза розробки була успішно завершена.

3. Побудова (Construction)

Більшість функціональних можливостей продукту реалізується під час фази будівництва. Перший зовнішній реліз системи та досягнення початкової функціональної готовності знаменують собою завершення етапу будівництва.

4. Впровадження (Transition)

Створення та передача фінальної версії продукту від розробника до клієнта відбувається під час фази впровадження. Вона охоплює навчання користувачів, контроль якості продукту та програму бета-тестування.

2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО КОМПЛЕКСУ

2.1 Проектування бази даних

Опис вхідних документів і повідомлень.

У базі даних «Поліклініка» використовуються наступні вхідні дані:

– інформація про клієнтів;

– інформація про лікарів;

– інформація про спеціалізацію лікарів;

– інформація про діагностування;

– інформація про користувачів системи;

– інформація про лікування.

Опис вихідних документів і повідомлень.

Вихідною інформацією є результати роботи запитів та збережувальних процедур на стороні бази даних для роботи додатку та усі дані, що відображаються на вікнах додатку в ході його експлуатації.

Інфологічна модель даних.

Процес створення формалізованої моделі предмета називається інформаційним проектуванням. Ці моделі створюються за допомогою загальноприйнятих мовних засобів, більшість з яких є графічними.

На етапі інфологічного проектування в ході збору інформації про предметну область потрібно з'ясувати:

1. основні об'єкти предметної області (об'єкти, про які повинна зберігатися інформація в БД);

2. атрибути об'єктів;

3. зв'язку між об'єктами;

4. основні запити до БД.

Наступні елементи є частиною реляційної моделі даних:

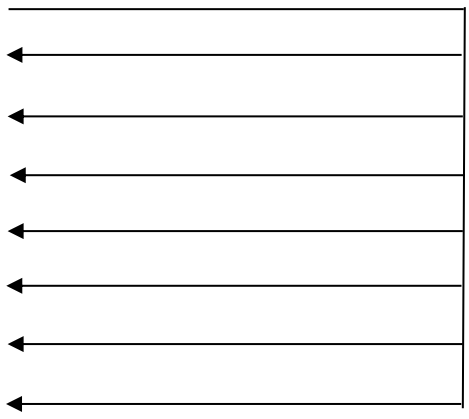
– Набір зв'язків складає структурний аспект (компонент) даних у базі даних.

– Аспект (компонент) цілісності: зв'язки (таблиці) задовольняють певним вимогам цілісності. Декларативні обмеження цілісності підтримуються реляційною моделлю даних на рівні домену, відношення та бази даних (для типів даних).

– Аспект (компонент) обробки (маніпулювання): оператори для маніпулювання відношеннями (реляційна алгебра, реляційне числення) підтримуються реляційною моделлю даних [14].

В БД «Поліклініка» в таблицях «Діагнози», «Лікарі», «Клієнти», «Спеціалізації», «Лікування», «Користувачі» між атрибутами і первинним ключем спостерігається функціональна залежність, так як значення ключа однозначно визначають значення інших атрибутів в даних таблицях.

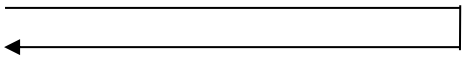
Таблиця 2.1 – Функціональні залежності між атрибутами «Лікарі»

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

У базі даних «Поліклініка» проведена нормалізація відносин:

Проаналізувавши таблицю «Клієнти», можна сказати, що вона знаходиться в першій нормальній формі, так як вона має первинний ключ, кожне поле таблиці представляє унікальний тип інформації, всі поля атомарні. Так само дана таблиця знаходиться і у 2НФ, так як вона задовольняє умовам 1НФ, а так же я переконалася в тому, що кожне поле функціонально залежить від первинного ключа, який ідентифікує вихідний об'єкт таблиці [15]. Таблиця «Клієнти» знаходиться в 3НФ, так як вона знаходиться у 2НФ і не містить транзитивних залежностей. Стовпці, які не є ключовими, залежать від первинного ключа таблиці і не залежать від всіх інших стовпців. Є можливість змінювати значення будь-якого поля (що не входить в первинний ключ) без впливу на дані інших полів.

Таблиця «Спеціалізації» аналогічно таблиці «Лікарі» знаходиться у всіх трьох нормальних формах.

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

2.2 Проектування архітектури програмної системи

Основні функції системи відображені на діаграмі варіантів використання, яка зображена на рисунку 2.1.

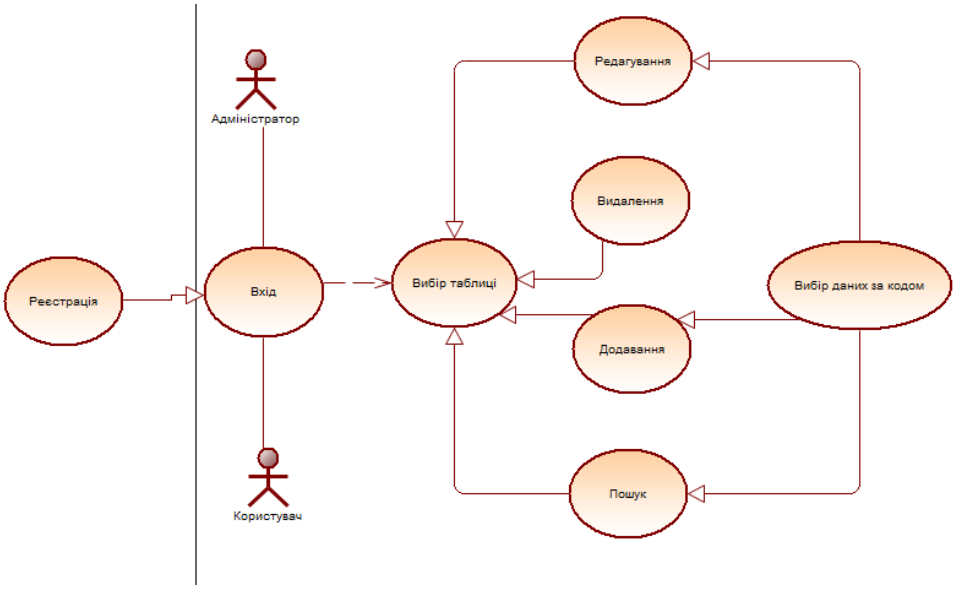


Рисунок 2.1 – Діаграма варіантів використання системи

Детальний опис основних функцій системи:

– Т1. Вхід в систему додатка відбуватиметься при введені унікального користувацького імені та відповідного паролю. Якщо користувач вперше в додатку то потрібно зареєструватись.

– Т2. Реєстрація відбувається за тим самим принципом, що і вхід, після реєстрації потрібно буде виконати вхід.

– Т3. 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: Т4 редагування – зміна обраного запису таблиці, та його оновлення; Т5 видалення – знищення вибраного запису з бази даних, попередньо підтвердивши дії у діалоговому вікні додатку; Т6 додавання – створення нового запису в базі даних з потрібною користувачеві

інформацією; Т7 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.

– Т8. 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 [16].

Матриця вимог відображена в таблиці 2.8.

Таблиця 2.8 – Матриця вимог проекту

		Прецедент	
		Адміністратор	Користувач
Вимога	Т1	X	X
	Т2	X	X
	Т3	X	X
	Т4	X	
	Т5	X	
	Т6	X	
	Т7	X	X
	Т8	X	X

Діаграма класів продемонстрована на рисунку 2.2.

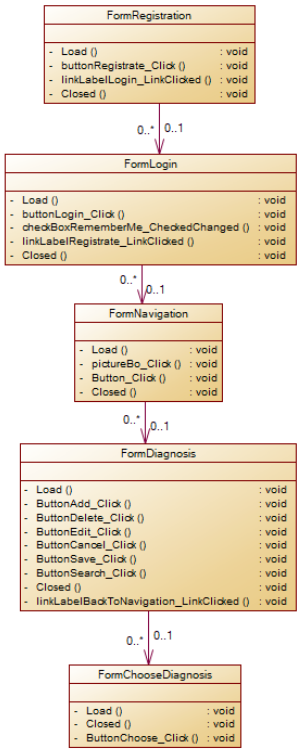


Рисунок 2.2 – Діаграма класів системи

Розгорнутий опис класів. В програмному рішені 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.

Перелік класів для збереження та надання інформації:

– 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. 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.

Відокремленою формою відображення класів є класи аналізу. Узагальнена діаграма класів аналізу зображена на рисунку 2.3. Діаграма саме узагальнена, тому що більшість класів мають різний вміст та інформаційну сутність. І ці класи поділяються на три групи:

– Перша група – унікальні класи, які будуть представлені повною мірою. FromNavigation, FormLogin, FornRegistration.

– Друга група – класи відображення форм додатку, що відображають різні таблиці системи та бази даних. FormTable.

– Третя група – представляє сукупність класів, що відповідають за передачу унікального поля в форму таблиці. FormChooseTable.

