Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

(повна назва)

Кафедра Інформатика

(повна назва)

**ПОЯСНЮВАЛЬНА ЗАПИСКА**

до курсової роботи

з дисципліни “ Організація баз даних та знань ”

за темою

“Розробка інформаційної системи „Магазин музичних інструментів”

з використанням СУБД \_SQL Server та мови запитів SQL

Виконав:

студент 2 курсу, групи ІТІНФ-20 -1

Самченко Станіслав Олександрович

(прізвище, ініціали)

Спеціальності 122 Комп’ютерні науки

(код і повна назва спеціальності)

Освітня програма Інформатика

(повна назва освітньої програми)

Керівник Яковлева О.В.

(посада, прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

(найменування вищого учбового закладу)

Факультет Інформаційно-аналітичних технологій та менеджменту

(повна назва)

Кафедра Інформатики

(повна назва)

Спеціальність 122 Комп’ютерні науки

(код і повна назва)

Освітня програма Інформатика

(повна назва освітньої програми)

Дисципліна Організація баз даних та знань

(повна назва освітньої програми)

Курс 2 Группа ІТІНФ-20 -1 Семестр 3 .

ЗАВДАННЯ

на курсовий проект (роботу) студента

Самченко Станіслав Олександрович

(прізвище, ім’я, по батькові)

1. Тема проекту (роботи) Розробка інформаційної системи „Магазин музичних інструментів” з використанням СУБД \_SQL Server та мови запитів SQL фф фф  
2. Строк здачі студентом закінченого проекту (роботи) 17.12.2021 .

3. Вихідні данні проекту (роботи) *мова запитів SQL, СУБД MS SQL Server, середовище MS SQL Server Management Studio, CASE – засіб візуального проектування даних Erwin, середовище розробки програмних систем Intellij IDEA 2021.2.2, мова є програмування Java, відомості про предметну область «*Магазин музичних інструментів*»* фф \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці

*1. Реляційні бази даних: функціональність, принципи проектування та використання*

*2. Проектування бази даних для ПО "Магазин музичних інструментів"*

*3. Проектування та розробка ІС для предметної області.*

5. Перелік графічного матеріала (з точним зазначенням обов’язкових креслень).

6. Дата видачі завдання 16.11.2021.

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № | Найменування етапів курсового проекту | Термін виконання етапів проекту | Примітки |
| 1 | *Отримання завдання* | *16.11.2021* | *Виконано* |
| 2 | *Вивчення теоретичного матеріалу, ознайомлення з програмним забезпеченням* | *17.11.21 – 27.11.21* | *Виконано* |
| 3 | *Розробка структури РБД* | *27.11.21 – 30.11.21* | *Виконано* |
| 4 | *Розробка функціональності ІС* | *02.12.21 – 14.12.21* | *Виконано* |
| 5 | *Тестування ІС* | *14.12.21 - 15.12.21* | *Виконано* |
| 6 | *Оформлення пояснювальної записки* | *24.12.21 – 28.12.21* | *Виконано* |
| 7 | *Захист курсової роботи* | *11.01.21* | *Виконано* |

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (підпис)

**РЕФЕРАТ/ABSTRACT**

Пояснювальна записка: с.44, рис.13, 3 додатка, 6 джерел.

БАЗА ДАНИХ, ПРЕДМЕТНА ОБЛАСТЬ, CASE – ЗАСІБ ПРОЕКТУВАННЯ, ER – ДІАГРАММА, ПРОГРАМА, ІНТЕРФЕЙС, ЛІКАРНЯ, ЛІКУВАННЯ, РЕЦЕПТ ЛІКІВ, ПАЦІЄНТ.

Медицина завжди була невід’ємною частиною нашого життя. Лікарі щодня рятують тисячі людей. При цьому, щоб стати лікарем, треба знати усі хвороби та їх симптоми, і як їх лікувати. Пам’ятати все це дуже важко, тому лікарі часто користуються медичними довідниками. Але у вік цифрових технологій ми можемо зробити програму, яка буде цією довідкою, і, що більше, намагатиметься сама ставити діагноз, що, безумовно допоможе лікарям.

Так само якщо зробити запис до лікаря у програмі, більше не буде необхідності йти для цього аж у лікарню або дзвонити.

Отже, такий додаток зможе використовувати як сам лікар, так і його пацієнти, що покращить їх зв’язок та дозволить лікарю, наприклад, досліджувати статистику захворювань серед його пацієнтів.

Мета: Спроектувати схему бази даних для предметної області «Довідник лікаря», на її основі розробити програму, яка буде реалізовувати функціонал та інтерфейс для керування даними у базі.

DATABASE, SUBJECT AREA, CASE - DESIGN MEANS, ER - DIAGRAM, PROGRAM, INTERFACE, HOSPITAL, TREATMENT, RECIPE OF MEDICINES, PATIENT.

Medicine has always been an integral part of our lives. Doctors save thousands of people every day. In order to become a doctor, you need to know all the diseases and their symptoms, and how to treat them. It is very difficult to remember all this, so doctors often use medical handbooks. However, in the age of digital technology, we can make a program that will be this reference, and, moreover, will try to diagnose itself, which will certainly help doctors.

Similarly, if you make an appointment with a doctor in the program, you will no longer need to go to the hospital or call.

Therefore, such an application can be used by both the doctor and his patients, which will improve their communication and allow the doctor, for example, to study the statistics of diseases among his patients.

Objective: To design a database scheme for the subject area «Doctor's handbook» based on it to develop a program that will implement the functionality and interface for data management in the database.

**ЗМІСТ**

[Вступ 7](#_Toc91359191)

[1 Реляційні бази даних: функціональність, принципи проектування та використання 8](#_Toc91359192)

[1.1 Історія виникнення реляційних БД 8](#_Toc91359193)

[1.2 Властивості реляційних БД 9](#_Toc91359194)

[1.3 Принципи використання реляційних БД 10](#_Toc91359195)

[1.4 Системи керування базами даних 12](#_Toc91359196)

[1.5 Можливості мови SQL 13](#_Toc91359197)

[1.6 Постановка задачі 15](#_Toc91359198)

[2 Проектування бази даних для ПО "Довідник лікаря" 16](#_Toc91359199)

[2.1 Специфікація вимог 16](#_Toc91359200)

[2.2 Розробка бізнес-правил для ПО: 17](#_Toc91359201)

[2.3 Побудова структури реляційної БД для ПО 18](#_Toc91359202)

[2.3.1 Логічна ER – діаграма ПО 18](#_Toc91359203)

[2.3.2 Перевірка на відповідність реляційній моделі даних 21](#_Toc91359204)

[2.4 Побудова моделі даних за допомогою Erwin 2020 21](#_Toc91359205)

[2.5 Фізичне проектування моделі даних за допомогою SQL Server 19 24](#_Toc91359206)

[2.6 Запити SQL для створення структури БД 26](#_Toc91359207)

[3 Проектування та розробка іс для предметної області 34](#_Toc91359208)

[3.1 Розробка дизайну користувача 34](#_Toc91359209)

[3.2 Реалізація необхідної функціональності для ІС 37](#_Toc91359210)

[3.2.1 Групування функціональностей ІС за принципом роботи 37](#_Toc91359211)

[3.2.2 Реалізація отримання даних та статистики 37](#_Toc91359212)

[3.2.3 Запити додавання даних 40](#_Toc91359214)

[Висновки 42](#_Toc91359218)

[Перелік посилань 43](#_Toc91359219)

[Додаток А Ілюстрація роботи ІС 44](#_Toc91359220)

[Додаток Б Завдання на курсову роботу 48](#_Toc91359221)

[Додаток В Інструкція користувача 50](#_Toc91359222)

# ВСТУП

Майже всі програми створюються для роботи з даними. При цьому дані мають десь зберігатись. Існує багато способів зберігання, але найрозповсюдженішим і найзручнішим є база даних.

База даних – це організована структура, яка призначена для зберігання, зміни та обробки взаємозалежної інформації, переважно великих обсягів. Це певний набір даних, які пов'язані між собою спільною ознакою або властивістю, та впорядковані, наприклад, за алфавітом. Головною перевагою БД є швидкість внесення та використання потрібної інформації. Завдяки спеціальним алгоритмам, які використовуються для баз даних, можна легко знаходити необхідні дані всього за декілька секунд. Також в базі даних існує певний взаємозв'язок інформації: зміна в одному рядку може спричинити зміни в інших рядках — часто це допомагає працювати з інформацією.

В довіднику лікаря мають зберігатись дані про всі хвороби і ліки, такий великий об’єм інформації зберігати не в базі дані дуже складно і неефективно. До того ж, потрібно також зберігати інформацію про всіх пацієнтів та про кожен прийом.

З базою даних все це різноманіття об’єктів та їх дані легко зберігати, а також отримувати необхідну частину інформації зі всієї купи, наприклад, лише про одного пацієнта.

