НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ

І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

_комп'ютерних наук_____

(назва кафедри)

_____ _доц., к.т.н. Голуб Б.Л._

(підпис) (ПІБ)

"____" червня 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему

«Інформаційна система відділення реєстратури в поліклініці»

Спеціальність 122 – «Комп'ютерні науки»

Гарант освітньої програми

_____д.е.н., професор_____ _____ _____Руденський Р.А.

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

_____д.е.н., професор_____ _____ _____Руденський Р.А.

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ _____Ануа О.В._____

(підпис) (ПІБ)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерних наук
_____ Голуб Б.Л.

"__" _____25 р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студенту

Ануа Ользі Вікторівні
(прізвище, ім'я, по батькові)

Спеціальність 122 – «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи:
~~Інформаційна система~~ відділення реєстратури в поліклініці

Затверджена наказом ректора НУБіП України від "25" 04 2025 р. № 699 "С"
Термін подання завершеної роботи на кафедру 2025.05.25
(рік, місяць, число)

Вихідні дані до бакалаврської кваліфікаційної роботи:
Опис програмного забезпечення

Перелік питань, які потрібно розробити:
1. Системний аналіз предметної області інформаційної системи
2. Проєктування інформаційного та програмного забезпечення
3. Розробка інформаційного та програмного забезпечення
4. Рекомендації щодо впровадження та експлуатації системи

Висновки

Дата видачі завдання "_____"_____20__р.

Керівник бакалаврської кваліфікаційної роботи

_____ _____ _____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Завдання прийняв до виконання _____
(підпис) (ПІБ студента)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 5

ВСТУП 6

ОСНОВНА ЧАСТИНА 9

1. СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 9

1.2 Постановка завдання 9
1.3 Огляд інформаційних джерел та існуючих рішень 12
Обґрунтування необхідності власної розробки 14
1.3 Моделювання предметної області 14
1.4 Висновки до розділу 1 21

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ 22

2.1 Логічна модель даних 22
2.2 Вибір системи управління інформаційною базою 28
2. 3 Створення інформаційної бази 29
2.4 Висновки до розділу 2 36

3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ 37

3.1 Організаційна Структура програмного забезпечення 37
3.2 Вибір інструментарію для створення прикладногопрограмного забезпечення 37
3.3 Алгоритмізація та програмування програмних модулів 38
3.4 Висновки до розділу 3 42

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ 44

4.1 Тестування системи 44
4.2 Вимоги до апаратного та програмного забезпечення 53
4.3 Склад інсталяційного пакету 53
4.4 Подальші перспективи розвитку программного продукту 57
4.5 Висновки до розділу 4 57

ВИСНОВКИ 58

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 59

ДОДАТКИ 61

ДОДАТОК А 61

ДОДАТОК Б 64

ДОДАТОК В 69

ДОДАТОК Д 71

ДОДАТОК Е 75

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення

БД – база даних

SQL – Structured Query Language

MS – Microsoft

СУБД – система керування базою даних

ВСТУП

Мета роботи — спростити та автоматизувати роботу працівника реєстратури поліклініки, використовуючи знання SQL та навички створення баз даних.

База даних – сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування [1]. В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти. Дані у базі організовують відповідно до моделі організації даних.

Бази даних є невід'ємною складовою будь-якої інформаційної системи. Вони дають змогу управляти великою кількістю даних, забезпечити швидкий доступ до них, а також зберігати їх у структурованому вигляді. Саме така система необхідна в реєстратурі поліклініки, де щодня обробляються десятки, а іноді й сотні запитів від пацієнтів.

У ході даної дипломної роботи було створено інформаційну систему, яка автоматизує роботу працівника реєстратури. Вона значно пришвидшує внесення, перегляд, редагування й пошук даних, з якими щодня має справу цей фахівець.

Актуальність теми пояснюється стрімким розвитком цифрових технологій. Зараз важко уявити ефективну роботу без комп'ютерної підтримки. Особливо це стосується медичної сфери, де важлива не тільки швидкість обробки інформації, але й її точність. Кожне робоче місце має свої особливості, тому універсальні рішення не завжди підходять. Натомість спеціалізоване програмне забезпечення здатне зробити роботу комфортнішою та більш продуктивною.

У цьому проєкті база даних була реалізована в середовищі Microsoft SQL Server

з використанням Management Studio 2022**.** Інтерфейс для роботи з даними створено у Microsoft Visual Studio 2022**.**

Проблематика та обґрунтування вибору теми:

Реєстратура — це місце, де щодня проходить величезна кількість пацієнтів. Тут постійно потрібно записувати людей на прийом, знаходити їхні медичні картки, оновлювати дані тощо. Якщо все це робити вручну — виникають помилки, дублікати, втрата інформації або просто довгі черги. А це — зайвий стрес як для працівників, так і для пацієнтів.

Уявіть просту ситуацію: пацієнт стоїть у черзі, а реєстратор шукає його дані в купі паперів. Це не тільки займає багато часу, а й може викликати непорозуміння — наприклад, якщо пацієнта випадково записали до не того лікаря чи на неправильний час. Усе це можна вирішити за допомогою автоматизації.

Сучасна програма дозволяє в лічені секунди знайти потрібну інформацію, уникнути помилок і спростити роботу. Інформаційна система — це не просто зручніше, це вже давно необхідність.

Архітектура системи та використовувані технології:

Для розробки системи були обрані сучасні й надійні інструменти від компанії Microsoft, які добре підходять для побудови подібного роду програм.

Основою системи стала база даних у Microsoft SQL Server. Це одна з найпотужніших СУБД, яка забезпечує збереження великих обсягів інформації, безпеку даних, швидкий доступ та гнучкість у налаштуванні.

Для розробки інтерфейсу користувача я використовувала середовище Microsoft Visual Studio 2022. Завдяки цьому програма вийшла зрозуміла та з чітким поділом функцій.

Мовою яку я обрала для розроби програми стала мова C#.

Завдяки такому підходу мені вдалось створити працездатну, налаштовувану та легко оновлювану систему, яка добре підходить для використання у відділенні реєстратури поліклініки.

Основний зміст дипломної роботи викладено на шістдесяти сторінках, доповнено сорока трьома ілюстраціями та трьома таблицями. Список використаної літератури містить сім джерел. Загальний обсяг роботи становить 60 сторінок.

ОСНОВНА ЧАСТИНА

1. СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.2 Постановка завдання

Необхідно розробити програмний продукт для автоматизації процесу обробки інформації для працівника реєстратури поліклініки для роботи з пацієнтами.

Основна мета створення цього продукту — зробити роботу реєстратора поліклініки простішою, зручнішою та менш затратною за часом.

Отже у процесі розробки ставились такі завдання:

Оптимізувати повсякденну роботу реєстратури - система має полегшити виконання рутинних дій, наприклад, пошук даних пацієнта, запис на прийом або редагування інформації. Це дозволить співробітнику зосередитись на спілкуванні з людьми, а не витрачати час на паперову бюрократію.

Забезпечити ефективну роботу з даними. Уся інформація про пацієнтів, лікарів, талони та медичні записи повинна бути впорядкована, зручно структурована та легко доступна. Це відкриває можливості для генерації звітів, аналізу навантаження лікарів і планування роботи установи загалом.

Контроль та облік медичних документів - програма допоможе точно відслідковувати видані талони, уникати дублювання записів і контролювати наявність вільного часу в лікарів.

Також система суттєво спрощує роботу з медичними картами — перегляд, внесення нових записів та зберігання історії лікування.

Програма має підтримувати стандартні дії для роботи з усіма сутностями:

Додавання, редагування та видалення даних. Це стосується всього — пацієнтів, лікарів, діагнозів, талонів, карток тощо.

Пошук записів у медичній карті пацієнта - можливість швидко знайти всю історію лікування за прізвищем або іншим параметром.

Пошук талонів за місяць. Це зручно, коли треба переглянути графік прийому конкретного лікаря або всі записи за період.

Цілі використання даного програмного продукту:

- Оптимізація роботи працівника реєстратури поліклініки
- Обробка даних для їх подальшого використання
- Контроль талонів та мед-карт

Необхідні сутності та їх властивості:

- Спеціалізація: код спеціалізації, назва спеціалізації
- Відділ: код відділу, назва відділу
- Кабінет: код кабінету, назва кабінету
- Лікар: код лікаря, телефон, прізвище, ім’я, по батькові, код відділу, код кабінету, код спеціалізації
- Талон: код талона, час, дата, код лікаря, код пацієнта
- Стать: код статі, назва статі
- Пацієнт: код пацієнта, телефон, прізвище, ім’я, по батькові, дата народження, адреса, стать
- Робочий графік лікаря: робочі дні, робочі часи, код лікаря
- Мед-карта: код мед-карти, дата, код діагнозу, код пацієнта, код лікаря

- Діагноз: код діагнозу, назва діагнозу, опис діагнозу

Інформація для програмного додатку:

- Вхідна інформація:
  a) Пацієнт
  b) Талон
  c) Мед-карта
- Базова інформація:
  a) Спеціалізація
  b) Відділ
  c) Кабінет
  d) Лікар
  e) Стать
  f) Робочий графік лікаря
  g) Діагноз
- Вихідна інформація:
  a) Список талонів на прийом за певний місяць
  b) Перелік записів мед-карти конкретного пацієнта
  c) Завантаженість лікарів

2. Операції з даними

- Додавання, редагування, видалення інформації
- Відбір записів мед-карти конкретного пацієнта
- Відбір талонів на прийом за певний місяць

1.3 Огляд інформаційних джерел та існуючих рішень

Для аналізу теми інформаційними джерелами слугували:

- наукові публікації з інформаційних систем охорони здоров’я;
- документація до систем Medoc, OpenMRS, GNU Health;
- технічні характеристики СУБД;
- державні нормативи щодо зберігання медичних даних;
- практичний досвід медичних працівників.

Існуючі програмні продукти:

На ринку присутні рішення, які частково або повністю реалізують функції реєстратури:

- Medoc, Doctor Eleks, MedWork — комерційні рішення, що мають широкий функціонал, але вимагають складної інсталяції, технічного обслуговування та оплати ліцензії.
- OpenMRS, GNU Health — безкоштовні open-source рішення, однак є складними для впровадження в умовах невеликих закладів і мають надлишковий функціонал.

Порівняльний аналіз існуючих систем представлений в таблиці 1:

Таблиця 1

Назва системи	**Тип**	**Ліцензія**	**Переваги**	**Недоліки**
Doctor Eleks	Комерційна	Платна	Широкий функціонал, інтеграція з eHealth	Висока ціна, складне впровадження
Medoc	Комерційна	Платна	Підтримка звітності, реєстрів	Невигідна для малих закладів
OpenMRS	Open Source	Безкоштовна	Гнучкість, спільнота розробників	Потрібні технічні знання для налаштування
GNU Health	Open Source	Безкоштовна	Розширений функціонал, підтримка SNOMED	Складна структура, потреба в доопрацюванні
Власна система	Індивідуальна	Безкоштовна	Простота, локальність, адаптація під потреби	Менший масштаб функцій у порівнянні з готовими

Переваги готових рішень:

- масштабованість;
- перевіреність на практиці;
- наявність підтримки.

Недоліки:

- складність налаштування;
- потреба у високих технічних ресурсах;
- складність локалізації;

- відсутність гнучкого налаштування під конкретну структуру поліклініки.

Обґрунтування необхідності власної розробки

З огляду на специфіку задачі, потребу в точному урахуванні структури конкретного закладу та обмеження у фінансах, вирішено реалізувати індивідуальну розробку, яка буде:

- відповідати всім функціональним вимогам користувача;
- простою у впровадженні та використанні;
- розгорнута локально, без потреби в зовнішньому сервері або інтернет-доступі;
- підтримувати лише ті функції, які дійсно потрібні — без надмірної складності.

У рамках роботи обрано інструменти, що забезпечують зручність розробки та стабільність роботи:

- Microsoft SQL Server для реалізації бази даних;
- C# (.NET Framework) + Windows Forms у середовищі Visual Studio — для побудови зручного клієнтського інтерфейсу.

Отже я думаю, що для поліклініки з обмеженим бюджетом та чіткими функціональними вимогами доцільним є створення адаптованого ПЗ.

1.3 Моделювання предметної області

1.3.1 Загальні відомості

Предметна область - сукупність пов'язаних між собою функцій, завдань управління, за допомогою яких досягається виконання поставлених цілей, це частина реального світу, що представляє інтерес для конкретного дослідження [2].

В моєму випадку досліджуваною предметною областю є робота поліклініки.

Після постановки завдання та збору інформації можна приступати до моделювання предметної області.

Цей етап є ключовим для процесу розробки програмного продукту, бо якщо почати розробку без нього, то це буде схоже на збірку кубика Рубіка зав’язаними очима. Це звичайно можливо, але займе набагато більше часу та спроб.

В межах даного етапу визначаються основні сутності, їхні характеристики, атрибути, методи та взаємозв’язки між ними, це дозволить створити цілісну і логічну модель системи.

Моделювання предметної області допомагає програмістам та іншим учасникам процесу розробки зрозуміти вимоги замовника або майбутнього користувача системи на глибшому рівні та передбачити всі сценарії її використання. Завдяки правильному проведенні етапу моделювання можна запобігти утворенню помилок на подальших етапах розробки, оскільки чим пізніше виявити помилку тим більше вона коштуватиме.

Після того як сформульовані функціональні та нефункціональні вимоги до системи, наступним етапом стає моделювання предметної області. І хоч здається, що головне вже зроблено — насправді цей етап не менш важливий. Саме тут потрібно розібратись, з чого взагалі складається система, які є сутності, чим вони відрізняються, як пов’язані між собою, і як усе це має працювати. Якщо коротко — створюється така собі модель, яка допомагає зрозуміти, що і для чого взагалі ми будуємо.

Чесно кажучи, це все моделювання — непроста штука. Але воно реально допомагає. Завдяки йому і розробники, і аналітики, і навіть замовники можуть на одному рівні зрозуміти, що має робити система, як саме її використовуватимуть, і що взагалі треба реалізувати. Якщо добре все змоделювати на початку — потім менше мороки під час реалізації, бо багато помилок і непорозумінь можна виявити ще на етапі обговорень.

Моделювання дозволяє:

- чітко сформувати структуру даних і спланувати архітектуру, яка нормально працюватиме з інформацією;
- описати бізнес-процеси так, щоб потім не довелось переробляти половину системи через якесь непорозуміння;
- краще налагодити комунікацію в команді — всі бачать одне й те саме, говорять про одне й те саме, і це вже щось;
- і, найголовніше, — дозволяє думати наперед: що буде, якщо бізнес розшириться, або зміняться вимоги.

Щоб все це зобразити, найчастіше використовують UML — уніфіковану мову моделювання. Раніше було багато підходів до того, як створювати ці моделі, і це тільки ускладнювало життя. А зараз є UML, і хоча це не завжди просто, зате є певний стандарт. Він містить багато діаграм і умовних позначень, які допомагають схематично зобразити, як все має працювати. Ну і головне — це зрозуміло не тільки програмісту, а й решті команди.
У моделювання є свої принципи, яких бажано дотримуватись.

Наприклад:

- Абстрагування — не чіплятись за зайві деталі, а виділяти головне.
- Модульність — треба ділити все на частини. Якщо треба буде щось змінити — краще правити один модуль, ніж ламати все.
- Ієрархія — має бути якась структура, а не хаос.
- Формалізація — використовувати чіткі позначення й правила, щоб не було двозначностей.
- Уніфікація — усі використовують ті самі підходи.

У результаті отримуємо модель, яка не тільки виглядає логічно, а ще й допоможе мені через рік не згадувати, що я там мала на увазі.

UML, як вже сказала, — це не просто набір схем. У версії 1.5 є аж 12 типів діаграм, і вони поділяються на три великі категорії:

- діаграми структури — показують, з чого складається система;
- діаграми поведінки — пояснюють, як система поводиться в різних ситуаціях;
- діаграми реалізації — про те як все це працює на рівні заліза.

Найчастіше обмежуються кількома основними:

- Діаграма прецедентів — що може робити користувач;
- Діаграма послідовності — як об’єкти взаємодіють між собою в часі;
- Діаграма діяльності — як виконується якийсь бізнес-процес.

У моєму випадку, для інформаційної системи роботи відділення реєстратури в поліклініці, я якраз і буду робити ці діаграми.

