СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном обществе огромное число предприятий прибегает к использованию персональных компьютеров для обработки и хранения разнообразной информации. Эти данные упорядочиваются в базах данных, которые играют важную роль в динамично-развивающемся мире технологий. По всей видимости, всё, с чем мы взаимодействуем ежедневно в нашей жизни, зафиксировано в какой-либо базе. Владение навыками работы с базами данных становится фундаментальным для работы с компьютерами, и специалисты в этой сфере становятся все более востребованными. Основные принципы информационной методологии сегодня основаны на идее, что информация должна быть систематизирована в базах данных для отображения динамически меняющегося мира и удовлетворения потребностей пользователей. Формирование и функционирование баз данных осуществляются с применением специализированных программных средств, известных как системы управления базами данных.

Ба́за да́нных – совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных [1].

Система управления базами данных (СУБД) – это набор инструментов, которые позволяют удобно управлять базами данных: удалять, добавлять, фильтровать и находить элементы, менять их структуру и создавать резервные копии [2].

Реляционные базы данных – это тип базы данных, в которой хранятся и систематизируются точки данных с определенными связями для быстрого доступа [3].

Цель курсового проекта заключается в разработке базы данных, обеспечивающей работу с информацией, касающейся структуры медицинских организаций.

Задачи проекта включают в себя систематизацию, закрепление и расширение теоретических и практических знаний по созданию баз данных, а также развитие навыков самостоятельной работы.

Актуальность проекта обусловлена неотложной потребностью учебных медицинских организаций в ведении учета пациентов и врачей.

В основе процесса создания базы данных лежат определенные принципы. Первый принцип состоит в том, чтобы избегать повторяющихся сведений (также называемых избыточными данными), поскольку они занимают много места и повышают вероятность появления ошибок и несоответствий. Второй принцип провозглашает важность правильности и полноты сведений [4].

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ВЫЯВЛЕНИЕ НЕОБХОДИМОГО НАБОРА СУЩНОСТЕЙ

Данная база существенно упрощает и совершенствует работу в здравоохранении. Базы данных в медицинской сфере играют ключевую роль в управлении пациентской информацией, позволяя эффективно хранить и обрабатывать огромные объемы медицинских данных. Это содействует быстрому доступу к истории болезни, что в свою очередь способствует более точным и оперативным медицинским решениям.

Базы данных – также упрощают административные процессы в медицинских учреждениях, автоматизируя учет пациентов, назначение врачей. Это снижает вероятность ошибок, связанных с ручным ведением медицинских записей, и значительно улучшает координацию медицинского персонала. Базы данных также способствуют улучшению обмена информацией между различными отделениями и специалистами, что в итоге повышает общую эффективность медицинского обслуживания в городе.

Перед началом разработки базы данных, необходимо определить основные цели и задачи для решаемой проблемы, после чего приступать к проектированию. Поэтому сформулируем краткое описание поставленной задачи.

Наименование задачи: Автоматизация работы медицинских организаций.

Цель работы: Накопление и систематизация информации о пациентах их болезней и рабочей активности каждого врача.

Базе данных можно задать следующие запросы:

1. Получить перечень и общее число врачей указанного профиля для конкретного медицинского учреждения, больницы, либо поликлиники, либо всех медицинских учреждений города.

2. Получить перечень и общее число врачей указанного профиля со степенью кандидата или доктора медицинских наук конкретного медицинского учреждения, либо больницы, либо поликлиники, либо всех медицинских учреждений города.

3. Получить перечень пациентов, наблюдающихся у врача в конкретном медицинском учреждении.

4. Получить данные о выработке (среднее число принятых пациентов в день) за указанный период для конкретного врача, либо всех врачей поликлиники, либо для всех врачей названного профиля.

5. Получить данные о загрузке (число пациентов, у которых врач в настоящее время является лечащим врачом) для указанного врача, либо всех врачей больницы, либо для всех врачей названного профиля.

Определим минимальный набор сущностей, необходимый для проектирования информационной системы медицинских организаций города. Основной задачей является учет медицинской информации и обеспечение

эффективного взаимодействия между различными подразделениями здравоохранения.

