СУБД «Больница»

**Содержание**

[Введение 2](#_Toc185839656)

[1 Проектирование базы данных 3](#_Toc185839657)

[1.1 Концептуальная модель 3](#_Toc185839658)

[1.2 Атрибуты, сущности и целлостность данных 6](#_Toc185839659)

[1.3 Построение физической модели 12](#_Toc185839660)

[2 Разработка приложения по работе с СУБД 20](#_Toc185839661)

[2.1 Назначение и основные функции приложения 20](#_Toc185839662)

[2.2 Структура приложения 22](#_Toc185839663)

[2.3 Тестирование программы 25](#_Toc185839664)

[Заключение 29](#_Toc185839665)

[Приложение А. Листинг программных модулей 30](#_Toc185839666)

# **Введение**

Сфера здравоохранения является одной из наиболее важных и социально значимых. В условиях постоянного роста объёма информации, связанной с пациентами, медицинскими записями, расписанием врачей и управлением ресурсами, использование современных систем управления базами данных (СУБД) становится необходимостью. Актуальность данной работы обусловлена необходимостью оптимизации работы больничных учреждений, что позволяет сократить время обработки данных, снизить число ошибок, улучшить качество обслуживания пациентов и повысить общий уровень медицинских услуг.

Цель исследования – разработка и анализ системы управления базами данных для автоматизации информационных процессов в больнице для улучшения работы персонала и качества обслуживания пациентов.

Для достижения поставленной цели в рамках курсовой работы были сформулированы следующие задачи:

1) Сравнить производительность СУБД и средств программирования.

2) Разработать модели данных.

3) Разработать базу данных.

4) Создать многопользовательское приложение.

Объект исследования – информационные процессы в больнице, связанные с регистрацией, хранением и обработкой медицинских данных.

Предмет исследования – система управления базами данных, предназначенная для автоматизации информационных процессов в больничных учреждениях.

# **1 Проектирование базы данных**

# **1.1 Концептуальная модель**

Программа "Больница" представляет собой интегрированную информационную систему для управления всеми аспектами больничной деятельности, включая взаимодействие с пациентами, медицинским персоналом, данными о лечении, а также административными и финансовыми процессами. Система построена как централизованная база данных с пользователями, работающими через интерфейс приложения. Это могут быть сотрудники больницы разных уровней: администраторы, врачи, бухгалтеры, фармацевты и другие специалисты. Система организована так, чтобы предоставить им доступ только к необходимым данным, гарантируя защиту личной и медицинской информации.

В ядре системы лежат тесно взаимосвязанные сущности. Таблица Patients является центральной для программы, так как все процессы обслуживания пациента зависят от информации, связанной с ним. Каждый пациент проходит регистрацию, при этом фиксируется его личная информация, возраст, контактные данные и данные об имеющемся страховом полисе. После регистрации пациент становится доступным для назначения встреч (Appointments), что осуществляют либо администраторы больницы, либо врачи через пользовательский интерфейс.

Назначенные встречи хранятся в таблице Appointments, где фиксируется время визита, причина обращения, а также статус встречи (например, запланирована, завершена или отменена). В момент встречи врач, связанный через таблицу Doctors, вносит данные в медицинскую карту пациента (MedicalRecords). Медицинская карта содержит важнейшую информацию о диагнозах, плане лечения, предписаниях и комментариях специалиста. Она облегчает отслеживание состояния пациента и помогает врачам принимать решения при дальнейших визитах.

Если пациенту требуется использование медикаментов, врач оформляет рецепт, данные о котором фиксируются в таблице Prescriptions. Эта таблица связана с таблицей Medications, где содержится информация о соответствующих лекарственных препаратах, их дозировке и побочных эффектах. Рецепт содержит чёткие указания по применению медикамента, что помогает снизить риски ошибок и недоразумений в процессе лечения. Выписанные лекарства могут быть проверены и выданы через аптеку больницы.

Финансовый аспект программы охватывается таблицей Payments, которая отслеживает все платёжные операции, связанные с пациентами. После предоставления услуг (например, консультаций, лечения или госпитализации), администраторы могут фиксировать поступление платежей. В таблицу заносятся такие данные, как сумма платежа, метод оплаты и дата. Эти сведения полезны для выполнения бухгалтерских операций и составления отчётов.

Для пациентов, нуждающихся в госпитализации, назначаются помещения, описанные в таблице Rooms. Администраторы и врачи могут проверять наличие свободных мест, распределять комнаты в зависимости от их типа и статуса занятости. Это упрощает систематизацию размещения и позволяет избегать конфликтов при распределении ресурсов.

Ключевую роль в системе играет персонал, информация о котором хранится в таблице Staff. Административный и медицинский персонал фиксируется в системе, включая их личные данные, должности и роли. Это необходимо для управления кадровым составом и распределения обязанностей. К примеру, администраторы отвечают за регистрацию пациентов, оформление документов и обработку платежей, в то время как врачи и медсёстры сосредотачиваются на их медицинском обслуживании.

Изображение выглядит как рисунок, диаграмма, зарисовка, Штриховая графика

Автоматически созданное описание

Рисунок 1 – ER-диаграмма концептуальной модели

Программа создаёт интегрированную среду, где каждая сущность выполняет свою роль в общей структуре оказания услуг. Пациент поступает в качестве центрального элемента системы, и вокруг него разворачивается обработка данных: регистрация, приём, диагностика, назначение лекарств или процедур, обработка финансовых операций и обеспечение палатами. Всё это взаимодействует через связные таблицы, поддерживая целостность данных и упрощая выполнение задач сотрудниками.

Реализация концепции способствует автоматизации всех основных процессов в больнице, минимизации человеческих ошибок, сокращению времени на обработку данных и повышению общего уровня удовлетворённости пациентов качеством обслуживания.

# **1.2 Атрибуты, сущности и целлостность данных**

Описание сущностей:

1. Patients (Пациенты):

- Описание: Хранит данные пациентов, включая личные и страховые данные.

- Ключевые атрибуты:

- patient\_id: Уникальный идентификатор.

- first\_name, last\_name, gender, date\_of\_birth: Личные данные пациента.

- insurance\_info: Страховая информация.

2. Doctors (Доктора):

- Описание: Информация о докторах, их специализациях и контактах.

- Ключевые атрибуты:

- doctor\_id: Уникальный идентификатор.

- specialization: Специализация доктора.

- phone\_number, email: Контакты.

3. Appointments (Назначения):

- Описание: Организует встречи между пациентами и докторами.

- Ключевые атрибуты:

- appointment\_id: Уникальный идентификатор.

- patient\_id, doctor\_id: Связь с пациентом и врачом.

- appointment\_date, reason\_for\_visit, status: Данные по встрече.

4. MedicalRecords (Медицинские записи):

- Описание: История лечения пациентов.

- Ключевые атрибуты:

- record\_id: Уникальный идентификатор записи.

- patient\_id, doctor\_id: Связь с пациентом и врачом.

- diagnosis, treatment\_plan, visit\_date, notes: Медицинские данные.

5. Prescriptions (Рецепты):

- Описание: Хранение информации о выписанных лекарствах.

- Ключевые атрибуты:

- prescription\_id: Уникальный идентификатор.

- record\_id: Ссылка на медицинскую запись.

- medication\_id: Ссылка на таблицу медикаментов.

- dosage\_instructions: Данные по дозировке.

6. Medications (Медикаменты):

- Описание: Каталог медикаментов, которые есть в больнице.

- Ключевые атрибуты:

- medication\_id: Уникальный идентификатор.

- dosage, side\_effects: Информация о лекарствах.

- photo: Для хранения изображения медикамента.

7. Rooms (Комнаты):

- Описание: Управление комнатами для пациентов.

- Ключевые атрибуты:

- room\_id: Уникальный идентификатор.

- room\_number: Номер комнаты.

- type: Тип комнаты (например, VIP, стандарт).

- availability\_status: Статус доступности комнаты (свободна, занята).

8. Staff (Персонал):

- Описание: Информация о других сотрудниках больницы (регистраторы, лаборанты и т.д.).

- Ключевые атрибуты:

- staff\_id: Уникальный идентификатор.

- position: Должность (например, администратор, уборщик).

9. Payments (Платежи):

- Описание: Управление финансовыми операциями пациентов.

- Ключевые атрибуты:

- payment\_id: Уникальный идентификатор.

- patient\_id: Связь с пациентом.

- amount: Сумма.

- payment\_method: Метод оплаты (наличные, карта).

10. Departments (Отделы):

- Описание: Организация больничных отделов.

- Ключевые атрибуты:

- department\_id: Уникальный идентификатор.

- name: Название (например, терапия, хирургия).

Связи между сущностями:

1. Пациенты — Назначения:

- Один пациент может иметь несколько назначений.

- Связь: patient\_id в Appointments.

2. Доктора — Назначения:

- Один доктор может проводить несколько встреч.

- Связь: doctor\_id в Appointments.

3. Назначения — Медицинские записи:

- Каждая медицинская запись связана с одной встречей.

- Связь: appointment\_id.

4. Медицинские записи — Рецепты:

- Каждая медицинская запись может иметь несколько рецептов.

- Связь: record\_id в Prescriptions.

5. Рецепты — Медикаменты:

- Один рецепт привязан к одному медикаменту.

- Связь: medication\_id в Prescriptions.

6. Пациенты — Платежи:

- Один пациент может совершать несколько платежей.

- Связь: patient\_id в Payments.

7. Доктора — Отделы:

- Один доктор принадлежит к одному отделу.

- Связь: department\_id.

8. Пациенты — Комнаты:

- Пациента можно назначить в комнату.

- Связь: patient\_id и room\_id (можно добавить таблицу размещения).

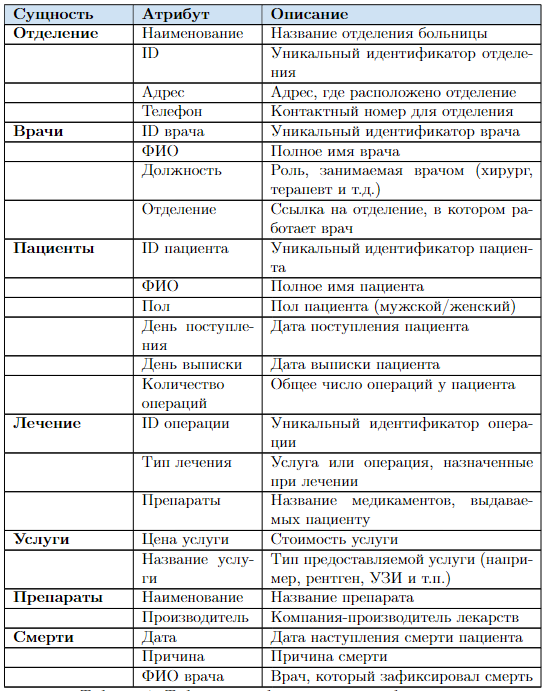


Рисунок 2 - Таблица атрибутов сущностей базы данных

* 1. **Построение физической модели**

Физическое проектирование - это процедура создания описания конкретной реализации БД с описанием структуры хранения данных, методов доступа к данным. В данном случае реализация была осуществлена в MS SQL.