# Реляційні бази даних: функціональність, принципи проектування та використання

## Історія виникнення реляційних БД

У перших відомих письмових свідоцтвах (приблизно 4000 р. до н.е.) описується облік царської скарбниці та податків у Шумері. Протягом наступних шести тисяч років спостерігається еволюція від глиняних таблиць до папірусу, потім до пергаменту і, нарешті, до паперу. Все це полегшувало зберігання даних. Було багато нововведень у поданні даних: бібліотеки, книги, паперові та друковані видання. Це були великі досягнення, але обробка інформації робилася вручну. [[3]](#_Перелік_посилань)

Вперше автоматизована обробка інформації з'явилася приблизно в 1800 році, коли Джеквард Лум (Jacquard Loom) почав розкроювати тканини за зразками, представленими перфокартами. Пізніше аналогічна технологія використовувалася в механічних піаніно. У 1890 р. Холлеріт (Herman Hollerith) використав технологію перфокарт для виконання перепису населення Сполучених Штатів. Його система містила запис для кожної сім'ї. Кожен запис даних представлявся як двійкова структура на перфокарті. Машини зводили підрахунки до таблиць за житловими кварталами, територіальними та адміністративними округами і штатами.

До 1955 року у багатьох компаній були цілі поверхи, призначені для зберігання перфокарт, це було багато в чому схоже на те, як у шумерських архівах зберігалися глиняні таблиці (рис 1.1). На інших поверхах розміщувалися шеренги перфораторів, сортувальників та табуляторів. Ці машини програмувалися шляхом перемонтування керуючих панелей, які керували деякими регістрами-накопичувачами та вибірково відтворювали карти на інших картах чи папері. Великі компанії обробляли та виробляли мільйони записів щоночі. Це було б неможливим при використанні ручних методів обробки.



Рис. 1.1 - 5 Мб даних або 62500 перфокарт

Електронні комп'ютери зі збереженими програмами були розроблені в 1940-х і на початку 1950-х років для виконання наукових та чисельних обчислень. Приблизно водночас компанія Univac розробила апаратуру магнітних стрічок, кожна з яких могла зберігати стільки інформації, скільки зберігалося в десяти тисячах перфокарт. Нові комп'ютери могли обробляти сотні записів за секунду, і їх можна було розмістити на меншій площі, ніж попереднє обладнання.

Ключовим компонентом цієї нової технології було програмне забезпечення. Було порівняно легко програмувати та використовувати комп'ютери. Було набагато простіше сортувати, аналізувати та обробляти дані із застосуванням таких мов, як COBOL та RPG. Почали з'являтися стандартні пакети для таких загальновживаних бізнес-додатків, як загальна бухгалтерія, розрахунок заробітної плати, ведення інвентарних відомостей та банківська діяльність.

Системи пакетної обробки транзакцій зберігали транзакції на картах або стрічках і збирали в пакети для подальшої обробки. Щодня ці пакети транзакцій сортувалися. Відсортовані транзакції зливалися зі збереженою на стрічці набагато більшою за розмірами базою даних (основним файлом) для нового основного файлу.

Пакетна обробка дозволяла дуже ефективно використовувати комп'ютери, але мала два серйозні обмеження. Якщо в транзакції була помилка, вона не розпізнавалася до вечірньої обробки основного файлу і могло знадобитися кілька днів для виправлення транзакції. Більш важливим є те, що бізнес не знав поточного стану бази даних, оскільки транзакції реально не оброблялися до ранку.

З кінця 1950-х лідируючі компанії з кількох областей промисловості почали вводити у використання системи баз даних з оперативними транзакціями; транзакції над оперативними базами даних оброблялися інтерактивному режимі. Оперативні бази даних зберігалися на магнітних дисках або барабанах.

1970 року у статті Теда Кодда (E.F. Codd) була змальована реляційна модель, що була альтернативою низькорівневому навігаційному інтерфейсу. Ідея реляційної моделі у тому, щоб одноманітно представляти і сутності, і зв'язки. Реляційна модель даних мала уніфіковану мову для визначення даних, навігації за даними та маніпулювання даними, а не окремими мовами для кожного з цих завдань. Ще важливіше те, що реляційна алгебра має справу з множинами записів (відносинами) як єдиним цілим, застосовуючи операції до множин записів цілком і роблячи множини записів в результаті.

Надихаючі ідеями Кодда протягом 1970-х дослідники з академії та індустрії експериментували з цим новим підходом до структуризації баз даних та забезпечення доступу до них, що обіцяло більш просте моделювання даних та прикладне програмування. Багато реляційних прототипів, розроблених протягом цього періоду, зійшлися на загальній моделі та мові. Дослідження в IBM Research, очолювані Тедом Коддом, Реймондом Бойсом (Raymond Boyce) та Доном Чемберліном (Don Chamberlin), та робота в Каліфорнійському університеті м. Берклі, якою керував Майкл Стоунбрейкер (Michael Stonebraker), породили мову SQL. Ця мова була вперше стандартизована у 1985 році. Сьогодні багато систем баз даних забезпечують інтерфейс SQL. Крім того, у всіх системах підтримуються власні розширення, що виходять за межі цього стандарту.

Реляційна модель виявилася добре придатною до використання в архітектурі клієнт-сервер, до паралельної обробки та графічних інтерфейсів користувача і до сьогодення є найбільш популярною.

## Властивості реляційних БД

Реляційні бази даних основані на реляційній моделі даних. Реляційна модель надає засоби опису даних на основі тільки їх природної структури, тобто без потреби введення додаткових структур для цілей машинного представлення. Іншими словами, подання даних не залежить від способу їх фізичної організації. Сама назва «реляційна» походить від англійського relation — «відношення». [[4]](#_Перелік_посилань)

До складу реляційної моделі даних зазвичай включають теорію нормалізації. Визначають три складові частини реляційної моделі даних:

* + - Структурна – усі елементи бази даних є відношеннями
    - Маніпуляційна – модель підтримує оператори маніпуляції з даними (реляційна алгебра, реляційне обчислення).
    - Цілісна – модель даних має відповідати умовам цілісності.

Відношення – фундаментальне явище у реляційних базах даних. Зобразити його можна у вигляді таблиці, де стовпці будуть атрибутами, а строки кортежами.

Зв'язки між різними таблицями встановлюються за допомогою зовнішніх ключів. Типи зв’язків, які можуть існувати між відношеннями (таблицями):

* Один до одного
* Один до багатьох
* Багато до багатьох

## Принципи використання реляційних БД

Робота з реляційними базами даних відбувається за допомогою засобів реляційної алгебри та реляційного обчислення (маніпуляційна частина).

Реляційна алгебра — відгалуження логіки першого порядку, множина відношень замкнених операторами. Оператори застосовуються до відношень, в результаті застосування отримується нове відношення. Реляційна алгебра заснована на теорії множин та деяких спеціальних операторах над відносинами. [[5]](#_Перелік_посилань)

Реляційне числення використовується з непроцедурною декларативною мовою запиту, в якій користувач не переймається процедурою отримання результатів. Користувач надає лише вимоги, а вихід надається без розуміння техніки пошуку. [[6]](#_Перелік_посилань)

В реляційній алгебрі Кодда визначено такі шість примітивних операторів: вибірка, проєкція, декартів добуток, об'єднання та різниця і перейменування. Шість операторів є фундаментальними в тому сенсі, що жоден із них не можна відкинути без втрати потужності. Багато інших операторів було визначено комбінацією цих шести. Серед найважливіших можна назвати: перетин множин, ділення та природне об'єднання.

Робота з базами даних можлива у фізичному та електронному варіанті. Фізично ми можемо прописувати формули реляційної алгебри, наприклад на папері та розуміти, які кортежі з відношень ми маємо отримати.

Другий варіант – електронний. У такому випадку маніпуляцій з даними виконується за допомогою систем керування базами даних (СУБД), які реалізують формули на певних мовах програмування.

## Системи керування базами даних

Система управління (керування) базами даних (СУБД) – це набір взаємопов'язаних даних (база даних) і програм для доступу до цих даних. СУБД дозволяє створювати, зберігати, організовувати, контролювати та адмініструвати інформацію в базі даних [[1]](#_Перелік_посилань).

СУБД можуть бути реляційними, ієрархічними, мережевими, об'єктно-орієнтованими, тощо. На рисунку 1.2 представлені приклади реляційних СУБД.

На сьогоднішній день реляційні бази даних, а відповідно й реляційні СУБД, є найбільш використовуваними у світі.

Усі розробники реляційних СУБД мають дотримуватись «12 правил Кодда», правил, які сформулював автор реляційної моделі Едгар Кодд у 1985 році.



Рисунок 1.2 – Приклади реляційних СУБД

## Можливості мови SQL

Мова SQL є невід’ємною частиною будь-якої СУБД, інструментом, що може поєднати усі її частини разом.

SQL – це мова інтерактивних запитів, мова програмування та адміністрування баз даних. Користувачі вводять команди SQL в інтерактивному режимі для вибірки даних та відображення їх на екрані, а також внесення змін до БД. Адміністратор БД використовує SQL для визначення структури бази даних та керування доступом до інформації.

Відповідно до загальноприйнятого стилю програмування, оператори (та інші зарезервовані слова) в SQL завжди слід писати великими літерами.

Оператори SQL поділяються на:

* + 1. оператори визначення даних (Data Definition Language, DDL);
       1. CREATE - створює об'єкти БД (саму базу, таблицю, подання, користувача і т. д.);
       2. ALTER - змінює об'єкти;
       3. DROP - видаляє об'єкти;
    2. оператори маніпуляції даними (Data Manipulation Language, DML);
       1. SELECT - читає дані, що задовольняють заданим умовам (DQL);
       2. INSERT - додає нові дані;
       3. UPDATE - змінює існуючі дані;
       4. DELETE - видаляє дані;
    3. оператори визначення доступу до даних (Data Control Language, DCL);
       1. GRANT - надає користувачеві (групі користувачів) дозволи на певні операції з об'єктом;
       2. REVOKE - відкликає раніше видані дозволи;
       3. DENY - задає заборону, яка завжди має пріоритет над дозволом;
    4. оператори управління транзакціями (Transaction Control Language, TCL);
       1. COMMIT - застосовує транзакцію;
       2. ROLLBACK - відкатує всі зміни, зроблені в контексті поточної транзакції;
       3. SAVEPOINT - ділить транзакцію на дрібніші ділянки.