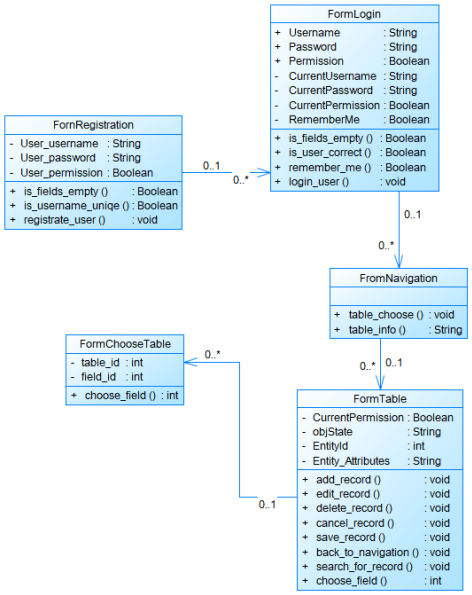


Рисунок 2.3 – Діаграма класів аналізу системи

При побудові діаграми класів аналізу дотримана основна норма – відповідність вмісту та суті атрибутів і функцій їх назвам. Додаткового опису усі елементи можуть бути реалізованими при необхідності.

Додаток не розділений на сотні класів. Декомпозицію проведено ієрархічно – спочатку систему було розбито на великі функціональні модулі/підсистеми, що описують її роботу в найзагальнішому вигляді. Потім, отримані модулі, проаналізовано більш детально і, в свою чергу, поділено на під-модулі або на об'єкти.

Перед тим як виділяти об'єкти було розділено систему на основні змістові блоки. Для невеликого додатка двох рівнів ієрархії цілком достатньо – система спочатку ділиться на підсистеми/пакети, а пакети діляться на класи.

Розподіл на модулі/підсистеми найкраще робити виходячи з тих завдань, які вирішує система. Основне завдання розбивається на складові для нього підзадачі, які можуть вирішуватися/виконуватися незалежно один від одного. Кожен модуль відповідає за вирішення якоїсь підзадачі і виконувати відповідну їй функцію. Крім функціонального призначення модуль характеризується також набором даних, необхідних йому для виконання його функції, тобто:

Модуль = Функція + Дані.

Таким чином, грамотна декомпозиція ґрунтується, насамперед, на аналізі функцій системи і необхідних для виконання цих функцій даних. Це яскраво відображається навіть в структурі проекту, тому що на кожну сутність таблицю створений окремий клас, що буде утримувати дані для операцій в цій сутності(рисунок 2.4).

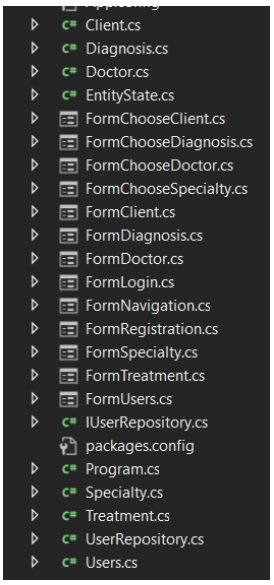


Рисунок 2.4 – Структура проекту

Діаграма послідовності. Діаграми послідовності показують, як об'єкти взаємодіють один з одним у хронологічному порядку. Ці діаграми, зокрема, відображають залучені об'єкти та послідовність повідомлень.

Інакше кажучи, діаграма послідовності показує часові аспекти передачі та отримання повідомлень об'єктом.

Діаграми послідовності можна використовувати для деталізації логіки варіантів використання і для того, щоб зробити діаграми прецедентів більш зрозумілими. Це чудовий метод для документування варіантів використання проекту. Діаграми послідовності зазвичай показують об'єкти в сценарії, які взаємодіють, повідомлення, якими вони обмінюються, і результати, які відповідають цим повідомленням.

На діаграмі послідовності відображаються лише ті об'єкти, які безпосередньо беруть участь у взаємодії.

На діаграмі послідовності пунктирна вертикальна лінія, з'єднана з одним об'єктом, являє собою лінію життя об'єкта. Лінія життя слугує візуальною підказкою того, як довго об'єкт є частиною системи і, таким чином, має можливість брати участь у всіх її взаємодіях [17]. Лінія життя об'єкта повинна простягатися від верхньої точки діаграми послідовності до нижньої, якщо він є постійною частиною системи.

Під час роботи об'єктно-орієнтованих систем певні об'єкти можуть перебувати в активному стані, безпосередньо виконуючи певні завдання, а можуть перебувати в пасивному стані, очікуючи на повідомлення від інших об'єктів. Фокус управління - це унікальне поняття в мові UML, яке використовується для явного виділення такої активності об'єкта. Фокус управління представляється у вигляді витягнутого вузького прямокутника, верхня сторона якого позначає початок фокусу управління, а нижня - його кінець. Прямокутник знаходиться під символом об'єкта, і якщо він активний на всьому протязі, то може зайняти місце лінії життя.

Кожна взаємодія в UML представляється набором повідомлень, за допомогою яких об'єкти, що беруть участь у ній, спілкуються один з одним. Повна інформація, що надсилається від одного об'єкта до іншого, називається повідомленням. Коли об'єкт отримує повідомлення, він починає виконувати певні дії, покликані допомогти йому вирішити конкретне завдання.

Діаграма послідовності проектованої системи зображена в Додатку А.

2.3 Розробка збережувальних процедур

При розробці додатків, основаних на принципі «клієнт – сервер», для полегшення виконання будь-яких операцій з даними використовуються механізми, за допомогою яких можна створювати підпрограми, що працюють на сервері і керують процесами обробки інформації. Ці механізми носять назву збережувальні процедури.

В кваліфікаційній роботі магістра було розроблено ряд збережувальних процедур, які використовуються клієнтською частиної системи для внесення та оновлення даних конкретної таблиці.

Код збережувальних процедур для внесення та оновлення даних спеціалізацій знаходиться в лістингу 2.1.

Лістинг 2.1 – 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 2.2.

Лістинг 2.2 – 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 2.3.

Лістинг 2.3 – 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 2.4.

Лістинг 2.4 – 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 2.5.

Лістинг 2.5 – 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()
```

2.4 Розробка механізмів управління даними в базі за допомогою тригерів

Унікальний клас збережених процедур, відомих як тригери, активується (запускається) у відповідь на зміни, внесені до даних таблиці. Тригери корисні для багатьох речей, таких як перевірка даних та виконання складних бізнес-правил. Здатність тригерів отримувати доступ до записів як до, так і після модифікації є дуже корисною функцією [19]. Вона дозволяє порівняти два записи і вирішити, який з них кращий.

Приклад створення та роботи тригера. Для таблиці лікування було розроблено тригер, який реагує при додаванні нового або оновлення існуючого запису в таблицю, якщо вартість буде менша нуля, тобто від'ємною, то дія не буде виконана. Код тригера знаходиться у лістингу 2.6.

Лістинг 2.6 – Код тригера

```
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 2.5.

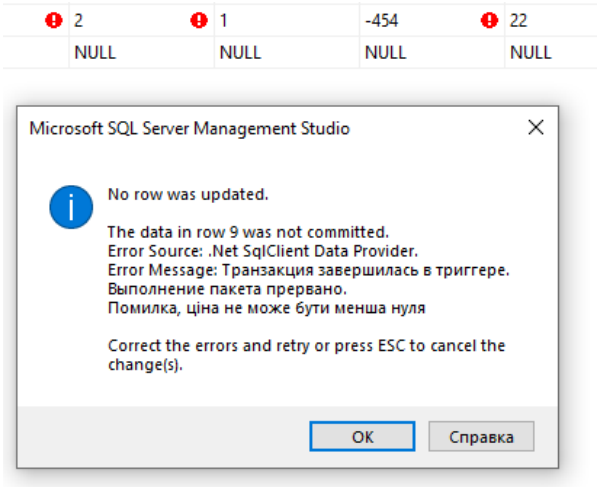


Рисунок 2.5 – Робота тригера

2.5 Розробка програмних рішень системи

Для десктопного застосунку з інтерфейсом для користувача, що базується на застосуванні бази даних надзвичайно важливо виконувати низку операцій, що буде дозволяти чотири операції, які вважаються необхідними для реалізації програми

постійного зберігання даних – це створення, читання, оновлення та видалення. Вони відомі під абревіатурою CRUD, яка походить зі сфери комп'ютерного програмування. Будь-який пристрій зберігання даних, наприклад, твердотільний накопичувач або жорсткий диск, який продовжує працювати після вимкнення живлення, називається постійним сховищем. З іншого боку, змінна пам'ять включає в себе такі речі, як пам'ять з довільним доступом і внутрішнє кешування, які зберігають інформацію, що буде видалена, коли вони втратять живлення.

Ключові визначення:

– Всі основні функції реляційних баз даних і програмного забезпечення для управління ними, таких як Oracle Database, Microsoft SQL Server, MySQL та інших, позначаються абревіатурою CRUD.