Скрипт создания базы данных:

SQL-код для создания базы данных с таблицами:

CREATE DATABASE Hospital;

USE Hospital;

CREATE TABLE Patients (

patient\_id INTEGER PRIMARY KEY AUTO\_INCREMENT,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

date\_of\_birth DATE,

gender VARCHAR(10),

address VARCHAR(255),

phone\_number VARCHAR(15),

email VARCHAR(100),

insurance\_info VARCHAR(255)

);

CREATE TABLE Appointments (

appointment\_id INTEGER PRIMARY KEY AUTO\_INCREMENT,

patient\_id INTEGER,

doctor\_id INTEGER,

appointment\_date DATETIME,

reason\_for\_visit VARCHAR(255),

status VARCHAR(50),

FOREIGN KEY (patient\_id) REFERENCES Patients(patient\_id),

FOREIGN KEY (doctor\_id) REFERENCES Doctors(doctor\_id)

);

CREATE TABLE Doctors (

doctor\_id INTEGER PRIMARY KEY AUTO\_INCREMENT,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

specialization VARCHAR(100),

phone\_number VARCHAR(15),

email VARCHAR(100)

);

CREATE TABLE MedicalRecords (

record\_id INTEGER PRIMARY KEY AUTO\_INCREMENT,

patient\_id INTEGER,

doctor\_id INTEGER,

visit\_date DATETIME,

diagnosis VARCHAR(255),

treatment\_plan VARCHAR(255),

notes TEXT,

FOREIGN KEY (patient\_id) REFERENCES Patients(patient\_id),

FOREIGN KEY (doctor\_id) REFERENCES Doctors(doctor\_id)

);

CREATE TABLE Prescriptions (

prescription\_id INTEGER PRIMARY KEY AUTO\_INCREMENT,

record\_id INTEGER,

medication\_id INTEGER,

dosage\_instructions VARCHAR(255),

FOREIGN KEY (record\_id) REFERENCES MedicalRecords(record\_id),

FOREIGN KEY (medication\_id) REFERENCES Medications(medication\_id)

);

CREATE TABLE Medications (

medication\_id INTEGER PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100),

dosage VARCHAR(50),

side\_effects VARCHAR(255),

photo BLOB

);

CREATE TABLE Payments (

payment\_id INTEGER PRIMARY KEY AUTO\_INCREMENT,

patient\_id INTEGER,

amount REAL,

payment\_date DATETIME,

payment\_method VARCHAR(50),

FOREIGN KEY (patient\_id) REFERENCES Patients(patient\_id)

);

CREATE TABLE Rooms (

room\_id INTEGER PRIMARY KEY AUTO\_INCREMENT,

room\_number INTEGER,

type VARCHAR(100),

availability\_status VARCHAR(50)

);

CREATE TABLE Staff (

staff\_id INTEGER PRIMARY KEY AUTO\_INCREMENT,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

position VARCHAR(100)

);

CREATE TABLE Departments (

department\_id INTEGER PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100)

);

ALTER TABLE Doctors ADD department\_id INTEGER;

ALTER TABLE Doctors ADD FOREIGN KEY (department\_id) REFERENCES Departments(department\_id);

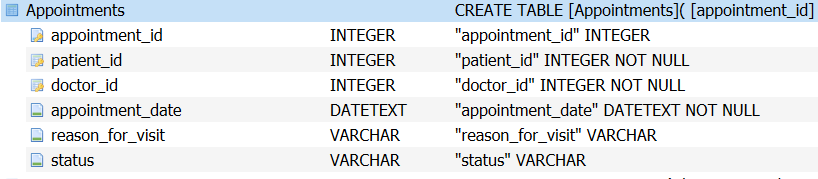


Рисунок 3 – Таблица Appointments

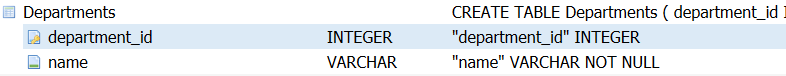


Рисунок 4 – Таблица Departments

Изображение выглядит как текст, Шрифт, число, линия

Автоматически созданное описание

Рисунок 5 – Таблица MedicalRecords

Изображение выглядит как текст, Шрифт, число, снимок экрана

Автоматически созданное описание

Рисунок 6 – Таблица Medications

Изображение выглядит как текст, Шрифт, число, снимок экрана

Автоматически созданное описание

Рисунок 7 – Таблица Patients

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 8 – Таблица Payments

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 9 – Таблица Prescriptions

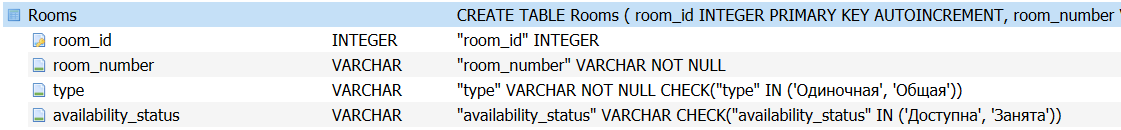


Рисунок 10 – Таблица Rooms

Изображение выглядит как текст, Шрифт, число, линия

Автоматически созданное описание

Рисунок 11 – Таблица Staff

Изображение выглядит как текст, диаграмма, Шрифт, План

Автоматически созданное описание

Рисунок 12 – UML-диаграмма базы данных

# **2 Разработка приложения по работе с СУБД**

# **2.1 Назначение и основные функции приложения**

Для разработки программного обеспечения в рамках выполнения данной задачи применяются различные инструменты и технологии, которые обеспечивают эффективное создание как клиентской, так и серверной части с базой данных. В качестве системы управления базами данных была выбрана MySQL. Данная СУБД широко используется благодаря своей производительности, устойчивости и возможности обработки реляционных данных. Основным преимуществом MySQL является её открытость и наличие большого количества встроенных функций, необходимых для работы с большими объемами информации. Для локального развертывания базы данных и администрирования использовалась утилита MySQL Workbench, которая позволяет визуализировать создаваемую структуру данных, осуществлять проектирование таблиц и выполнение SQL-запросов. В случае необходимости веб-ориентированного управления базой данных может быть применен PhpMyAdmin.

Среда разработки Visual Studio 2022 выбрана для написания основного кода приложения. Данный программный комплекс является универсальной интегрированной средой разработки, которая поддерживает множество языков программирования, включая C#. Использование C# в сочетании с технологией ASP.NET Core позволяет создавать как клиентские, так и серверные части программного обеспечения, обеспечивая взаимодействие с базой данных через встроенные библиотеки и пакеты, например, такие как MySQL Data Connector или Entity Framework. Удобство Visual Studio также заключается в интеграции с системами контроля версий и автоматизацией процесса сборки приложений.

Backend-часть приложения, отвечающая за обработку запросов и взаимодействие с базой данных, реализуется с помощью стандартных технологий, принятых в промышленной разработке. Для обеспечения доступности данных используются операции CRUD (создание, чтение, обновление и удаление записей), которые реализуются посредством SQL-запросов. Кодом на языке C# обеспечивается подключение к базе данных MySQL, обработка SQL-запросов и возвращение результата клиенту. Это достигается за счет использования стандартных библиотек .NET Framework или .NET Core.

Если в рамках реализации требуется создание графического пользовательского интерфейса, возможно использование технологий WPF (Windows Presentation Foundation). WPF позволяет проектировать современный, интуитивно понятный фронтенд для настольных приложений с поддержкой сложных графических интерфейсов. Для создания веб-интерфейса могут применяться такие инструменты, как Blazor или React.js, обеспечивающие взаимодействие с серверной частью посредством API.

На этапе тестирования программы предусматривается использование локального сервера. Для этого могут быть задействованы программные среды, такие как XAMPP или WAMP, которые обеспечивают локальный запуск базы данных MySQL и обработку запросов. Это дает возможность отлаживать приложение в изолированной среде без необходимости разворачивать его на реальном сервере.

Основные функции приложения включают в себя добавление, обновление и удаление записей, связанных с пациентами и врачами, с использованием команд базы данных SQLite. При загрузке формы приложения происходит инициализация базы данных, после чего загружаются данные о записях назначений, а также списки пациентов и врачей, которые отображаются в соответствующих комбобоксах. Это позволяет пользователю выбирать существующих пациентов и врачей для создания или редактирования записей о назначениях. При добавлении новой записи приложение проверяет, заполнены ли все необходимые поля, и валидирует введенные данные с помощью регулярных выражений. Это обеспечивает корректность введенной информации, такой как имена пациентов и врачей, дата визита и причина обращения. Если все данные введены правильно, запись добавляется в базу данных, и пользователю отображается сообщение об успешном добавлении. Функция обновления записи работает аналогично, но вместо добавления новой записи происходит изменение существующей. Пользователь может выбрать запись из таблицы, после чего данные загружаются в поля ввода, где их можно редактировать. После внесения изменений и подтверждения, обновленная информация сохраняется в базе данных. Удаление записей также реализовано через интерфейс, где пользователь может выбрать запись и удалить ее из базы данных. Приложение уведомляет пользователя о результате операции, будь то успешное удаление или необходимость выбора записи для удаления. Таким образом, приложение обеспечивает полный цикл работы с медицинскими записями, начиная от их создания и заканчивая редактированием и удалением, при этом акцентируя внимание на валидации данных и удобстве взаимодействия с пользователем.

# **2.2 Структура приложения**

Приложение для управления медицинскими записями в больнице имеет четкую и логичную структуру, которая обеспечивает эффективное взаимодействие между пользователем и базой данных. В центре приложения находится форма, которая служит интерфейсом для выполнения различных операций, таких как добавление, обновление и удаление записей о пациентах и врачах.

Основные компоненты приложения включают в себя классы, отвечающие за различные аспекты функциональности. Класс `Appointments` управляет записями о приёмах, обеспечивая загрузку данных из базы, отображение их в интерфейсе и обработку действий пользователя, таких как добавление новых записей или удаление существующих. Этот класс использует экземпляр `ClassProvide`, который реализует паттерн Singleton для управления соединением с базой данных SQLite. Это позволяет избежать создания нескольких соединений и обеспечивает централизованный доступ к базе данных. Класс `ClassProvide` отвечает за установление и управление соединением с базой данных.

Он инициализирует соединение при создании экземпляра и предоставляет методы для получения этого соединения, а также для его закрытия. Это обеспечивает надежное управление ресурсами и предотвращает утечки памяти. В дополнение к классу `Appointments`, приложение включает другие классы, такие как `Doctors`, `Departments`, и `Payments`, каждый из которых имеет свою собственную логику и интерфейс для управления соответствующими данными.

Эти классы взаимодействуют с базой данных аналогичным образом, используя SQL-запросы для выполнения операций CRUD (создание, чтение, обновление, удаление). Интерфейс пользователя состоит из различных элементов управления, таких как текстовые поля, комбобоксы и таблицы данных, которые позволяют пользователю вводить информацию и получать обратную связь. Валидация данных осуществляется с помощью регулярных выражений, что гарантирует, что введенные данные соответствуют ожидаемым форматам, прежде чем они будут отправлены в базу данных. Таким образом, структура приложения организована вокруг классов, которые инкапсулируют логику работы с данными и интерфейсом, обеспечивая четкое разделение ответственности.

Это позволяет легко расширять функциональность приложения и поддерживать его в будущем.

Для визуализации структуры приложения можно использовать UML-диаграммы объектов и компонентов. UML-диаграмма объектов будет представлять классы и их взаимосвязи, показывая, как они взаимодействуют друг с другом, в то время как UML-диаграмма компонентов отразит архитектуру приложения, включая основные модули и их интерфейсы. Эти диаграммы помогут лучше понять архитектурные решения, принятые при разработке приложения, и упростят процесс его дальнейшего развития.

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Автоматически созданное описание

Рисунок 13 – UML-диаграмма объектов

Изображение выглядит как текст, диаграмма, линия, Шрифт

Автоматически созданное описание

Рисунок 14 – UML-диаграмма компонентов

# **2.3 Тестирование программы**

Тестирование включает в себя проверку всех функций, таких как добавление, обновление и удаление записей, а также валидацию пользовательского ввода.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 15 - Главное окно

На рисунке 15 представлено главное окно приложения. Оно служит центральной точкой взаимодействия пользователя с системой. В этом окне отображаются основные функции, такие как добавление, редактирование и удаление записей о пациентах и врачах.