Сущность «Больница» представляет собой основное учреждение, состоящее из различных корпусов. Каждый корпус в свою очередь имеет отделения, специализирующиеся на лечении конкретных заболеваний. Важными атрибутами сущности «Отделение» являются количество палат и коек, а также привязка к конкретному корпусу.

Для эффективной работы медицинского персонала необходимо учесть сущность «Врач», которая включает в себя информацию о ФИО врача, его специализации и степени (кандидат или доктор медицинских наук). Степень врача влияет на его академическое звание, а также возможность консультирования в нескольких медицинских учреждениях.

Сущность «Пациент» представляет собой информацию о людях, получающих медицинскую помощь. Важными атрибутами являются ФИО пациента. Для подробного учета состояния здоровья каждого пациента создается сущность «Карта», содержащая информацию о диагнозах и лечащем враче.

Также создаются сущности «Степень» для хранения информации о научных степенях врачей и «Пользователь» для управления правами доступа к системе, включая администраторские права.

Таблица 1.1 – Атрибуты сущностей

Название сущности	Атрибуты сущности
Hospital	id Name Address
Corpus	Id Name IdHospital
Otdelenie	Id Name NumberPalats NumberBads IdCorpus
Bolezn	Id Name IdOtdelenie
Degree	Id Name

Продолжение таблицы 1.1

Название сущности	Атрибуты сущности
Doctor	Id Name Spacialization Degree
Patcient	Id Name
Karta	Id Vrema_Postuplenia IdBolezni IdPatcient
Policlinic	Id Name IdHospital
DoctorH	IdDoctor IdHospital
DoctorP	IdDoctor IdPoliclinic
Users	Id userlogin password AdministratorRights

Анализ предметной области является неотъемлемой частью процесса разработки программного обеспечения. Это процесс изучения и понимания специфики конкретной предметной области, в которой будет существовать и функционировать разрабатываемая система или приложение. [5].

2 ПОСТРОЕНИЕ СХЕМЫ РЕЛЯЦИОННОЙ БД

2.1 Построение набора необходимых отношений базы данных

Для построения структуры реляционной базы данных требуется выделить набор отношений, которые будут формировать основу базы данных. Все необходимые данные для хранения в базе будут охвачены этим набором отношений. Исходя из представленных сущностей и их атрибутов, указанных в таблице 1.1, можно определить соответствующий набор отношений. На рисунке 2.1 представлены отношения, которые будут использоваться в базе данных информационной системы медицинских организаций города.