Оператор SQL складається з зарезервованих слів і назв користувача.

Зарезервовані слова є постійною частиною SQL і мають фіксоване значення. Їх слід записувати точно так, як це встановлено.

Назви користувача - слова, що задаються користувачем відповідно до синтаксичних правил і є ідентифікаторами або іменами різних об'єктів бази даних.

Синтаксичні правила для назв користувача:

* + - включає малі та великі літери латинського алфавіту (A-Z, a-z), цифри (0-9) та символ підкреслення (\_);
    - починається з літери;
    - може мати довжину до 128 символів;
    - не може містити пробіли.

## Постановка задачі

Отже, необхідно розробити реляційну модель для предметної області «Довідник лікаря», дотримуючись заданих бізнес-правил та вимог реляційної моделі даних.

На основі розробленої моделі створити фізичну базу даних за допомогою СУБД SQL Server.

На базі БД розробити програму, написану мовою програмування С#, яка реалізуватиме графічний інтерфейс для взаємодії із базою даних.

Програма має реалізовувати наступні задачі:

* записування пацієнта на прийом до лікаря;
* виведення рецепту ліків для пацієнта;
* виведення інформації про всі прийоми для лікаря;
* виведення статистики по усім прийомам у вигляді діаграм для лікаря.

# Проектування бази даних для ПО "Довідник лікаря"

## Специфікація вимог

Основні можливості системи:

* + - можливість увійти у ролі лікаря або пацієнта;
    - реалізовувати можливості, які надаються кожному з них.

можливості пацієнта:

* + - записатись до лікаря;
    - отримати рецепт ліків на основі опису симптомів своєї хвороби.

можливості лікаря:

* + - передивлятись інформацію про кожний прийом
    - передивлятись статистику:

1. за захворюваннями – 5 найчастіших захворювань серед пацієнтів лікаря;
2. за пацієнтами – 5 пацієнтів, що звертаються за допомогою найчастіше;
3. за медикаментами – 5 препаратів, що найчастіше потрапляють у рецепти.

## Розробка бізнес-правил для ПО

Функціонал розробленої системи має працювати враховуючи бізнес-правила.

Бізнес-правила для предметної області «Довідник лікаря»:

1. Лікар має пацієнтів;
2. У пацієнта, хвороби, медикаменту та прийому унікальні коди у базі;
3. Пацієнти можуть записуватись на прийом до лікаря багато разів;
4. Пацієнт характеризується іменем та номером телефону, тому пацієнти з однаковими іменами та різними номерами вважаються різними людьми;
5. Діагноз ставиться автоматично, основуючись на кількості співпадінь симптомів хвороби, записаних у довіднику, та симптомі, описаних пацієнтом у скарзі;
6. Рецепт формується для визначеної діагнозом хвороби;
7. Для кожного медичного препарату формується окремий рецепт.

## Побудова структури реляційної БД для ПО

### Логічна ER – діаграма ПО

Схема "сутність-зв'язок" (ER-діаграма) - це різновид блок-схеми, де показано, як різні "сутності" (люди, об'єкти, концепції тощо) пов'язані між собою всередині системи.

Сутності зображуються прямокутником, який містить ім'я сутності. Атрибути сутності зображуються у вигляді овалу, з'єднаного з відповідним прямокутником. Назви ключових атрибуті підкреслюються. Зв'язки між сутностями позначаються ромбом. Учасники зв'язку зв’язані з ромбом лінією. Для позначення типу зв'язку використовуються символи "1" та "М" (один та багато). Також для цього можуть використовуватись одинарні та подвійні стрілки.

На рисунку 2.1 наведена логічна ER – діаграма предметної області. На рисунку 2.2 наведена ER - модель бази даних у нотації UML.