Для створення діаграм я використовувала програму Draw.io

1.3.2 Діаграма прецедентів

Нижче на рис. 1 зображена діаграма прецедентів предметної області:

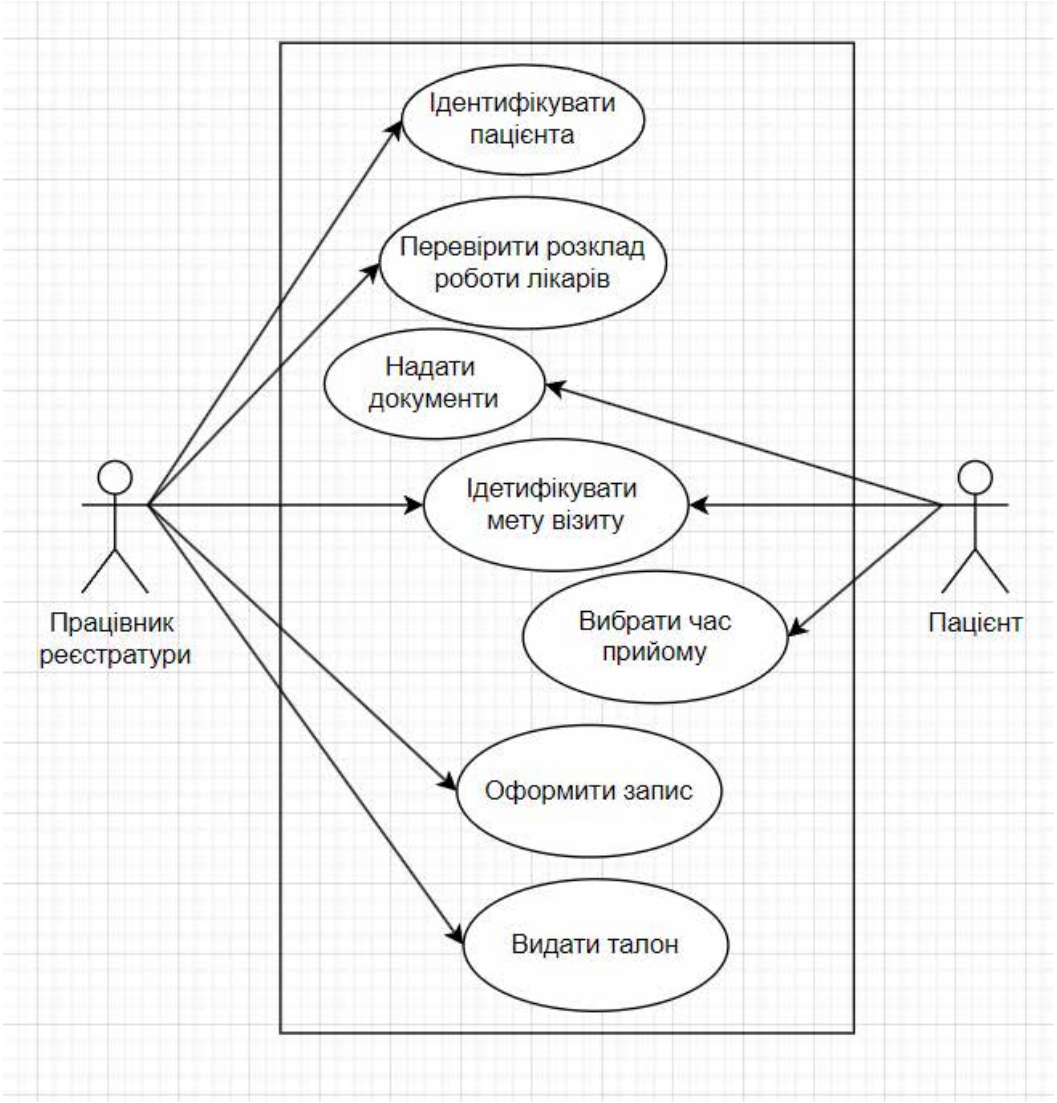

Рис. 1 Діаграма прецедентів предметної області

1.3.3 Діаграма активності

Створену діаграму активності можна подивитись нижче на рис.2

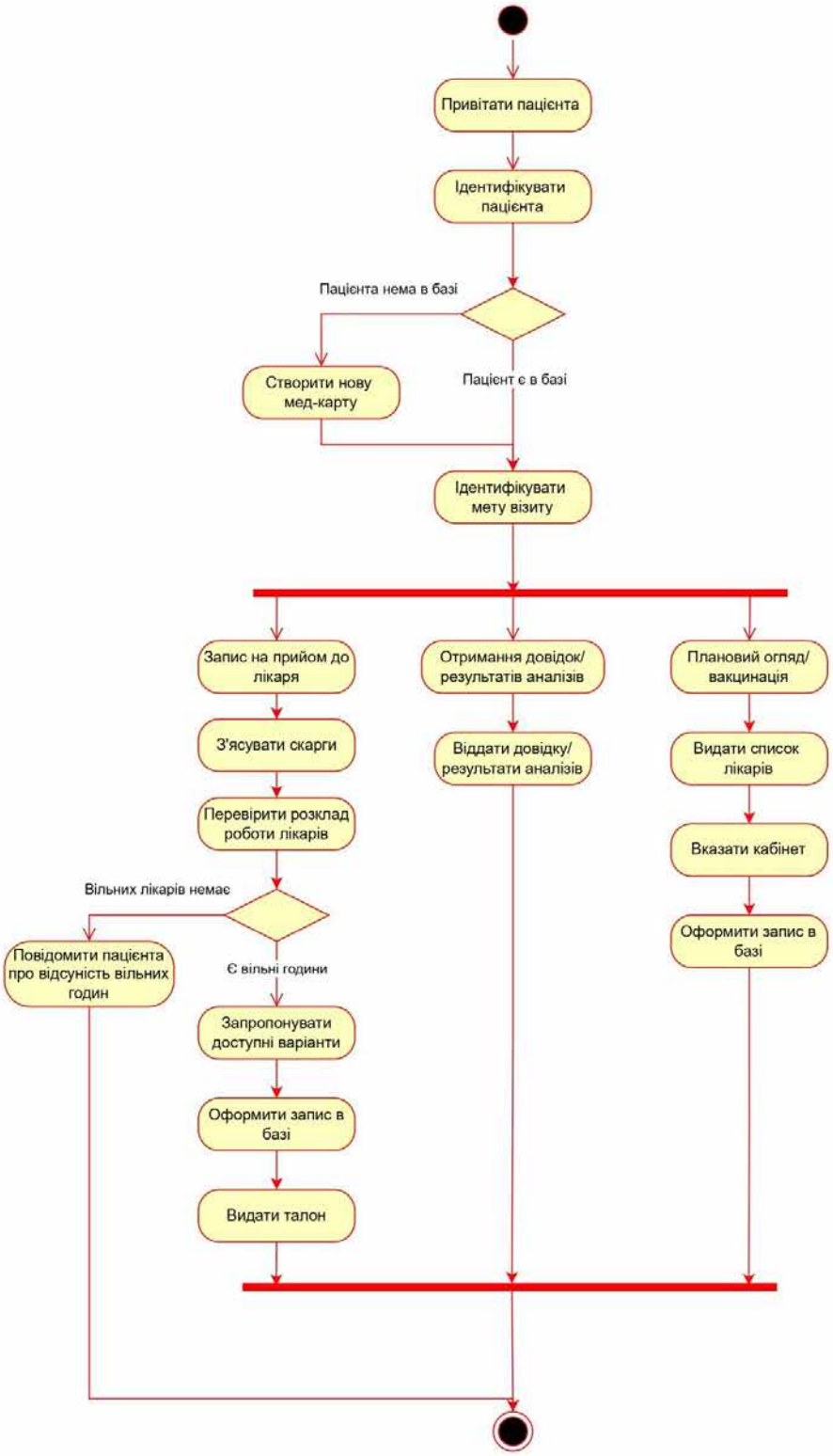

Рис. 2 Діаграма активності

1.3.4 Діаграма послідовності

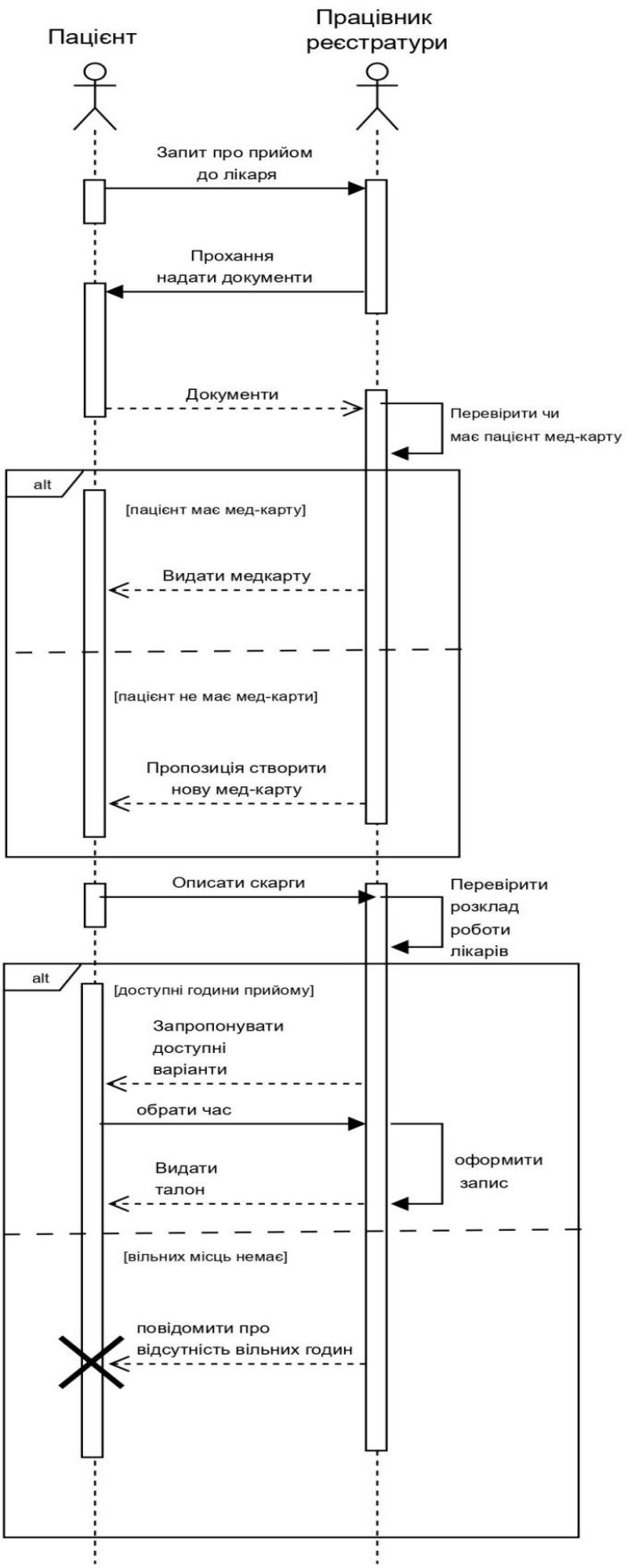

Рис.3 Діаграма послідовості (предметної області)

1.4 Висновки до розділу 1

У цьому розділі проведено глибокий аналіз предметної області діяльності працівника реєстратури поліклініки. Результатом стало виявлення ключових сутностей, з якими працює персонал — пацієнти, лікарі, медичні картки, талони на прийом, графіки роботи. Проаналізовано зв’язки між об'єктами: наприклад, кожен лікар має кілька талонів, але кожен талон належить лише одному пацієнту.

На основі виявлених зв’язків сформовано вимоги до майбутньої інформаційної системи. Визначено набір функцій, які має реалізовувати програмне забезпечення: облік пацієнтів, планування запису, формування звітів, контроль доступу до медичної інформації. Також обґрунтовано необхідність автоматизації в умовах підвищеного навантаження та великого обсягу ручної роботи.

Загалом, системний аналіз дав змогу визначити логічні межі майбутньої системи, формалізувати її призначення, а також закласти основу для ефективного проектування, що розглядатиметься в наступних розділах.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних

Для побудови діаграми використовувалися такий програмний продукти як ERwin Data Modeler. На рис.4 зображено логічну схему бази даних «Інформаціїна система роботи відділення реєстратури в поліклініці».

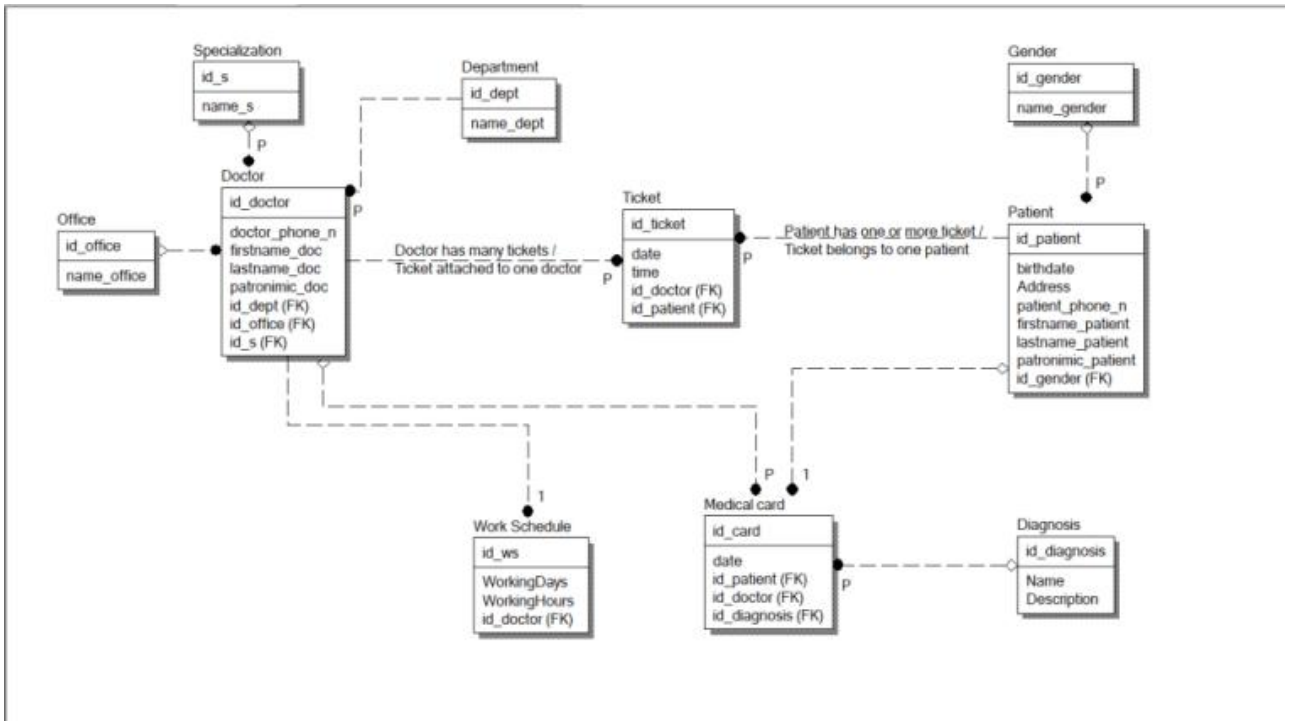

Рис. 4 Логічна модель бази даних

Структура інформації представлена наступним чином:

1. Умовно-постійна інформація:

- Спеціалізація
- Відділ
- Кабінет
- Лікар
- Стать
- Робочий графік лікаря
- Діагноз

2. Робоча інформація:

- Пацієнт
- Талон
- Мед-карта

Кожна сутність має свої унікальні поля, що напряму належать ключовим полям. Всі дані БД мають бути нормалізовані. Розглянемо дане поняття та кожну сутність.

Нормальна форма — властивість відношення в реляційної моделі даних, що характеризує його з точки зору надмірності, яка потенційно може призвести до логічних помилкових результатів вибірки або зміни даних. Нормальна форма визначається як сукупність вимог, яким має задовольняти відношення [3].

Правило Кодда №1 - інформація повинна бути представлена у вигляді даних, що зберігаються у комірках. Дані, що зберігаються у комірках, повинні бути атомарними. Порядокрядків в реляційній таблиці не повинен впливати на зміст даних. Таким чином, вся інформація в базі даних повинна бути представлена виключно на логічному рівні і лише одним способом - у вигляді значень, що містяться в таблицях. Фактично це неформальне визначення реляційної бази даних [4].

Перша нормальна форма(1NF).

Відношення знаходиться в 1NF тоді і тільки тоді, коли всі його атрибути є атомарними. Значення атрибуту вважається атомарним, якщо воно є неподільним у всіх застосуваннях [5].

Друга нормальна форма(2NF).

Відношення знаходиться в 2NF, якщо воно знаходиться в 1NF і кожен його не первинний атрибут функціонально повно залежить від первинного ключа [5].

Третя нормальна форма (3NF).

Відношення R знаходиться в третій нормальній формі (3NF) у тому і тільки у

тому випадку, якщо знаходиться в 2NF і кожен не ключовий атрибут залежить від первинного ключа [5].

Для створеної схеми зображеної на рис.1 виконуються умови першої нормальної форми, тобто:

У сутності Спеціалізація атрибути код спеціалізації(містить тільки один код), Дата_реєстрації (містить тільки одну дату конкретної реєстрації), назва спеціалізації (містить тільки одну назву спеціалізації) є атомарними.

- У сутності Відділ атрибути Код відділу (унікальний ідентифікатор відділу), Назва відділу є атомарними.
- У сутності Кабінет атрибути, Код кабінету (унікальний ідентифікатор кабінету), Назва кабінету є атомарними.
- У сутності Лікар атрибути Код лікаря (унікальний ідентифікатор лікаря), Телефон (контактний номер телефону лікаря), Прізвище (прізвище лікаря), Ім'я (ім'я лікаря), По батькові (по батькові лікаря), Код відділу (посилання на відділ, до якого належить лікар) - Код кабінету (посилання на кабінет, в якому працює лікар) - Код спеціалізації (посилання на спеціалізацію, до якої належить лікар) є атомарними.
- У сутності Талон атрибути Код талона (унікальний ідентифікатор талона), Час (час запису на прийом), Дата (дата запису на прийом), Код лікаря (посилання на лікаря, до якого стосується талон), Код пацієнта (посилання на пацієнта, якому належить талон) є атомарними
- У сутності Стать атрибути Код статі (унікальний ідентифікатор статі), Назва статі є атомарними
- У сутності Пацієнт атрибути Код пацієнта (унікальний ідентифікатор пацієнта), Телефон (контактний номер телефону пацієнта), Прізвище (прізвище пацієнта), Ім'я (ім'я пацієнта), По батькові (по батькові пацієнта), Дата народження (дата народження пацієнта), Адреса (адреса пацієнта), Код статі (посилання на стать пацієнта) є атомарними

- У сутності Робочий графік лікаря атрибути Робочі дні (дні, коли лікар приймає), Робочі часи (часи, коли лікар приймає), Код лікаря (посилання на лікаря, до якого належить графік) є атомарними
- У сутності Мед-карта атрибути Код мед-карти (унікальний ідентифікатор медичної карти) - Дата (дата створення медичної карти), Код діагнозу (посилання на діагноз, встановлений для пацієнта), Код пацієнта (посилання на пацієнта, якому належить медична карта), Код лікаря (посилання на лікаря, який створив медичну карту) є атомарними
- У сутності Діагноз атрибути Код діагнозу (унікальний ідентифікатор діагнозу) - Назва діагнозу (назва конкретного діагнозу) - Опис діагнозу (детальний опис діагнозу) є атомарними

Для даної схеми виконуються умови другої нормальної форми так як у кожній сутності кожен неключовий атрибут повністю залежить від первинного ключа.