– За допомогою чотирьох функцій CRUD можна виконувати різні види операцій над конкретними даними бази даних.

– Багато програм, які покладаються на реляційні бази даних, широко використовують операції CRUD.

– Можливості агрегації журналів можна використовувати для відстеження загального обсягу команд CRUD з плином часу, кореляції результатів з іншими важливими метриками, а також для визначення та усунення причин низької продуктивності.

Підприємства, які ведуть облік своїх клієнтів, рахунків, платежів, медичної інформації та іншої інформації, потребують апаратного та програмного забезпечення для зберігання даних, яке може зберігати інформацію протягом тривалого часу. Зазвичай цю інформацію впорядковують у базах даних, які, по суті, є колекціями даних, упорядкованих і доступних в електронному вигляді. Існує багато різновидів баз даних, зокрема об'єктно-орієнтовані, графові та ієрархічні бази даних. Реляційна база даних, яка є найпоширенішим типом баз даних, складається з даних, які розміщені в таблицях у вигляді рядків і стовпців і пов'язані з іншими таблицями з додатковими даними за допомогою системи ключових слів, що включає первинні ключі та зовнішні ключі.

Отож далі буде виконана практика реалізації CRUD з обраними технологіями Windows Forms та C# з SQL Server.

Для створення потрібної таблиці, виконуємо код T-SQL у SQL Server Management Studio (лістинг 2.7).

Лістинг 2.7 - Код T-SQL для створення таблиці Contacts

```
CREATE TABLE [dbo].[Contacts](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [FullName] [nvarchar](100) NULL,
    [Email] [varchar](100) NULL,
    [Address] [nvarchar](100) NULL,
 CONSTRAINT [PK_Contacts] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
      GO
```

Відкриваємо файл app.config, а потім додаємо рядок підключення до бази даних, змінюємо рядок підключення на основі свого користувача та пароля сервера sql (лістинг 2.8).

Лістинг 2.8 - Код файлу app.config

```
<connectionStrings>
    <add name="cn" connectionString="Data Source=(localdb)\LocalDB;Initial
    Catalog=paid_clinic;Integrated Security=True;"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

Створюємо метод InitData, що дозволить отримувати контактні дані з таблиці контактів (лістинг 2.9).

Лістинг 2.9 - Код методу InitData

```
void InitData()
{
    using (SqlConnection cn = new SqlConnection(_connectionString))
    {
        DataTable dt = new DataTable("Customer");
        SqlDataAdapter dataAdapter = new SqlDataAdapter("select * from contacts",
cn);
        dataAdapter.Fill(dt);
        dataGridView.DataSource = dt;
    }
}
```

Створення методу Clear, дозволить очистити текстові дані в елементі керування текстовим полем (лістинг 2.10).

Лістинг 2.10 - Код методу Clear

```
void Clear()
{
```

```
    txtFullName.Text = string.Empty;
    txtEmail.Text = string.Empty;
    txtAddress.Text = string.Empty;
}
```

Додавання обробника події клацання до кнопки «Вставити» дозволяє вставляти дані з текстового поля до таблиці контактів (лістинг 2.11).

Лістинг 2.11 - Код обробника події клацання до кнопки «Вставити»

```
private void btnInsert_Click(object sender, EventArgs e)
{
    using (SqlConnection cn = new SqlConnection(_connectionString))
    {
        if (cn.State == ConnectionState.Closed)
            cn.Open();
        using (SqlCommand cmd = new SqlCommand("insert into contacts(fullname,
email, address) values(@FullName, @Email, @Address)", cn))
        {
            cmd.Parameters.AddWithValue("@FullName", txtFullName.Text);
            cmd.Parameters.AddWithValue("@Email", txtEmail.Text);
            cmd.Parameters.AddWithValue("@Address", txtAddress.Text);
            cmd.ExecuteNonQuery();
            InitData();
            Clear();
        }
    }
}
```

Додавання обробника події клацання до кнопки «Оновити» дозволяє оновлювати контактні дані з елемента керування текстовим полем (лістинг 2.12).

Лістинг 2.12 - Код обробника події клацання до кнопки «Оновити»

```
private void btnUpdate_Click(object sender, EventArgs e)
{
    using (SqlConnection cn = new SqlConnection(_connectionString))
    {
        if (cn.State == ConnectionState.Closed)
            cn.Open();
        using (SqlCommand cmd = new SqlCommand("update contacts set FullName =
@FullName, Email = @Email, Address = @Address where Id = @Id", cn))
        {
            DataGridViewRow row = dataGridView.Rows[dataGridView.CurrentRow.Index]
as DataGridViewRow;
            if (row != null)
            {
                cmd.Parameters.AddWithValue("Id", row.Cells[0].Value);
                cmd.Parameters.AddWithValue("@FullName", txtFullName.Text);
                cmd.Parameters.AddWithValue("@Email", txtEmail.Text);
                cmd.Parameters.AddWithValue("@Address", txtAddress.Text);
                cmd.ExecuteNonQuery();
                InitData();
            }
        }
    }
}
```

Додавання обробника подій CellClick до DataGridView дозволяє заповнювати дані з DataGridView в елемент керування TextBox (лістинг 2.13).

Лістинг 2.13 - Код обробника подій CellClick до DataGridView

```
private void dataGridView_CellClick(object sender, DataGridViewCellEventArgs e)
{
    DataGridViewRow row = dataGridView.Rows[dataGridView.CurrentRow.Index] as
DataGridViewRow;
    if (row != null)
    {
        txtFullName.Text = row.Cells[1].Value.ToString();
        txtEmail.Text = row.Cells[2].Value.ToString();
        txtAddress.Text = row.Cells[3].Value.ToString();
    }
}
```

Додавання обробника події клацання до кнопки «Видалити» дозволяє видалити контактні дані в таблиці контактів (лістинг 2.14).

Лістинг 2.14 - Код обробника події клацання до кнопки «Видалити»

```
private void btnDelete_Click(object sender, EventArgs e)
{
    DataGridViewRow row = dataGridView.Rows[dataGridView.CurrentRow.Index] as
DataGridViewRow;
    if (row != null)
    {
        if (MessageBox.Show("Are you sure want to delete this record?", "Message",
MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
        {
            using (SqlConnection cn = new SqlConnection(_connectionString))
            {
                if (cn.State == ConnectionState.Closed)
                    cn.Open();
                using (SqlCommand cmd = new SqlCommand("delete from contacts where
Id = @Id", cn))
                {
                    cmd.Parameters.AddWithValue("Id", row.Cells[0].Value);
                    cmd.ExecuteNonQuery();
                    InitData();
                    Clear();
                }
            }
        }
    }
}
```

Додавання обробника події KeyPress до елемента керування текстовим полем пошуку дає змогу вловлювати клавішу enter (лістинг 2.15).

Лістинг 2.15 - Код обробника події KeyPress

```
private void txtSearch_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == (char)13)
    {
        using (SqlConnection cn = new SqlConnection(_connectionString))
        {
            DataTable dt = new DataTable("Customer");
            SqlDataAdapter dataAdapter = new SqlDataAdapter("select * from
contacts where fullname like @Search", cn);
            dataAdapter.SelectCommand.Parameters.AddWithValue("@Search",
$"%{txtSearch.Text}%");
            dataAdapter.Fill(dt);
```