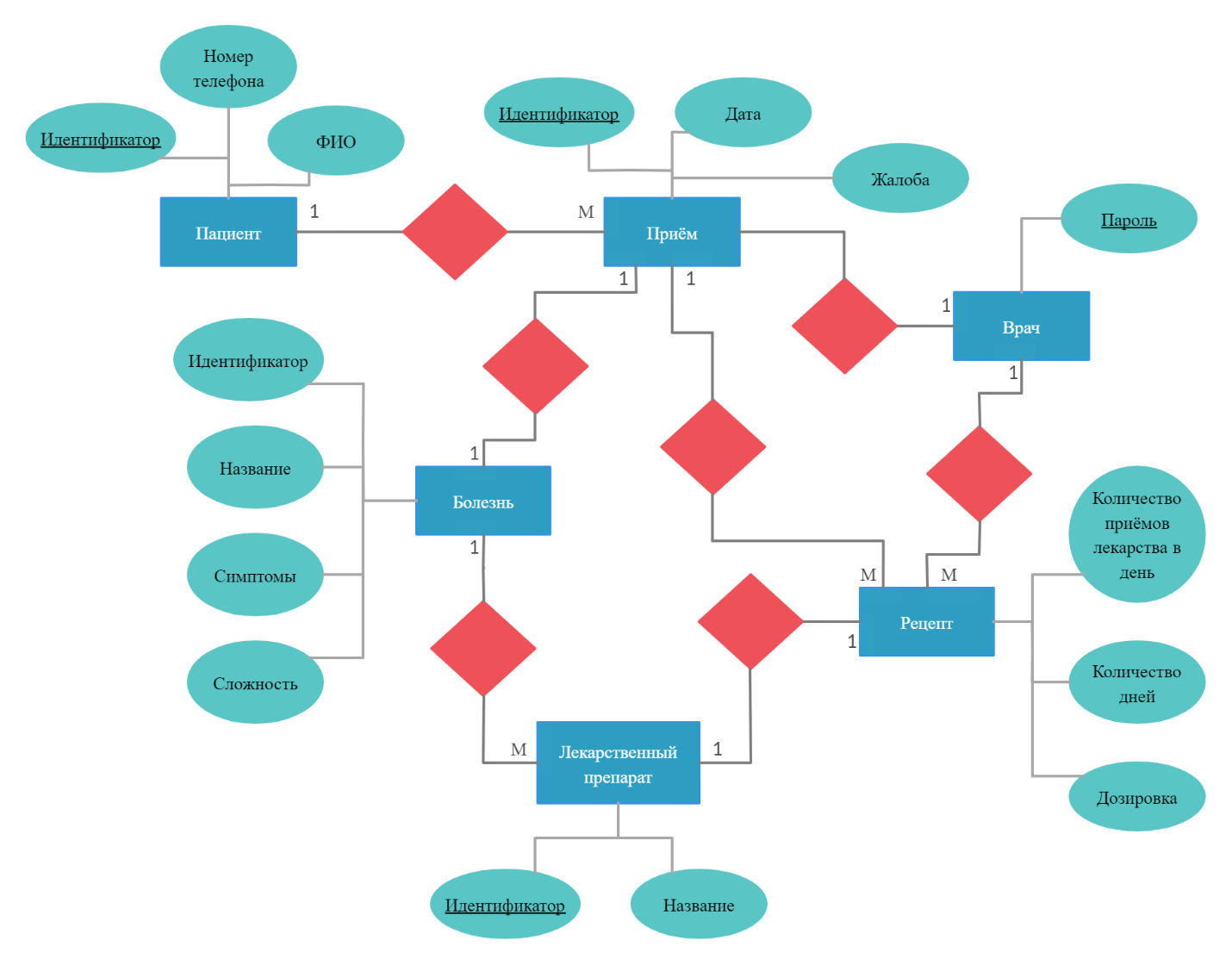


Рисунок 2.1 – Логічна ER – діаграма ПО

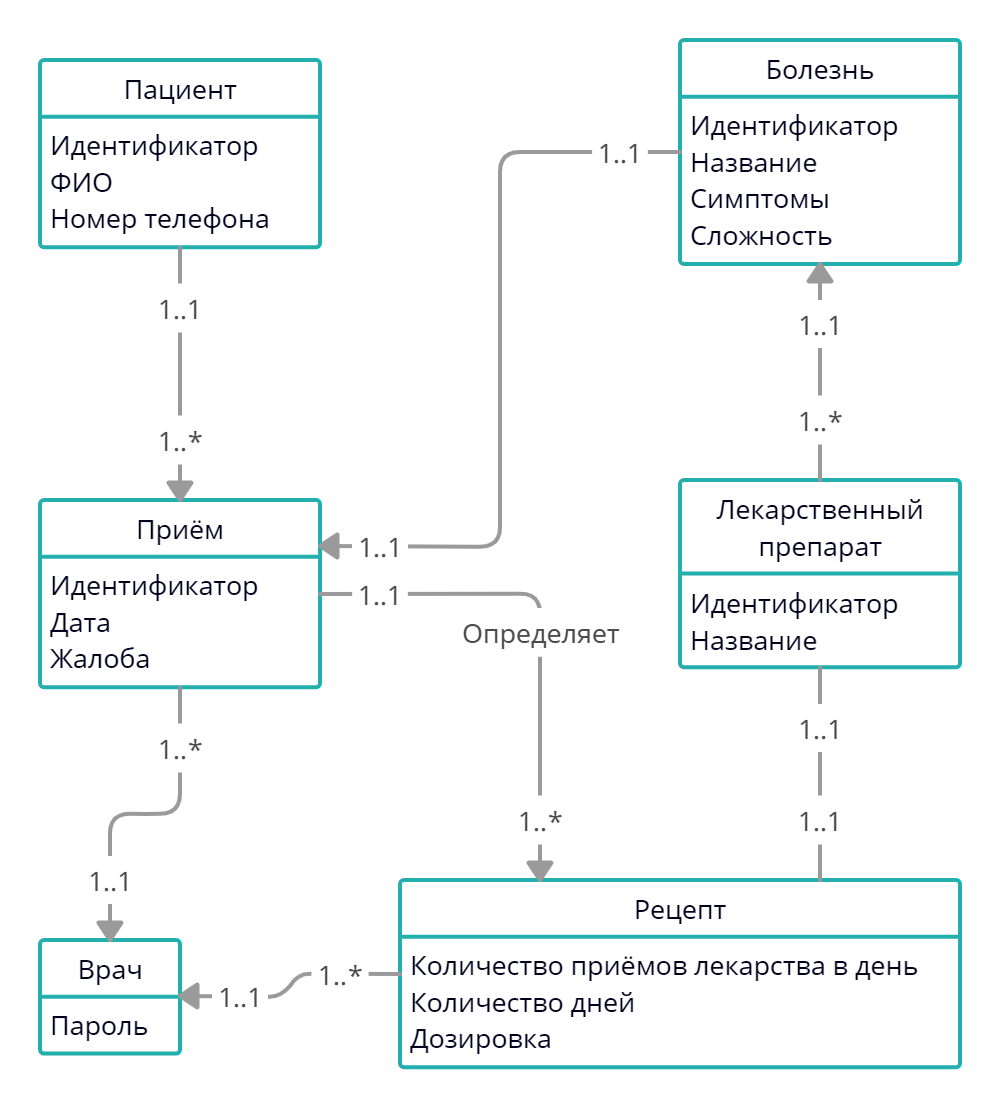


Рисунок 2.2 – UML– діаграма ПО

### Перевірка на відповідність реляційній моделі даних

Перетворення двосторонніх зв'язків типу M:N – відсутні.

Перетворення зв'язків із атрибутами – відсутні.

Перетворення складних зв'язків – відсутні.

Перетворення багатозначних атрибутів – відсутні.

Перетворення рекурсивних зв'язків M:N – відсутні.

Модель бази даних відповідає потребам третьої нормальної форми [[2]](#_Перелік_посилань).

Додатковий аналіз:

Перевірка зв'язків типу 1:1 – не утворюють пасток.

Аналіз рекурсивних зв'язків 1:1- відсутні.

## Побудова моделі даних за допомогою Erwin 2020

На рисунках 2.3, 2.4 представлені логічна та фізичні моделі побудови моделі даних за допомогою Case-засобу візуального проектування Erwin 2020.