Главное окно также включает в себя элементы управления, такие как комбобоксы для выбора пациентов и врачей, текстовые поля для ввода информации и таблицу для отображения записей. Это окно вызывается при запуске приложения и обеспечивает пользователю доступ ко всем необходимым функциям.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 16 - Окно добавления записи

Это окно позволяет пользователю вводить данные о пациенте, враче, дате приема и причине визита. Оно вызывается при нажатии кнопки "Добавить" на главном окне и обеспечивает удобный интерфейс для ввода информации.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 17 - Окно платежи

Пользователь может выбрать запись из таблицы и внести изменения в данные. Это окно открывается при двойном щелчке на строке таблицы и позволяет обновить информацию о приеме.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 18 – Пациенты

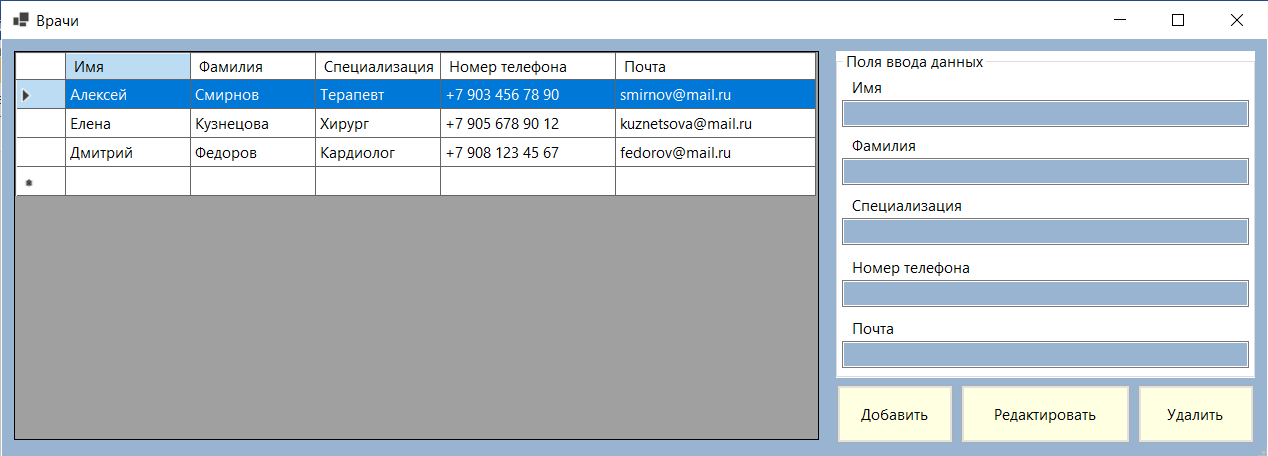


Рисунок 19 – Врачи

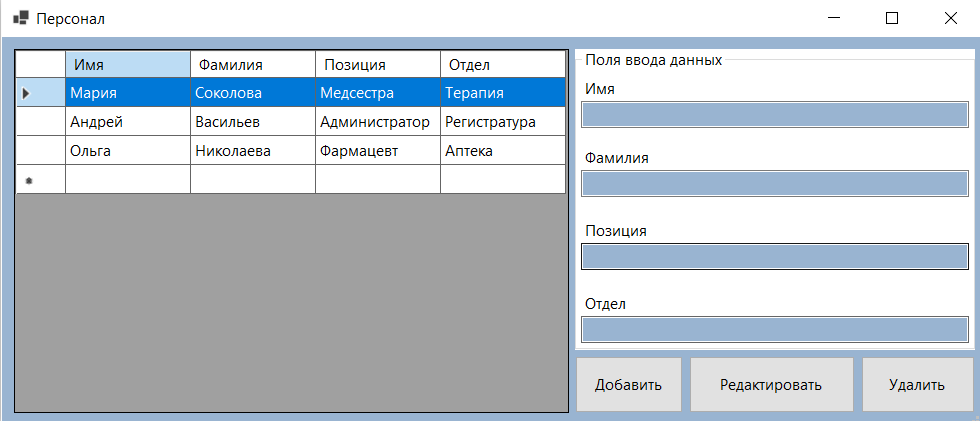


Рисунок 20 – Персонал

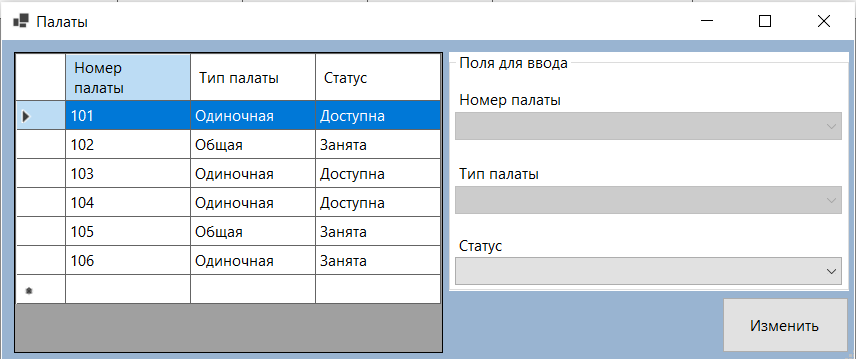


Рисунок 21 – Палаты

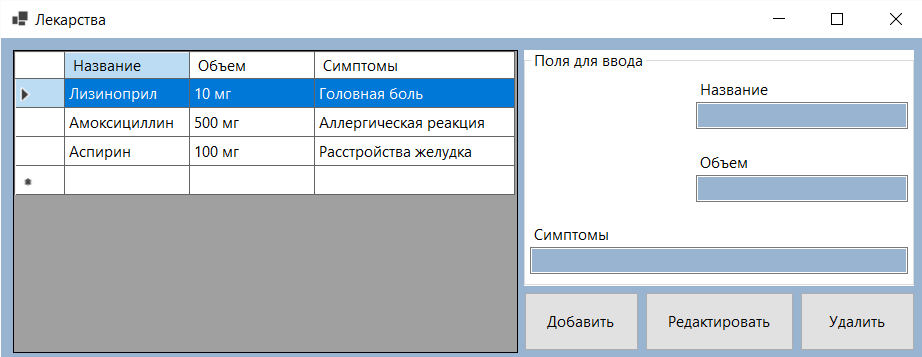


Рисунок 22 – Лекарства

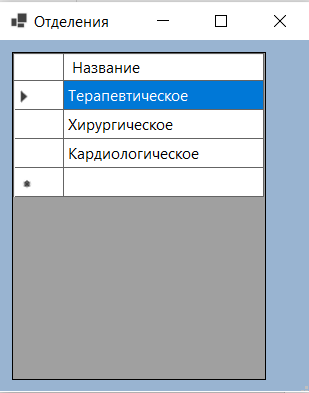


Рисунок 23 – Отделения

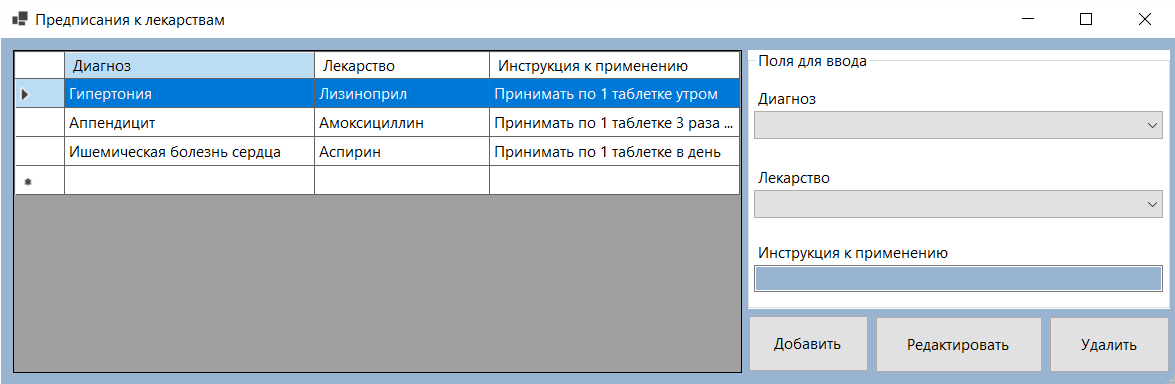


Рисунок 24 – Предписания к лекарствам

# **Заключение**

В рамках данной курсовой работы была исследована необходимость внедрения системы управления базами данных для автоматизации процессов в больнице. В ходе выполнения работы был проведён анализ требований, разработана структура базы данных и описаны основные сценарии работы с системой.

Результатом исследования стало создание базовой концепции СУБД, позволяющей эффективно управлять данными пациентов, записями на приём, медицинскими картами и прочими аспектами работы больницы. Реализация подобной системы способствует оптимизации времени, позволяет избежать ошибок в работе медицинского персонала и повышает качество оказываемых услуг.

Таким образом, поставленная цель работы была достигнута, а предъявленные задачи выполнены, что подтверждает практическую и теоретическую значимость разработанной базы данных для больничных учреждений.

# **Приложение А. Листинг программных модулей**

**Appointments.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SQLite;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Hospital

{

    public partial class Appointments : Form

    {

        public Appointments()

        {

            InitializeComponent();

        }

        private ClassProvide \_db;

        string id;

        private void Appointments\_Load(object sender, EventArgs e)

        {

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn = \_db.GetConnection();

            string query = "SELECT appointment\_id, Patients.last\_name as lnp, Doctors.last\_name lnd, appointment\_date, reason\_for\_visit, status FROM Appointments INNER JOIN Patients ON Appointments.patient\_id = Patients.patient\_id INNER JOIN Doctors ON Appointments.doctor\_id = Doctors.doctor\_id";

            SQLiteCommand cmd = new SQLiteCommand(query, conn);

            SQLiteDataReader reader = cmd.ExecuteReader();

            while (reader.Read())

            {

                dataGridView1.Rows.Add(reader["appointment\_id"], reader["lnp"], reader["lnd"], reader["appointment\_date"], reader["reason\_for\_visit"], reader["status"]);

            }

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn1 = \_db.GetConnection();

            string query1 = "SELECT DISTINCT last\_name FROM Patients";

            SQLiteCommand cmd1 = new SQLiteCommand(query1, conn1);

            SQLiteDataReader reader1 = cmd1.ExecuteReader();

            while (reader1.Read())

            {

                string data = reader1.GetString(0);

                comboBox1.Items.Add(data);

            }

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn2 = \_db.GetConnection();

            string query2 = "SELECT DISTINCT last\_name FROM Doctors";

            SQLiteCommand cmd2 = new SQLiteCommand(query2, conn2);

            SQLiteDataReader reader2 = cmd2.ExecuteReader();

            while (reader2.Read())

            {

                string data = reader2.GetString(0);

                comboBox2.Items.Add(data);

            }

            comboBox3.Items.Add("Запланированный");

            comboBox3.Items.Add("Отмененный");

            comboBox3.Items.Add("Завершенный");

        }

        private void button2\_Click(object sender, EventArgs e)

        {

            if (dataGridView1.SelectedRows.Count > 0)

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string id = dataGridView1.SelectedRows[0].Cells["Column1"].Value.ToString();

                string sql = "DELETE FROM Appointments WHERE appointment\_id = @id";

                SQLiteCommand cmd = new SQLiteCommand(sql, conn);

                cmd.Parameters.AddWithValue("@id", id);

                cmd.ExecuteNonQuery();

                dataGridView1.Refresh();

                MessageBox.Show("Запись удалена из базы!");

            }

            else

            {

                MessageBox.Show("Выберите запись для удаления!");

            }

        }

        private void button1\_Click(object sender, EventArgs e)

        {

            string pt = comboBox1.Text;

            string dc = comboBox2.Text;

            string ad = dateTimePicker1.Text;

            string rfv = textBox1.Text;

            string st = comboBox3.Text;

            if (string.IsNullOrEmpty(pt) || string.IsNullOrEmpty(dc) || string.IsNullOrEmpty(ad) || string.IsNullOrEmpty(rfv) || string.IsNullOrEmpty(st))

            {

                MessageBox.Show("Заполните все поля!");