Також виконуються умови третьої нормальної форми, так як відсутнізв’язки багато до багатьох.

Отже тепер визначено, що логічна модель відповідає вимогам реляційного підходу та приведена до третьої нормальної форми(3NF).

Фізична модель теж була побудована в ERwin Data Modeler.

Для цього я відкрила уже створену раніше логічну модель в в середовищі ERWin, перейшла на фізичний рівень моделювання та за допомогою інструмента Forward Engineer(рис.5) згенерувала фізичну модель бази даних.

Згенерована фізична модель представлена на рис. 7.

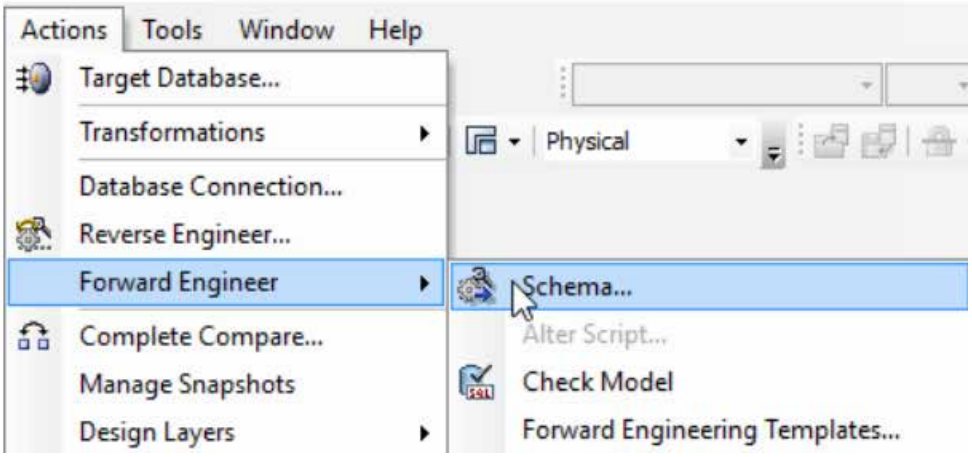

Рис. 5 Forward Engineer tool

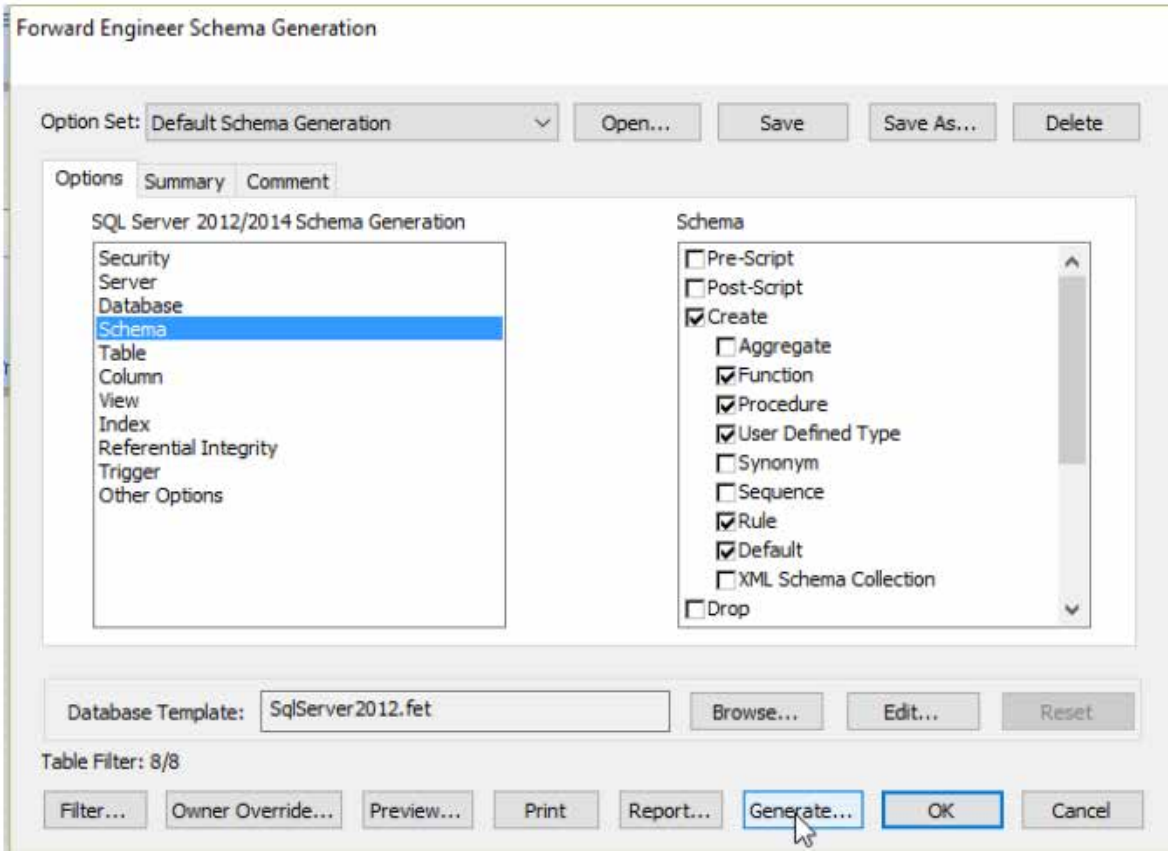

Рис. 6 Генерація моделі

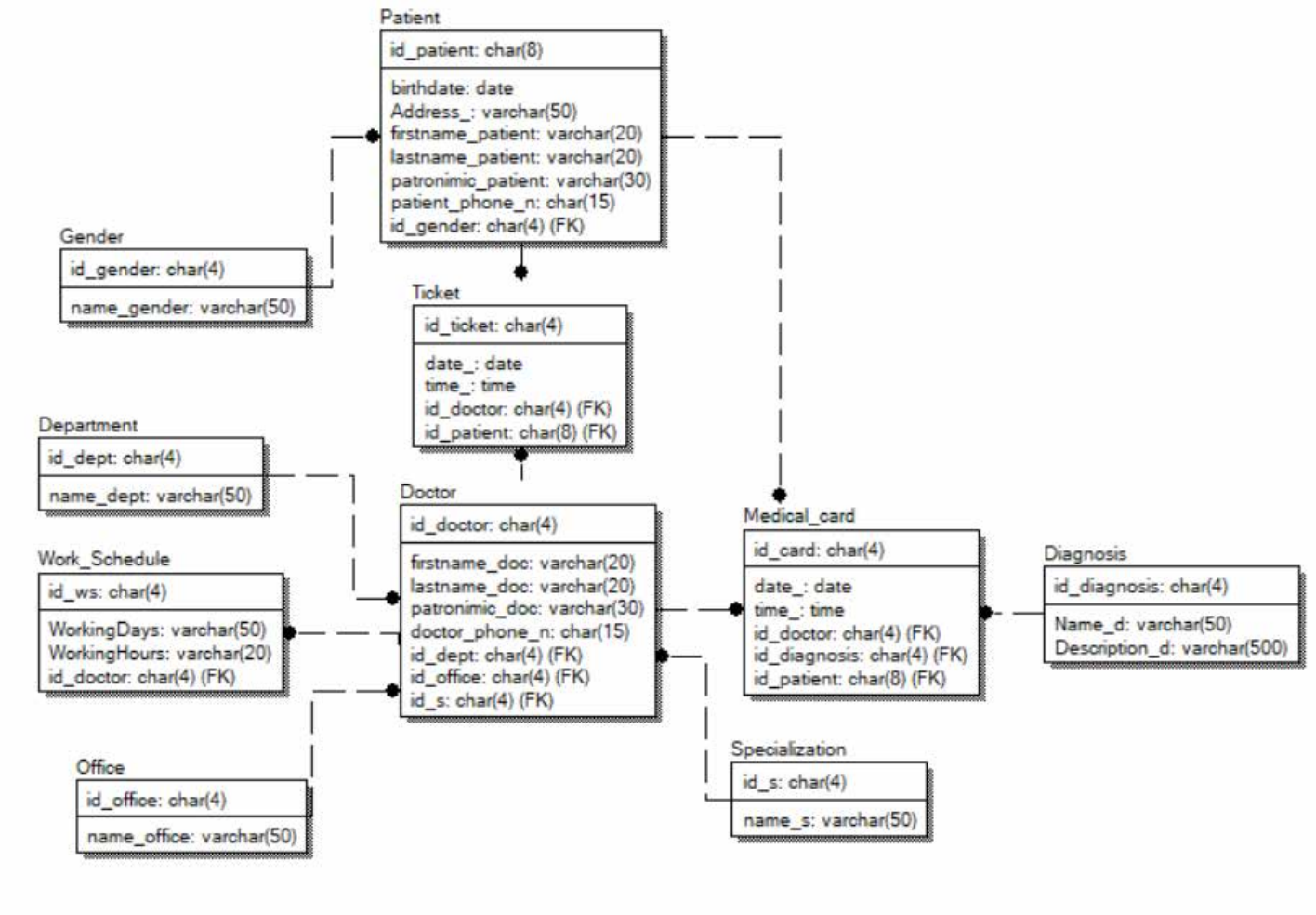

Рис. 7 Фізична модель бази даних

2.2 Вибір системи управління інформаційною базою

Порівняння СУБД:

Перед вибором СУБД було проведено порівняльний аналіз. Він представлений в таб. 2:

Таблиця 2

СУБД	**Продуктив-ність**	**Підтримка збережених процедур**	**Інтеграція з .NET**	**Вартість**	**Підтримка транзакцій**
MS SQL Server	Висока	Є	Ідеальна	Умовно безкоштовна (Express)	Повна
MySQL	Середня	Часткова	Через додаткові бібліотеки	Безкоштовна	Часткова
PostgreSQL	Висока	Є	Добра	Безкоштовна	Повна
SQLite	Низька	Обмежена	Добра	Безкоштовна	Часткова
Oracle	Висока	Є	Часткова	Платна	Повна

Мною було обрано MS SQL Server, оскільки він:

- має зручний GUI (SSMS);
- підтримує транзакції, ролі, процедури;
- інтегрується з .NET без додаткових драйверів;
- Express-версія є безкоштовною.

2.2.1 Обґрунтування типу бази даних

Існують три основні типи моделей баз даних:

- Реляційні БД — таблиці з чіткими зв’язками (SQL, MySQL, PostgreSQL).
- Об’єктно-реляційні БД — доповнені об’єктами та методами (Oracle, PostgreSQL).
- Документо-орієнтовані БД — сховища JSON/NoSQL-документів (MongoDB, CouchDB).

Вибрано реляційну модель, оскільки:

- структура системи чітко описується таблицями (пацієнти, талони тощо);
- зручно формалізувати зв’язки за допомогою зовнішніх ключів;
- потрібна сувора валідація типів та цілісності даних;
- стандартна SQL-мова дозволяє легко формувати звіти.

2. 3 Створення інформаційної бази

2.3.1 Створення бази даних

В якості інструменту для створення БД було обрано СКБД MS SQL Server.

Microsoft SQL Server – система керування базами даних, розроблена корпорацією Microsoft. Основна мова запитів – Transact-SQL, створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI/ISO структурованої мови запитів SQL з розширеннями [6].

Команда створення бази даних зображена на рис.8

```
create database PolyclinicDB
```

Рис. 8 Створення бази даних PolyclinicDB

2.3.2 Створення таблиць та зв’язків

Усі об’єкти бази даних були створені за допомогою SQL-запитів. Для створення таблиць необхідно використати команду CREATE TABLE вказати_назву_таблиці (перелік атрибутів та їх типів). Перед цим варто вказати

назву бази даних, для якої створюється таблиця.

На прикладі таблиць «Відділ» та «Лікар» розглянемо процес створення за допомогою команд SQL, які показані на рис.9 та рис.10. Команди для створення інших таблиць представлені у ДОДАТОК А.

```
use PolyclinicDB
go
IF EXISTS (SELECT name FROM sys.objects
 WHERE name = 'Department' AND type_desc = 'USER_TABLE')
drop table Department
go
create table Department
(id_dept char(4) NOT NULL primary key, name_dept varchar(50))
```

Рис.9 Створення таблиці «Відділ»

```
IF EXISTS (SELECT name FROM sys.objects
 WHERE name = 'Doctor' AND type_desc = 'USER_TABLE')
drop table Doctor
go
create table Doctor
(id_doctor char(4) NOT NULL primary key,
firstname_doc varchar(20),
lastname_doc varchar(20),
patronimic_doc varchar(30),
doctor_phone_n char(15),
id_dept char(4) NOT NULL,
id_office char(4) NOT NULL,
id_s char(4) NOT NULL,
CONSTRAINT id_dept_FK FOREIGN KEY (id_dept)
REFERENCES Department (id_dept),
CONSTRAINT id_office_FK FOREIGN KEY (id_office)
REFERENCES Office (id_office),
CONSTRAINT id_s_FK FOREIGN KEY (id_s)
REFERENCES Specialization (id_s)
)
```

Рис.10 Створення таблиці «Лікар»

Як результат роботи запитів на рис.11 наведена діаграма, яка показує всі атрибути таблиць та їхні ключові поля. Всі зв’язки між таблицями були реалізовані за допомогою команди:

```
FOREIGN KEY Назва_поля_яке_стає_ключовим REFERENCES Назва_таблиці(Назва_поля_на_яке_посилаються).
```

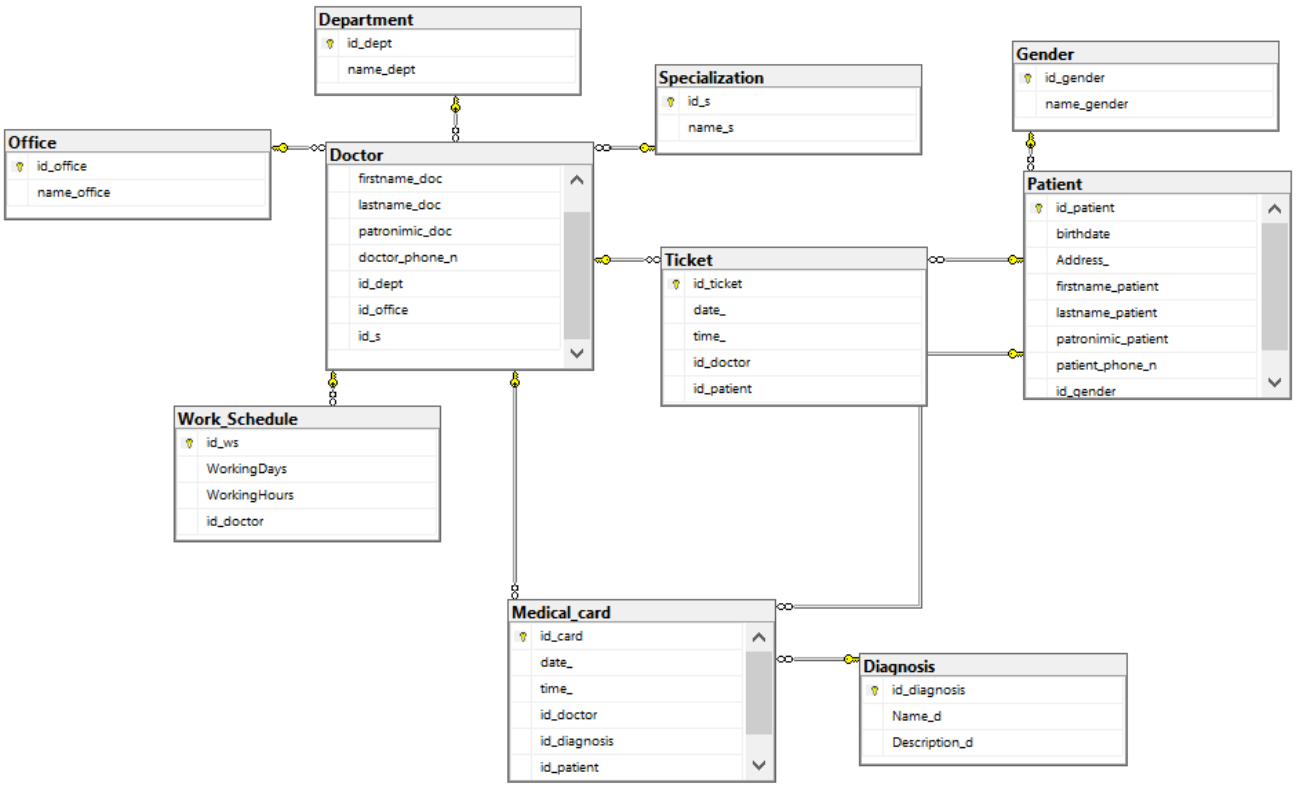

Рис.11 Діаграма бази даних, яка відображає таблиці та зв’язки між ними

2.3.3 Створення уявлень

Уявлення – це віртуальна таблиця. На відміну від таблиць в осередку бази даних уявлення зберігає не дані, а запит для відбору даних з однієї або декількох таблиць або уявлень.