```
                dataGridView.DataSource = dt;
            }
        }
}
```

На прикладі CRUD (Create, Read, Update, Delete) C# було виконано простий спосіб додавання, читання, оновлення, видалення даних із datagridview у програму C# Windows Forms. Результат у вигляді форми зображено на рисунку 2.6.

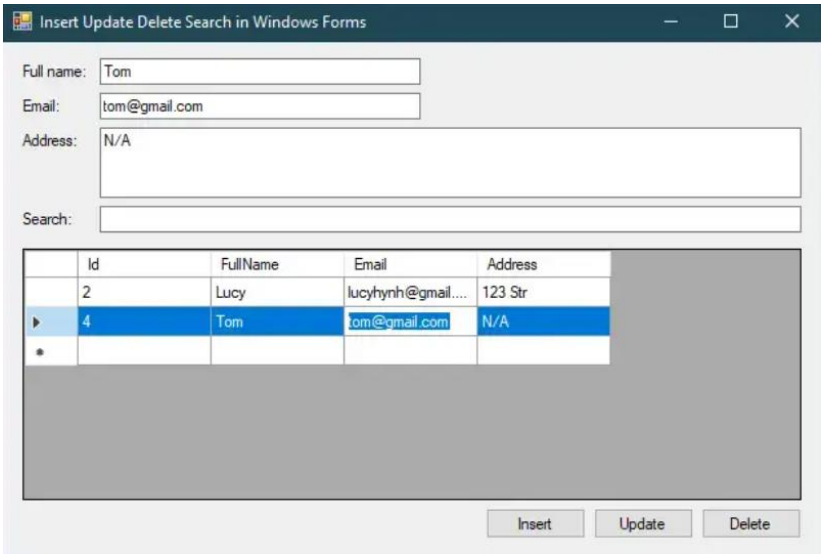


Рисунок 2.6 – Вигляд CRUD Windows Forms C#

Елемент управління DataGridView спрощує процес визначення візуальних аспектів комірок і форматування відображення значень комірок. Комірки слугують фундаментальними одиницями взаємодії в DataGridView, всі вони походять від базового класу DataGridViewCell. Кожна комірка в елементі управління може мати свій власний унікальний стиль, що охоплює такі елементи, як формат тексту, колір фону, колір переднього плану і шрифт [19]. Тим не менш, зазвичай декілька клітинок мають спільні характеристики стилю. За замовчуванням тип даних для властивості Value клітинки - об'єкт.

DataGridView чи не найкраще підходить для відображення записів з бази даних, що необхідно для створення десктопного застосунку.

Створюємо форму з елементом DataGridView та заповнюємо дані з готового статичного джерела (рисунок 2.7).

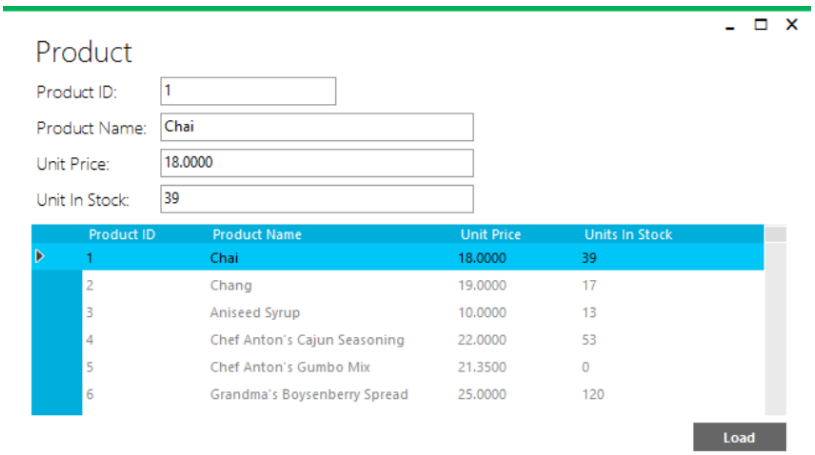


Рисунок 2.7 – Використання DataGridView на практиці

Також в майбутньому застосунку обов'язково планується створення форм для реєстрації та входу користувачів для автентифікації в системі управління даними поліклініки. Для практики створення подібного виду форм було створено тестовий вигляд (рисунок 2.8).

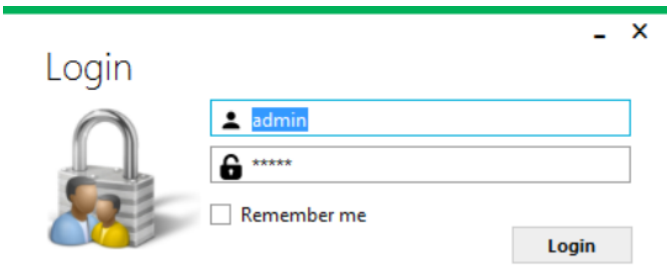


Рисунок 2.8 – Форма автентифікації в системі

2.6 Дослідна експлуатація та тестування готової програми поліклініки

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 2.9.

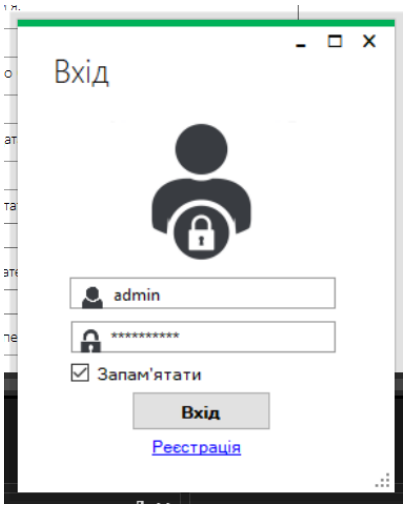


Рисунок 2.9 – Форма входу

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

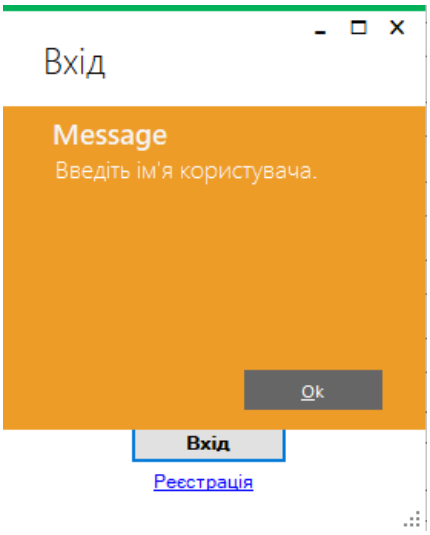


Рисунок 2.10 – Повідомлення про відсутність імені користувача

При введені невірних даних та натиснені на кнопку входу також буде виведено відповідне повідомлення, яке продемонстровано на рисунку 2.11.

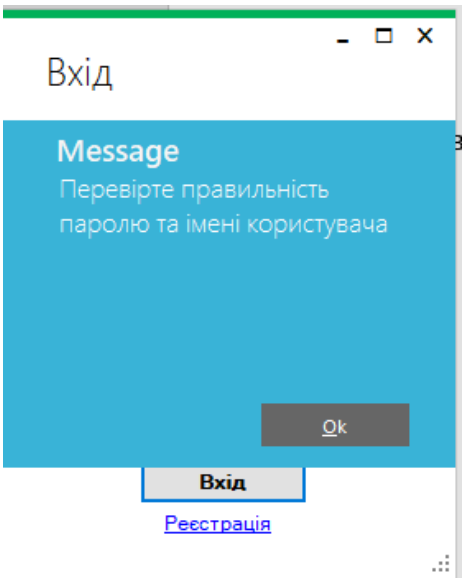


Рисунок 2.11 – Повідомлення про неправильність даних входу

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 2.12.

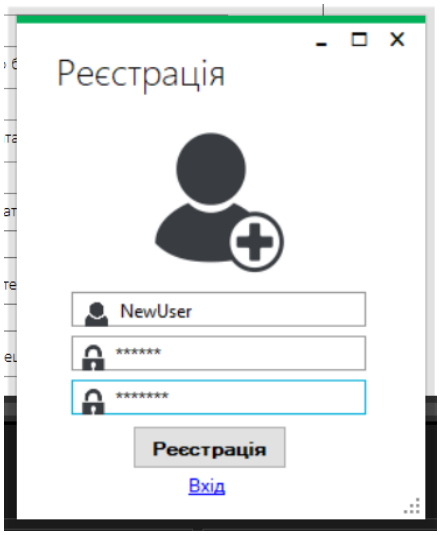


Рисунок 2.12 – Форма реєстрації

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 2.13.

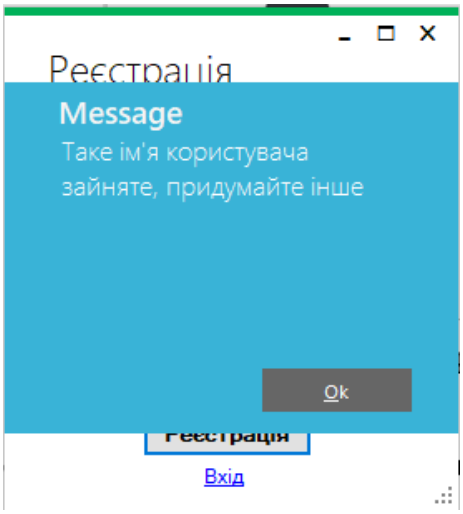


Рисунок 2.13 – Повідомлення про зайнятість введеного імені користувача

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 2.14.

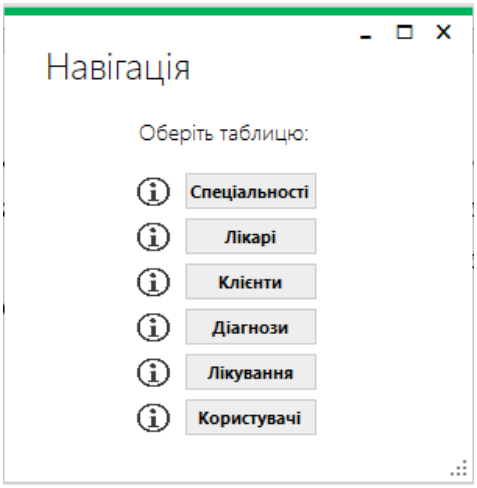


Рисунок 2.14 – Форма навігації

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 2.15.

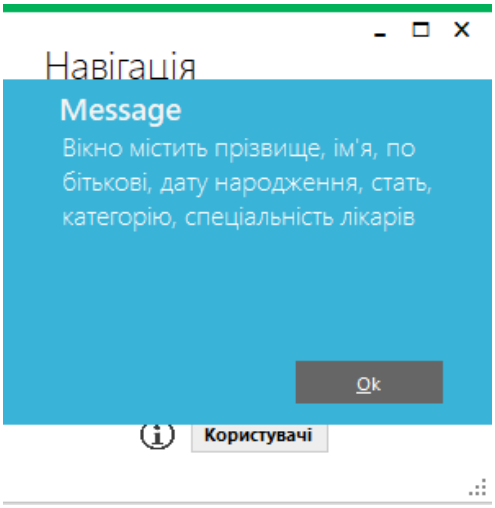


Рисунок 2.15 – Вигляд інформування про таблицю

При натисненні кнопки Лікарі на формі навігації буде відкрито вікно для управління відповідною таблицею. Вигляд відкритої форми Лікарі продемонстровано на рисунку 2.16.

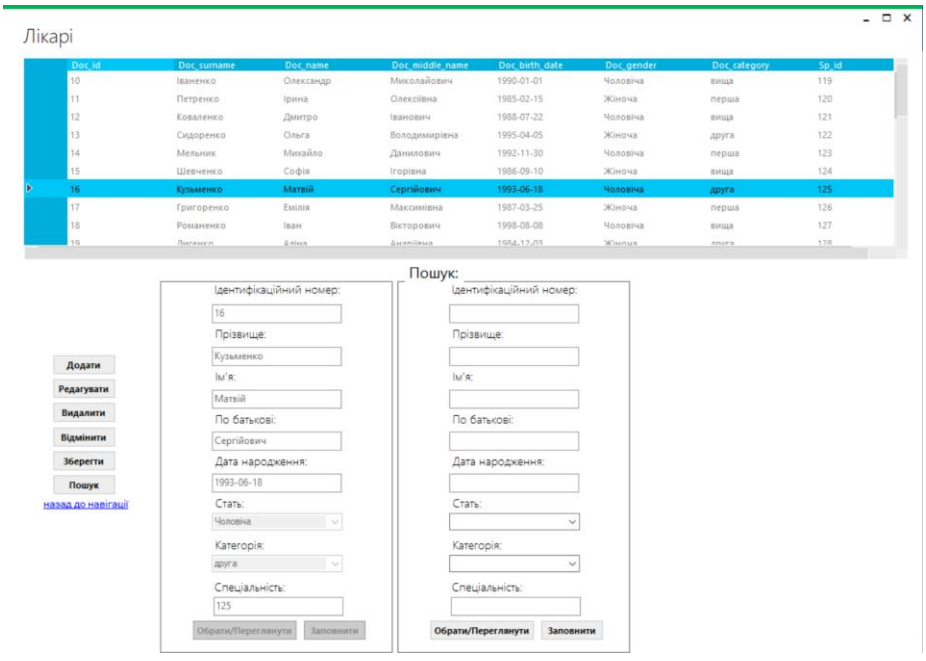


Рисунок 2.16 – Вигляд форми управління таблицею Лікарі

Усі інші форми призначенні для управління таблицями бази даних виглядають подібним чином, відрізняються лише кількістю та змістом текстових полів. На формі присутні наступні елементи: поле для відображення інформації з бази даних, панель, що містить текстові поля для редагування, додавання та зміну обраного запису, панель з полями для пошуку відповідного запису у полі даних бази. Кнопки виконують відповідно до їх назви функції.

Панелі змінюють свою властивість активносі при натиснені певних кнопок, що надають доступ до полі відповідної панелі. При натиснені кнопок додати, редагувати активується панель для цих операцій та леактивується панель пошуку. Кнопка пошуку активовує панель пошуку відповідно. Кнопка відмінити виконує деактивацію панелей та анулює попередньо введені критерії пошуку відображаючу у полі даних усі існуючі в базі.

Також присутнє текстове посилання для повернення до форми навігації та вибрання наступної необхідної користувачу таблиці.

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 2.17.

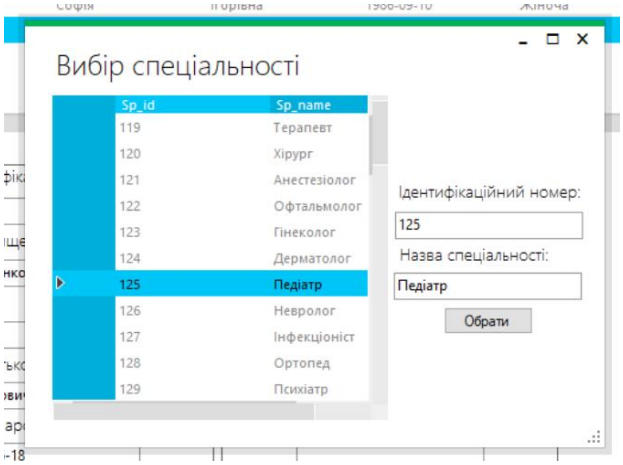


Рисунок 2.17 – Вигляд форми побічного вибору спеціальності

На формі знаходиться вже знайоме поле відображення даних з бази, відповідні до таблиці текстові поля та кнопка обрати, яка виконує запам’ятовування вибіру у глобальній змінні та буде доступна на поточній формі таблиці, після натиснення кнопки форма побічного вибору закривається. Настпне натиснення на формі таблиці кнопки Заповнити внесе обраний код запису у відповідне текстове поле.

Деякі текстові поля недоступні для редагування у тих чи інших обставинах роботи додатку для запобігання внесення інформації неправильно формату чи невірного значення, що може нашкодити правильності та точності роботи додатку, як його клієнтській частині так і серверній.

З будь-якої форми можна перейти на потрібну іншу потрібну форму. Додаток закривається також з будь-якої точки роботи додатку натиснувши на кнопку закриття вікна. Приклад пошуку за статтю лікара зображено на рисунку 2.18.

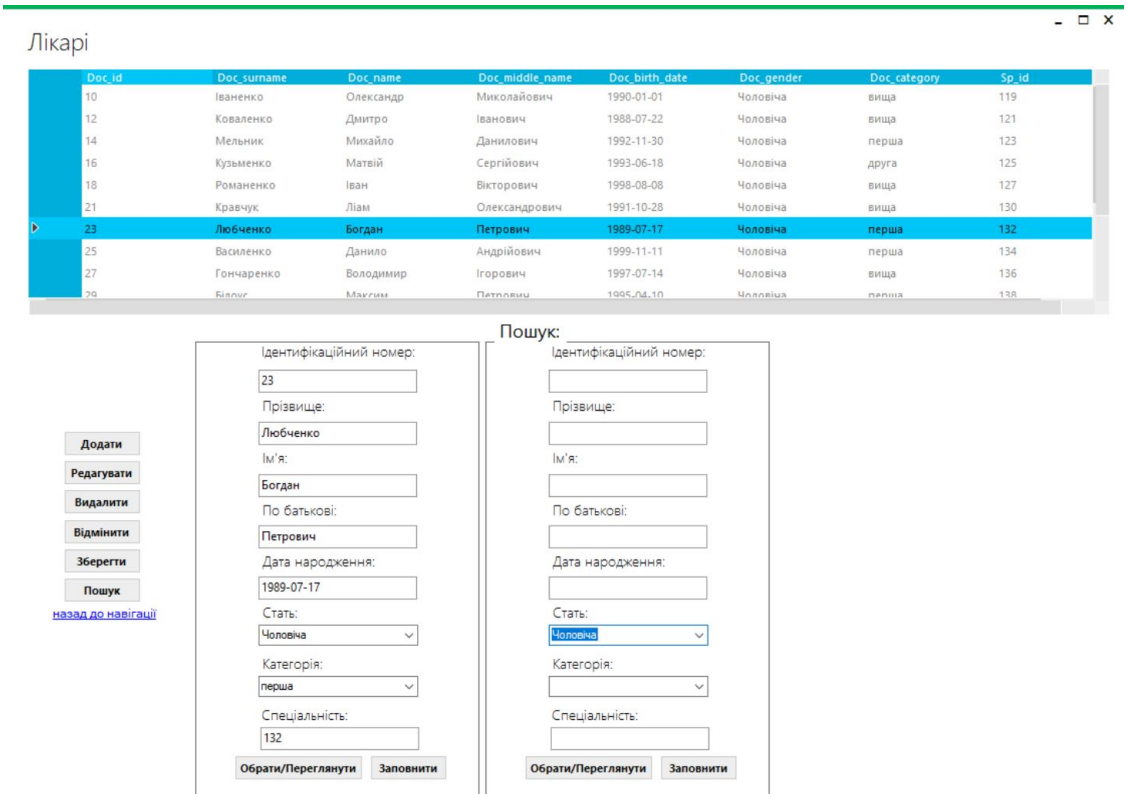


Рисунок 2.18 – Приклад застосування пошуку за статтю лікара

Після тестування додатку не було виявлено помилок, які б перешкоджали запланованій та правильній роботі системи.

Діаграма розгортання десктопного додатку прямо дорівнює діаграмі клієнт-сервера(рисунок 2.19).

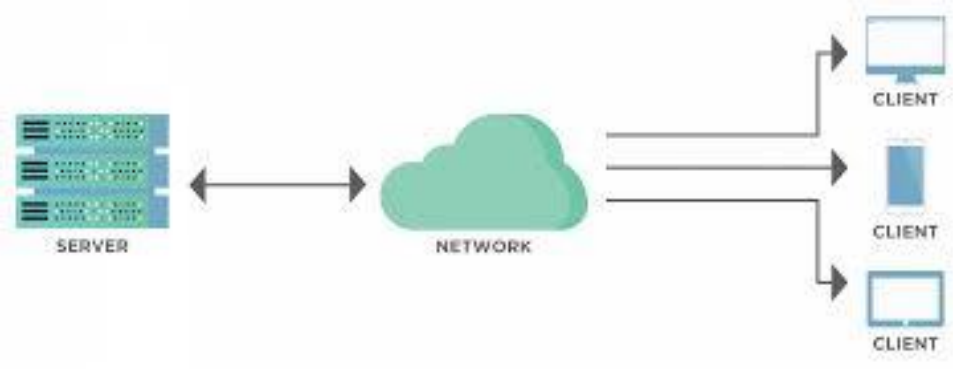


Рисунок 2.19 – Діаграма розгортання додатку

Уся система містить сервер, де зберігаються усі дані, певні протоколи та підключення, щоб підключитись до сервера та клієнтські комп’ютери, які отримують дані з сервера та працюють з ними.