            }

            else if (!Regex.IsMatch(pt, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Пациента правильно!");

            }

            else if (!Regex.IsMatch(dc, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Врача правильно!");

            }

            else if (!Regex.IsMatch(ad, "^[0-9а-яА-Я .-]+$"))

            {

                MessageBox.Show("Заполните Дату визита правильно!");

            }

            else if (!Regex.IsMatch(rfv, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Причину визита правильно!");

            }

            else if (!Regex.IsMatch(st, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Статус правильно!");

            }

            else

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string query = "INSERT INTO Appointments (patient\_id, doctor\_id, appointment\_date, reason\_for\_visit, status) \r\nVALUES ((SELECT patient\_id FROM Patients WHERE last\_name = @pt), (SELECT doctor\_id FROM Doctors WHERE last\_name = @dc), @ad, @rfv, @st);";

                SQLiteCommand command = new SQLiteCommand(query, conn);

                command.Parameters.AddWithValue("@pt", pt);

                command.Parameters.AddWithValue("@dc", dc);

                command.Parameters.AddWithValue("@ad", ad);

                command.Parameters.AddWithValue("@rfv", rfv);

                command.Parameters.AddWithValue("@st", st);

                command.ExecuteNonQuery();

                MessageBox.Show("Запись добавлена в базу!");

            }

        }

        private void button3\_Click(object sender, EventArgs e)

        {

            if (string.IsNullOrEmpty(comboBox1.Text) || string.IsNullOrEmpty(comboBox2.Text) || string.IsNullOrEmpty(dateTimePicker1.Text) || string.IsNullOrEmpty(textBox1.Text) || string.IsNullOrEmpty(comboBox3.Text))

            {

                MessageBox.Show("Заполните все поля!");

            }

            else if (!Regex.IsMatch(comboBox1.Text, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Пациента правильно!");

            }

            else if (!Regex.IsMatch(comboBox2.Text, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Врача правильно!");

            }

            else if (!Regex.IsMatch(dateTimePicker1.Text, "^[0-9а-яА-Я .-]+$"))

            {

                MessageBox.Show("Заполните Дату визита правильно!");

            }

            else if (!Regex.IsMatch(textBox1.Text, "^[а-яА-Я, -]+$"))

            {

                MessageBox.Show("Заполните Причину визита правильно!");

            }

            else if (!Regex.IsMatch(comboBox3.Text, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Статус правильно!");

            }

            else

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string query = "UPDATE Appointments SET patient\_id = (SELECT patient\_id FROM Patients WHERE last\_name = @pt), doctor\_id = (SELECT doctor\_id FROM Doctors WHERE last\_name = @dc), appointment\_date = @ad, reason\_for\_visit = @rfv, status = @st WHERE appointment\_id = @id";

                SQLiteCommand command = new SQLiteCommand(query, conn);

                command.Parameters.AddWithValue("@id", id);

                command.Parameters.Add("@pt", DbType.String).Value = comboBox1.Text;

                command.Parameters.Add("@dc", DbType.String).Value = comboBox2.Text;

                command.Parameters.Add("@ad", DbType.String).Value = dateTimePicker1.Text;

                command.Parameters.Add("@rfv", DbType.String).Value = textBox1.Text;

                command.Parameters.Add("@st", DbType.String).Value = comboBox3.Text;

                command.ExecuteNonQuery();

                MessageBox.Show("Запись успешно изменена!");

            }

        }

        private void dataGridView1\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

        {

            if (dataGridView1.SelectedRows.Count > 0)

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                id = dataGridView1.SelectedRows[0].Cells["Column1"].Value.ToString();

                string query = "SELECT appointment\_id, Patients.last\_name as lnp, Doctors.last\_name lnd, appointment\_date, reason\_for\_visit, status FROM Appointments INNER JOIN Patients ON Appointments.patient\_id = Patients.patient\_id INNER JOIN Doctors ON Appointments.doctor\_id = Doctors.doctor\_id WHERE appointment\_id = @id";

                SQLiteCommand cmd = new SQLiteCommand(query, conn);

                cmd.Parameters.AddWithValue("@id", id);

                SQLiteDataReader reader = cmd.ExecuteReader();

                if (reader.Read())

                {

                    string lnp = reader["lnp"].ToString();

                    string lnd = reader["lnd"].ToString();

                    string appointment\_date = reader["appointment\_date"].ToString();

                    string reason\_for\_visit = reader["reason\_for\_visit"].ToString();

                    string status = reader["status"].ToString();

                    comboBox1.Text = lnp;

                    comboBox2.Text = lnd;

                    dateTimePicker1.Text = appointment\_date;

                    textBox1.Text = reason\_for\_visit;

                    comboBox3.Text = status;

                }

            }

        }

    }

}

**ClassProvide.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.SQLite;

namespace Hospital

{

    internal class ClassProvide

    {

        private static ClassProvide \_instance;

        private SQLiteConnection \_connection;

        private ClassProvide()

        {

            \_connection = new SQLiteConnection("Data Source=D:\hospital.db");

            \_connection.Open();

        }

        public static ClassProvide GetInstance()

        {

            if (\_instance == null)

            {

                \_instance = new ClassProvide();

            }

            return \_instance;

        }

        public SQLiteConnection GetConnection()

        {

            return \_connection;

        }

        public void CloseConnection()

        {

            if (\_connection != null)

            {

                \_connection.Close();

            }

        }

    }

}

**Departments.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SQLite;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Hospital

{

    public partial class Departments : Form

    {

        public Departments()

        {

            InitializeComponent();

        }

        private ClassProvide \_db;

        private void Departments\_Load(object sender, EventArgs e)

        {

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn = \_db.GetConnection();

            string query = "SELECT department\_id, name FROM Departments";

            SQLiteCommand cmd = new SQLiteCommand(query, conn);

            SQLiteDataReader reader = cmd.ExecuteReader();

            while (reader.Read())

            {

                dataGridView1.Rows.Add(reader["department\_id"], reader["name"]);

            }

        }

    }

}

**Doctors.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SQLite;

using System.Drawing;

using System.Linq;

using System.Runtime.Intrinsics.X86;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml.Linq;

namespace Hospital

{

    public partial class Doctors : Form

    {

        public Doctors()

        {

            InitializeComponent();

        }

        private ClassProvide \_db;

        string id;

        private void Doctors\_Load(object sender, EventArgs e)

        {

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn = \_db.GetConnection();

            string query = "SELECT doctor\_id, first\_name, last\_name, specialization, phone\_number, email FROM Doctors";

            SQLiteCommand cmd = new SQLiteCommand(query, conn);

            SQLiteDataReader reader = cmd.ExecuteReader();

            while (reader.Read())

            {

                dataGridView1.Rows.Add(reader["doctor\_id"], reader["first\_name"], reader["last\_name"], reader["specialization"], reader["phone\_number"], reader["email"]);

            }

        }

        private void button3\_Click(object sender, EventArgs e)

        {

            if (dataGridView1.SelectedRows.Count > 0)

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string id = dataGridView1.SelectedRows[0].Cells["Column1"].Value.ToString();

                string sql = "DELETE FROM Doctors WHERE doctor\_id = @id";

                SQLiteCommand cmd = new SQLiteCommand(sql, conn);

                cmd.Parameters.AddWithValue("@id", id);

                cmd.ExecuteNonQuery();

                dataGridView1.Refresh();

                MessageBox.Show("Запись удалена из базы!");

            }

            else

            {

                MessageBox.Show("Выберите запись для удаления!");

            }

        }

        private void button1\_Click(object sender, EventArgs e)

        {

            string name = textBox1.Text;

            string fam = textBox2.Text;

            string sp = textBox3.Text;

            string pn = textBox4.Text;

            string em = textBox5.Text;

            if (string.IsNullOrEmpty(name) || string.IsNullOrEmpty(fam) || string.IsNullOrEmpty(sp) || string.IsNullOrEmpty(pn) || string.IsNullOrEmpty(em))

            {

                MessageBox.Show("Заполните все поля!");

            }

            else if (!Regex.IsMatch(name, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Имя правильно!");

            }

            else if (!Regex.IsMatch(fam, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Фамилию правильно!");

            }

            else if (!Regex.IsMatch(sp, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Специализацию правильно!");

            }

            else if (!Regex.IsMatch(pn, "^[0-9 +]+$"))

            {

                MessageBox.Show("Заполните Номер телефона правильно!");

            }

            else if (!Regex.IsMatch(em, "^[A-Za-z0-9 .@+]+$"))

            {

                MessageBox.Show("Заполните Почту правильно!");

            }

            else

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string query = "INSERT INTO Doctors (first\_name, last\_name, specialization, phone\_number, email) \r\nVALUES (@name, @fam, @sp, @pn, @em);";

                SQLiteCommand command = new SQLiteCommand(query, conn);

                command.Parameters.AddWithValue("@name", name);

                command.Parameters.AddWithValue("@fam", fam);

                command.Parameters.AddWithValue("@sp", sp);

                command.Parameters.AddWithValue("@pn", pn);

                command.Parameters.AddWithValue("@em", em);

                command.ExecuteNonQuery();

                MessageBox.Show("Запись добавлена в базу!");

            }

        }

        private void button2\_Click(object sender, EventArgs e)

        {

            if (string.IsNullOrEmpty(textBox1.Text) || string.IsNullOrEmpty(textBox2.Text) || string.IsNullOrEmpty(textBox3.Text) || string.IsNullOrEmpty(textBox4.Text) || string.IsNullOrEmpty(textBox5.Text))

            {

                MessageBox.Show("Заполните все поля!");

            }

            else if (!Regex.IsMatch(textBox1.Text, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Имя правильно!");

            }

            else if (!Regex.IsMatch(textBox2.Text, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Фамилию правильно!");

            }

            else if (!Regex.IsMatch(textBox3.Text, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Специализацию правильно!");

            }

            else if (!Regex.IsMatch(textBox4.Text, "^[0-9 +]+$"))

            {

                MessageBox.Show("Заполните Номер телефона правильно!");

            }

            else if (!Regex.IsMatch(textBox5.Text, "^[A-Za-z0-9 .@+]+$"))

            {

                MessageBox.Show("Заполните Почту правильно!");

            }

            else

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string query = "UPDATE Doctors SET first\_name = @fn, last\_name = @ln, specialization = @sp, phone\_number = @pn, email = @em WHERE doctor\_id = @id";

                SQLiteCommand command = new SQLiteCommand(query, conn);

                command.Parameters.AddWithValue("@id", id);

                command.Parameters.Add("@fn", DbType.String).Value = textBox1.Text;

                command.Parameters.Add("@ln", DbType.String).Value = textBox2.Text;

                command.Parameters.Add("@sp", DbType.String).Value = textBox3.Text;

                command.Parameters.Add("@pn", DbType.String).Value = textBox4.Text;

                command.Parameters.Add("@em", DbType.String).Value = textBox5.Text;

                command.ExecuteNonQuery();

                MessageBox.Show("Запись успешно изменена!");

            }

        }

        private void dataGridView1\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

        {

            if (dataGridView1.SelectedRows.Count > 0)

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                id = dataGridView1.SelectedRows[0].Cells["Column1"].Value.ToString();

                string query = "SELECT doctor\_id, first\_name, last\_name, specialization, phone\_number, email FROM Doctors WHERE doctor\_id = @id";

                SQLiteCommand cmd = new SQLiteCommand(query, conn);

                cmd.Parameters.AddWithValue("@id", id);

                SQLiteDataReader reader = cmd.ExecuteReader();

                if (reader.Read())

                {

                    string fn = reader["first\_name"].ToString();

                    string ln = reader["last\_name"].ToString();

                    string sp = reader["specialization"].ToString();

                    string pn = reader["phone\_number"].ToString();

                    string em = reader["email"].ToString();

                    textBox1.Text = fn;

                    textBox2.Text = ln;

                    textBox3.Text = sp;

                    textBox4.Text = pn;

                    textBox5.Text = em;