На рис.12 показано створення уявлення, яке містить інформацію проробочий графік лікарів, яка буде корисною для формування звітності.

```
use PolyclinicDB
go
IF EXISTS (SELECT name FROM sys.all_views
WHERE name = 'Workload_of_the_doctors')
drop view Workload_of_the_doctors
go
create view Workload_of_the_doctors AS
Select firstname_doc, lastname_doc, patronimic_doc,
id_ticket, date_, time_, firstname_patient,
lastname_patient, patronimic_patient
From Doctor, Ticket, Patient
Where Doctor.id_doctor = Ticket.id_doctor And Patient.id_patient = Ticket.id_patient
```

Рис.12 Створення уявлення

SQL-запити для створення інших уявлень наведені у ДОДАТОК Б.

2.3.4 Створення процедур та тригерів

Збережена процедура MS SQL Server – це набір операторів мови T-SQL. Збережена процедура зберігається у базі даних і є об’єктом цієї бази, так само, як таблиці, уявлення, користувачі. Вона виконується на сервері, що значно пришвидшує роботу.

На рис.13 наведено приклад створення процедури, яка додає записи до таблиці Patient

```
CREATE PROCEDURE AddPatient
    @id_patient CHAR(8),
    @birthdate DATE,
    @Address_ VARCHAR(50),
    @firstname_patient VARCHAR(20),
    @lastname_patient VARCHAR(20),
    @patronimic_patient VARCHAR(30),
    @patient_phone_n CHAR(15),
    @id_gender CHAR(4)
AS
BEGIN
    INSERT INTO Patient (id_patient, birthdate, Address_, firstname_patient, lastname_patient,
    patronimic_patient, patient_phone_n, id_gender)
    VALUES (@id_patient, @birthdate, @Address_, @firstname_patient, @lastname_patient,
    @patronimic_patient, @patient_phone_n, @id_gender)
END
```

Рис.13 Створення збереженої процедури AddPatient

SQL-запити для створення інших процедур наведені у ДОДАТОК В.

Тригери – це спеціальний тип збережених процедур, які запускаються автоматично при виконанні певних дій з таблицями бази даних [7]. Це може бути при виконанні команд INSERT, UPDATE, DELETE. Кожний тригер прив’язується до конкретної таблиці.

На рис.14 наведений приклад створення тригера для запобігання дублюванню пацієнтів.

```
USE [PolyclinicDB]
GO
/****** Object:  Trigger [dbo].[PreventDuplicateNames]    Script Date: 01.06.2023 01:24:44 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [dbo].[PreventDuplicateNames]
ON [dbo].[Patient]
AFTER INSERT, UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    -- Check if there are any duplicate records with the same firstname_patient, lastname_patient, and patronimic_patient
    IF EXISTS (
        SELECT 1
        FROM Patient p1
        INNER JOIN inserted i ON p1.firstname_patient = i.firstname_patient
            AND p1.lastname_patient = i.lastname_patient
            AND p1.patronimic_patient = i.patronimic_patient
        WHERE p1.id_patient <> i.id_patient -- Exclude the currently inserted/updated record
    )
    BEGIN
        -- If a duplicate record is found, rollback the transaction and display an error message
        RAISERROR('Такий пацієнт вже існує.', 16, 1);
        ROLLBACK TRANSACTION;
    END;
END;
```

Рис.14 Створення тригера

2.3.5 Створення користувачів та надання привілеїв

Для правильного керування БД створюють ролі, яким надають певні привілеї. MS SQL Server надає можливість створення ролей користувачів тарозподілення рівнів доступу. Для бази даних «АРМ працівника аптеки» було створено два користувачі:

- receptionist
- viewer

SQL-запит для створення ролей користувачів наведено на рис.15.

```
use PolyclinicDB
go
Drop user if exists viewer
Drop user if exists receptionist
create login viewer
with password = 'view58%'
create user viewer for login viewer
go
create login receptionist
with password = 'admin28!'
create user receptionist for login receptionist
```

Рис.15 Створення ролей користувачів

Для кожної ролі були надані певні привілеї, які визначали його можливості управління базою даних. Привілеї ролей вказані на рис.16:

```
use PolyclinicDB
go
grant select on Department to receptionist
grant select on Diagnosis to receptionist
grant select on Doctor to receptionist
grant select on Gender to receptionist
grant select, insert, delete, update on Medical_card to receptionist
grant select on office to receptionist
grant select , insert,  delete, update on Patient to receptionist
grant select on Specialization to receptionist
grant select, insert, delete, update on Ticket to receptionist
grant select on Work_Schedule to receptionist
GRANT EXECUTE ON dbo.AddPatient TO receptionist
GRANT EXECUTE ON dbo.UpdatePatient TO receptionist
GRANT EXECUTE ON dbo.GetMC TO receptionist
```

Рис.16 Надання прав ролі receptionist

Код для надання привілеїв усім ролям надано у ДОДАТОК Г.

2.4 Висновки до розділу 2

В результаті розробки інформаційного забезпечення було сформовано структуру бази даних, що повністю відображає логіку функціонування реєстратури. Створено логічну модель даних із застосуванням нормалізації до третьої нормальної форми, що дозволило уникнути надмірності та забезпечити цілісність збережених даних.

Моделювання дозволило сформувати чітке уявлення про структуру майбутньої системи. У процесі моделювання були визначені основні сутності, такі як відділи, лікарі, пацієнти, медичні картки, спеціалізації тощо, та встановлено логічні зв’язки між ними — все для того, щоб система працювала послідовно.

Під час проектування я особливо зосередилась на уніфікації даних і підтримці цілісності, щоб у майбутньому було простіше розширювати функціонал без кардинальних змін у структурі бази. Створені SQL-подання суттєво спрощують отримання звітної інформації, а збережені процедури й тригери реалізують частину бізнес-логіки прямо на рівні бази даних.

Загалом можна сказати, що інформаційна модель вийшла логічно цілісною, структурованою і технічно готовою до подальшої реалізації. Вона охоплює як внутрішню організацію даних, так і потенційну інтеграцію з іншими модулями системи.

3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Організаційна Структура програмного забезпечення

Програмне забезпечення складається з 3-ох частин:

- Інтерфейс користувача
- Встановлення зв’язку з базою даних
- Перегляд звітної інформації

Для зручності відображення та обробки інформації для користувача створюється інтерфейс. Для даного програмного продукту інтерфейс надає можливість працювати з базою даних за допомогою зручних елементів керування та навігації.

Зв’язок інтерфейсу з базою даних дає можливість використовувати всі можливості, які передбачені в програмному продукті та його меті. Саме тому, як зазначалось раніше, він відіграє найголовнішу роль.

Для зручного відображення інформації, яка потребує обробки, групування та подальшого використання користувачем використовуються звіти.

3.2 Вибір інструментарію для створення прикладного програмного забезпечення

Для створення програмного продукту «АРМ працівника аптеки» було використане таке середовище розробки як Microsoft Visual Studio та мову програмування C#.

Microsoft Visual Studio – серія продуктів фірми Майкрософт, які включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-

службияк в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.

3.3 Алгоритмізація та програмування програмних модулів

3.3.1. Реалізація інтерфейсу користувача

Для створення інтерфейсу користувача були використані такі компоненти:

- Label - елемент керування, який дає можливість створювати заголовки.написи на формі.
- Button - елемент керування, який виконує певні дії при натисканні (кнопка).
- TextBox – елемент для введення інформації. Також може використовуватись для відображення інформації. У проекті використовується в розділі «Додавання».
- ComboBox – елемент, який може зберігати список значень (дуже зручно використовувати, коли необхідно вибрати значення із завчасно відомих). Використовується для редагування даних таблиць.
- DataGridView – незамінний елемент для відображення таблиць.
- PictureBox – основний та найбільш зручний засіб відображення графічних зображень. Використовується в основному для оформлення інтерфейсу
- ReportViewer – елемент, який необхідний для відображення та зручної роботи зі звітами.
- DateTimePicker – є елементом керування, який використовується в програмуванні для вибору дати і часу з графічним інтерфейсом користувача.
- Panel – елемент для оформлення форм.

3.3.2. Забезпечення зв’язку інтерфейсу з базою даних

Реалізація зв’язку інтерфейсу з базою даних була здійснена за допомогою Wizard.exe та ConnectionString. Wizard.exe - це програма для формування ConnectionString.

Усі SQL-запити виконуються через SqlCommand, а результати обробляються у DataGridView.

Зчитування файла ConnectionString безпосередньо в програмі зображено на рис.17:

```
private string GetConnectionString()
{
    string fileName = "DBConnectionString";
    string filePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, fileName);

    if (!File.Exists(filePath))
    {
        return null;
    }

    try
    {
        return File.ReadAllText(filePath);

    }
    catch (Exception ex)
    {
        MessageBox.Show($"Помилка зчитування файлу підключення: {ex.Message}");
        return null;
    }
}
```

Рис.17 Зчитування файла ConnectionString

Алгоритм додавання нових записів у таблицю «Талон» на рис 18:

Рис. 18 Алгоритм додавання нових записів у таблицю «Талон»

Функції додавання, редагування та відображення інформації є основою функціоналу інтерфейсу користувача. Розглянемо усі функції на прикладі таблиці «Талон».

На рис.19, 20, 21 показані відповідно реалізації додавання, редагування та видалення записів у таблицю «Талон».

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    if (IdTick.Text == "" || cbidpat.Text == "" || date.Text == "" || time.Text == "" || cbIdDoc.Text == "" )
    {
        MessageBox.Show("Бракує даних про пацієнта");
    }
    else
    {
        try
        {
            SqlCommand cmd = new SqlCommand("insert into Ticket(id_ticket, date_, time_, id_doctor" +
                ", id_patient) values(@idT,@d,@t,@idD,@idP)", connection);
            cmd.Parameters.AddWithValue("@idT", IdTick.Text.Substring(0, 4));
            cmd.Parameters.AddWithValue("@d", date.Text);
            cmd.Parameters.AddWithValue("@t", time.Text);
            cmd.Parameters.AddWithValue("@idD", cbIdDoc.Text.Substring(0, 4));
            cmd.Parameters.AddWithValue("@idP", cbidpat.Text.Substring(0, 8));
            reset();
            cmd.ExecuteNonQuery();
            displaytik();
            MessageBox.Show("Талон збережений");
        }
        catch (Exception Ex)
        {

            MessageBox.Show(Ex.Message); ;
        }

    }
}
```

Рис.19 Реалізація додавання записів у таблицю «Талон».

```
private void button2_Click(object sender, EventArgs e)
{
    if (IdTick.Text == "" || cbidpat.Text == "" || date.Text == "" || time.Text == "" || cbIdDoc.Text == "")
    {
        MessageBox.Show("Бракує даних про пацієнта");
    }
    else
    {
        try
        {
            SqlCommand cmd = new SqlCommand("Update Ticket set date_=@d, time_=@t, id_doctor=@idD, id_patient=@idP " +
                "Where id_ticket=@idT", connection);
            cmd.Parameters.AddWithValue("@idT", IdTick.Text.Substring(0, 4));
            cmd.Parameters.AddWithValue("@d", date.Text);
            cmd.Parameters.AddWithValue("@t", time.Text);
            cmd.Parameters.AddWithValue("@idD", cbIdDoc.Text.Substring(0, 4));
            cmd.Parameters.AddWithValue("@idP", cbidpat.Text.Substring(0, 8));
            reset();
            cmd.ExecuteNonQuery();
            displaytik();
            MessageBox.Show("Талон оновлений");
        }
        catch (Exception Ex)
        {

            MessageBox.Show(Ex.Message); ;
        }

    }
}
```

Рис. 20 Редагування записів таблиці "Талон"