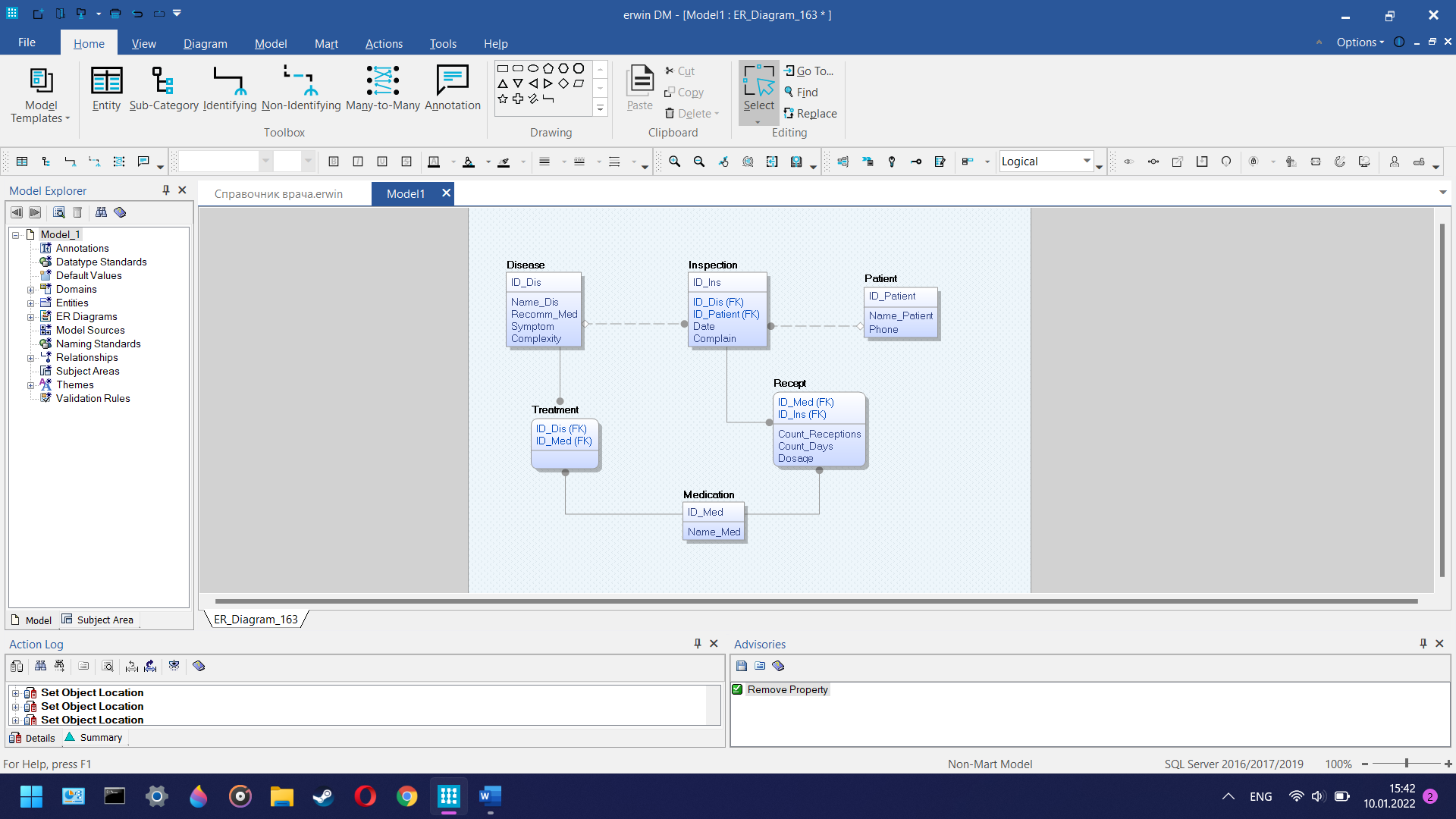


Рисунок 2.3 – Логічна модель даних

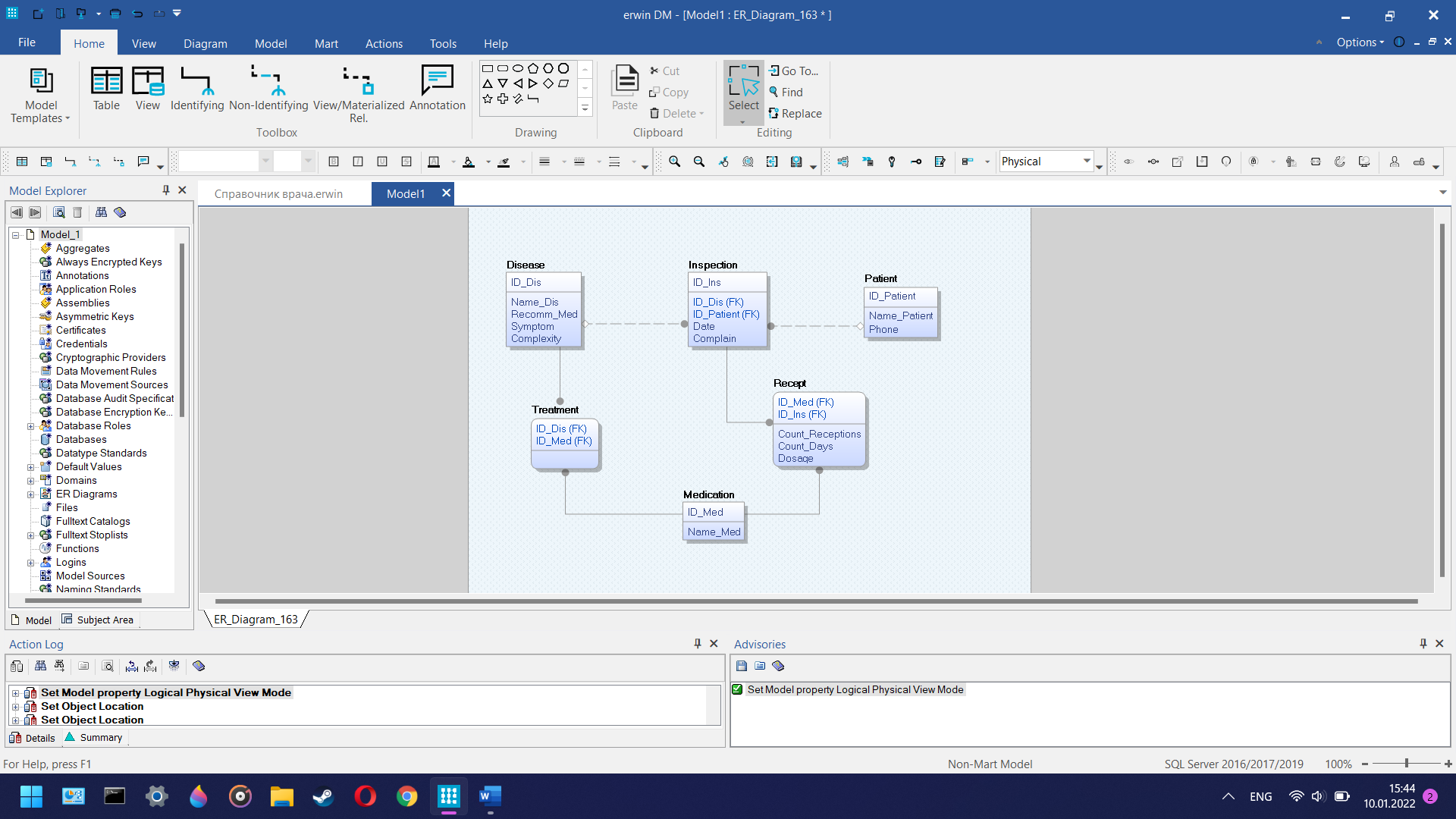


Рисунок 2.4 – Фізична модель даних

## Фізичне проектування моделі даних за допомогою SQL Server 2019

Щоб продемонструвати структуру бази даних, найкраще підходить графічне зображення у вигляді діаграми бази даних. SQL Server автоматично генерує таку діаграму. На ній видно всі таблиці, їх стовпці, а також ключі зі зв’язками.

Атрибути, що є первинними ключами, позначаються значком ключа, а при наявності зовнішніх ключів можна побачити зв’язок між таблицями.

На рисунку 2.5 представлена схема бази даних, що згенерована за допомогою СУБД SQL Server 2019.



Рисунок 2.4 – схема бази даних, згенерована за допомогою СУБД SQL Server 2019

## Запити SQL для створення структури БД

У лістингу 2.1 наведено скрипт для створення структури БД у SQL Server з заповненням таблиць хвороб та ліків та лікування.

Лістинг 2.1 - Запити DDL та DML SQL

USE DoctorsHandbook

create table Disease (

ID\_Dis int identity(1,1) primary key,

Name\_Dis varchar(50),

Recomm\_Med varchar(50),

Symptom varchar(100),

Complexity int

CONSTRAINT [R\_1] CHECK (Complexity BETWEEN 1 AND 3)

)

create table Patient (

ID\_Patient int identity(1,1) primary key,

Name\_Patient varchar(50),

Phone varchar(13)

)

create table Medication (

ID\_Med int identity(1,1) primary key,

Name\_Med varchar(70)

)

CREATE TABLE Treatment (

ID\_Med INT NOT NULL,

ID\_Dis INT NOT NULL,

CONSTRAINT [XPKTreatment] PRIMARY KEY CLUSTERED (ID\_med ASC, ID\_Dis ASC),

CONSTRAINT [R\_2] FOREIGN KEY (ID\_Dis) REFERENCES Disease (ID\_Dis),

CONSTRAINT [R\_3] FOREIGN KEY (ID\_Med) REFERENCES Medication (ID\_Med)

)

CREATE TABLE Inspection (

ID\_Ins INT identity(1,1) NOT NULL,

ID\_Patient INT NULL,

ID\_Dis INT NOT NULL,

[Date] DATETIME NULL,

Complain text NULL,

CONSTRAINT [XPKInspection] PRIMARY KEY CLUSTERED (ID\_Ins ASC),

CONSTRAINT [R\_4] FOREIGN KEY (ID\_Dis) REFERENCES Disease (ID\_Dis),

CONSTRAINT [R\_5] FOREIGN KEY (ID\_Patient) REFERENCES Patient (ID\_Patient)

);

CREATE TABLE Recept (

ID\_Ins INT NOT NULL,

ID\_Med INT NOT NULL,

Count\_receptions INT NULL,

Count\_days INT NULL,

Dosage INT NULL,

CONSTRAINT [XPKRecept] PRIMARY KEY CLUSTERED (ID\_Ins ASC, ID\_Med ASC),

CONSTRAINT [R\_6] FOREIGN KEY (ID\_Ins) REFERENCES Inspection (ID\_Ins),

CONSTRAINT [R\_7] FOREIGN KEY (ID\_Med) REFERENCES Medication (ID\_Med),

CONSTRAINT [R\_8] CHECK (Dosage BETWEEN 1 AND 3),

CONSTRAINT [R\_9] CHECK (Count\_days BETWEEN 7 AND 21),

CONSTRAINT [R\_10] CHECK (Count\_receptions BETWEEN 1 AND 4)

);

INSERT INTO Medication

VALUES

('Амиксин'),

('Атоксил гель'),

('Ветрянка стоп'),

('Гель антисептический'),

('Герпевир'),

('Глицин'),

('Ибупрофен'),

('Мультигрип'),

('Нокспрей'),

('Нурофен'),

('Повязка гидрогелевая'),

('Салфетка противоожог'),

('Селен'),

('Стрепсилс'),

('Сустафор'),

('Флуцар-Дарница'),

('Цитрамон-дарница'),

('Черный уголь'),

('Викс Актив'),

('Терафлю'),

('Линекс'),

('Нимесил'),

('Эргоферон'),

('Орасепт'),

('Адвантан'),

('Проспан'),

('Карнитин'),

('Витамиа A'),

('Витамин B'),

('Витамин C'),

('Витамин D');

INSERT INTO Disease

VALUES

('ОРВИ', 'Амиксин', 'боль в горле/температура/насморк', 1),

('Легкая краснуха', 'Ибупрофен', 'сыпь/чесотка', 1),

('Тяжелая краснуха', 'Каринитин', 'сыпь/температура/чесотка', 2),

('Грипп', 'Мультигрип', 'головная боль/кашель/насморк/температура', 2),

('Легкое отравление', 'Черный уголь', 'тошнота/боль в желудке', 1),

('Тяжелое отравление', 'Атоксил гель', 'тошнота/боль в животе/понос/температура', 2),

('Воспаление суставов', 'Сустафор', 'боль в колене/боль в локте/боль в кисти', 1),

('Ветрянка', 'Ветрянка стоп', 'сыпь/чесотка', 1),

('Герпес на губах', 'Герпевир', 'белые пузырики на губах', 1),

('Герпес на теле', 'Селен', 'белые пузырики на теле', 2),

('Лишай', 'Флуцар-Дарница', 'пятно на теле', 1),

('Бессоница', 'Глицин', 'плохой сон/усталость/головная боль', 1),

('Ожог первой степени', 'Салфетка противоожог', 'красное пятно', 1),

('Ожог второй степени', 'Повязка гидрогелевая', 'красное пятно/боль', 2),

('Ожог третей степени', 'Гель антисептический', 'красное пятно/боль/температура', 3);

INSERT INTO Treatment

VALUES

(1, 1),

(2, 6),

(3, 8),

(4, 15),

(5, 9),

(6, 12),

(7, 2),

(8, 4),

(9, 1),

(9, 4),

(10, 1),

(10, 3),

(10, 4),

(10, 6),

(10, 7),

(11, 14),

(12, 13),

(13, 10),

(14, 1),

(15, 7),

(16, 2),

(16, 3),

(16, 8),

(16, 11),

(17, 4),

(17, 12),

(18, 5),

(19, 1),

(19, 4),

(20, 1),

(21, 5),

(21, 6),

(22, 5),

(22, 6),

(22, 7),

(22, 14),

(22, 15),

(23, 4),

(24, 1),

(25, 6),

(26, 4),

(27, 2),

(27, 3),

(27, 8),

(27, 9),

(27, 10),

(27, 12),

(27, 15),

(28, 2),

(28, 3),

(28, 8),

(28, 9),

(28, 10),

(28, 11),

(28, 13),

(28, 14),

(28, 15),

(29, 5),

(29, 6),

(29, 7),

(29, 8),

(29, 9),

(29, 10),

(29, 11),

(30, 1),

(30, 4),

(31, 7);

# Проектування та розробка ДОДАТКУ для предметної області

## Розробка дизайну користувача

Програма написана на мові програмування C# [[4]](#_Перелік_посилань) у середовищі розробки Visual Studio 2022 з використанням інтерфейсу програмування додатків Windows Forms. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows, тому з ним дуже легко розробляти додатки з інтерфейсом.

Для роботи з базою даних було використано бібліотеку Microsoft.Data.SqlClient, яка дозволяє відкрити з’єднання з базою даних на конкретному сервері, після чого у ній можна виконати будь-який запит.

## Реалізація необхідної функціональності для додатка

### Групування функціональностей додатка за принципом роботи

Усі функції програми, направлені на оборку даних, звертаються до бази даних, використовуючи відповідні запити. Ці функції можна розділити на дві групи.

Перша група – це функції отримання даних. За допомогою їх можна

отримати дані для формування рецепту та виведення статистичних діаграм й інформації про всі прийоми.