3 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

3.1 Охорона праці

При розробці бази даних та системи для контролю та управління взаємовідносинами із потенційними клієнтами інженер-програміст використовує персональні електронно-обчислювальні машини. Тому виконання правил з охорони праці є важливим фактором для запоруки безпеки та здоров‘я.

Питання охорони праці регулюються певними законодавчими та нормативно-правовими актами, які, зокрема, визначають обов’язки роботодавця із забезпечення робітникам комфортних та безпечних умов для здійснення роботи. Ці обов’язки, а також права робітників на таких умовах праці передбачені частиною 2 ст.2 і частиною 1 ст.21 КЗпП, а також ст.13 Закону України «Про охорону праці», у яких визначаються основні положення з реалізації конституційного права робітників.

В залежності від розміру компанії та кількості працівників які задіяні в розробці системи знаходження критичного шляху служба охорони праці може мати різні розміри та форми.

Підпорядковується служба охорони праці згідно із законодавством безпосередньо роботодавцеві.

Проте роботодавець може доручити функціональне управління (кураторство) діяльністю служби іншій посадовій особі, скажімо, головному інженерові, заступникові директора з охорони праці тощо. Функції служби охорони праці:

1. Підготовка проектів наказів (розпоряджень) з питань охорони праці і внесення їх на розгляд роботодавцю. Проведення спільно з представниками інших структурних підрозділів і за участю представників професійної спілки підприємства.

2. Проведення з працівниками вступного інструктажу з питань охорони праці та супутніх інструктажів.

3. Ведення обліку та проведення аналізу причин виробничого травматизму, професійних захворювань, аварій на виробництві, заподіяної ними шкоди.

4. Забезпечення належного оформлення і зберігання документації з питань охорони праці, а також своєчасної передачі її до архіву для тривалого зберігання згідно з установленим порядком.

5. Складання звітності з охорони праці за встановленими формами.

6. Складання за участю керівників підрозділів підприємства переліків професій, посад і видів робіт, на які повинні бути розроблені інструкції з охорони праці, що діють в межах підприємства, надання методичної допомоги під час їх розроблення.

7. Інформування працівників про основні вимоги законів, інших нормативно-правових актів та актів з охорони праці, що діють в межах підприємства.

8. Розгляд питань про підтвердження наявності небезпечної виробничої ситуації, що стала причиною відмови працівника від виконання дорученої роботи відповідно до законодавства (у разі необхідності).

9. Організаційне забезпечення підрозділів нормативно-правовими актами з охорони праці та актами з охорони праці, що діють в межах підприємства, посібниками, навчальними матеріалами з цих питань.

На перший погляд, робота за комп'ютером здається безпечною, але саме легковажність до неї може призвести до певних проблем у здоров'ї людини. Професія програміста та інших фахівців IT-технологій пов'язана з колосальним розумовим напруженням. Розробники – настільки захоплені люди, що навіть відволікаючись від роботи над проектом, продовжують думати про роботу. Нерідко відпочинком вони вважають паралельну заміну основної діяльності, наприклад, читання профільної літератури, верстку сайтів, вивчення нових мов програмування.

Працюючи за персональний комп’ютером ступінь розумової напруги значно зростає. Навантаження також спричиняє емоційне вигорання, є високим показником напруженості зору і навантаженням на м‘язи рук при використанні клавіатури. У певній мірі, підвищення ефективності і результативності праці

робітника залежить від факторів втомлюваності чи бадьорості. Згідно з Державними санітарними правилами і нормами роботи з візуальними дисплейними терміналами електронно обчислювальних машин ДСанПІН 3.3.2.007-98 відповідність розташування та організація робочого місця з ВДТ повинна відповідати ергономічними вимогами, враховуючи особливість трудової діяльності.

При розробці системи для контролю та управління взаємовідносинами із потенційними клієнтами повинні враховуватись існуючі санітарні нормативи освітлення, вимоги до параметрів мікроклімату (температура, відносна вологість), ступеня і сили вібрації, звукового шуму і вогнестійкості приміщення. Конкретні показники зазначених санітарних норм є у інструкції "Державні санітарні правила і норми роботи з візуальними дисплейними терміналами (ВДТ) електронно-обчислювальних машин" ДСанПіН 3.3.2.007-98.

Для об'єктивного оцінювання умов праці робочого місця здійснюється його атестація, а також «Гігієнічна класифікація праці». Дані показників загрозливих, шкідливих чинників залежать від самого виробничого середовища та гостроти напруження трудового процесу [21].

Розробка системи для кваліфікаційної роботи магістра, а саме бази даних та графічного інтерфейсу для взаємодії користувача з нею відбувається в офісному приміщенні, в якому знаходяться комп'ютеризовані робочі місця та експлуатується комп'ютерна техніка інженерами-програмістами. Під час планування приміщення слід дотримуватись державних нормативно-правових актів та законів. Згідно ДБН В.2.5-28:2018 приміщення, в якому розробляється система оцінки вартості житлового будівництва, повинно мати природне і штучне освітлення. Природне освітлення має здійснюватися через світлові прорізи, орієнтовані переважно на північ чи північний схід, і забезпечувати коефіцієнт природної освітленості (КПО) не нижче, ніж 1,5%. Також у приміщеннях глибиною 6 м та більше доцільно застосовувати на вікнах спеціальні світло відбивні екрани та жалюзі, що перерозподіляють світловий потік в глибину приміщення.