                }

            }

        }

    }

}

**Form1.cs**

using System.Data;

using System.Data.SQLite;

using System.Runtime.Intrinsics.Arm;

using System.Text.RegularExpressions;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Hospital

{

    public partial class Form1 : Form

    {

        public Form1()

        {

            InitializeComponent();

        }

        private ClassProvide \_db;

        string id;

        private void �����ToolStripMenuItem\_Click(object sender, EventArgs e)

        {

            Departments dp = new Departments();

            dp.ShowDialog(this);

        }

        private void �������ToolStripMenuItem\_Click(object sender, EventArgs e)

        {

            Doctors dc = new Doctors();

            dc.ShowDialog(this);

        }

        private void ��������ToolStripMenuItem\_Click(object sender, EventArgs e)

        {

            Patients pt = new Patients();

            pt.ShowDialog(this);

        }

        private void ��������ToolStripMenuItem\_Click(object sender, EventArgs e)

        {

            Staff st = new Staff();

            st.ShowDialog(this);

        }

        private void ���������ToolStripMenuItem\_Click(object sender, EventArgs e)

        {

            Medications md = new Medications();

            md.ShowDialog(this);

        }

        private void ������ToolStripMenuItem\_Click(object sender, EventArgs e)

        {

            Rooms rm = new Rooms();

            rm.ShowDialog(this);

        }

        private void �������ToolStripMenuItem\_Click(object sender, EventArgs e)

        {

            Payments pm = new Payments();

            pm.ShowDialog(this);

        }

        private void ����������������������ToolStripMenuItem\_Click(object sender, EventArgs e)

        {

            Prescriptions pr = new Prescriptions();

            pr.ShowDialog(this);

        }

        private void �������������ToolStripMenuItem\_Click(object sender, EventArgs e)

        {

            Appointments appointments = new Appointments();

            appointments.ShowDialog(this);

        }

        private void Form1\_Load(object sender, EventArgs e)

        {

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn = \_db.GetConnection();

            string query = "SELECT \r\n    MedicalRecords.record\_id as id, \r\n    Patients.last\_name as lnp, \r\n    Doctors.last\_name as lnd, \r\n    MedicalRecords.visit\_date as vd, \r\n    MedicalRecords.diagnosis as d, \r\n    MedicalRecords.treatment\_plan tp, \r\n    MedicalRecords.notes as n \r\nFROM \r\n    MedicalRecords\r\nINNER JOIN \r\n    Patients ON MedicalRecords.patient\_id = Patients.patient\_id\r\nINNER JOIN \r\n    Doctors ON MedicalRecords.doctor\_id = Doctors.doctor\_id;";

            SQLiteCommand cmd = new SQLiteCommand(query, conn);

            SQLiteDataReader reader = cmd.ExecuteReader();

            while (reader.Read())

            {

                dataGridView1.Rows.Add(reader["id"], reader["lnp"], reader["lnd"], reader["vd"], reader["d"], reader["tp"], reader["n"]);

            }

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn1 = \_db.GetConnection();

            string query1 = "SELECT DISTINCT last\_name FROM Patients";

            SQLiteCommand cmd1 = new SQLiteCommand(query1, conn1);

            SQLiteDataReader reader1 = cmd1.ExecuteReader();

            while (reader1.Read())

            {

                string data = reader1.GetString(0);

                comboBox1.Items.Add(data);

            }

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn2 = \_db.GetConnection();

            string query2 = "SELECT DISTINCT last\_name FROM Doctors";

            SQLiteCommand cmd2 = new SQLiteCommand(query2, conn2);

            SQLiteDataReader reader2 = cmd2.ExecuteReader();

            while (reader2.Read())

            {

                string data = reader2.GetString(0);

                comboBox2.Items.Add(data);

            }

        }

        private void dataGridView1\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

        {

            if (dataGridView1.SelectedRows.Count > 0)

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                id = dataGridView1.SelectedRows[0].Cells["Column1"].Value.ToString();

                string query = "SELECT \r\n    MedicalRecords.record\_id as id, \r\n    Patients.last\_name as lnp, \r\n    Doctors.last\_name as lnd, \r\n    MedicalRecords.visit\_date as vd, \r\n    MedicalRecords.diagnosis as d, \r\n    MedicalRecords.treatment\_plan tp, \r\n    MedicalRecords.notes as n \r\nFROM \r\n    MedicalRecords\r\nINNER JOIN \r\n    Patients ON MedicalRecords.patient\_id = Patients.patient\_id\r\nINNER JOIN \r\n    Doctors ON MedicalRecords.doctor\_id = Doctors.doctor\_id WHERE record\_id = @id";

                SQLiteCommand cmd = new SQLiteCommand(query, conn);

                cmd.Parameters.AddWithValue("@id", id);

                SQLiteDataReader reader = cmd.ExecuteReader();

                if (reader.Read())

                {

                    string lnp = reader["lnp"].ToString();

                    string lnd = reader["lnd"].ToString();

                    string vd = reader["vd"].ToString();

                    string d = reader["d"].ToString();

                    string tp = reader["tp"].ToString();

                    string n = reader["n"].ToString();

                    comboBox1.Text = lnp;

                    comboBox2.Text = lnd;

                    dateTimePicker1.Text = vd;

                    textBox1.Text = d;

                    textBox2.Text = tp;

                    textBox3.Text = n;

                }

            }

        }

        private void button3\_Click(object sender, EventArgs e)

        {

            if (dataGridView1.SelectedRows.Count > 0)

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string id = dataGridView1.SelectedRows[0].Cells["Column1"].Value.ToString();

                string sql = "DELETE FROM MedicalRecords WHERE record\_id = @id";

                SQLiteCommand cmd = new SQLiteCommand(sql, conn);

                cmd.Parameters.AddWithValue("@id", id);

                cmd.ExecuteNonQuery();

                dataGridView1.Refresh();

                MessageBox.Show("������ ������� �� ����!");

            }

            else

            {

                MessageBox.Show("�������� ������ ��� ��������!");

            }

        }

        private void button1\_Click(object sender, EventArgs e)

        {

            string pt = comboBox1.Text;

            string dc = comboBox2.Text;

            string vd = dateTimePicker1.Text;

            string dg = textBox1.Text;

            string tp = textBox2.Text;

            string nt = textBox3.Text;

            if (string.IsNullOrEmpty(pt) || string.IsNullOrEmpty(dc) || string.IsNullOrEmpty(vd) || string.IsNullOrEmpty(dg) || string.IsNullOrEmpty(tp) || string.IsNullOrEmpty(nt))

            {

                MessageBox.Show("��������� ��� ����!");

            }

            else if (!Regex.IsMatch(pt, "^[�-��-� -]+$"))

            {

                MessageBox.Show("��������� �������� ���������!");

            }

            else if (!Regex.IsMatch(dc, "^[�-��-� -]+$"))

            {

                MessageBox.Show("��������� ����� ���������!");

            }

            else if (!Regex.IsMatch(vd, "^[�-��-�0-9 .-]+$"))

            {

                MessageBox.Show("��������� ���� ������ ���������!");

            }

            else if (!Regex.IsMatch(dg, "^[�-��-� .-]+$"))

            {

                MessageBox.Show("��������� ������� ���������!");

            }

            else if (!Regex.IsMatch(tp, "^[�-��-� .-]+$"))

            {

                MessageBox.Show("��������� ���� ������� ���������!");

            }

            else if (!Regex.IsMatch(nt, "^[�-��-� .-]+$"))

            {

                MessageBox.Show("��������� ������ ���������!");

            }

            else

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string query = "INSERT INTO MedicalRecords (patient\_id, doctor\_id, visit\_date, diagnosis, treatment\_plan, notes) \r\nVALUES ((SELECT patient\_id FROM Patients WHERE last\_name = @pt), (SELECT doctor\_id FROM Doctors WHERE last\_name = @dc), @vd, @dg, @tp, @nt);";

                SQLiteCommand command = new SQLiteCommand(query, conn);

                command.Parameters.AddWithValue("@pt", pt);

                command.Parameters.AddWithValue("@dc", dc);

                command.Parameters.AddWithValue("@vd", vd);

                command.Parameters.AddWithValue("@dg", dg);

                command.Parameters.AddWithValue("@tp", tp);

                command.Parameters.AddWithValue("@nt", nt);

                command.ExecuteNonQuery();

                MessageBox.Show("������ ��������� � ����!");

            }

        }

        private void button2\_Click(object sender, EventArgs e)

        {

            if (string.IsNullOrEmpty(comboBox1.Text) || string.IsNullOrEmpty(comboBox2.Text) || string.IsNullOrEmpty(dateTimePicker1.Text) || string.IsNullOrEmpty(textBox1.Text) || string.IsNullOrEmpty(textBox2.Text) || string.IsNullOrEmpty(textBox3.Text))

            {

                MessageBox.Show("��������� ��� ����!");

            }

            else if (!Regex.IsMatch(comboBox1.Text, "^[�-��-� -]+$"))

            {

                MessageBox.Show("��������� �������� ���������!");

            }

            else if (!Regex.IsMatch(comboBox2.Text, "^[�-��-� -]+$"))

            {

                MessageBox.Show("��������� ����� ���������!");

            }

            else if (!Regex.IsMatch(dateTimePicker1.Text, "^[�-��-�0-9 .-]+$"))

            {

                MessageBox.Show("��������� ���� ������ ���������!");

            }

            else if (!Regex.IsMatch(textBox1.Text, "^[�-��-� .-]+$"))

            {

                MessageBox.Show("��������� ������� ���������!");

            }

            else if (!Regex.IsMatch(textBox2.Text, "^[�-��-� ,.-]+$"))

            {

                MessageBox.Show("��������� ���� ������� ���������!");

            }

            else if (!Regex.IsMatch(textBox3.Text, "^[�-��-� .-]+$"))

            {

                MessageBox.Show("��������� ������ ���������!");

            }

            else

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string query = "UPDATE MedicalRecords SET patient\_id = (SELECT patient\_id FROM Patients WHERE last\_name = @pt), doctor\_id = (SELECT doctor\_id FROM Doctors WHERE last\_name = @dc), visit\_date = @vd, diagnosis = @dg, treatment\_plan = @tp, notes = @nt WHERE record\_id = @id";

                SQLiteCommand command = new SQLiteCommand(query, conn);

                command.Parameters.AddWithValue("@id", id);

                command.Parameters.Add("@pt", DbType.String).Value = comboBox1.Text;

                command.Parameters.Add("@dc", DbType.String).Value = comboBox2.Text;

                command.Parameters.Add("@vd", DbType.String).Value = dateTimePicker1.Text;

                command.Parameters.Add("@dg", DbType.String).Value = textBox1.Text;

                command.Parameters.Add("@tp", DbType.String).Value = textBox2.Text;

                command.Parameters.Add("@nt", DbType.String).Value = textBox3.Text;

                command.ExecuteNonQuery();

                MessageBox.Show("������ ������� ��������!");

            }

        }

    }

}

**Medications.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.Entity.Core.Common.CommandTrees.ExpressionBuilder;

using System.Data.SQLite;

using System.Drawing;

using System.Linq;

using System.Net;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Hospital

{

    public partial class Medications : Form

    {

        public Medications()

        {

            InitializeComponent();

        }

        private ClassProvide \_db;

        string id;

        private byte[] imageBytes;

        private void Medications\_Load(object sender, EventArgs e)

        {

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn = \_db.GetConnection();

            string query = "SELECT medication\_id, name, dosage, side\_effects FROM Medications";

            SQLiteCommand cmd = new SQLiteCommand(query, conn);

            SQLiteDataReader reader = cmd.ExecuteReader();

            while (reader.Read())

            {

                dataGridView1.Rows.Add(reader["medication\_id"], reader["name"], reader["dosage"], reader["side\_effects"]);