```
private void button3_Click(object sender, EventArgs e)
{
    SqlCommand cmd = new SqlCommand("Delete from Ticket Where id_ticket=@idT", connection);
    cmd.Parameters.AddWithValue("@idT", IdTick.Text.Substring(0, 4));
    cmd.ExecuteNonQuery();
    reset();
    displaytik();
    MessageBox.Show("Талон видалено");
}
```

Рис. 21 Видалення записів з таблиці «Талон»

3.4 Висновки до розділу 3

У цьому розділі реалізовано повнофункціональне прикладне програмне забезпечення, яке відповідає вимогам інформаційної системи роботи відділення в реєстратурі. Застосовано архітектуру 3-tier із поділом на інтерфейс користувача, бізнес-логіку та рівень доступу до даних. Такий підхід забезпечує масштабованість, зручність підтримки коду та можливість подальшої інтеграції з іншими сервісами.

Інтерфейс реалізовано у вигляді Windows Forms-додатка із зручною навігацією та підтримкою ролей користувачів. Забезпечено повний набір CRUD-функціоналу

для основних сутностей, авторизацію, фільтрацію записів, формування звітів через компонент ReportViewer. Використано сучасні підходи до безпеки — шифрування паролів, параметризовані запити, розмежування доступу.

Таким чином, програмне забезпечення є функціонально завершеним, стабільним, захищеним та придатним до практичного використання в умовах медичного закладу малого або середнього масштабу.

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Система пройшла повний цикл функціонального тестування, який охоплює сценарії описані в таб.3:

Таблиця 3

№	**Сценарій**	Результат
1	Авторизація користувача	Успішна, з перевіркою логіну/пароля
2	Додавання нового пацієнта	Дані зберігаються у БД Виведення повідомлення: «Пацієнт збережений»
3	Редагування пацієнта	Дані оновлюються коректно Виведення повідомлення: «Запис оновлено»
4	Запис на прийом через талон	Талон зберігається в таблиці Ticket.Виведення повідомлення: «Талон збережений»
5	Генерація звіту по медкарті	Інформація відображається згідно з фільтром
6	Вибір талонів за місяцем	Дані фільтруються по даті
7	Відмова в повторному записі пацієнта (тригер)	Попередження видається коректно
8	Перевірка прав доступу користувачів	*viewer* не може редагувати/видаляти. Виводиться повідомлення.

При запуску програми перед користувачем відкривається вікноавторизації (рис.22).

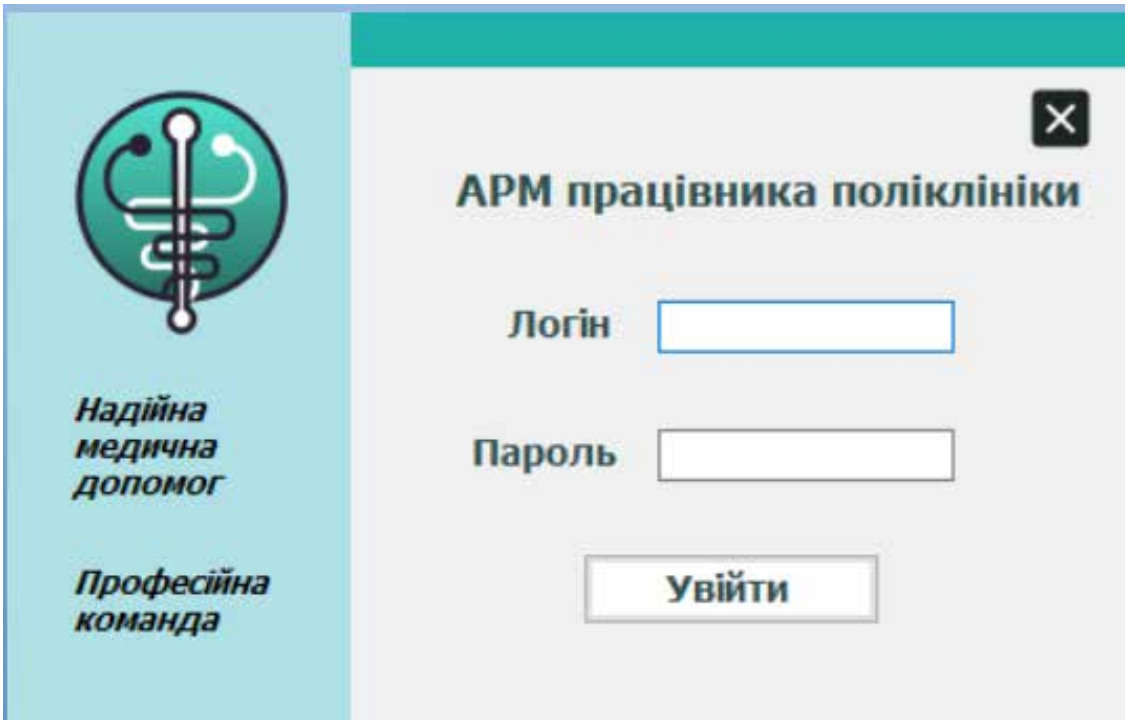

Рис. 22 Вікно авторизації

Після проходження авторизації користувач потрапляє на форму «Головне меню» (рис. 23)

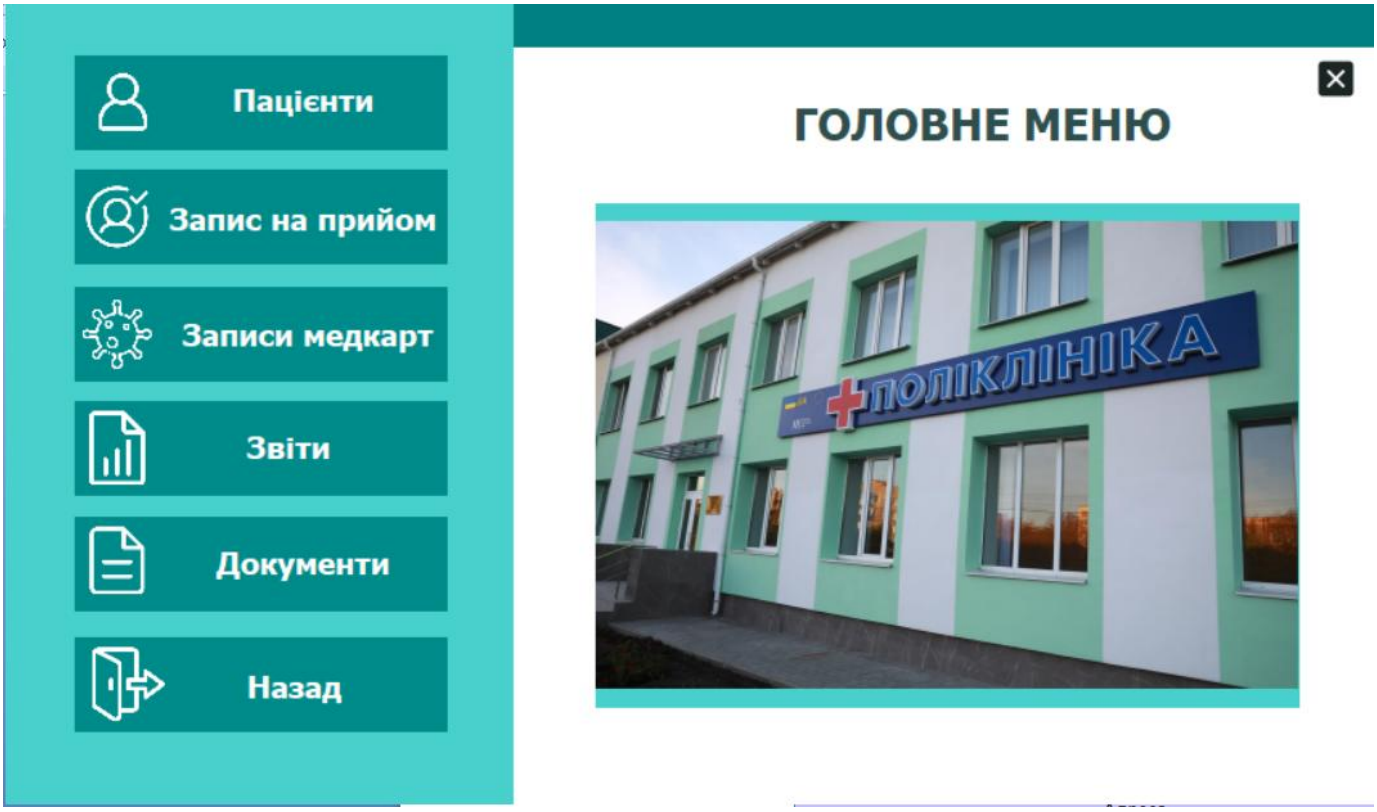

Рис. 23 Вікно «Головне меню»

Операція додавання записів у таблицю «Пацієнт» (рис.24-27):

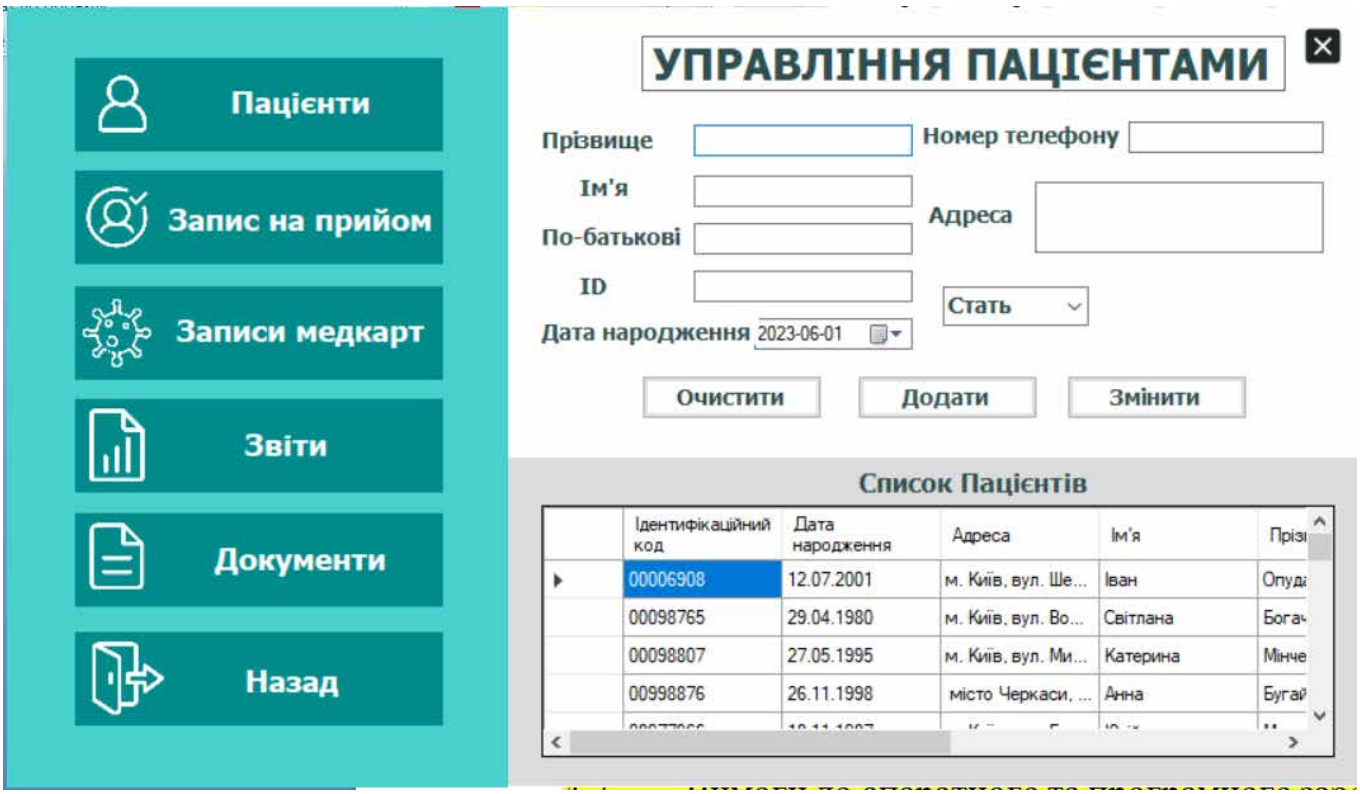

Рис.24 Перегляд та додавання записів таблиці «Пацієнт»

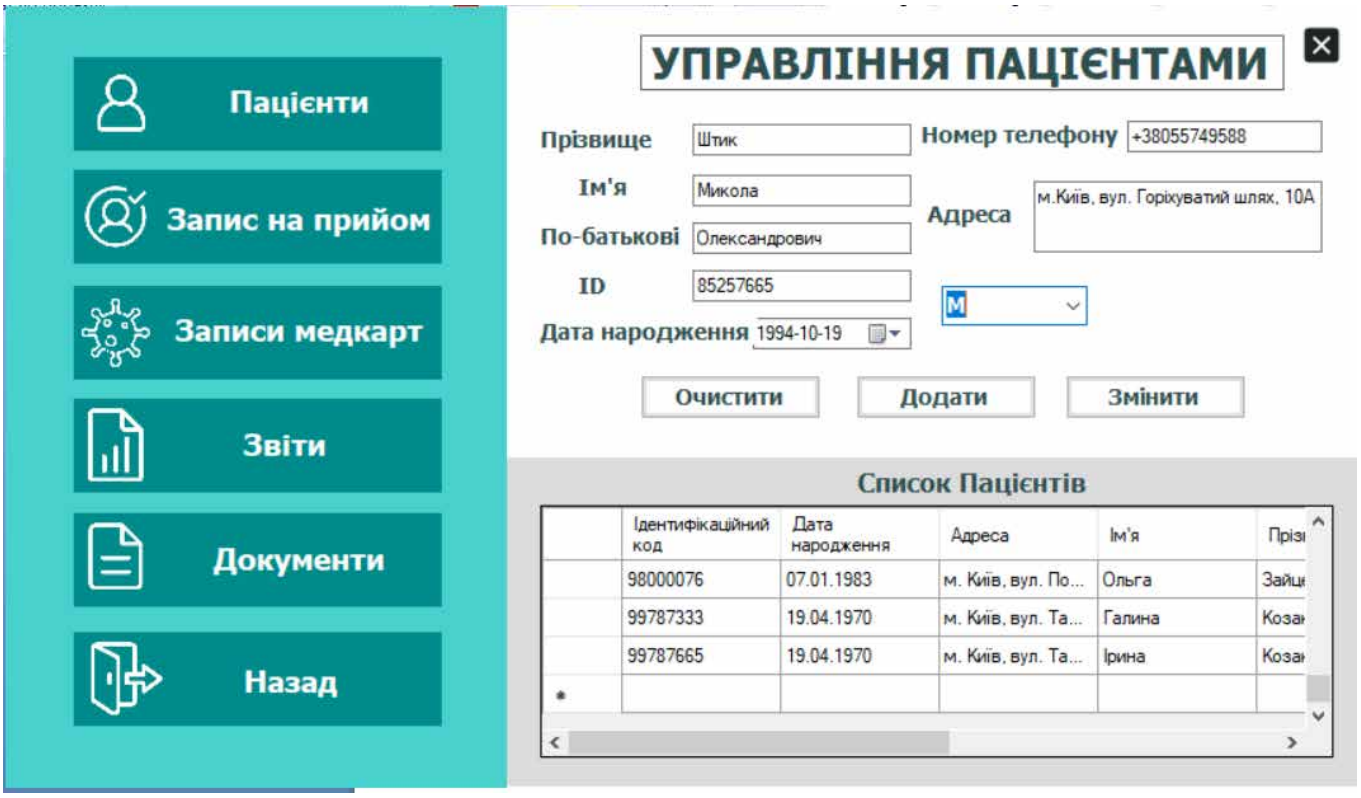

Рис.25 Введення даних

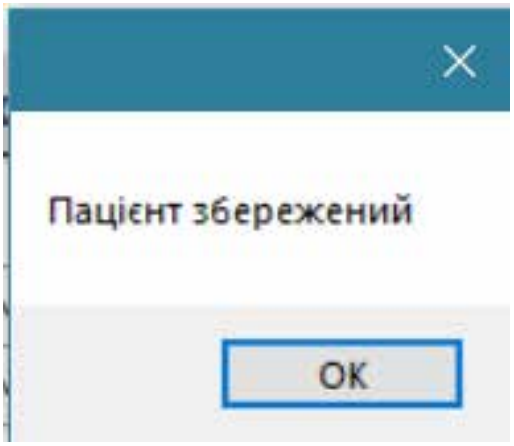

Рис.26 Повідомлення про успішне додавання

Рис.27 Доданий пацієнт

Редагування на рис. 28:

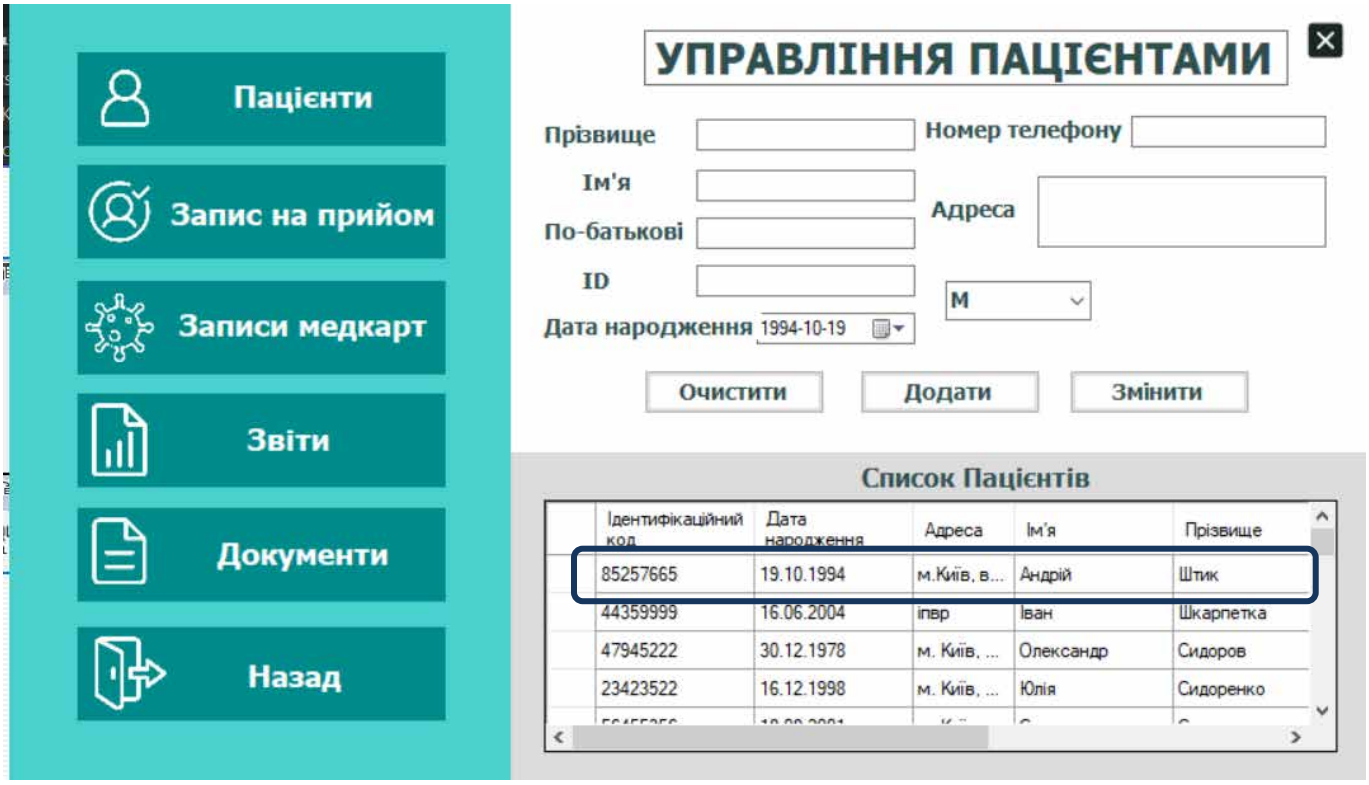

Рис. 28 Відредагований запис

Перегляд та додавання та записів таблиці «Талон» (рис.29-31):

Додавання:

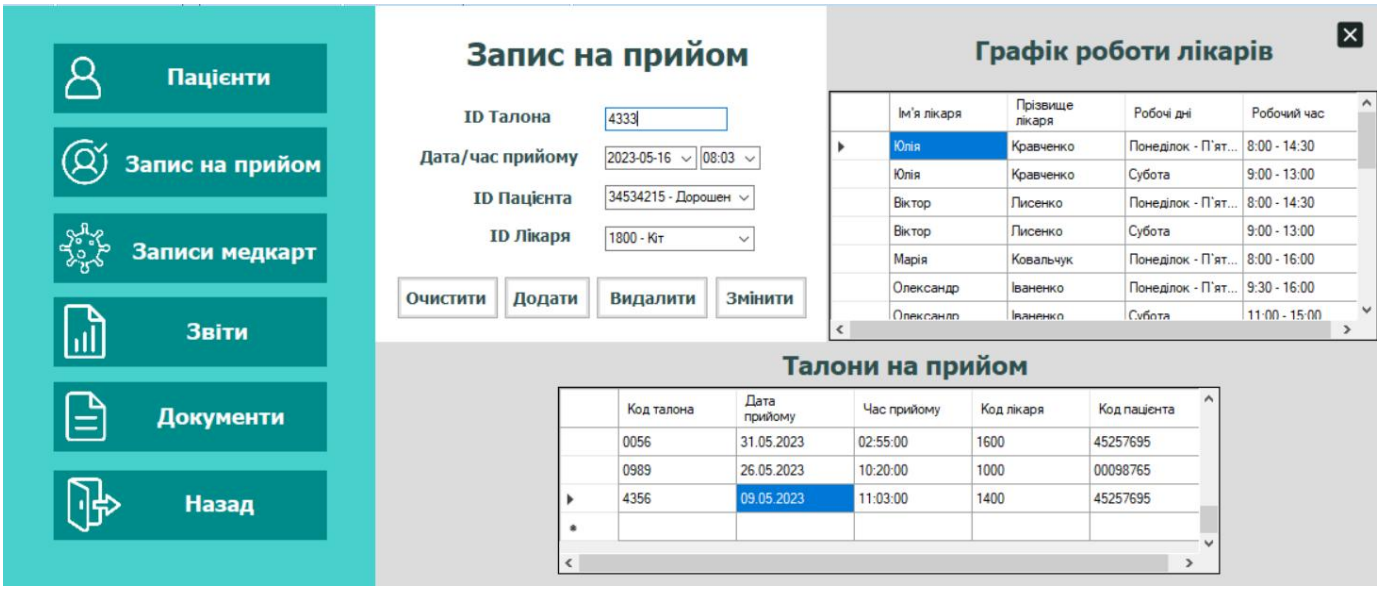

Рис. 29 Введення необхідної інформації

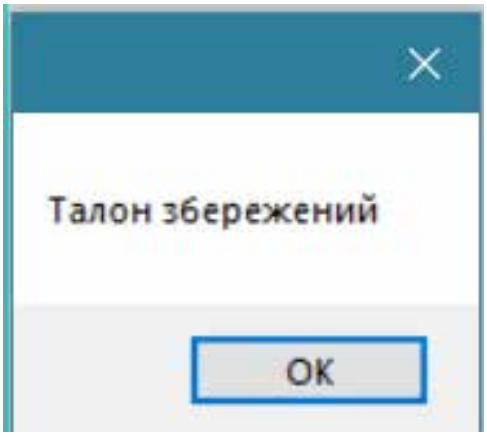

Рис. 30 Повідомлення про успішне додання талона

Талони на прийом

Код талона	Дата прийому	Час прийому	Код лікаря	Код пацієнта
0989	26.05.2023	10:20:00	1000	00098765
4333	16.05.2023	08:03:00	1800	34534215
4356	09.05.2023	11:03:00	1400	45257695

Рис. 31 Доданий Талон

Редагування записів таблиці «Талон» зображено на рис. 32-33:

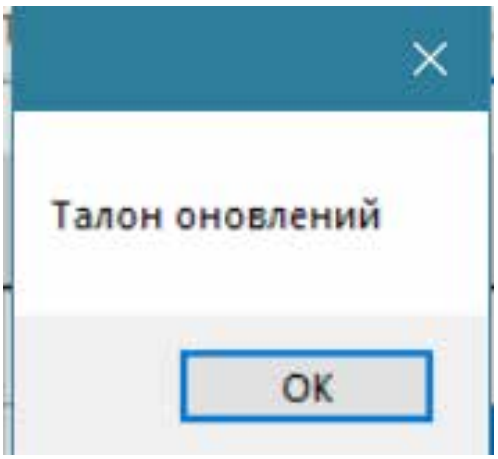

Рис. 32 Повідомлення про успішне оновлення інформації

Рис.33 Оновлена інформація

Видалення на рис. 34:

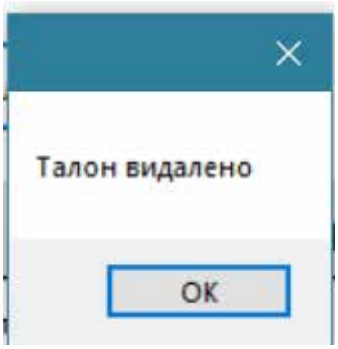

Рис 34 Повідомлення про успішне видалення запису з таблиці "Талон"

Перегляд , додавання та редагування записів таблиці «Мед-карта» (вікно на рис 35) мають аналогічну процедуру:

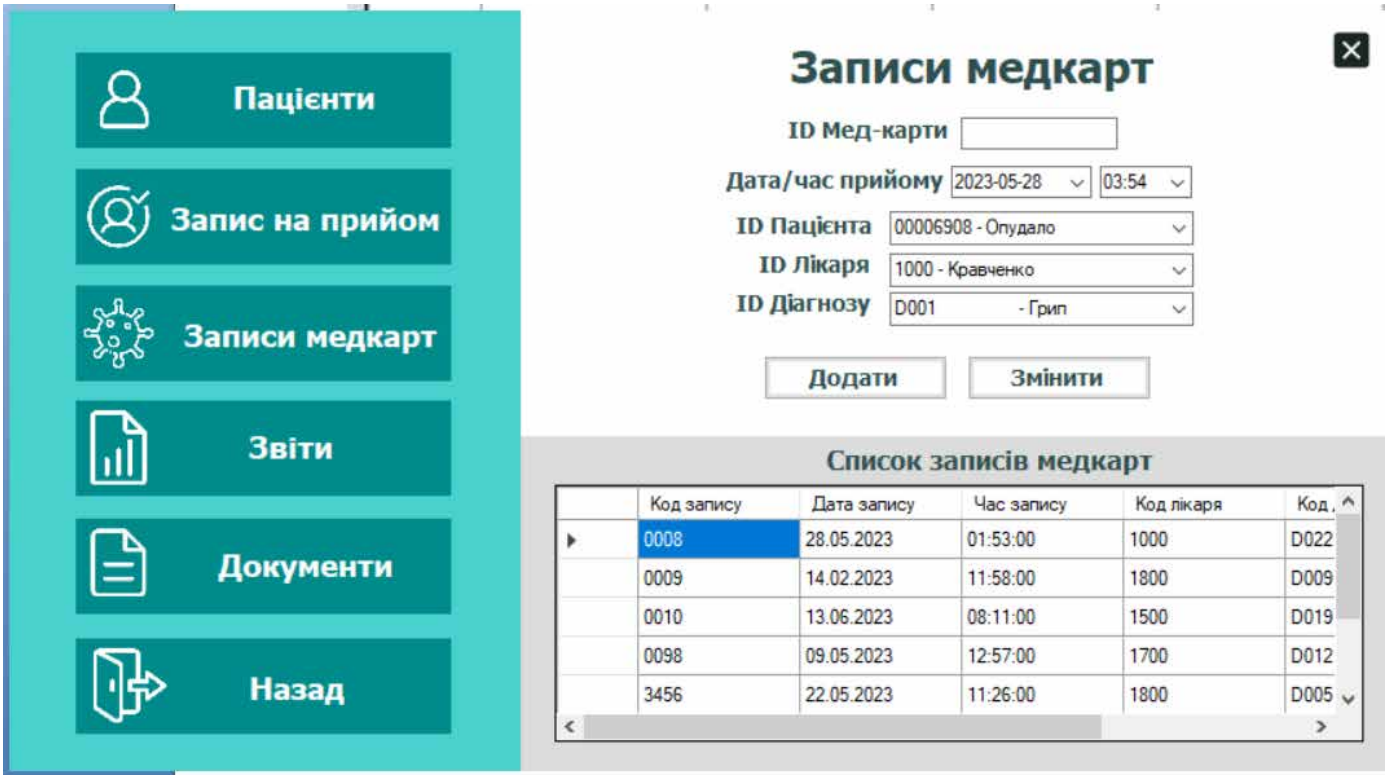

Рис. 35 Вікно «Записи Мед-карт»

Процес додавання та редагування записів аналогічний до попереднього вікна.

Вікно «Документи» (рис. 36) містить важливу для роботи умовно постійну інформацію:

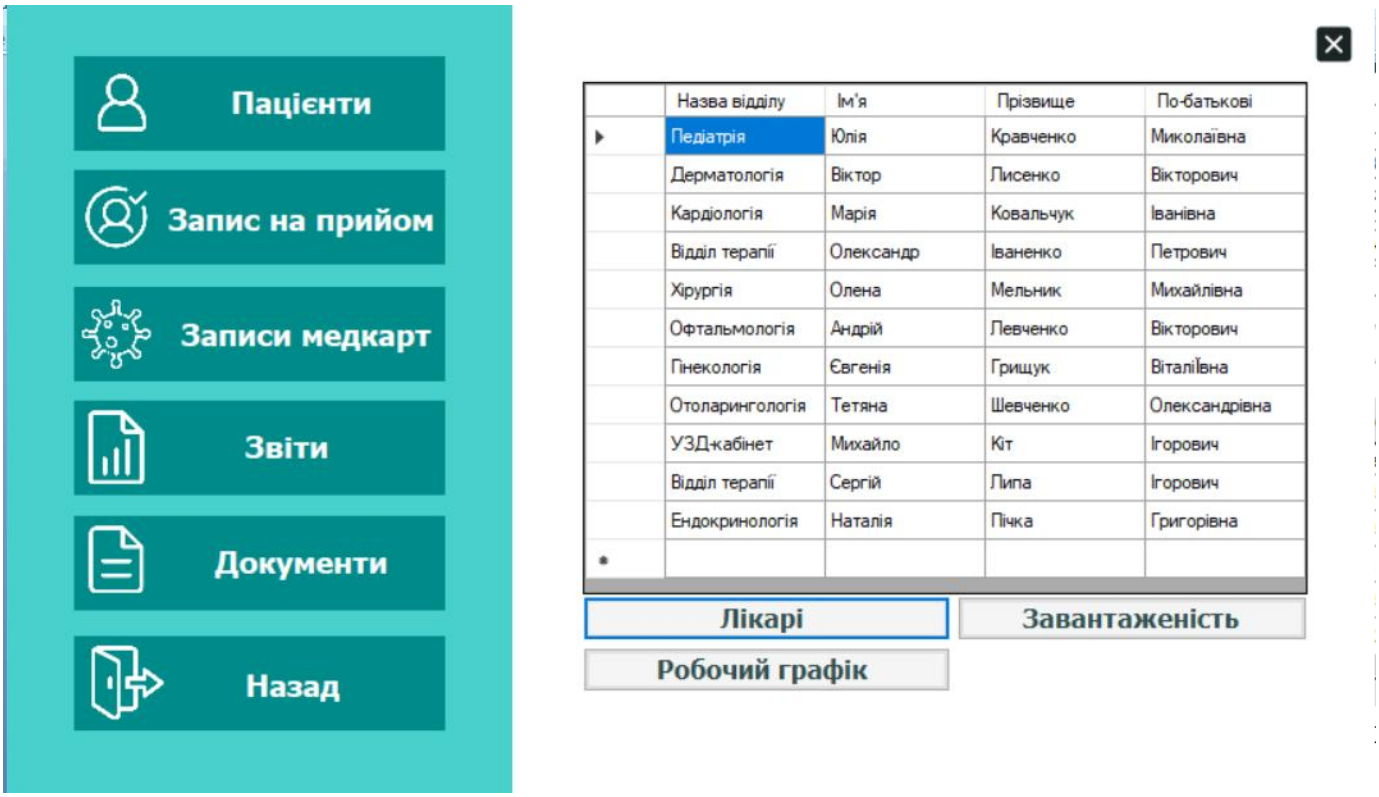

Рис. 36 Вікно з документацією

Результат роботи звітів зображені на рис 37-38

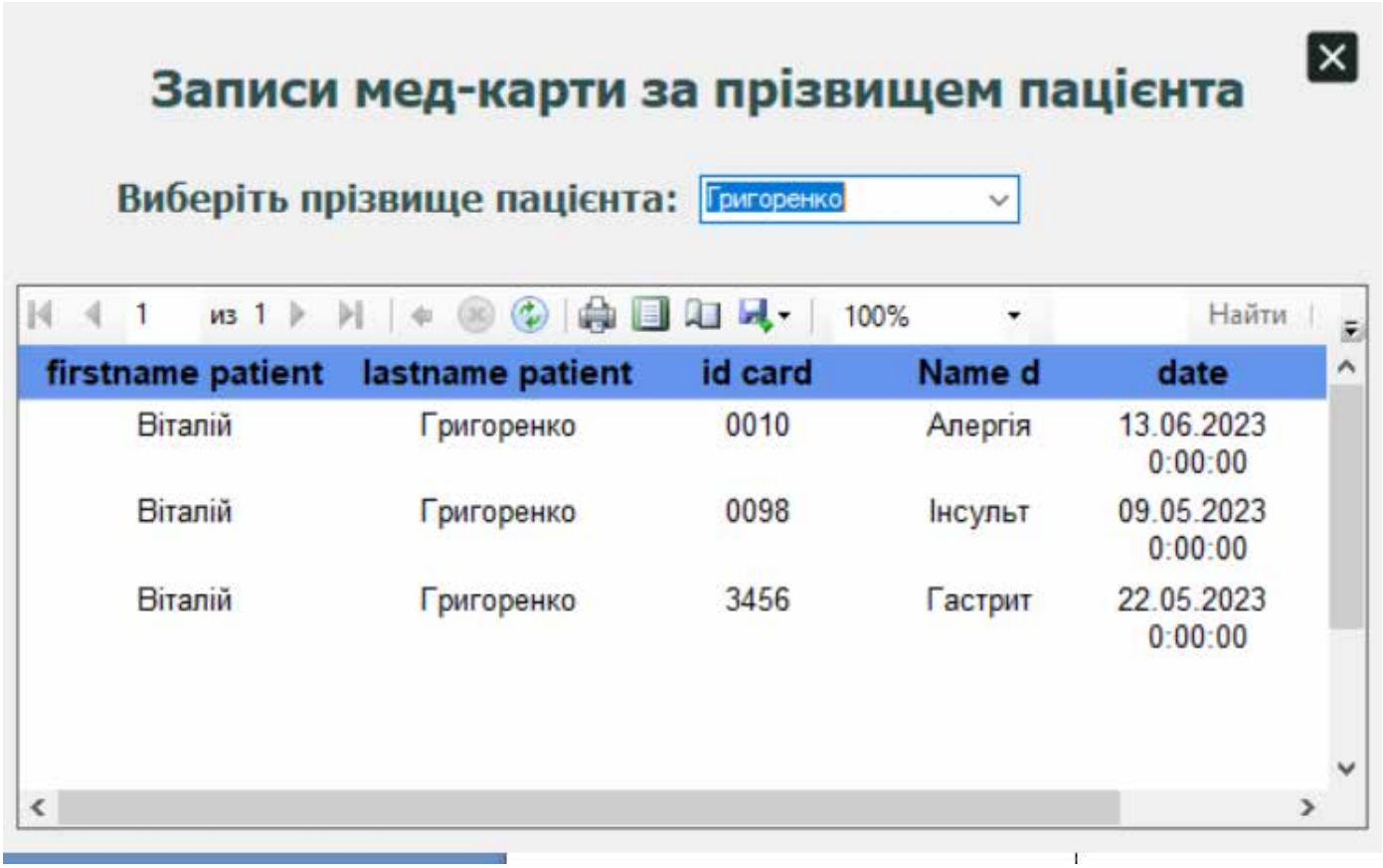

Рис 37 Звіт «Записи мед карти за прізвищем пацієнта»

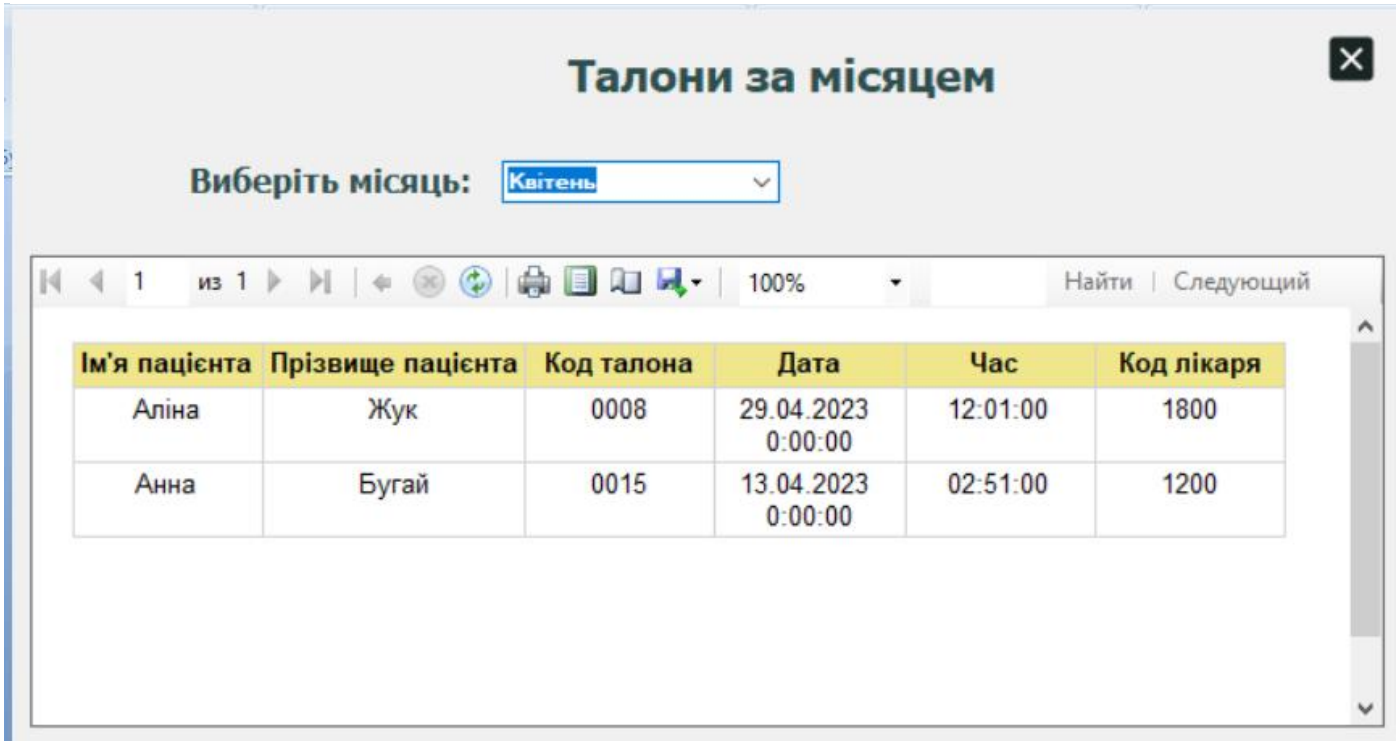

Рис 38 Звіт «Записи на прийом до лікаря за місяцем»

4.2 Вимоги до апаратного та програмного забезпечення

Мінімальні технічні вимоги до апаратного забезпечення користувача для роботи з програмою:

- Процесор класу Intel Pentium G4560;
- Об’єм оперативної пам’яті не менше 500 мб;
- Мінімальна пропускна здатність каналу зв’язку – 128 Kbit/s;
- Об’єм жорсткого диску не менше 20GB

Вимоги до системного програмного забезпечення:

Для статичної роботи програмного продукту потрібно мати таке програмне забезпечення :

- Microsoft SQL Server 2017 (або вище);
- Операційна система: Windows 7 (або вище).

4.3 Склад інсталяційного пакету

Для розгортання даного прикладного програмного забезпечення було створено інсталяційний пакет, який служить для встановлення програми та необхідні програмні компоненти, які потрібні для її роботи. До складу цього пакету входять наступні компоненти:

- Прикладне програмне забезпечення, що було створено в рамках дипломного проекту;
- Microsoft .NET Framework 4.7.2. Даний програмний компонент необхідно буде автоматично встановлено при розгортанні програми;
- Програма для створення бази даних на сервері;

Порядок встановлення інсталяційного пакета:

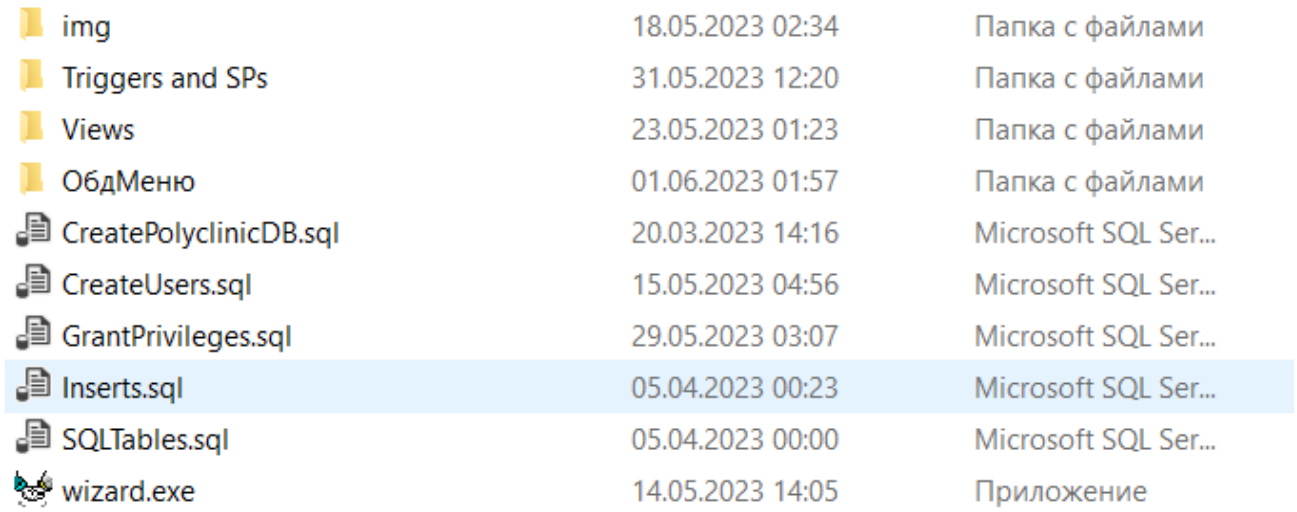

Рис. 39 Запити

3. Завантажити на сервер та виконати запити у такому порядку:
    - CreatePolyclinicDB.sql
    - SQLTables.sql
    - Inserts.sql
    - SQLTables.sql
    - CreateUsers.sql
    - GrantPrivileges.sql

А також наступні уявлення (рис. 40), процедури та тригери (рис. 41).

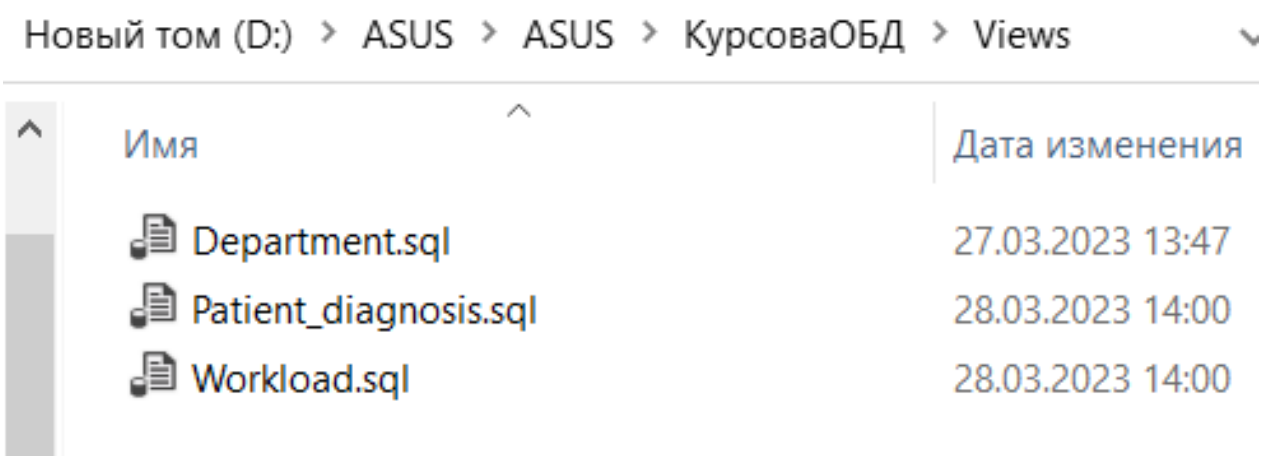

Рис. 40 уявлення які входять в інсталяційний пакет

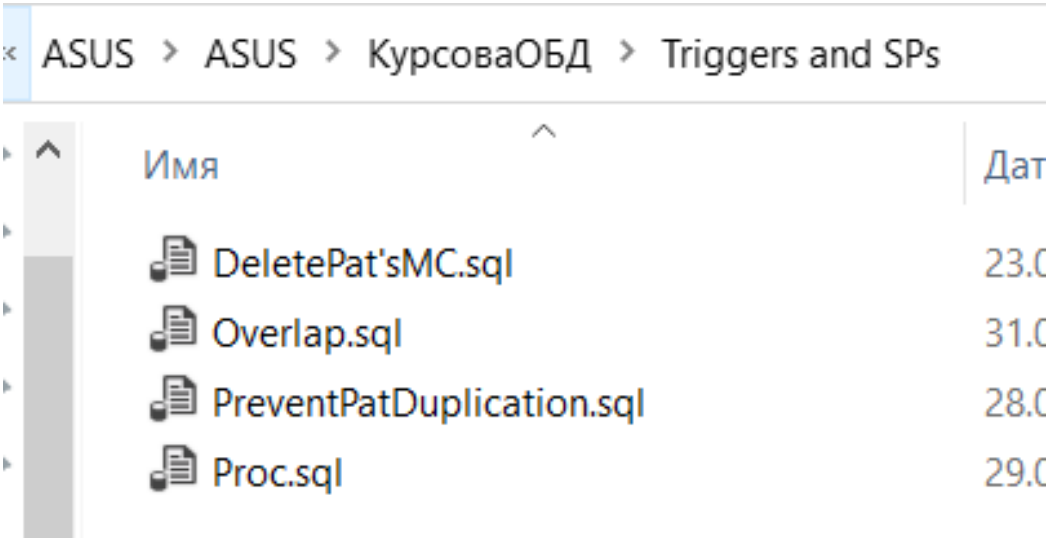

Рис. 41 процедури та тригери, що входять в інсталяційний пакет

Далі запустити програму Wizard.exe та сформувати файл з ConnectionString.

Після цього запустити .exe Файл програми з папки ОбдМеню.

Рис. 42 .exe файл

Діаграма пакетів системи зображена на рис. 43:

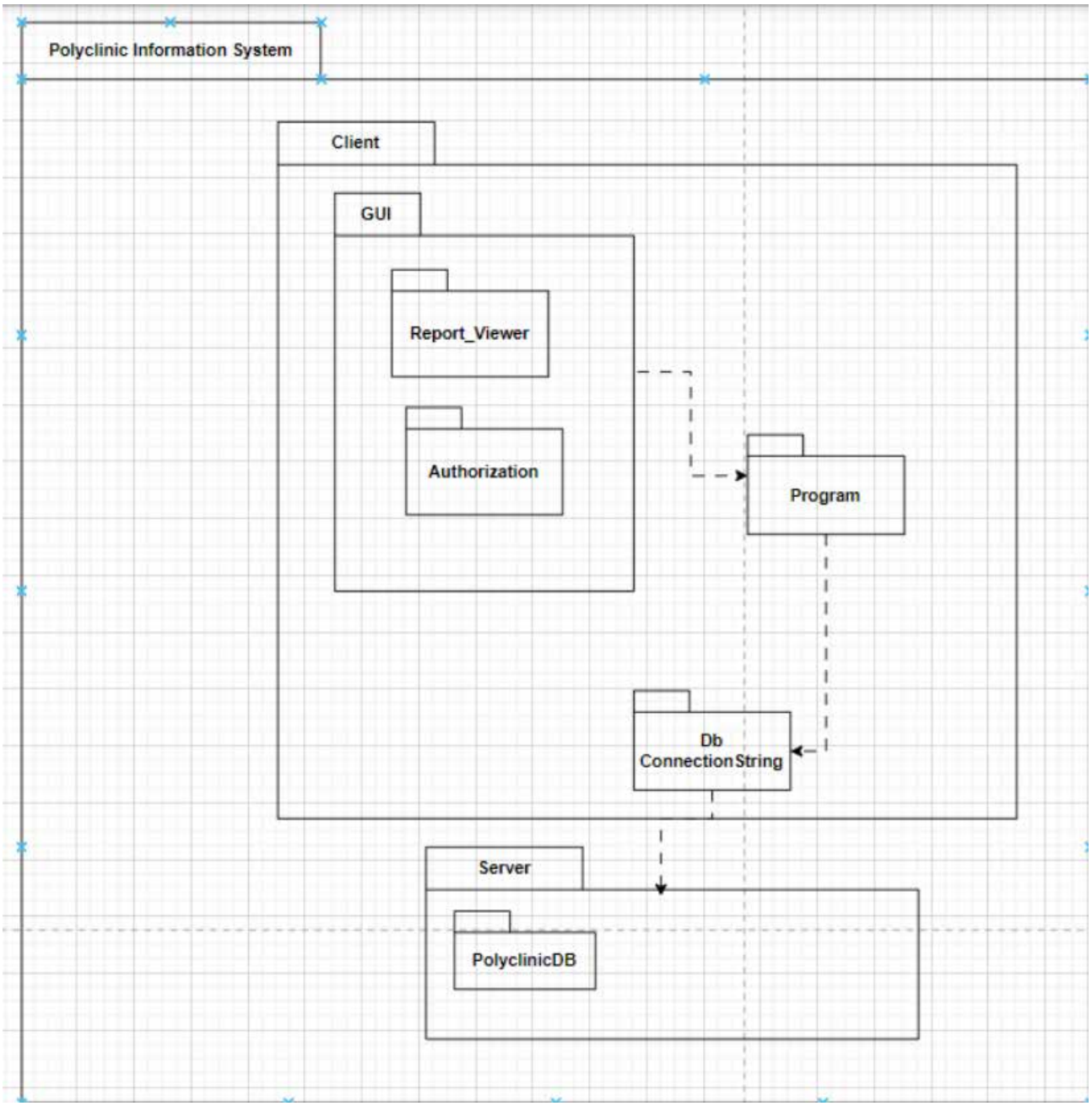

Рис. 43 Діаграма пакетів

4.4 Подальші перспективи розвитку программного продукту

Розроблена система має потенціал до вдосконалення:

- реалізація web-версії з хмарним зберіганням;
- інтеграція з електронними медичними картами eHealth;
- впровадження модуля штучного інтелекту для прогнозування завантаженості лікарів;
- локалізація на кілька мов.

4.5 Висновки до розділу 4

Розроблена система була апробована в умовах, максимально наближених до реальної експлуатації, з урахуванням специфіки роботи медичного персоналу. Проведено тестування ключових модулів, у тому числі авторизації, роботи з базою, формування звітності та захисту даних. Усі виявлені недоліки були виправлені на етапі апробації, після чого система визнана придатною до впровадження.

ВИСНОВКИ

Під час виконання дипломної роботи було проведено аналіз роботи фахівця(реєстратора), для визначення його потреб та вимог до програмного забезпечення. Відповідно до цих вимог була розроблена модель даних у 3-й нормальній формі, потім на основі неї розроблена база даних. Разом з базою даних були створені сутності, уявлення, користувачі, тригери, збережені процедури, транзакції. Інтерфейс користувача розроблений в Microsoft Visual Studio, використовуючи мову програмування C#, реалізовано його зв’язок з базою даних.

Під час тестової роботи з програмним продуктом були визначені та виправлені помилки, протестовані всі функції, які має виконувати програмний продукт: введення, редагування, видалення даних, їх обробка. Під час роботи були розвинені навички створення бази даних для конкретної галузі, створення інтерфейсу, який був би зрозумілий та легкий у використані. Прикладна програма дає змогу працювати з базою даних ,тобто переглядати, додавати, редагувати, видаляти дані, які зберігаються в базі даних.

Отже можна зробити висновок, що був розроблений зручний програмний продукт, який пов’язаний з базою даних, і полегшує роботу реєстратора з пацієнтами та їх даними.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ISO/IEC 2382:2015, Information technology — Vocabulary — Part 1: Terms and definitions.
   URL: https://www.iso.org/obp/ui/en/#iso:std:iso-iec:2382:ed-1:v2:en (Дата звернення: 15.05.2025).
2. Предметна область, моделювання предметної області
   URL: https://learn.microsoft.com/uk-ua/visualstudio/IDe/create-csharp-winform-visual-studio?view=vs-2022 (Дата звернення: 16.05.2025).
3. Реляційна модель даних. Основна термінологія. Види ключів. Цілісність реляційних даних.
   URL: https://learn.microsoft.com/uk-ua/visualstudio/IDe/create-csharp-winform-visual-studio?view=vs-2022 (Дата звернення: 17.05.2025).
4. Мулеса О.Ю. Основи мови запитів SQL. – Ужгород, 2015. – 48 с
   URL: https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/8868/1/sql.pdf
   (Дата звернення: 18.05.2025).
5. Лекція 6. Нормалізація
   URL: https://elib.lntu.edu.ua/sites/default/files/elib_upload/%D0%95%D0%9D%D0%9F_%D0%A1%D0%B0%D0%B2%D0%B0%D1%80%D0%B8%D0%BD_%D0%9B%D0%B5%D0%BF%D0%BA%D0%B8%D0%B9/teoretic/lec6.html
   (Дата звернення: 18.05.2025).
6. Методичні вказівки до лабораторних робіт з дисципліни «Бази даних і знань»
   URL: https://elib.lntu.edu.ua/sites/default/files/elib_upload/%D0%95%D0%9D%D0%9F_%D0%A1%D0%B0%D0%B2%D0%B0%D1%80%D0%B8%D0%BD_%D0%9B%D0%B5%D0%BF%D0%BA%D0%B8%D0%B9/teoretic/lec6.html
   (Дата звернення: 20.05.2025).
7. Системи баз даних та знань. Книга 2 уривок

URL: https://magnolia.lviv.ua/wp-content/uploads/2024/01/Systemy-baz-danykh-ta-znan-Knyha-2_uryvok.pdf76 (Дата звернення: 20.05.2025).

ДОДАТКИ

ДОДАТОК А

Створення таблиць:

```
use PolyclinicDB
go
IF EXISTS (SELECT name FROM sys.objects
 WHERE name = 'Department' AND type_desc = 'USER_TABLE')
drop table Department
go
create table Department
(id_dept char(4) NOT NULL primary key, name_dept varchar(50))
go
IF EXISTS (SELECT name FROM sys.objects
 WHERE name = 'Specialization' AND type_desc = 'USER_TABLE')
drop table Specialization
go
create table Specialization
(id_s char(4) NOT NULL primary key, name_s varchar(50))
go
IF EXISTS (SELECT name FROM sys.objects
 WHERE name = 'Office' AND type_desc = 'USER_TABLE')
drop table Office
go
create table Office
(id_office char(4) NOT NULL primary key, name_office
varchar(50))
go
IF EXISTS (SELECT name FROM sys.objects
 WHERE name = 'Gender' AND type_desc = 'USER_TABLE')
drop table Gender
go
create table Gender
(id_gender char(4) NOT NULL primary key, name_gender
varchar(50))
go
IF EXISTS (SELECT name FROM sys.objects
 WHERE name = 'Diagnosis' AND type_desc = 'USER_TABLE')
drop table Diagnosis
go
create table Diagnosis
(id_diagnosis char(4) NOT NULL primary key,
Name_d varchar(50),
```