Друга група – функції додавання даних, з ними можна додавати до бази даних нові об’єкти, такі як, прийом, пацієнт або рецепт.

### Реалізація отримання даних та статистики

Об’єкти Прийом та Рецепт мають свої відповідні класи у коді, до яких зчитуються усі дані із запиту. Якщо дані мають відповідати певним умовам, то у запиті додається оператор WHERE, який вибирає дані за відповідним критерієм. Якщо ж треба відсортувати їх, то використовується групування GROUP BY після чого сортування ORDER BY. Приклади запитів наведено на лістингах 3.1 – 3.4:

Лістинг 3.1 – Отримати дані про всі прийоми (ідентифікатор, ім’я пацієнта, назва хвороби, дата прийому та скарга пацієнта). Так як таблиці зв’язані за ідентифікаторами, можна легко отримати ім’я пацієнта та назву хвороби, використавши операцію з’єднання таблиць JOIN за цими ідентифікаторами. Також з повних дати та часу беремо лише дату певному форматі за допомогою операції FORMAT.

SELECT Inspection.ID\_Ins, Patient.Name\_Patient, Disease.Name\_Dis, FORMAT(Inspection.Date, N'dd.MM.yyyy') AS Date, Inspection.Complain

FROM Inspection

JOIN Patient ON (Inspection.ID\_Patient = Patient.ID\_Patient)

JOIN Disease ON (Inspection.ID\_Dis = Disease.ID\_Dis)

Лістинг 3.2 – Отримати кількість захворювань кожною хворобою та відсортувати за спаданням.

SELECT Disease.Name\_Dis, COUNT(Disease.Name\_Dis) AS Count

FROM Inspection

JOIN Disease ON (Inspection.ID\_Dis = Disease.ID\_Dis)

GROUP BY Disease.Name\_Dis

ORDER BY Count DESC

Лістинг 3.3 – Отримати ідентифікатори та симптоми усіх хвороб (для пошуку співпадінь між симптомами та скаргою пацієнта).

SELECT ID\_Dis, Symptom FROM Disease

Лістинг 3.4 – Отримати кількість разів, коли медикамент рекомендується та потрапляє у рецепт, та відсортувати за спаданням.

SELECT Medication.Name\_Med, COUNT(Medication.Name\_Med) AS Count

FROM Recept

JOIN Medication ON (Recept.ID\_Med = Medication.ID\_Med)

GROUP BY Medication.Name\_Med

ORDER BY Count DESC

На рисунку 3.4 наведена статистика у вигляді діаграми до запиту «Кількість разів, коли медикамент рекомендується та потрапляє у рецепт»

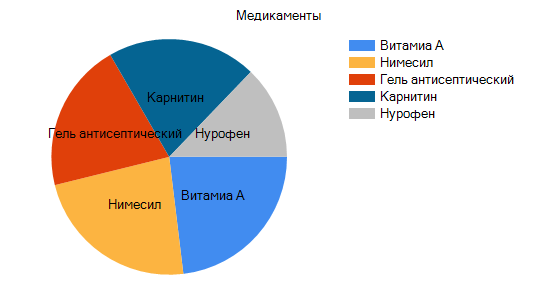


Рисунок 3.4 – Статистика до лістинга 3.4

### Запити додавання даних

Лістинги 3.5 – 3.6 показують приклади реалізації додавання даних:

Лістинг 3.5 – Додавання інформації про прийом до бази даних

INSERT INTO Inspection (ID\_Patient, ID\_Dis, Date, Complain) " +

"SELECT ID\_Patient, " + id\_dis + " AS ID\_Dis, '" + datetime.ToString("yyyy-MM-dd HH:mm:ss.fff") + "' AS Date, '" + complaint + "' AS Complain " +

"FROM Patient " +

"WHERE Name\_Patient='" + name\_patient + "' " + "AND Phone='" + phone + "'"

Лістинг 3.6 – Додавання пацієнта

INSERT INTO Patient " +

"VALUES (" + "'" + name\_patient + "'" + ", " + "'" + phone + "'" + ")"

# 

# ВИСНОВКИ

У рамках моєї курсової роботи була розроблена реляційна модель для предметної області «Довідник лікаря».

Також було розроблено додаток, в якому реалізовані такі можливості:

* + - вхід у ролі пацієнта та лікаря;
    - для пацієнта: запис до лікаря та отримання рецепту лікування;
    - для лікаря: перегляд інформації про всі свої прийоми та статистики по хворобам/пацієнтам/медикаментам.

При реалізації курсової роботи були поглиблені навички проектування реляційних моделей даних, написання SQL запитів мовами DDL, DML, програмування на мові C#. Отримано навички роботи з інтерфейсом програмування додатків Windows Forms.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Клайн, К. SQL. Довідник/К. Клайн. - 2-ге вид. - М.: КУДИЦЬ - ОБРАЗ, 2006. - 832 с.
2. Конноллі Т. Бази даних. Проектування, реалізація та супровід. Теорія та практика: пров. з англ. / Т. Конноллі, К. Біг. – 3-тє видання. - М.: Вільямс, 2003. - 1440 с.
3. Грей, Дж. Управління даними: минуле, сьогодення і майбутнє
4. Стаття про реляційні бази даних - <https://sites.google.com/site/bazidanihsubdaccess/bazi-danih-subd/ierarhicna-model-danih/relacijna-model-baz-danih>
5. Стаття про реляційну алгебру - <https://uk.wikipedia.org/wiki/Реляційна_алгебра>
6. Стаття про реляційне обчислення - https://uk.theastrologypage.com/relational-calculus

ДОДАТОК А   
  
**ІЛЮСТРАЦІЯ РОБОТИ ДОДАТКУ**

На рисунках А.1 – А.3 продемонстровано процес записування на прийом до лікаря та отримання рецепту лікування.