            }

        }

        private void dataGridView1\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

        {

            if (dataGridView1.SelectedRows.Count > 0)

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                id = dataGridView1.SelectedRows[0].Cells["Column1"].Value.ToString();

                string query = "SELECT medication\_id, name, dosage, side\_effects, photo FROM Medications WHERE medication\_id = @id;";

                SQLiteCommand cmd = new SQLiteCommand(query, conn);

                cmd.Parameters.AddWithValue("@id", id);

                SQLiteDataReader reader = cmd.ExecuteReader();

                if (reader.Read())

                {

                    string nm = reader["name"].ToString();

                    string ds = reader["dosage"].ToString();

                    string se = reader["side\_effects"].ToString();

                    textBox1.Text = nm;

                    textBox2.Text = ds;

                    textBox3.Text = se;

                    byte[] pht = (byte[])reader["photo"];

                    MemoryStream ms = new MemoryStream(pht);

                    pictureBox1.Image = System.Drawing.Image.FromStream(ms);

                    pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;

                    pictureBox1.Refresh();

                }

            }

        }

        private void button3\_Click(object sender, EventArgs e)

        {

            if (dataGridView1.SelectedRows.Count > 0)

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string id = dataGridView1.SelectedRows[0].Cells["Column1"].Value.ToString();

                string sql = "DELETE FROM Medications WHERE medication\_id = @id";

                SQLiteCommand cmd = new SQLiteCommand(sql, conn);

                cmd.Parameters.AddWithValue("@id", id);

                cmd.ExecuteNonQuery();

                dataGridView1.Refresh();

                MessageBox.Show("Запись удалена из базы!");

            }

            else

            {

                MessageBox.Show("Выберите запись для удаления!");

            }

        }

        private void button1\_Click(object sender, EventArgs e)

        {

            string nm = textBox1.Text;

            string ds = textBox2.Text;

            string se = textBox3.Text;

            if (string.IsNullOrEmpty(nm) || string.IsNullOrEmpty(ds) || string.IsNullOrEmpty(se))

            {

                MessageBox.Show("Заполните все поля!");

            }

            else if (!Regex.IsMatch(nm, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Название правильно!");

            }

            else if (!Regex.IsMatch(ds, "^[а-яА-Я0-9 -]+$"))

            {

                MessageBox.Show("Заполните Объем правильно!");

            }

            else if (!Regex.IsMatch(se, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Симптомы правильно!");

            }

            else

            {

                using (OpenFileDialog openFileDialog = new OpenFileDialog())

                {

                    openFileDialog.Filter = "Image files (\*.jpg;\*.jpeg;\*.png)|\*.jpg;\*.jpeg;\*.png";

                    if (openFileDialog.ShowDialog() == DialogResult.OK)

                    {

                        imageBytes = File.ReadAllBytes(openFileDialog.FileName);

                        \_db = ClassProvide.GetInstance();

                        SQLiteConnection conn = \_db.GetConnection();

                        string query = "INSERT INTO Medications (name, dosage, side\_effects, photo) \r\nVALUES (@nm, @ds, @se, @phot);";

                        SQLiteCommand command = new SQLiteCommand(query, conn);

                        command.Parameters.AddWithValue("@nm", nm);

                        command.Parameters.AddWithValue("@ds", ds);

                        command.Parameters.AddWithValue("@se", se);

                        command.Parameters.AddWithValue("@phot", imageBytes);

                        command.ExecuteNonQuery();

                        MessageBox.Show("Запись добавлена в базу!");

                    }

                }

            }

        }

        private void button2\_Click(object sender, EventArgs e)

        {

            if (string.IsNullOrEmpty(textBox1.Text) || string.IsNullOrEmpty(textBox2.Text) || string.IsNullOrEmpty(textBox3.Text))

            {

                MessageBox.Show("Заполните все обязательные поля (\*)!");

            }

            else if (!Regex.IsMatch(textBox1.Text, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Название правильно!");

            }

            else if (!Regex.IsMatch(textBox2.Text, "^[а-яА-Я0-9 -]+$"))

            {

                MessageBox.Show("Заполните Объем правильно!");

            }

            else if (!Regex.IsMatch(textBox3.Text, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Симптомы правильно!");

            }

            else

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string query = "UPDATE Medications SET name = @nm, dosage = @ds, side\_effects = @se WHERE medication\_id = @id";

                SQLiteCommand command = new SQLiteCommand(query, conn);

                command.Parameters.AddWithValue("@id", id);

                command.Parameters.Add("@nm", DbType.String).Value = textBox1.Text;

                command.Parameters.Add("@ds", DbType.String).Value = textBox2.Text;

                command.Parameters.Add("@se", DbType.String).Value = textBox3.Text;

                command.ExecuteNonQuery();

                MessageBox.Show("Запись успешно изменена!");

            }

        }

    }

}

**Patients.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SQLite;

using System.Drawing;

using System.Linq;

using System.Runtime.Intrinsics.X86;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml.Linq;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Hospital

{

    public partial class Patients : Form

    {

        public Patients()

        {

            InitializeComponent();

        }

        private ClassProvide \_db;

        string id;

        private void Patients\_Load(object sender, EventArgs e)

        {

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn = \_db.GetConnection();

            string query = "SELECT patient\_id, first\_name, last\_name, date\_of\_birth, gender, address, phone\_number, email, insurance\_info FROM Patients";

            SQLiteCommand cmd = new SQLiteCommand(query, conn);

            SQLiteDataReader reader = cmd.ExecuteReader();

            while (reader.Read())

            {

                dataGridView1.Rows.Add(reader["patient\_id"], reader["first\_name"], reader["last\_name"], reader["date\_of\_birth"], reader["gender"], reader["address"], reader["phone\_number"], reader["email"], reader["insurance\_info"]);

            }

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn3 = \_db.GetConnection();

            string query3 = "SELECT DISTINCT gender FROM Patients";

            SQLiteCommand cmd3 = new SQLiteCommand(query3, conn3);

            SQLiteDataReader reader3 = cmd3.ExecuteReader();

            while (reader3.Read())

            {

                string data = reader3.GetString(0);

                comboBox1.Items.Add(data);

            }

        }

        private void button3\_Click(object sender, EventArgs e)

        {

            if (dataGridView1.SelectedRows.Count > 0)

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string id = dataGridView1.SelectedRows[0].Cells["Column1"].Value.ToString();

                string sql = "DELETE FROM Patients WHERE patient\_id = @id";

                SQLiteCommand cmd = new SQLiteCommand(sql, conn);

                cmd.Parameters.AddWithValue("@id", id);

                cmd.ExecuteNonQuery();

                dataGridView1.Refresh();

                MessageBox.Show("Запись удалена из базы!");

            }

            else

            {

                MessageBox.Show("Выберите запись для удаления!");

            }

        }

        private void button1\_Click(object sender, EventArgs e)

        {

            string fn = textBox1.Text;

            string ln = textBox6.Text;

            string db = dateTimePicker1.Text;

            string gn = comboBox1.Text;

            string ad = textBox5.Text;

            string np = textBox8.Text;

            string em = textBox3.Text;

            string str = textBox4.Text;

            if (string.IsNullOrEmpty(fn) || string.IsNullOrEmpty(ln) || string.IsNullOrEmpty(db) || string.IsNullOrEmpty(gn) || string.IsNullOrEmpty(ad) || string.IsNullOrEmpty(np) || string.IsNullOrEmpty(em) || string.IsNullOrEmpty(str))

            {

                MessageBox.Show("Заполните все поля!");

            }

            else if (!Regex.IsMatch(fn, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Имя правильно!");

            }

            else if (!Regex.IsMatch(ln, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Фамилию правильно!");

            }

            else if (!Regex.IsMatch(db, "^[0-9а-яА-Я .-]+$"))

            {

                MessageBox.Show("Заполните Дату рождения правильно!");

            }

            else if (!Regex.IsMatch(gn, "^[а-яА-Я]+$"))

            {

                MessageBox.Show("Заполните Пол правильно!");

            }

            else if (!Regex.IsMatch(ad, "^[а-яА-Я .,0-9-]+$"))

            {

                MessageBox.Show("Заполните Адрес правильно!");

            }

            else if (!Regex.IsMatch(np, "^[0-9 +]+$"))

            {

                MessageBox.Show("Заполните Номер телефона правильно!");

            }

            else if (!Regex.IsMatch(em, "^[0-9 a-zA-Z @.]+$"))

            {

                MessageBox.Show("Заполните Почту правильно!");

            }

            else if (!Regex.IsMatch(str, "^[а-яА-Я 0-9]+$"))

            {

                MessageBox.Show("Заполните Страховку правильно!");

            }

            else

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string query = "INSERT INTO Patients (first\_name, last\_name, date\_of\_birth, gender, address, phone\_number, email, insurance\_info) \r\nVALUES (@fn, @ln, @db, @gn, @ad, @np, @em, @str);";

                SQLiteCommand command = new SQLiteCommand(query, conn);

                command.Parameters.AddWithValue("@fn", fn);

                command.Parameters.AddWithValue("@ln", ln);

                command.Parameters.AddWithValue("@db", db);

                command.Parameters.AddWithValue("@gn", gn);

                command.Parameters.AddWithValue("@ad", ad);

                command.Parameters.AddWithValue("@np", np);

                command.Parameters.AddWithValue("@em", em);

                command.Parameters.AddWithValue("@str", str);

                command.ExecuteNonQuery();

                MessageBox.Show("Запись добавлена в базу!");

            }

        }

        private void dataGridView1\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

        {

            if (dataGridView1.SelectedRows.Count > 0)

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                id = dataGridView1.SelectedRows[0].Cells["Column1"].Value.ToString();

                string query = "SELECT patient\_id, first\_name, last\_name, date\_of\_birth, gender, address, phone\_number, email, insurance\_info FROM Patients WHERE patient\_id = @id";

                SQLiteCommand cmd = new SQLiteCommand(query, conn);

                cmd.Parameters.AddWithValue("@id", id);

                SQLiteDataReader reader = cmd.ExecuteReader();

                if (reader.Read())

                {

                    string fn = reader["first\_name"].ToString();

                    string ln = reader["last\_name"].ToString();

                    string db = reader["date\_of\_birth"].ToString();

                    string gn = reader["gender"].ToString();

                    string ad = reader["address"].ToString();

                    string pn = reader["phone\_number"].ToString();

                    string em = reader["email"].ToString();

                    string str = reader["insurance\_info"].ToString();

                    textBox1.Text = fn;

                    textBox6.Text = ln;

                    dateTimePicker1.Text = db;

                    comboBox1.Text = gn;

                    textBox5.Text = ad;

                    textBox8.Text = pn;

                    textBox3.Text = em;

                    textBox4.Text = str;

                }

            }

        }

        private void button2\_Click(object sender, EventArgs e)

        {

            if (string.IsNullOrEmpty(textBox1.Text) || string.IsNullOrEmpty(textBox3.Text) || string.IsNullOrEmpty(textBox4.Text) || string.IsNullOrEmpty(textBox8.Text) || string.IsNullOrEmpty(textBox6.Text) || string.IsNullOrEmpty(textBox5.Text) || string.IsNullOrEmpty(comboBox1.Text) || string.IsNullOrEmpty(dateTimePicker1.Text))

            {

                MessageBox.Show("Заполните все поля!");

            }

            else if (!Regex.IsMatch(textBox1.Text, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Имя правильно!");

            }

            else if (!Regex.IsMatch(textBox6.Text, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Фамилию правильно!");

            }

            else if (!Regex.IsMatch(dateTimePicker1.Text, "^[0-9а-яА-Я .-]+$"))

            {

                MessageBox.Show("Заполните Дату рождения правильно!");

            }

            else if (!Regex.IsMatch(comboBox1.Text, "^[а-яА-Я]+$"))

            {

                MessageBox.Show("Заполните Пол правильно!");

            }

            else if (!Regex.IsMatch(textBox5.Text, "^[а-яА-Я .,0-9-]+$"))