Під час роботи за комп’ютером інженер-програміст піддається впливу небезпечних та шкідливих здоров’ю чинників. Психофізіологічні чинники: напруження уваги, пам’яті та зору; значна кількість інформації, яку необхідно обробити за одиницю часу; в деяких випадках монотонність завдань. Фізичні чинники: збільшений рівень інфрачервоного, ультрафіолетового та рентгенівського випромінювання; відсутність рівномірного розподілу яскравості на робочому місці. Для зменшення впливу негативних чинників, під час розробки програмної системи, було правильно розміщено робочі місця та комп’ютерне обладнання. Це питання регулюється широким спектром нормативно-правових актів. Законом України “Про охорону праці” передбачено обов’язки роботодавця, які описують необхідність забезпечення безпечних та комфортних умов праці.

За таких умов зростає роль та значення охорони праці як системи правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, що спрямовані на збереження здоров’я і працездатності людини в процесі праці [22].

Таким чином, система управління базою даних з графічним інтерфейсом для взаємодії з користувачем, виконана з вимог техніки безпеки. Правильна організація робочого простору є важливим та доцільним елементом, для забезпечення якого необхідно уникати незручності в розташуванні засобів праці.

3.2 Безпека в надзвичайних ситуаціях

Уряд, який надає першочергового значення зміцненню обороноздатності нашої держави, неодноразово підкреслював, що оборонна міць держави не обмежується високим рівнем боєготовності та оснащення Збройних Сил; вона також тісно пов'язана з високим рівнем економічного розвитку країни та готовністю населення і економіки до захисту від зброї масового знищення.

Здатність економічних об'єктів країни виробляти товари в запланованому обсязі та асортименті в мирний і воєнний час, а також швидко відновлювати виробництво в разі незначних або середніх пошкоджень, перебоїв у співпраці та ланцюгах поставок широко розуміється як міра їхньої стійкості [20].

Збереження та належне функціонування чотирьох основних компонентів сучасного виробництва наведених нижче - є передумовою здатності суб'єкта господарювання виробляти товари:

– робітники та інший виробничий персонал;

– технологічно оснащені будівлі та споруди;

– енерго-, водо-, паливо-, обладнання та ремонтна база;

– виробнича система та коопераційні зв'язки з іншими об'єктами.

Найважливішим завданням для підвищення стійкості суб'єкта господарювання є захист робітників і службовців у мирний і воєнний час. Оскільки робітники і службовці є основним джерелом продуктивності, захист і збереження цієї сили є ключовим фактором, що визначає стійкість економіки.

Під час військових конфліктів руйнуються будівлі та інші споруди, а також знищується працездатне населення - першоджерело виробництва. Як наслідок, найважливішим завданням серед усіх заходів, спрямованих на зміцнення стійкості національної економіки, є якнайшвидше забезпечення захисту робітників і членів їхніх сімей.

У нинішніх умовах існує три основні методи захисту робітників і службовців від зброї масового ураження:

– Людей слід укривати в захисних спорудах, таких як бомбосховища або протирадіаційні укриття.

– Працівники та їхні сім'ї також повинні бути евакуйовані.

– Необхідно носити засоби індивідуального захисту, а також вживати заходів обережності від радіації, хімікатів і бактерій, залежно від ситуації.

Підприємство вважається стійким, якщо воно може в разі надзвичайної ситуації виробляти продукцію в запланованому обсязі та асортименті, а також

може швидко відновити виробництво власними силами в разі незначних або середніх пошкоджень чи перебоїв у постачанні матеріалів.

Завдяки завчасному здійсненню низки інженерних, технологічних та організаційних кроків, спрямованих на зменшення руйнівного впливу зброї масового знищення та створення умов для швидкої ліквідації наслідків, економічні об'єкти можуть стати більш стійкими під час війни або надзвичайних ситуацій. Для того, щоб забезпечити безперебійне відновлення перерваного виробництва, підготовчі заходи проводяться заздалегідь. Це включає в себе організацію процесу реконструкції в різних аспектах, таких як підготовка ремонтних бригад, збір необхідних матеріалів і техніки, а також гарантування її надійної безпеки.

Для підвищення стійкості виробничих будівель і споруд, верстатів і технологічного обладнання, а також інженерних та енергетичних систем, інженерно-технічні заходи зазвичай включають в себе цілий ряд дій.

Технологічні заходи, змінюючи технологічний процес, забезпечують стійкість об'єкта, прискорюючи виробництво та усуваючи ймовірність вторинних впливів.

Організаційні заходи передбачають розробку та підготовку планів дій керівництва, командно-диспетчерського складу, штабів, служб та підрозділів цивільного захисту для забезпечення безпеки працівників підприємства та виконання інших невідкладних завдань, відновлення виробництва та випуску продукції на збережених об'єктах.

Для виробництва товарів необхідні електроенергія, вода, паливо, сировина та інші матеріали, а також технічні засоби. Доступність цих ресурсів для бізнесу визначає, чи зможе він продовжувати нормально функціонувати під час війни. Це досягається шляхом реалізації заходів, які сприяють підвищенню цілісності транспортних мереж, енергетичних та інженерних мереж, джерел постачання, а також надійному захисту необхідних запасів палива, сировини, напівфабрикатів, комплектуючих та інших запасів.

Це одна з найважливіших передумов для своєчасного та успішного створення проекту під час розробки системи електропостачання. Повна зупинка об'єкта може

бути наслідком переривання регулярного електропостачання окремих виробничих ділянок об'єкта. Щоб гарантувати надійне електропостачання під час війни, під час проектування та будівництва слід враховувати такі основні вимоги, що випливають із завдань цивільної оборони, щоб гарантувати надійне електропостачання.