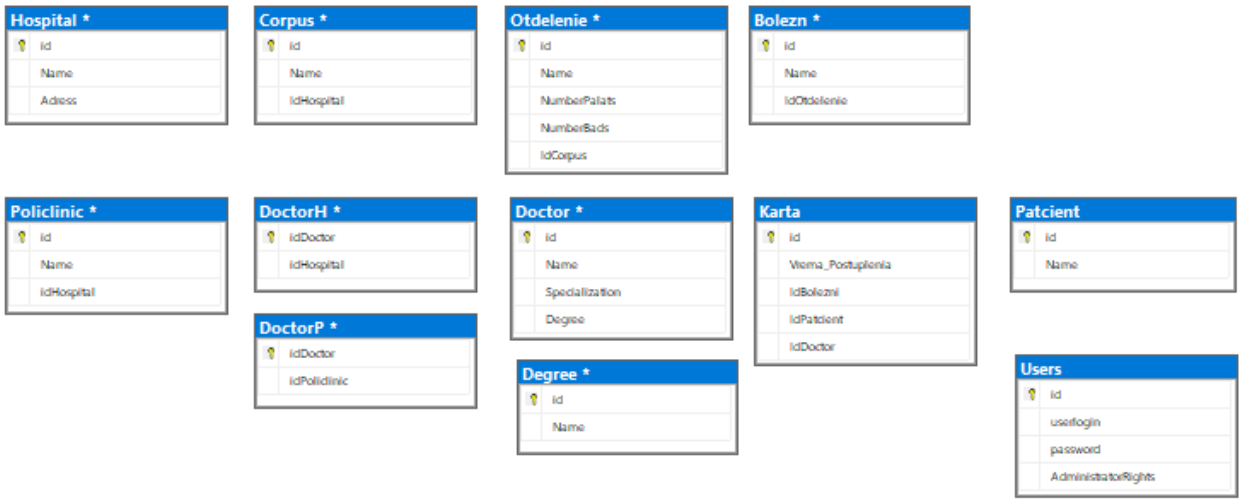

Рисунок 2.1 – Набор необходимых отношений базы данных

При построении схемы акцент был сделан на тщательном проектировании отношений в базе данных, придерживаясь принципов реляционной модели. Особое внимание уделяется избеганию отношений многие ко многим, что способствует упорядоченному и эффективному хранению данных. Реляционная структура базы данных обеспечивает легкость в обработке информации и повышает ее целостность, что является ключевым аспектом при создании базы данных для медицинских организаций.

2.2 Задание первичных и внешних ключей определенных отношений

Первичный ключ в базе данных выполняет важную роль, обеспечивая уникальную идентификацию каждой записи и устанавливая связи между таблицами. В контексте взаимосвязанных таблиц первичный ключ из родительской таблицы становится внешним ключом в дочерней. Этот внешний ключ в дочерней таблице ссылается на соответствующие данные родительской таблицы, обеспечивая эффективное организацию и связность данных в базе.

Первичные и внешние ключи отношений представлены в таблице 2.1.

№ п/п	Название таблицы	Первичный ключ	Внешние ключи
1	Hospital	id	-
2	Corpus	id	IdHospital
3	Otdelenie	id	IdCorpus
4	Bolezn	id	IdOtdelenie
5	Degree	id	-
6	Doctor	id	Degree
7	Patcient	id	-
8	Karta	id	IdBolezni IdPatcient IdDoctor
9	Policlinic	id	idHospital
10	DoctorH	idDoctor	idHospital
11	DoctorP	idDoctor	idPoliclinic
12	Users	id	-

2.3 Приведение отношений БД к третьей нормальной форме

Нормализация в контексте баз данных – это процесс организации структуры данных с целью устранения избыточности и зависимостей между данными, что способствует более эффективному хранению, обновлению и управлению информацией. Процесс нормализации включает в себя несколько нормальных форм (1НФ, 2НФ, 3НФ и т.д.), каждая из которых устанавливает определенные правила для организации данных.

Первая нормальная форма (1НФ) гарантирует атомарность данных. Вторая нормальная форма (2НФ) требует полной зависимости не ключевых атрибутов от всего составного первичного ключа. Третья нормальная форма (3НФ) устраняет транзитивные зависимости, обеспечивая, чтобы каждый атрибут зависел только от первичного ключа. Высшие нормальные формы (4НФ, 5НФ и так далее) решают более сложные случаи зависимостей данных.

Процесс нормализации начинается с анализа предметной области и определения сущностей и их атрибутов. Затем создается начальная таблица, и с помощью правил нормализации эта таблица приводится к более высоким нормальным формам. Нормализация обеспечивает эффективное хранение данных, предотвращает избыточность и аномалии, а также облегчает поддержание и модификацию базы данных.

2.4 Определение ограничений целостности для внешних ключей отношений и для отношений в целом

Согласованность базы данных определяется соответствием внутренней логики, структуры и установленных правил информации в базе данных. В этом контексте ограничения целостности представляют собой правила, накладываемые на возможные состояния базы данных. Системы управления базами данных (СУБД) не могут осуществлять проверку каждого вводимого значения в базу данных, но существуют инструменты, такие как триггеры, проверки и уникальность, которые помогают уменьшить вероятность нарушения целостности данных.

Основной задачей СУБД является обеспечение согласованности, правильности и точности данных в базе данных в любой момент времени. Этот процесс также называется поддержанием целостности базы данных.

Важно отметить разницу между задачами обеспечения целостности базы данных и защиты ее от несанкционированного доступа. Поддержание целостности базы данных означает защиту данных от ошибочных действий пользователей или случайных воздействий. В обоих случаях нарушения целостности базы данных обусловлены непреднамеренными факторами.