            {

                MessageBox.Show("Заполните Адрес правильно!");

            }

            else if (!Regex.IsMatch(textBox8.Text, "^[0-9 +]+$"))

            {

                MessageBox.Show("Заполните Номер телефона правильно!");

            }

            else if (!Regex.IsMatch(textBox3.Text, "^[0-9 a-zA-Z @.]+$"))

            {

                MessageBox.Show("Заполните Почту правильно!");

            }

            else if (!Regex.IsMatch(textBox4.Text, "^[а-яА-Я 0-9]+$"))

            {

                MessageBox.Show("Заполните Страховку правильно!");

            }

            else

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string query = "UPDATE Patients SET first\_name = @fn, last\_name = @ln, date\_of\_birth = @db, gender = @gn, address = @ad, phone\_number = @pn, email = @em, insurance\_info = @str WHERE patient\_id = @id";

                SQLiteCommand command = new SQLiteCommand(query, conn);

                command.Parameters.AddWithValue("@id", id);

                command.Parameters.Add("@fn", DbType.String).Value = textBox1.Text;

                command.Parameters.Add("@ln", DbType.String).Value = textBox6.Text;

                command.Parameters.Add("@db", DbType.String).Value = dateTimePicker1.Text;

                command.Parameters.Add("@gn", DbType.String).Value = comboBox1.Text;

                command.Parameters.Add("@ad", DbType.String).Value = textBox5.Text;

                command.Parameters.Add("@pn", DbType.String).Value = textBox8.Text;

                command.Parameters.Add("@em", DbType.String).Value = textBox3.Text;

                command.Parameters.Add("@str", DbType.String).Value = textBox4.Text;

                command.ExecuteNonQuery();

                MessageBox.Show("Запись успешно изменена!");

            }

        }

    }

}

**Payments.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SQLite;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Hospital

{

    public partial class Payments : Form

    {

        public Payments()

        {

            InitializeComponent();

        }

        private ClassProvide \_db;

        string id;

        private void Payments\_Load(object sender, EventArgs e)

        {

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn = \_db.GetConnection();

            string query = "SELECT payment\_id, Patients.last\_name as lnp, amount, payment\_date, payment\_method FROM Payments INNER JOIN Patients ON Payments.patient\_id = Patients.patient\_id";

            SQLiteCommand cmd = new SQLiteCommand(query, conn);

            SQLiteDataReader reader = cmd.ExecuteReader();

            while (reader.Read())

            {

                dataGridView1.Rows.Add(reader["payment\_id"], reader["lnp"], reader["amount"], reader["payment\_date"], reader["payment\_method"]);

            }

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn1 = \_db.GetConnection();

            string query1 = "SELECT DISTINCT last\_name FROM Patients";

            SQLiteCommand cmd1 = new SQLiteCommand(query1, conn1);

            SQLiteDataReader reader1 = cmd1.ExecuteReader();

            while (reader1.Read())

            {

                string data = reader1.GetString(0);

                comboBox1.Items.Add(data);

            }

            comboBox2.Items.Add("Наличные");

            comboBox2.Items.Add("Кредитная карта");

            comboBox2.Items.Add("Страхование");

        }

        private void button3\_Click(object sender, EventArgs e)

        {

            if (dataGridView1.SelectedRows.Count > 0)

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string id = dataGridView1.SelectedRows[0].Cells["Column1"].Value.ToString();

                string sql = "DELETE FROM Payments WHERE payment\_id = @id";

                SQLiteCommand cmd = new SQLiteCommand(sql, conn);

                cmd.Parameters.AddWithValue("@id", id);

                cmd.ExecuteNonQuery();

                dataGridView1.Refresh();

                MessageBox.Show("Запись удалена из базы!");

            }

            else

            {

                MessageBox.Show("Выберите запись для удаления!");

            }

        }

        private void button1\_Click(object sender, EventArgs e)

        {

            string pc = comboBox1.Text;

            string am = textBox1.Text;

            string pd = dateTimePicker1.Text;

            string pm = comboBox2.Text;

            if (string.IsNullOrEmpty(pc) || string.IsNullOrEmpty(am) || string.IsNullOrEmpty(pd) || string.IsNullOrEmpty(pm))

            {

                MessageBox.Show("Заполните все поля!");

            }

            else if (!Regex.IsMatch(pc, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Пациента правильно!");

            }

            else if (!Regex.IsMatch(am, "^[0-9 -]+$"))

            {

                MessageBox.Show("Заполните Сумму правильно!");

            }

            else if (!Regex.IsMatch(pd, "^[0-9а-яА-Я .-]+$"))

            {

                MessageBox.Show("Заполните Дату оплаты правильно!");

            }

            else if (!Regex.IsMatch(pm, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Способ оплаты правильно!");

            }

            else

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string query = "INSERT INTO Payments (patient\_id, amount, payment\_date, payment\_method) \r\nVALUES ((SELECT patient\_id FROM Patients WHERE last\_name = @pc), @am, @pd, @pm);";

                SQLiteCommand command = new SQLiteCommand(query, conn);

                command.Parameters.AddWithValue("@pc", pc);

                command.Parameters.AddWithValue("@am", am);

                command.Parameters.AddWithValue("@pd", pd);

                command.Parameters.AddWithValue("@pm", pm);

                command.ExecuteNonQuery();

                MessageBox.Show("Запись добавлена в базу!");

            }

        }

        private void button2\_Click(object sender, EventArgs e)

        {

            if (string.IsNullOrEmpty(comboBox1.Text) || string.IsNullOrEmpty(textBox1.Text) || string.IsNullOrEmpty(dateTimePicker1.Text) || string.IsNullOrEmpty(comboBox2.Text))

            {

                MessageBox.Show("Заполните все поля!");

            }

            else if (!Regex.IsMatch(comboBox1.Text, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Пациента правильно!");

            }

            else if (!Regex.IsMatch(textBox1.Text, "^[0-9 -]+$"))

            {

                MessageBox.Show("Заполните Сумму правильно!");

            }

            else if (!Regex.IsMatch(dateTimePicker1.Text, "^[0-9а-яА-Я .-]+$"))

            {

                MessageBox.Show("Заполните Дату оплаты правильно!");

            }

            else if (!Regex.IsMatch(comboBox2.Text, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Способ оплаты правильно!");

            }

            else

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string query = "UPDATE Payments SET patient\_id = (SELECT patient\_id FROM Patients WHERE last\_name = @pc), amount = @am, payment\_date = @pd, payment\_method = @pm WHERE payment\_id = @id";

                SQLiteCommand command = new SQLiteCommand(query, conn);

                command.Parameters.AddWithValue("@id", id);

                command.Parameters.Add("@pc", DbType.String).Value = comboBox1.Text;

                command.Parameters.Add("@am", DbType.String).Value = textBox1.Text;

                command.Parameters.Add("@pd", DbType.String).Value = dateTimePicker1.Text;

                command.Parameters.Add("@pm", DbType.String).Value = comboBox2.Text;

                command.ExecuteNonQuery();

                MessageBox.Show("Запись успешно изменена!");

            }

        }

        private void dataGridView1\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

        {

            if (dataGridView1.SelectedRows.Count > 0)

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                id = dataGridView1.SelectedRows[0].Cells["Column1"].Value.ToString();

                string query = "SELECT payment\_id, Patients.last\_name as lnp, amount, payment\_date, payment\_method FROM Payments INNER JOIN Patients ON Payments.patient\_id = Patients.patient\_id WHERE payment\_id = @id";

                SQLiteCommand cmd = new SQLiteCommand(query, conn);

                cmd.Parameters.AddWithValue("@id", id);

                SQLiteDataReader reader = cmd.ExecuteReader();

                if (reader.Read())

                {

                    string lnp = reader["lnp"].ToString();

                    string amount = reader["amount"].ToString();

                    string payment\_date = reader["payment\_date"].ToString();

                    string payment\_method = reader["payment\_method"].ToString();

                    comboBox1.Text = lnp;

                    textBox1.Text = amount;

                    dateTimePicker1.Text = payment\_date;

                    comboBox2.Text = payment\_method;

                }

            }

        }

    }

}

**Prescriptions.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SQLite;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Hospital

{

    public partial class Prescriptions : Form

    {

        public Prescriptions()

        {

            InitializeComponent();

        }

        private ClassProvide \_db;

        string id;

        private void Prescriptions\_Load(object sender, EventArgs e)

        {

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn = \_db.GetConnection();

            string query = "SELECT prescription\_id, MedicalRecords.diagnosis as dia, Medications.name as nm, dosage\_instructions FROM Prescriptions INNER JOIN MedicalRecords ON Prescriptions.record\_id = MedicalRecords.record\_id INNER JOIN Medications ON Prescriptions.medication\_id = Medications.medication\_id";

            SQLiteCommand cmd = new SQLiteCommand(query, conn);

            SQLiteDataReader reader = cmd.ExecuteReader();

            while (reader.Read())

            {

                dataGridView1.Rows.Add(reader["prescription\_id"], reader["dia"], reader["nm"], reader["dosage\_instructions"]);

            }

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn1 = \_db.GetConnection();

            string query1 = "SELECT DISTINCT name FROM Medications";

            SQLiteCommand cmd1 = new SQLiteCommand(query1, conn1);

            SQLiteDataReader reader1 = cmd1.ExecuteReader();

            while (reader1.Read())

            {

                string data = reader1.GetString(0);

                comboBox2.Items.Add(data);

            }

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn2 = \_db.GetConnection();

            string query2 = "SELECT DISTINCT diagnosis FROM MedicalRecords";

            SQLiteCommand cmd2 = new SQLiteCommand(query2, conn2);

            SQLiteDataReader reader2 = cmd2.ExecuteReader();

            while (reader2.Read())

            {

                string data = reader2.GetString(0);

                comboBox1.Items.Add(data);

            }

        }

        private void button2\_Click(object sender, EventArgs e)

        {

            if (dataGridView1.SelectedRows.Count > 0)

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string id = dataGridView1.SelectedRows[0].Cells["Column1"].Value.ToString();

                string sql = "DELETE FROM Prescriptions WHERE prescription\_id = @id";

                SQLiteCommand cmd = new SQLiteCommand(sql, conn);

                cmd.Parameters.AddWithValue("@id", id);

                cmd.ExecuteNonQuery();

                dataGridView1.Refresh();

                MessageBox.Show("Запись удалена из базы!");

            }

            else

            {

                MessageBox.Show("Выберите запись для удаления!");

            }

        }

        private void dataGridView1\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

        {

            if (dataGridView1.SelectedRows.Count > 0)

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                id = dataGridView1.SelectedRows[0].Cells["Column1"].Value.ToString();

                string query = "SELECT prescription\_id, MedicalRecords.diagnosis as dia, Medications.name as nm, dosage\_instructions FROM Prescriptions INNER JOIN MedicalRecords ON Prescriptions.record\_id = MedicalRecords.record\_id INNER JOIN Medications ON Prescriptions.medication\_id = Medications.medication\_id WHERE prescription\_id = @id";

                SQLiteCommand cmd = new SQLiteCommand(query, conn);

                cmd.Parameters.AddWithValue("@id", id);

                SQLiteDataReader reader = cmd.ExecuteReader();

                if (reader.Read())

                {

                    string dia = reader["dia"].ToString();

                    string nm = reader["nm"].ToString();

                    string dosage\_instructions = reader["dosage\_instructions"].ToString();

                    comboBox1.Text = dia;

                    comboBox2.Text = nm;

                    textBox1.Text = dosage\_instructions;

                }

            }

        }

        private void button1\_Click(object sender, EventArgs e)

        {

            string dg = comboBox1.Text;

            string nm = comboBox2.Text;

            string dosage\_instructions = textBox1.Text;

            if (string.IsNullOrEmpty(dg) || string.IsNullOrEmpty(nm) || string.IsNullOrEmpty(dosage\_instructions))