```
Description_d varchar(500))
go
IF EXISTS (SELECT name FROM sys.objects
 WHERE name = 'Doctor' AND type_desc = 'USER_TABLE')
drop table Doctor
go
create table Doctor
(id_doctor char(4) NOT NULL primary key,
firstname_doc varchar(20),
lastname_doc varchar(20),
patronimic_doc varchar(30),
doctor_phone_n char(15),
id_dept char(4) NOT NULL,
id_office char(4) NOT NULL,
id_s char(4) NOT NULL,
CONSTRAINT id_dept_FK FOREIGN KEY (id_dept)
REFERENCES Department (id_dept),
CONSTRAINT id_office_FK FOREIGN KEY (id_office)
REFERENCES Office (id_office),
CONSTRAINT id_s_FK FOREIGN KEY (id_s)
REFERENCES Specialization (id_s)
)
go
IF EXISTS (SELECT name FROM sys.objects
 WHERE name = 'Patient' AND type_desc = 'USER_TABLE')
drop table Patient
go
create table Patient
(id_patient char(8) NOT NULL primary key,
birthdate date,
Address_ varchar(50),
firstname_patient varchar(20),
lastname_patient varchar(20),
patronimic_patient varchar(30),
patient_phone_n char(15),
id_gender char(4) NOT NULL,
CONSTRAINT id_gender_FK FOREIGN KEY (id_gender)
REFERENCES Gender (id_gender)
)
go
IF EXISTS (SELECT name FROM sys.objects
 WHERE name = 'Work_Schedule' AND type_desc = 'USER_TABLE')
drop table Work_Schedule
go
```