Целостность базы данных может быть нарушена из-за сбоя оборудования, программных ошибок или неправильных действий пользователей. Обеспечение целостности данных гарантирует качество данных в таблице.

В процессе проектирования таблиц важно определить допустимые значения для каждого столбца и выбрать методы обеспечения целостности данных, включая сущностную, доменную и ссылочную целостность.

Связи между сущностями представлены в таблице 2.2.

Таблица 2.2 – Связи между сущностями

№ связи	Родительская связь		Дочерняя сущность		Тип связи
	Название	Атрибут	Название	Атрибут	
1	Hospital	id	Corpus	IdHospital	Один ко многим (1-∞)
2	Corpus	id	Otdelenie	IdCorpus	Один ко многим (1-∞)
3	Otdelenie	id	Bolezn	IdOtdelenie	Один ко многим (1-∞)
4	Degree	id	Doctor	Degree	Один ко многим (1-∞)
5	Bolezn	id	Karta	IdBolezni	Один ко многим (1-∞)
6	Patcient	id	Karta	IdPatcient	Один ко многим (1-∞)
7	Doctor	id	Karta	IdDoctor	Один ко многим (1-∞)
8	Hospital	id	Policlinic	idHospital	Один ко многим (1-∞)
9	Hospital	idDoctor	DoctorH	idHospital	Один ко многим (1-∞)
10	Policlinic	idHospital	DoctorP	idPoliclinic	Один ко многим (1-∞)

Сщность Hospital связана с Corpus по внешнему ключу IdHospital в Corpus, который ссылается на первичный ключ (id) в таблице Hospital.

Сущность Corpus связана с Otdelenie по внешнему ключу IdCorpus в Otdelenie, который ссылается на первичный ключ (id) в таблице Corpus.

Сущность Otdelenie связана с Bolezn по внешнему ключу IdOtdelenie в Bolezn, который ссылается на первичный ключ (id) в таблице Otdelenie.

Сущность Doctor связана с Degree по внешнему ключу Degree в Doctor, который ссылается на первичный ключ (id) в таблице Degree.

Сущность Karta связана с Bolezn, Patcient и Doctor по внешним ключам IdBolezni, IdPatcient и IdDoctor соответственно.

Сущность Hospital связана с Policlinic по внешнему ключу idHospital в Policlinic, который ссылается на первичный ключ (id) в таблице Hospital.

Сущность Hospital связана с DoctorH по внешнему ключу idHospital в DoctorH, который ссылается на первичный ключ (idDoctor) в таблице DoctorH.

Сущность Policlinic связана с DoctorP по внешнему ключу idPoliclinic в DoctorP, который ссылается на первичный ключ (id) в таблице Policlinic.

2.5 Графическое представление связей между внешними и первичными ключами

После выполнения нормализации, выявления первичных и внешних ключей, определения связей между таблицами, была получена схема реляционной базы данных, представленная в приложении А. На ней изображаются все отношения базы данных, а также связей между внешними и первичными ключами.

3 СОЗДАНИЕ СПРОЕКТИРОВАННОЙ БАЗЫ ДАННЫХ

Для развертывания планируемой информационной системы в области медицинских учреждений был выбран фреймворк для управления данными MS SQL Server 2022. Microsoft SQL Server представляет собой современную систему управления реляционными базами данных, разработанную корпорацией Microsoft. Она оперирует языком запросов Transact-SQL, являющимся процедурным расширением стандарта SQL, разработанным Microsoft и Sybase. Transact-SQL, эффективно применяемый для обработки данных различного объема, поддерживается масштабируемостью от персональных до крупных корпоративных баз данных, конкурируя успешно на рынке с другими системами управления базами данных.

В контексте медицинской информационной системы, подобной системе SQL Server, охватывает различные аспекты хранения и обработки данных о пациентах, врачах и медицинских учреждениях. Медицинская система будет ориентирована на хранение и обработку данных, связанных с больницами, корпусами, отделениями, врачами и пациентами.

Таблица Hospital содержит основные сведения о больнице. Её структура приведена в таблице 3.1.