До основних вимог систем електропостачання відносять

– Енергетичні системи, що складаються з електростанцій, які працюють на різних видах палива, повинні забезпечувати електроенергією. Великі електростанції повинні бути розташовані щонайменше в двох радіусах від великих міст і одна від одної, щоб запобігти потенційним руйнуванням.

– Великі міста та інші установи, які продовжують функціонувати під час війни, повинні забезпечуватися електроенергією з двох окремих джерел. У випадку, якщо об'єкт живиться від одного джерела, має бути щонайменше два входи, що надходять з різних сторін.

– Стабільність трансформаторної підстанції повинна дорівнювати або перевищувати стабільність об'єкта, і вони повинні бути надійно захищені. Незалежні електричні кабелі, закопані на глибині від 0,8 до 1,2 метра, повинні забезпечувати електроенергією виробничі майданчики.

– Розвиток незалежних резервних джерел живлення є вкрай необхідним. Для цього можна використовувати мобільні електростанції на кораблях і залізничних платформах, електростанції малої потужності, не підключені до електромережі, тощо.

– Невеликі стаціонарні електростанції слід передбачати як резервні об'єкти при проектуванні систем електропостачання.

ВИСНОВКИ

Використовуючи методологію RUP та мову програмування C#, в цій роботі було проведено ретельний аналіз та розробку бази даних та десктопного додатку для поліклініки. Основною метою проекту була розробка ефективної автоматизованої системи, яка буде керувати всіма аспектами діяльності поліклініки.

Результатом дослідження є база даних, що містить інформацію про пацієнтів та персонал, а також інші важливі дані для повсякденної роботи медичного закладу. Ефективний доступ до даних та їх обробка забезпечується за допомогою фреймворку Entity Framework Core та інтеграції з MS SQL Server Management Studio. Десктопний додаток має простий у використанні інтерфейс, який враховує унікальні вимоги роботи клініки. Він був створений за допомогою мови програмування C#. В процесі розробки були створені прикладні програми, DDL-скрипти, моделі даних, а також ретельний контроль за ефективністю системи.

Зростаюча популярність автоматизованих систем управління охороною здоров'я підкреслює актуальність теми. Застосування отриманих результатів може покращити догляд за пацієнтами, підвищити продуктивність клініки та заощадити ресурси. Інші медичні заклади можуть використовувати інформацію та досвід, викладені в цій роботі, для подальшої розробки та вдосконалення подібних систем. Вивчення та застосування сучасних методологій розробки програмного забезпечення, а також перевірка їх ефективності в медичній галузі принесли значну користь від цієї роботи.

Роботу можна вважати успішною та перспективною в галузі автоматизації управління медичним закладом, оскільки основні завдання, поставлені перед нею, були успішно виконані, а високий рівень професіоналізму та ретельне опрацювання всіх етапів розробки дозволило досягти поставлених цілей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Haigh, T. (2014). T-SQL Querying (4th ed.). Microsoft Press. This book focuses specifically on T-SQL, the dialect of SQL used in Microsoft SQL Server, and provides in-depth coverage of advanced features and optimization techniques.

2) 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", 2016. – 175 С. С.: іл. Isbn (укр.)

3) Freeman, B., & Bates, E. (2008). Head First C#: A learner's guide to object-oriented programming with C# and.NET. O'Reilly Media. This visual and engaging book introduces C# and object-oriented programming principles in a fun and accessible way.

4) Troelsen, A. C. (2017). Pro C# 7 with .NET and the Core Framework: Mastering Modern Application Development. Apress. This comprehensive textbook covers all aspects of C# programming, including object-oriented programming, advanced language features, and .NET libraries.

5) Scharpf, M., & Antonovich, P. (2020). C# 9 in a Nutshell: The concise guide to the new features of C# 9 and .NET 5. O'Reilly Media. This book provides a concise overview of the new features introduced in C# 9 and .NET 5, making it a valuable resource for experienced developers.

6) Pressman, R. S., & Maxim, B. (2020). Software Engineering: A Practitioner's Approach. McGraw-Hill Education. This comprehensive textbook covers the entire software engineering process, from requirements gathering to deployment and maintenance.

7) Hunt, A., & Thomas, D. (2000). The Pragmatic Programmer: From Journeyman to Master. Addison-Wesley. This practical guide offers valuable advice on the software development life cycle, code review, testing, and professional practice.

8) SFML. Вікіпедія - вільна енциклопедія: веб-сайт. URL: https://uk.wikipedia.org/wiki/Windows_Forms.

9) Методичні вказівки до виконання атестаційної роботи магістра за спеціальністю 121 – Інженерія програмного забезпечення (Освітньопрофесійна програма - «Програмне забезпечення систем», Освітньо-наукова програма - «Інженерія програмного забезпечення») для студентів усіх форм навчання / Упор.: М.Р. Петрик, Д.М. Михалик, О.Ю. Петрик, Г.Б. Цуприк - Тернопіль: ТНТУ, 2020- 51с.

10) Martin, R. C. (2008). Clean Code: A Handbook of Agile Software Craftsmanship. Addison-Wesley. This classic book emphasizes code clarity, maintainability, and best practices for writing robust and sustainable software.

11) Грекул В.І., Денищенко Г.Н., Коровкіна Н. Л. Проектування інформаційних систем [Текст] / В.І. Грекул – М.: Інтернет-університет інформаційних технологій – ІНТУІТ.Ру, 2005.