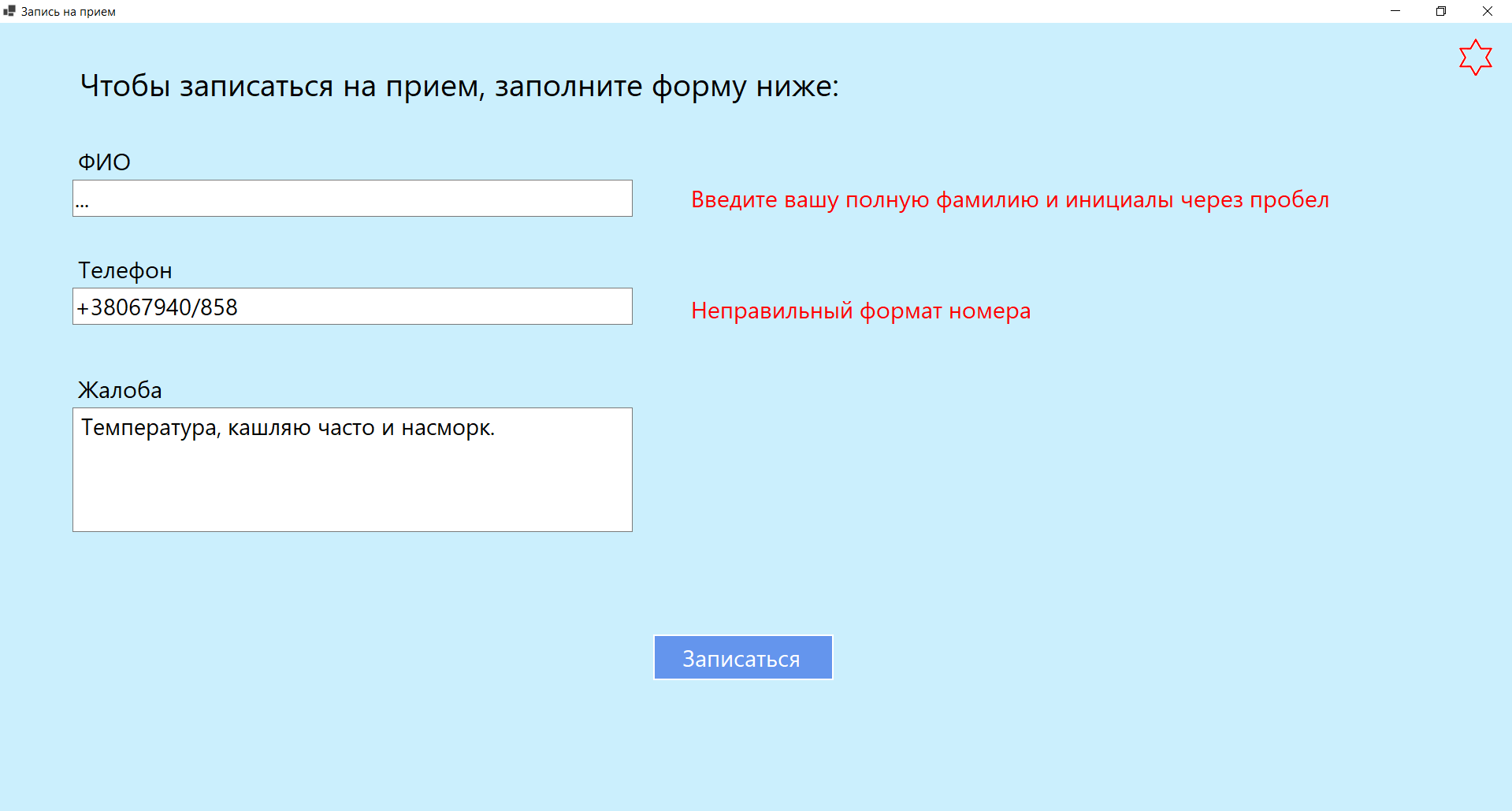


Рисунок А.1 – Неправильно введено ПІБ та телефон

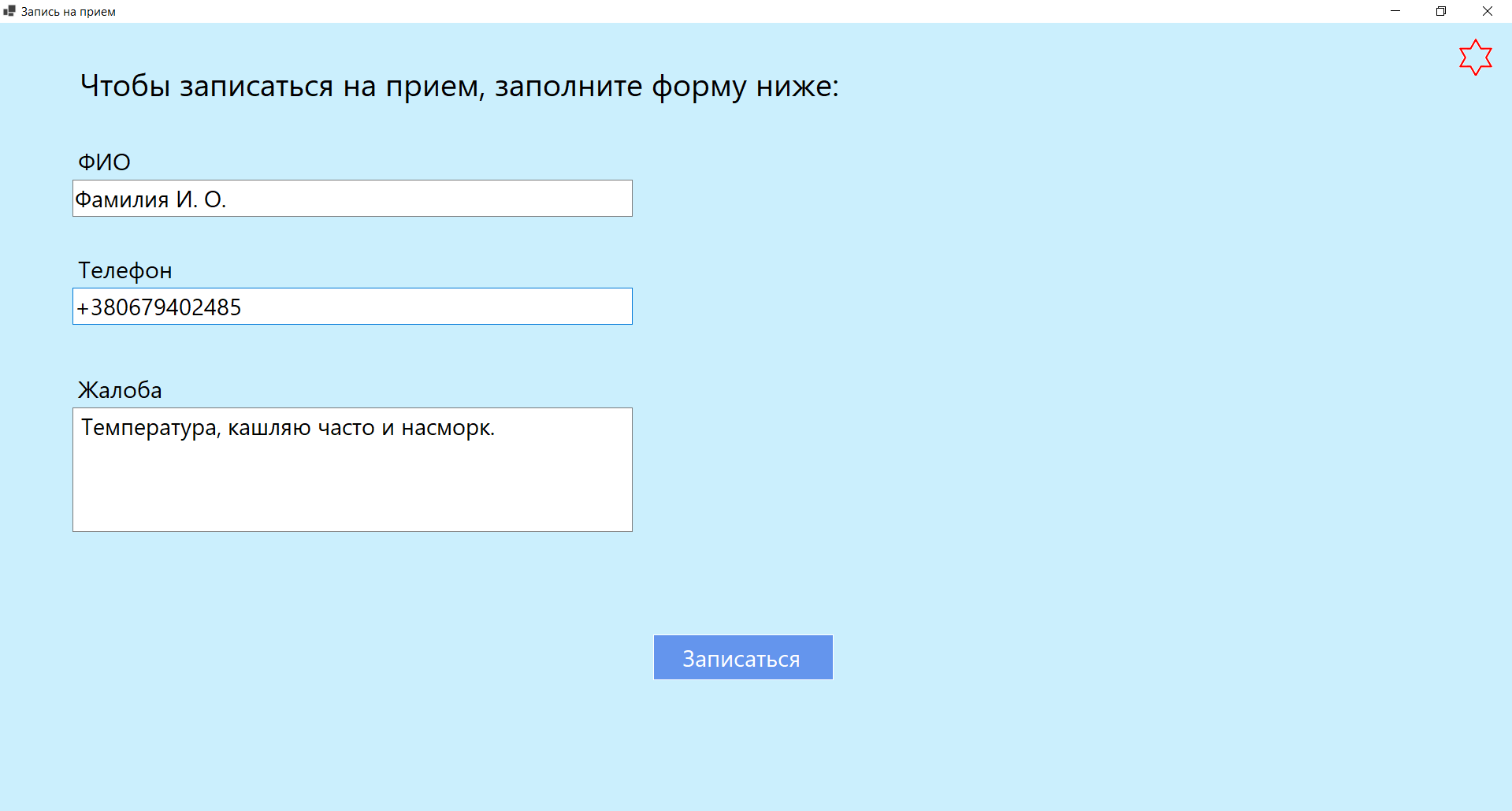


Рисунок А.2 – Дані введено правильно – автоматичне перенаправлення на сторінку з рецептом

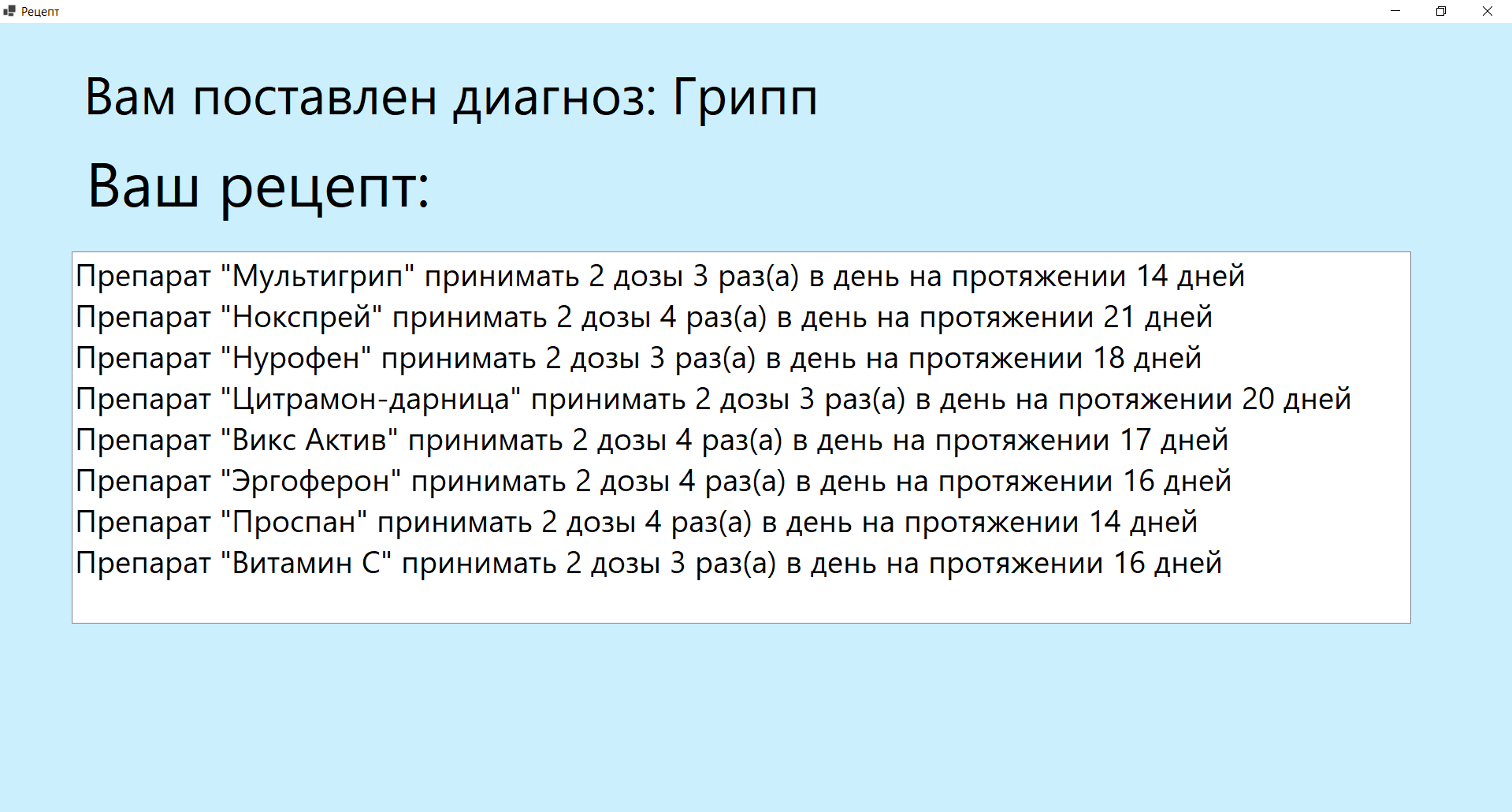


Рисунок А.3 – Отриманий рецепт

На рисунках А.4 – А.5 продемонстровано процес входу, як лікар, та сторінку лікаря