Таблица 3.1 – Характеристика атрибутов таблицы Hospital

Имя атрибута	Тип	NULL	Описания
id	int	-	Первичный ключ
Name	nvarchar(40)	-	Название больницы
Address	nvarchar(40)	-	Адрес больницы

Таблица Corpus содержит основные сведения о корпусе. Её структура приведена в таблице 3.2.

Таблица 3.2 – Характеристика атрибутов таблицы Corpus

Имя атрибута	Тип	NULL	Описания
id	int	-	Первичный ключ
Name	nvarchar(40)	-	Название корпуса
IdHospital	int	-	Больница в которой корпус находится

Таблица Otdelenie содержит основные сведения об отделении. Её структура приведена в таблице 3.3.

Таблица 3.3 – Характеристика атрибутов таблицы Otdelenie

Имя атрибута	Тип	NULL	Описания
id	int	-	Первичный ключ
Name	nvarchar(40)	-	Название отделения
NumberPalats	int	+	Количество палат
NumberBads	int	+	Количество кроватей
IdCorpus	int	-	Корпус в котором находится отделение

Таблица Bolezn содержит основные сведения о болезни. Её структура приведена в таблице 3.4.

Таблица 3.4 – Характеристика атрибутов таблицы Bolezn

Имя атрибута	Тип	NULL	Описания
id	int	-	Первичный ключ
Name	nvarchar(40)	-	Название болезни
IdOtdelenie	int	-	Отделение которое специализируется на этой болезни

Таблица Degree содержит основные сведения о ученой степени врача. Её структура приведена в таблице 3.5.

Таблица 3.5 – Характеристика атрибутов таблицы Degree

Имя атрибута	Тип	NULL	Описания
id	int	-	Первичный ключ
Name	nvarchar(40)	-	Название ученой степени

Таблица Doctor содержит основные сведения о врача. Её структура приведена в таблице 3.6.

Таблица 3.6 – Характеристика атрибутов таблицы Doctor

Имя атрибута	Тип	NULL	Описания
id	int	-	Первичный ключ
Name	nvarchar(40)	-	ФИО врача
Specialization	nvarchar(40)	-	Специализация врача
Degree	int	-	Его ученая степень

Таблица Patcient содержит основные сведения о пациенте. Её структура приведена в таблице 3.7.

Таблица 3.7 – Характеристика атрибутов таблицы Patcient

Имя атрибута	Тип	NULL	Описания
id	int	-	Первичный ключ
Name	nvarchar(40)	-	ФИО пациента

Таблица Karta содержит основные сведения о болезнях пациента. Её структура приведена в таблице 3.8.

Таблица 3.8 – Характеристика атрибутов таблицы Karta

Имя атрибута	Тип	NULL	Описания
id	int	-	Первичный ключ
Vrema_Postuplenia	date	+	Время поступления
IdBolezni	int	-	Болезнь пациента
IdPatcient	int	-	Пациент
IdDoctor	int	-	Доктор

Таблица Policlinic содержит основные сведения о поликлиники. Её структура приведена в таблице 3.9.

Таблица 3.9 – Характеристика атрибутов таблицы Policlinic

Имя атрибута	Тип	NULL	Описания
id	int	-	Первичный ключ
Name	nvarchar(40)	-	Название поликлиники
idHospital	int	+	Больница, к которой она привязана

Таблица DoctorH содержит основные сведения об отношении доктора к больнице. Её структура приведена в таблице 3.10.

Таблица 3.10 – Характеристика атрибутов таблицы DoctorH

Имя атрибута	Тип	NULL	Описания
idDoctor	int	-	Доктор часть составного ключа
idHospital	int	-	Больница часть составного ключа

Таблица DoctorP содержит основные сведения об отношении доктора к поликлинике. Её структура приведена в таблице 3.11.

Таблица 3.11 – Характеристика атрибутов таблицы DoctorP

Имя атрибута	Тип	NULL	Описания
idDoctor	int	-	Доктор часть составного ключа
idPoliclinic	int	-	Поликлиника часть составного ключа

Таблица Users содержит основные сведения о пользователях. Её структура приведена в таблице 3.12.