```
create table Work_Schedule
(id_ws char(4) NOT NULL primary key,
WorkingDays varchar(50),
WorkingHours varchar(20),
id_doctor char(4) NOT NULL,
CONSTRAINT FK_id_doctor FOREIGN KEY (id_doctor)
REFERENCES Doctor (id_doctor)
)
go
IF EXISTS (SELECT name FROM sys.objects
 WHERE name = 'Ticket' AND type_desc = 'USER_TABLE')
drop table Ticket
go
create table Ticket
(id_ticket char(4) NOT NULL primary key,
date_ date,
time_ time,
id_doctor char(4) NOT NULL,
id_patient char(8) NOT NULL,
CONSTRAINT id_doctor_FK FOREIGN KEY (id_doctor)
REFERENCES Doctor (id_doctor),
CONSTRAINT id_patient_FK FOREIGN KEY (id_patient)
REFERENCES Patient (id_patient)
)
go
IF EXISTS (SELECT name FROM sys.objects
 WHERE name = 'Medical_card' AND type_desc = 'USER_TABLE')
drop table Medical_card
go
create table Medical_card
(id_card char(4) NOT NULL primary key,
date_ date,
time_ time,
id_doctor char(4) NOT NULL,
id_diagnosis char(4) NOT NULL,
id_patient char(8) NOT NULL,
CONSTRAINT id_doctorFK FOREIGN KEY (id_doctor)
REFERENCES Doctor (id_doctor),
CONSTRAINT id_patientFK FOREIGN KEY (id_patient)
REFERENCES Patient (id_patient),
CONSTRAINT id_diagnosisFK FOREIGN KEY (id_diagnosis)
REFERENCES Diagnosis (id_diagnosis)
)
```

ДОДАТОК Б

Транзакції. Заповнення умовно-постійної інформації :

```
use PolyclinicDB
Go
Begin transaction
Insert into Gender values
('F','Жіноча'),
('M','Чоловіча')
Go
Commit
Go
Begin transaction
Insert into Specialization values
('0010','Педіатр'),
('0011','Дерматолог'),
('0012','Кардіолог'),
('0013','Терапевт'),
('0014','Хірург'),
('0015','Офтальмолог'),
('0016','Гінеколог'),
('0017','ЛОР'),
('0018','УЗІ-спеціаліст'),
('0019','Сімейний лікар'),
('0020','Ендокринолог')
Go
Commit
Go
Begin transaction
Insert into Department values
('0100','Педіатрія'),
('0110','Дерматологія'),
('0120','Кардіологія'),
('0130','Відділ терапії'),
('0140','Хірургія'),
('0150','Офтальмологія'),
('0160','Гінекологія'),
('0170','Отоларингологія'),
('0180','УЗД-кабінет'),
('0190','Лабораторія'),
('0200','Ендокринологія')
```

```
Go
Commit
Go
Begin transaction
Insert into Office values
('0030','Кабінет педіатра'),
('0031','Кабінет дерматолога'),
('0032','Кабінет кардіолога'),
('0033','Кабінет терапевта'),
('0034','Кабінет хірурга'),
('0035','Кабінет офтальмолога'),
('0036','Кабінет гінеколога'),
('0037','Кабінет ЛОРа'),
('0038','УЗД-кабінет'),
('0039','Кабінет сімейного лікаря'),
('0040','Кабінет ендокринолога')
Go
Commit
Go
Begin transaction
Insert into Diagnosis values
('D001', 'Грип', 'Респіраторна інфекційна хвороба, яку
викликають віруси грипу.'),
('D002', 'Ангіна', 'Захворювання горла, яке супроводжується
болями, запаленням та набряком.'),
('D003', 'Цукровий діабет', 'Хронічне захворювання, викликане
недостатнім виробленням або використанням інсуліну.'),
('D004', 'Гіпертонія', 'Хвороба, яка характеризується
підвищеним артеріальним тиском.'),
('D005', 'Гастрит', 'Захворювання шлунку, яке характеризується
запаленням слизової оболонки.'),
('D006', 'Пневмонія', 'Захворювання легенів, яке
характеризується запаленням повітряних мішків.'),
('D007', 'Мігрень', 'Хвороба, яка характеризується набряком та
болями голови.'),
('D008', 'Артрит', 'Захворювання суглобів, яке
характеризується запаленням та болями.'),
('D009', 'Астма', 'Хронічне захворювання дихальних шляхів, яке
характеризується звуженням повітряних шляхів.'),
('D010', 'Гіпотиреоз', 'Хвороба щитоподібної залози, яка
характеризується недостатньою продукцією гормонів щитоподібної
залози.'),
```

```
('D011', 'Гемофілія', 'Наслідкове захворювання, яке
характеризується порушенням згортання крові.'),
('D012', 'Інсульт', 'Захворювання мозку, яке характеризується
порушенням кровообігу та нервової діяльності.'),
('D013', 'Діарея', 'Розлад шлунково-кишкового тракту, який
характеризується частим виділенням рідкого калу.'),
('D014', 'Анемія', 'Патологічний стан, який характеризується
зниженням кількості еритроцитів та гемоглобіну в крові.'),
('D015', 'Катаракта', 'Захворювання очей, яке характеризується
погіршенням зіниць та зору.'),
('D016', 'Ожиріння', 'Хронічне захворювання, яке
характеризується надмірним накопиченням жиру в організмі.'),
('D017', 'Інфекція сечовивідних шляхів', 'Захворювання, яке
характеризується запаленням сечових шляхів.'),
('D018', 'Гепатит', 'Захворювання печінки, яке
характеризується запаленням та пошкодженням тканин.'),
('D019', 'Алергія', 'Хвороба, яка характеризується відповіддю
імунної системи на певні речовини (алергени).'),
('D020', 'Кір', 'Вірусне захворювання, яке характеризується
висипанням на шкірі та лихоманкою.'),
('D021', 'Бронхіт', 'Захворювання дихальних шляхів, яке
характеризується запаленням бронхів та кашлем.'),
('D022', 'Онкозахворювання', 'Хвороба, яка характеризується
незвичайними клітинами, що зростають та розмножуються
неконтрольовано.'),
('D023', 'Ларингіт', 'Захворювання горла та голосових зв`язок,
яке характеризується запаленням та набряком.'),
('D024', 'Менінгіт', 'Інфекційне захворювання оболонок
головного мозку, яке характеризується запаленням та порушенням
нервової системи.'),
('D025', 'Коньюнктивіт', 'Захворювання очей, яке
характеризується запаленням тонкої плівки, що покриває білок
та внутрішню поверхню повіки.'),
('D026', 'Нарколепсія', 'Хронічне захворювання нервової
системи, яке характеризується нестачею енергії та
нездоланністю заснути в найбільш несподіваний момент.'),
('D027', 'Туберкульоз', 'Інфекційне захворювання, яке
характеризується ураженням легень та інших органів.'),
('D028', 'Малярія', 'Інфекційне захворювання, яке
характеризується пошкодженням еритроцитів та порушенням роботи
органів.'),
```

```
('D029', 'Епілепсія', 'Хвороба, яка характеризується
порушенням роботи нервової системи та періодичними приступами
нестерпної болі.'),
('D030', 'Обструктивна бронхітна хвороба', 'Хвороба дихальних
шляхів, яка характеризується зменшенням провідності повітря
через бронхи та проблемами з подихом.'),
('D031', 'Дизентерія', 'Інфекційне захворювання кишечника, яке
характеризується сильним діареєю та болями у животі.'),
('D032', 'Енцефаліт', 'Інфекційне захворювання головного
мозку, яке характеризується запаленням та порушенням нервової
системи.'),
('D033', 'Ревматоїдний артрит', 'Хронічне запальне
захворювання суглобів, яке характеризується болісністю,
набряком та деформацією.'),
('D034', 'Скарлатина', 'Інфекційне захворювання, яке
характеризується висипанням на шкірі, болем у горлі та
лихоманкою.'),
('D035', 'Тиреоїдит', 'Захворювання щитовидної залози, яке
характеризується запаленням та порушенням її функцій.'),
('D036', 'Тромбофлебіт', 'Захворювання, яке характеризується
утворенням тромбів у венах та запаленням їх стінок.'),
('D037', 'Холера', 'Інфекційне захворювання кишкової системи,
яке характеризується різкою діареєю та швидкою втратою води і
солей.'),
('D038', 'Депресія', 'Психічне захворювання, яке
характеризується порушенням настрою, енергетики та
мислення.'),
('D039', 'Гострий панкреатит', 'Захворювання підшлункової
залози, при якому відбувається її запалення.'),
('D040', 'Хламідіоз', 'Інфекційне захворювання, яке
передається статевим шляхом та характеризується запаленням
слизової оболонки.'),
('D041', 'Кандидоз', 'Грибкове захворювання, яке може вражати
шкіру, слизові оболонки, нігті, волосся.'),
('D042', 'Геморой', 'Захворювання прямої кишки, при якому
виникають вузлики та запалення.'),
('D043', 'Сколіоз', 'Деформація хребта, при якій відбувається
вигин у бічну сторону.'),
('D044', 'Псоріаз', 'Хронічне захворювання шкіри, при якому
виникають червоні плями з білими лусочками.'),
('D045', 'Сказ', 'Вірусне захворювання, яке передається від
тварин на людину та характеризується психічними та
неврологічними порушеннями.'),
```

```
('D046', 'Цинга', 'Паразитарне захворювання, при якому виникає
свербіж та підвищення температури.'),
('D047', 'Цистит', 'Захворювання сечового міхура, при якому
виникає запалення.'),
('D048', 'Тиф', 'Інфекційне захворювання, яке характеризується
лихоманкою, головним болем та діареєю.'),
('D049', 'Отит', 'Захворювання вуха, при якому відбувається
запалення середнього вуха.'),
('D050', 'Отруєння', 'Стан, що виникає при вживанні токсичних
речовин, що призводить до різноманітних порушень у
функціонуванні організму.')
Go
Commit
Go
Begin transaction
Insert into Work_Schedule values
('D100','Понеділок - П`ятниця','8:00 - 14:30','1000'),
('D120','Понеділок - П`ятниця','8:00 - 14:30','1100'),
('D140','Понеділок - П`ятниця','8:00 - 16:00','1200'),
('D150','Понеділок - П`ятниця','9:30 - 16:00','1300'),
('D170','Понеділок - П`ятниця','9:00 - 15:30','1400'),
('D190','Понеділок - П`ятниця','8:00 - 14:30','1500'),
('D210','Понеділок - П`ятниця','9:00 - 15:30','1600'),
('D230','Понеділок - П`ятниця','9:30 - 16:00','1700'),
('D250','Понеділок - П`ятниця','11:30 - 18:00','1800'),
('D270','Понеділок - П`ятниця','8:00 - 15:00','1900'),
('D290','Понеділок - П`ятниця','12:00 - 18:30','2000'),
Go
Commit
```

ДОДАТОК В

Команди створення уявлень:
Діагнози пацієнтів:

```
use PolyclinicDB
go
IF EXISTS (SELECT name FROM sys.all_views
WHERE name = 'Patient_diagnosis')
drop view Patient_diagnosis
go
create view Patient_diagnosis AS
Select  firstname_patient, lastname_patient,
patronimic_patient, id_card, Name_d
From Patient, Medical_card, Diagnosis
Where Patient.id_patient = Medical_card.id_patient
And Diagnosis.id_diagnosis = Medical_card.id_diagnosis
```

Графік завантаженості лікарів:

```
use PolyclinicDB
go
IF EXISTS (SELECT name FROM sys.all_views
WHERE name = 'Workload_of_the_doctors')
drop view Workload_of_the_doctors
go
create view Workload_of_the_doctors AS
Select firstname_doc, lastname_doc, patronimic_doc,
id_ticket, date_, time_, firstname_patient,
lastname_patient, patronimic_patient
From Doctor, Ticket, Patient
Where Doctor.id_doctor = Ticket.id_doctor And
Patient.id_patient = Ticket.id_patient
```

Лікарі відділів:

```
use PolyclinicDB
go
IF EXISTS (SELECT name FROM sys.all_views
WHERE name = 'Doctors_from_the_department')
```

```
drop view Doctors_from_the_department
go
create view Doctors_from_the_department AS
Select name_dept, firstname_doc, lastname_doc, patronimic_doc
From Doctor, Department
Where Department.id_dept = Doctor.id_dept
```

ДОДАТОК Д

Тригери:

```
/Тригер, що запобігає накладанню записів до лікаря

USE [PolyclinicDB]
GO
/****** Object:  Trigger [dbo].[PreventDuplicateRecords]
Script Date: 31.05.2023 11:55:30 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
Create TRIGGER [dbo].[PreventDuplicateRecords]
ON [dbo].[Ticket]
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @id_doctor char(4);
    DECLARE @date_ date;
    DECLARE @time_ time;

    -- Get the values of id_doctor, date_, and time_ for the
inserted row
    SELECT @id_doctor = id_doctor, @date_ = date_, @time_ =
time_
    FROM inserted;

    -- Check if there are any existing records with the same
id_doctor, date_ and time
    IF EXISTS (
        SELECT 1
        FROM Ticket
        WHERE id_doctor = @id_doctor
        AND date_ = @date_
        AND ABS(DATEDIFF(minute, time_, @time_)) < 30
        )
        BEGIN
        -- If a duplicate record is found, rollback the
transaction and display an error message
```

```
            RAISERROR('В цього лікаря вже є запис на такий
час.', 16, 1);
            ROLLBACK TRANSACTION;
        END;
    END;
END;
/Тригер, що запобігає дублюванню пацієнтів

USE [PolyclinicDB]
GO
/****** Object:  Trigger [dbo].[PreventDuplicateNames]
Script Date: 01.06.2023 03:14:55 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [dbo].[PreventDuplicateNames]
ON [dbo].[Patient]
AFTER INSERT, UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    -- Check if there are any duplicate records with the same
firstname_patient, lastname_patient, and patronimic_patient
    IF EXISTS (
        SELECT 1
        FROM Patient p1
        INNER JOIN inserted i ON p1.firstname_patient =
i.firstname_patient
            AND p1.lastname_patient = i.lastname_patient
            AND p1.patronimic_patient = i.patronimic_patient
        WHERE p1.id_patient <> i.id_patient -- Exclude the
currently inserted/updated record
    )
    BEGIN
        -- If a duplicate record is found, rollback the
transaction and display an error message
        RAISERROR('Такий пацієнт вже існує.', 16, 1);
        ROLLBACK TRANSACTION;
    END;
END;
```

Збережені процедури:

```
/Додавання записів до таблиці Пацієнт:
USE [PolyclinicDB]
GO
/****** Object:  StoredProcedure [dbo].[AddPatient]    Script
Date: 01.06.2023 03:17:37 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[AddPatient]
    @id_patient CHAR(8),
    @birthdate DATE,
    @Address_ VARCHAR(50),
    @firstname_patient VARCHAR(20),
    @lastname_patient VARCHAR(20),
    @patronimic_patient VARCHAR(30),
    @patient_phone_n CHAR(15),
    @id_gender CHAR(4)
AS
BEGIN
    INSERT INTO Patient (id_patient, birthdate, Address_,
firstname_patient, lastname_patient, patronimic_patient,
patient_phone_n, id_gender)
    VALUES (@id_patient, @birthdate, @Address_,
@firstname_patient, @lastname_patient, @patronimic_patient,
@patient_phone_n, @id_gender)
END
/Редагування записів таблиці Пацієнт:

USE [PolyclinicDB]
GO
/****** Object:  StoredProcedure [dbo].[UpdatePatient]
Script Date: 01.06.2023 03:17:50 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[UpdatePatient]
    @id_patient CHAR(8),
    @birthdate DATE,
    @Address_ VARCHAR(50),
```

```
    @firstname_patient VARCHAR(20),
    @lastname_patient VARCHAR(20),
    @patronimic_patient VARCHAR(30),
    @patient_phone_n CHAR(15),
    @id_gender CHAR(4)
AS
BEGIN
    UPDATE Patient set birthdate=@birthdate,
Address_=@Address_, firstname_patient=@firstname_patient,
lastname_patient=@lastname_patient,
patronimic_patient=@patronimic_patient,
patient_phone_n=@patient_phone_n, id_gender=@id_gender where
id_patient=@id_patient
END
```

ДОДАТОК Е

Створення ролей та надання прав:

```
 --Запити для
 створення користувача
use PolyclinicDB
go
Drop user if exists viewer
Drop user if exists receptionist
create login viewer
with password = 'view58%'
create user viewer for login viewer
go
create login receptionist
with password = 'admin28!'
      create user receptionist for login receptionist

 --Надання прав
 користувачу receptionist
use PolyclinicDB
go
grant select on Department to receptionist
grant select on Diagnosis to receptionist
grant select on Doctor to receptionist
grant select on Gender to receptionist
grant select, insert, delete, update on Medical_card to
receptionist
grant select on office to receptionist
grant select , insert,  delete, update on Patient to
receptionist
grant select on Specialization to receptionist
grant select, insert, delete, update on Ticket to receptionist
grant select on Work_Schedule to receptionist
GRANT EXECUTE ON dbo.AddPatient TO receptionist
GRANT EXECUTE ON dbo.UpdatePatient TO receptionist
GRANT EXECUTE ON dbo.GetMC TO receptionist
go


 --Надання прав користувачу viewer
grant select on Department to viewer
grant select on Diagnosis to viewer
```

```
grant select on Doctor to viewer
grant select on Gender to viewer
grant select on Medical_card to viewer
grant select on office to viewer
grant select on Patient to viewer
grant select on Specialization to viewer
grant select on Ticket to viewer
grant select on Work_Schedule to viewer
go
```