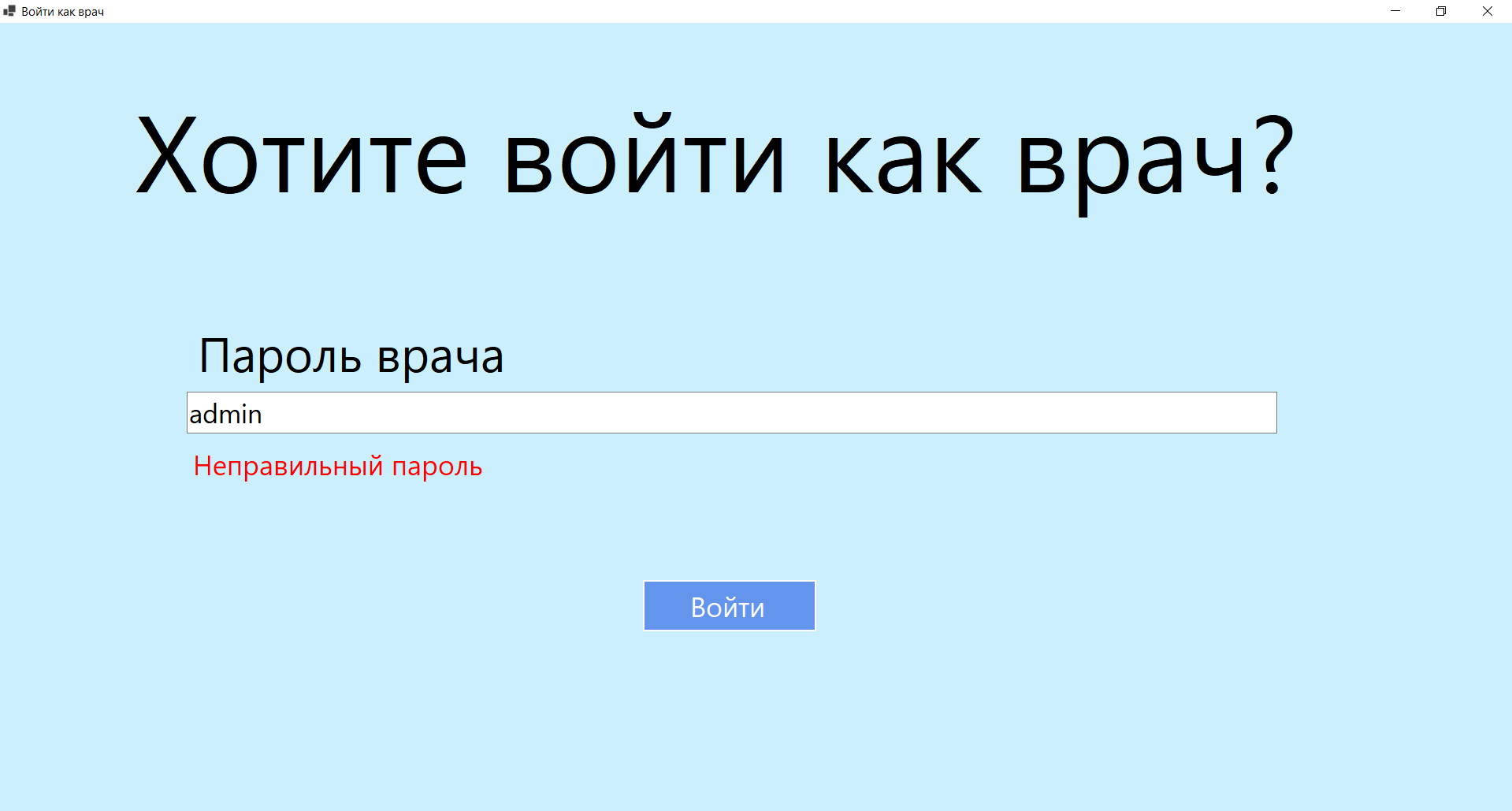


Рисунок А.4 – Введено неправильний пароль

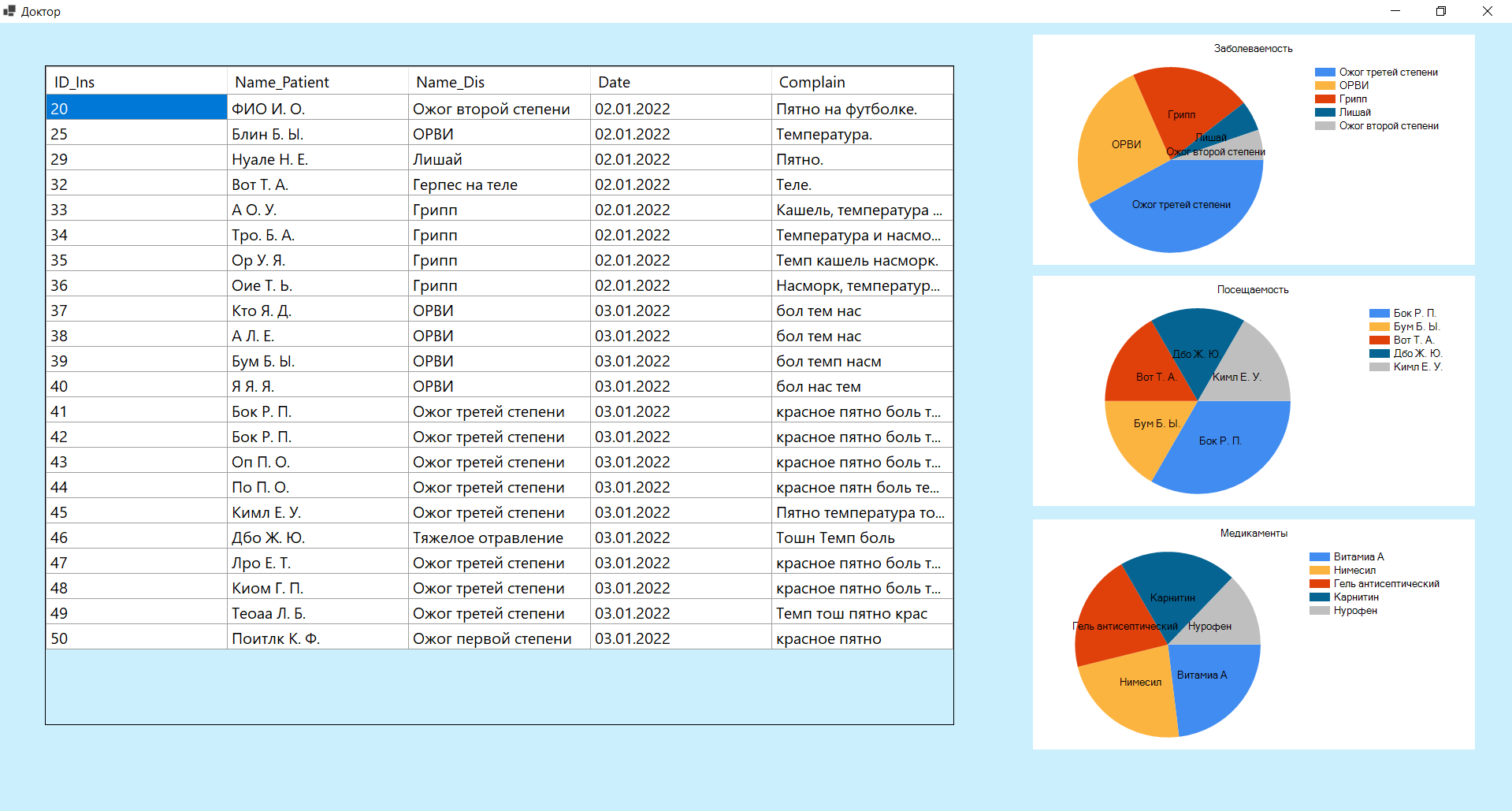


Рисунок А.5 – Після введення правильного паролю можемо побачити дані, доступні лікарю

# Додаток Б Завдання на курсову роботу

Метою курсової роботи є створення додатку для предметної області «Довідник лікаря».

Нижче наведений опис даної предметної області, потреби користувачів до даних та функціональності додатку.

У предметній області наявні такі об'єкти як лікар, пацієнт, рецепт ліків. Пацієнт може записатись до лікаря та отримати рецепт ліків, а лікар може бачити статистику за весь час своєї роботи.

Кожен пацієнт характеризується ім’ям та номером телефону. Якщо імена пацієнтів співпадають, але при цьому в них різні номери телефонів, кожен з них матиме свій ідентифікатор.

Для досягнення мети курсової роботи необхідно вирішити такі завдання:

- вивчити теорію проектування реляційних БД на основі побудови ER-моделі, основні принципи нормалізації БД і принципи побудови запитів до БД (мови DQL SQL, DML SQL, DDL SQL);

- ознайомитись з наступними інструментами проектування та програмування ІС: CASE-засобом візуального проектування даних Erwin, MS Visual Studio;

- застосувати отримані теоретичні та практичні знання для розробки ІC для ПЗ «Довідник лікаря»:

- вивчивши опис ПЗ та вимоги до функціональності ІС, розробити бізнес-правила та глосарій;

- на основі бізнес-правил розробити ER-модель та відобразити її за допомогою ER-діаграми;

- розробити модель даних за допомогою CASE-засобу візуального проектування Erwin;

- сформувати структуру БД у СУБД;

- розробити дизайн ІС;

- розробити програмний код для забезпечення необхідної функціональності ІС;

- Провести тестування ІС;

- відобразити процес та результати розробки ІС у пояснювальній записці до курсової роботи.

Для розробки ІС необхідно використовувати такі інструменти:

Erwin - CASE-засіб проектування БД;

MS SQL Server - СУБД;

Visual Studio, C# - середовище та мова розробки програмного забезпечення.

Windows Forms, Microsoft.Data.SqlClient - додаткові бібліотеки для середовища розробки.

# Додаток В Інструкція користувача

У ролі пацієнта:

Введіть ваші прізвище та ініціали у форматі Прізвище І. Б.;

Введіть ваш номер телефону;

Введіть скаргу (що не так у вас зі здоров’ям);

Натисніть «Записаться» і, якщо проблем нема, отримайте ваш рецепт.

У ролі лікаря:

Натисніть на кнопку «Увійти як лікар» (червона зірка у правому верхньому кутку:six-pointed-star)

Введіть пароль лікаря (1111) та натисніть увійти.

Можете передивлятись інформацію про всі прийоми та статистику.