            {

                MessageBox.Show("Заполните все поля!");

            }

            else if (!Regex.IsMatch(nm, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Диагноз правильно!");

            }

            else if (!Regex.IsMatch(nm, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Лекарство правильно!");

            }

            else if (!Regex.IsMatch(dosage\_instructions, "^[0-9а-яА-Я .-]+$"))

            {

                MessageBox.Show("Заполните Инструкцию к применению правильно!");

            }

            else

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string query = "INSERT INTO Prescriptions (record\_id, medication\_id, dosage\_instructions) \r\nVALUES ((SELECT record\_id FROM MedicalRecords WHERE diagnosis = @dg), (SELECT medication\_id FROM Medications WHERE name = @nm), @dosage\_instructions);";

                SQLiteCommand command = new SQLiteCommand(query, conn);

                command.Parameters.AddWithValue("@dg", dg);

                command.Parameters.AddWithValue("@nm", nm);

                command.Parameters.AddWithValue("@dosage\_instructions", dosage\_instructions);

                command.ExecuteNonQuery();

                MessageBox.Show("Запись добавлена в базу!");

            }

        }

        private void button3\_Click(object sender, EventArgs e)

        {

            if (string.IsNullOrEmpty(comboBox1.Text) || string.IsNullOrEmpty(comboBox2.Text) || string.IsNullOrEmpty(textBox1.Text))

            {

                MessageBox.Show("Заполните все поля!");

            }

            else if (!Regex.IsMatch(comboBox1.Text, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Диагноз правильно!");

            }

            else if (!Regex.IsMatch(comboBox2.Text, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Лекарство правильно!");

            }

            else if (!Regex.IsMatch(textBox1.Text, "^[0-9а-яА-Я .-]+$"))

            {

                MessageBox.Show("Заполните Инструкцию к применению правильно!");

            }

            else

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string query = "UPDATE Prescriptions SET record\_id = (SELECT record\_id FROM MedicalRecords WHERE diagnosis = @dg), medication\_id = (SELECT medication\_id FROM Medications WHERE name = @nm), dosage\_instructions = @dosage\_instructions WHERE prescription\_id = @id";

                SQLiteCommand command = new SQLiteCommand(query, conn);

                command.Parameters.AddWithValue("@id", id);

                command.Parameters.Add("@dg", DbType.String).Value = comboBox1.Text;

                command.Parameters.Add("@nm", DbType.String).Value = comboBox2.Text;

                command.Parameters.Add("@dosage\_instructions", DbType.String).Value = textBox1.Text;

                command.ExecuteNonQuery();

                MessageBox.Show("Запись успешно изменена!");

            }

        }

    }

}

**Program.cs**

namespace Hospital

{

    internal static class Program

    {

        /// <summary>

        ///  The main entry point for the application.

        /// </summary>

        [STAThread]

        static void Main()

        {

            // To customize application configuration such as set high DPI settings or default font,

            // see https://aka.ms/applicationconfiguration.

            ApplicationConfiguration.Initialize();

            Application.Run(new Form1());

        }

    }

}

**Rooms.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SQLite;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Hospital

{

    public partial class Rooms : Form

    {

        public Rooms()

        {

            InitializeComponent();

        }

        private ClassProvide \_db;

        string id;

        private void Rooms\_Load(object sender, EventArgs e)

        {

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn = \_db.GetConnection();

            string query = "SELECT room\_id, room\_number, type, availability\_status FROM Rooms";

            SQLiteCommand cmd = new SQLiteCommand(query, conn);

            SQLiteDataReader reader = cmd.ExecuteReader();

            while (reader.Read())

            {

                dataGridView1.Rows.Add(reader["room\_id"], reader["room\_number"], reader["type"], reader["availability\_status"]);

            }

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn3 = \_db.GetConnection();

            string query3 = "SELECT DISTINCT availability\_status FROM Rooms";

            SQLiteCommand cmd3 = new SQLiteCommand(query3, conn3);

            SQLiteDataReader reader3 = cmd3.ExecuteReader();

            while (reader3.Read())

            {

                string data = reader3.GetString(0);

                comboBox3.Items.Add(data);

            }

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn1 = \_db.GetConnection();

            string query1 = "SELECT DISTINCT type FROM Rooms";

            SQLiteCommand cmd1 = new SQLiteCommand(query1, conn1);

            SQLiteDataReader reader1 = cmd1.ExecuteReader();

            while (reader1.Read())

            {

                string data = reader1.GetString(0);

                comboBox2.Items.Add(data);

            }

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn2 = \_db.GetConnection();

            string query2 = "SELECT DISTINCT room\_number FROM Rooms";

            SQLiteCommand cmd2 = new SQLiteCommand(query2, conn2);

            SQLiteDataReader reader2 = cmd2.ExecuteReader();

            while (reader2.Read())

            {

                string data = reader2.GetString(0);

                comboBox1.Items.Add(data);

            }

        }

        private void button1\_Click(object sender, EventArgs e)

        {

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn = \_db.GetConnection();

            string query = "UPDATE Rooms SET room\_number = @rn, type = @tp, availability\_status = @as WHERE room\_id = @id";

            SQLiteCommand command = new SQLiteCommand(query, conn);

            command.Parameters.AddWithValue("@id", id);

            command.Parameters.Add("@rn", DbType.String).Value = comboBox1.Text;

            command.Parameters.Add("@tp", DbType.String).Value = comboBox2.Text;

            command.Parameters.Add("@as", DbType.String).Value = comboBox3.Text;

            command.ExecuteNonQuery();

            MessageBox.Show("Запись успешно изменена!");

        }

        private void dataGridView1\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

        {

            if (dataGridView1.SelectedRows.Count > 0)

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                id = dataGridView1.SelectedRows[0].Cells["Column1"].Value.ToString();

                string query = "SELECT room\_id, room\_number, type, availability\_status FROM Rooms WHERE room\_id = @id";

                SQLiteCommand cmd = new SQLiteCommand(query, conn);

                cmd.Parameters.AddWithValue("@id", id);

                SQLiteDataReader reader = cmd.ExecuteReader();

                if (reader.Read())

                {

                    string rn = reader["room\_number"].ToString();

                    string tp = reader["type"].ToString();

                    string avs = reader["availability\_status"].ToString();

                    comboBox1.Text = rn;

                    comboBox2.Text = tp;

                    comboBox3.Text = avs;

                }

            }

        }

    }

}

**Staff.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SQLite;

using System.Drawing;

using System.Linq;

using System.Runtime.Intrinsics.Arm;

using System.Runtime.Intrinsics.X86;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml.Linq;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Hospital

{

    public partial class Staff : Form

    {

        public Staff()

        {

            InitializeComponent();

        }

        private ClassProvide \_db;

        string id;

        private void Staff\_Load(object sender, EventArgs e)

        {

            \_db = ClassProvide.GetInstance();

            SQLiteConnection conn = \_db.GetConnection();

            string query = "SELECT staff\_id, first\_name, last\_name, position, department FROM Staff";

            SQLiteCommand cmd = new SQLiteCommand(query, conn);

            SQLiteDataReader reader = cmd.ExecuteReader();

            while (reader.Read())

            {

                dataGridView1.Rows.Add(reader["staff\_id"], reader["first\_name"], reader["last\_name"], reader["position"], reader["department"]);

            }

        }

        private void button3\_Click(object sender, EventArgs e)

        {

            if (dataGridView1.SelectedRows.Count > 0)

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string id = dataGridView1.SelectedRows[0].Cells["Column1"].Value.ToString();

                string sql = "DELETE FROM Staff WHERE staff\_id = @id";

                SQLiteCommand cmd = new SQLiteCommand(sql, conn);

                cmd.Parameters.AddWithValue("@id", id);

                cmd.ExecuteNonQuery();

                dataGridView1.Refresh();

                MessageBox.Show("Запись удалена из базы!");

            }

            else

            {

                MessageBox.Show("Выберите запись для удаления!");

            }

        }

        private void button1\_Click(object sender, EventArgs e)

        {

            string fn = textBox1.Text;

            string ln = textBox2.Text;

            string pos = textBox3.Text;

            string dep = textBox4.Text;

            if (string.IsNullOrEmpty(fn) || string.IsNullOrEmpty(ln) || string.IsNullOrEmpty(pos) || string.IsNullOrEmpty(dep))

            {

                MessageBox.Show("Заполните все поля!");

            }

            else if (!Regex.IsMatch(fn, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Имя правильно!");

            }

            else if (!Regex.IsMatch(ln, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Фамилию правильно!");

            }

            else if (!Regex.IsMatch(pos, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Позицию правильно!");

            }

            else if (!Regex.IsMatch(dep, "^[а-яА-Я]+$"))

            {

                MessageBox.Show("Заполните Отдел правильно!");

            }

            else

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string query = "INSERT INTO Staff (first\_name, last\_name, positions, department) \r\nVALUES (@fn, @ln, @pos, @dep);";

                SQLiteCommand command = new SQLiteCommand(query, conn);

                command.Parameters.AddWithValue("@fn", fn);

                command.Parameters.AddWithValue("@ln", ln);

                command.Parameters.AddWithValue("@pos", pos);

                command.Parameters.AddWithValue("@dep", dep);

                command.ExecuteNonQuery();

                MessageBox.Show("Запись добавлена в базу!");

            }

        }

        private void dataGridView1\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

        {

            if (dataGridView1.SelectedRows.Count > 0)

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                id = dataGridView1.SelectedRows[0].Cells["Column1"].Value.ToString();

                string query = "SELECT staff\_id, first\_name, last\_name, position, department FROM Staff WHERE staff\_id = @id";

                SQLiteCommand cmd = new SQLiteCommand(query, conn);

                cmd.Parameters.AddWithValue("@id", id);

                SQLiteDataReader reader = cmd.ExecuteReader();

                if (reader.Read())

                {

                    string fn = reader["first\_name"].ToString();

                    string ln = reader["last\_name"].ToString();

                    string sp = reader["position"].ToString();

                    string pn = reader["department"].ToString();

                    textBox1.Text = fn;

                    textBox2.Text = ln;

                    textBox3.Text = sp;

                    textBox4.Text = pn;

                }

            }

        }

        private void button2\_Click(object sender, EventArgs e)

        {

            if (string.IsNullOrEmpty(textBox1.Text) || string.IsNullOrEmpty(textBox2.Text) || string.IsNullOrEmpty(textBox3.Text) || string.IsNullOrEmpty(textBox4.Text))

            {

                MessageBox.Show("Заполните все поля!");

            }

            else if (!Regex.IsMatch(textBox1.Text, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Имя правильно!");

            }

            else if (!Regex.IsMatch(textBox2.Text, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Фамилию правильно!");

            }

            else if (!Regex.IsMatch(textBox3.Text, "^[а-яА-Я -]+$"))

            {

                MessageBox.Show("Заполните Позицию правильно!");

            }

            else if (!Regex.IsMatch(textBox4.Text, "^[а-яА-Я]+$"))

            {

                MessageBox.Show("Заполните Отдел правильно!");

            }

            else

            {

                \_db = ClassProvide.GetInstance();

                SQLiteConnection conn = \_db.GetConnection();

                string query = "UPDATE Staff SET first\_name = @fn, last\_name = @ln, position = @po, department = @dep WHERE staff\_id = @id";

                SQLiteCommand command = new SQLiteCommand(query, conn);

                command.Parameters.AddWithValue("@id", id);

                command.Parameters.Add("@fn", DbType.String).Value = textBox1.Text;

                command.Parameters.Add("@ln", DbType.String).Value = textBox2.Text;

                command.Parameters.Add("@po", DbType.String).Value = textBox3.Text;

                command.Parameters.Add("@dep", DbType.String).Value = textBox4.Text;

                command.ExecuteNonQuery();

                MessageBox.Show("Запись успешно изменена!");

            }

        }

    }

}