12) Програмування і математичне моделювання : підручник для студ. вищих навч. закл. / І. О. Хвищун ; Львівський національний ун-т ім. Івана Франка. - Київ 72 : Ін Юре : Видавничий центр Львівського національного університету ім. Івана Франка, 2007. - 544 с.: рис. - ISBN 978-966-613-516-5. - ISBN 978-966-313-315-5

13) Виноградська А.М. Технологія комерційного підприємництва: навч. посібник / А.М.Виноградська. – До.: Центр навчальної літератури, 2006. – 780 с.

14) Standard Glossary of Terms used in Software Testing [Digital resource] – Mode of access: URL: https://www.astqb.org/documents/Glossary-of-SoftwareTesting-Terms-v3.pdf – Last access: 2023. – Title from the screen.

15) Robert Cecil Marin. Clean Architecture: A Craftsman’s Guide to Software Structure and Design [Text] – 2017 – 367 p.

16) Software Testing Life Cycle – Different Stages of Testing [Digital resource] – Mode of access: URL: https://www.edureka.co/blog/software-testing-life-cycle/ – Last access: 2020. – Title from the screen.

17) Robert Cecil Martin. Clean Code [Text] – 2008 – 94 p.

18) Why is software testing necessary? [Digital resource] – Mode of access: URL: http://tryqa.com/why-is-testing-necessary/ – Last access: 2023. – Title from the screen.

19) Google Tech Dev Guide [Digital resource] – Mode of access: URL: https://techdevguide.withgoogle.com/ – Last access: 2023. – Title from the screen.

20) Охорона праці в галузі комп'ютингу: підручник / Л. А. Катренко, А. В. Катренко ; [за наук. ред. В. В. Пасічника] ; М-во освіти і науки, молоді та спорту України. — Л. : Магнолія 2006, 2012. — 544 с

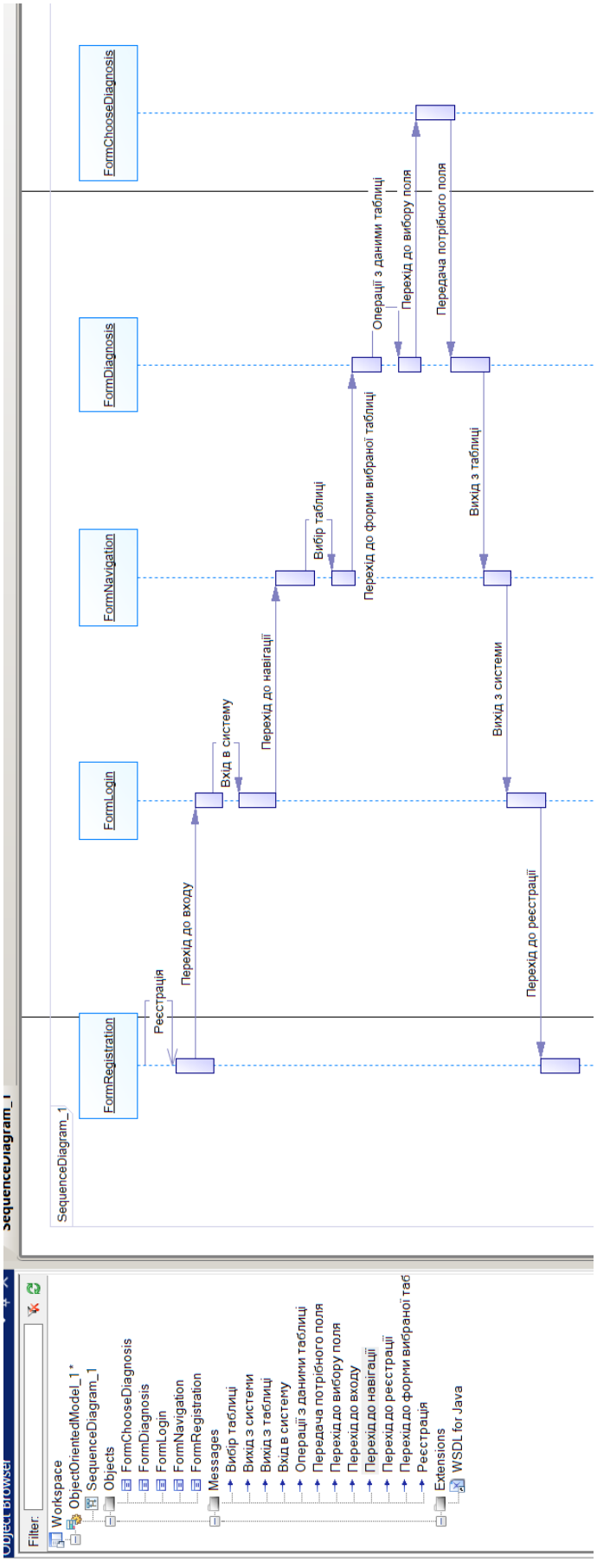
21) Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями [Електронний ресурс] – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508-18#n14.

22) Русаловський А. В. Правові та організаційні питання охорони праці: Навч. посіб. – 4-те вид. – К.: Університет «Україна», 2009. – 295 с.

ДОДАТКИ

Додаток А

Діаграма послідовності



Додаток Б

Публікація в науковому виданні

УДК 004.41

В. Рудак, Ю. Стоянов, канд. техн. наук
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ПРОЕКТУВАННЯ ТА РОЗРОБКА БАЗИ ДАНИХ ДЛЯ ПОЛІКЛІНІКИ З ВИКОРИСТАННЯМ МЕТОДОЛОГІЇ RUP ТА МОВИ ПРОГРАМУВАННЯ C#

V. Rudak, Y. Stoyanov, Ph.D.

DESIGN AND DEVELOPMENT OF DATABASE FOR POLYCLINIC USING RUP METHODOLOGY AND C# PROGRAMMING LANGUAGE

Сьогодні ведення бізнесу без комп'ютера просто неможливо уявити. У таких управлінських сферах, як бухгалтерський облік, закупівлі, управління асортиментом та складом, комп'ютери вже давно стали основною опорою. Тим не менш, сучасний бізнес потребує набагато ширшого використання інформаційних технологій в управлінні підприємством. Для того, щоб бути першим, інтуїції, особистого досвіду менеджера та капіталу вже недостатньо. Будь-який компетентний менеджер повинен постійно контролювати різноманітні види фінансово-господарської діяльності, включаючи торгівлю, виробництво та надання послуг, щоб приймати рішення в умовах невизначеності та ризику. Отже, сучасна управлінська стратегія передбачає інвестиції в інформаційні технології.

Створення інформаційної системи для платної поліклініки, яка дозволить будь-якому користувачеві легко знайти необхідну інформацію про будь-якого лікаря, клієнта або іншу потрібну інформацію, є основною вимогою проектованої системи. Інформаційна система поліклініки міститиме інформацію про лікарів, пацієнтів, діагнози, спеціалізації лікарів та методи лікування. А також інформацію про користувачів інформаційної системи, які матимуть доступ до системи, та бічні ключі від більшості доступних на даний момент таблиць, що запобігає необхідності доступу до кожної таблиці окремо. Дані можна переглядати, додавати, редагувати, шукати та видаляти за допомогою інструментів та інтерфейсу системи.

Дана тема є актуальною, оскільки завдяки розвитку інформаційних технологій вся паперова документація тепер може бути переведена в електронний формат, що дозволяє працівникам реєстратури швидко знаходити інформацію про пацієнтів, дзвінки та кабінети. Для створення автоматизованої інформаційної системи поліклініка найкращими за дослідженням є використання мови програмування C#, Windows Forms, MetroFramework, Dapper. Базу даних буде створено за допомогою SQL Server Management Studio 2019, SQL, Transact-SQL.

Зростає попит на складні програмні системи. Все більше і більше складних процесів можна автоматизувати, оскільки обчислювальні потужності стають доступнішими та продуктивнішими. При створенні складних інформаційних систем проектування в першу чергу корисне тим, що звільняє час для нудних, повторюваних завдань, щоб можна було сконцентруватися на вирішенні творчих завдань. Тому проектування буде виконано згідно усіх стандартів методології RUP.

Література

1. Страуструп Бьярне. Програмування. Принципи і практика з використанням C#. Київ : Фоліо, 2018. – 1328 С.

2. Довбуш Г. Visual C# на прикладах/ Галина Довбуш, Анатолий Хомоненко. Київ : Фоліо, 2015. – 528 С.

Додаток В

Програмний код

Лістинг В1 – 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"].ConnectionString))
                {
                    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 користувацьких
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 В2 – 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"].ConnectionString))
                {
                    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 = "Hi" });
                        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 В3 – 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 В4 – 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"].ConnectionString))
                {
                    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();
        }
    }
}
```