Таблица 3.12 – Характеристика атрибутов таблицы Users

Имя атрибута	Тип	NULL	Описания
id	int	-	Первичный ключ
userlogin	nvarchar(40)	-	Логин
password	nvarchar(40)	-	Пароль
AdministratorRights	BIT	-	Права

4 ЗАПИСЬ ВЫРАЖЕНИЙ, УКАЗАННЫХ В ВАРИАНТЕ ЗАДАНИЯ НА ЯЗЫКЕ SQL

Запрос 1: получить перечень и общее число врачей указанного профиля для конкретного медицинского учреждения, больницы, либо поликлиники, либо всех медицинских учреждений города.

Для первой части SQL запрос представлен в листинге 4.1

Листинг 4.1 – Запрос 1

```
SELECT
    Doctor.Name AS Доктор,
    Doctor.Specialization AS Специализация,
    Degree.Name AS Ученая_степень
FROM Doctor
JOIN Degree ON Doctor.Degree = Degree.id
LEFT JOIN DoctorH ON Doctor.id = DoctorH.idDoctor
LEFT JOIN Hospital ON DoctorH.idHospital = Hospital.id
WHERE Hospital.Name = 'Название_больницы';
```

Для второй части запрос представлен в листинге 4.2

Листинг 4.2 – Запрос 1

```
SELECT
    Doctor.Name AS Доктор,
    Doctor.Specialization AS Специализация,
    Degree.Name AS Ученая_степень,
    Hospital.Name AS Больница,
    Policlinic.Name AS Поликлиника
FROM Doctor
JOIN Degree ON Doctor.Degree = Degree.id
LEFT JOIN DoctorH ON Doctor.id = DoctorH.idDoctor
LEFT JOIN Hospital ON DoctorH.idHospital = Hospital.id
LEFT JOIN DoctorP ON Doctor.id = DoctorP.idDoctor
LEFT JOIN Policlinic ON DoctorP.idPoliclinic = Policlinic.id;
```

Запрос 2: получить перечень и общее число врачей указанного профиля со степенью кандидата или доктора медицинских наук для конкретного медицинского учреждения. Запрос представлен в листинге 4.3

Листинг 4.3 – Запрос 2

```
SELECT
     Doctor.Name AS Имя_врача,
     Doctor.Specialization AS Специализация,
     Degree.Name AS Ученая_степень,
     Hospital.Name AS Больница,
     Policlinic.Name AS Поликлиника,
     COUNT(Doctor.id) OVER(PARTITION BY Degree.Name, Hospital.Name)
     AS Количество_врачей
FROM Doctor
JOIN Degree ON Doctor.Degree = Degree.id
```

```
LEFT JOIN DoctorH ON Doctor.id = DoctorH.idDoctor
```

Продолжение листинга 4.3

```
LEFT JOIN Hospital ON DoctorH.idHospital = Hospital.id
LEFT JOIN DoctorP ON Doctor.id = DoctorP.idDoctor
LEFT JOIN Policlinic ON DoctorP.idPoliclinic = Policlinic.id;
```

Запрос 3: получить перечень пациентов, наблюдающихся у врача указанного профиля в конкретной больнице. Запрос предоставлен в листинге 4.4

Листинг 4.4 – Запрос 3

```
SELECT
     Patcient.Name AS Пациент,
     Doctor.Name AS Доктор,
     Doctor.Specialization AS Специализация,
     Hospital.Name AS Больница
FROM Karta
JOIN Patcient ON Karta.IdPatcient = Patcient.id
JOIN Doctor ON Karta.IdDoctor = Doctor.id
JOIN DoctorH ON Doctor.id = DoctorH.idDoctor
JOIN Hospital ON DoctorH.idHospital = Hospital.id;
```

Запрос 4: получить данные о выработке (среднее число принятых пациентов в день) за указанный период для конкретного врача, либо всех врачей поликлиники, либо для всех врачей названного профиля. Запрос предоставлен в листинге 4.5

Листинг 4.5 – Запрос 4

```
SELECT
     Doctor.Name AS Доктор,
     Karta.Vrema_Postuplenia AS Дата,
     SUM(1) AS Пациент
FROM Karta
JOIN Doctor ON Karta.IdDoctor = Doctor.Id
GROUP BY Karta.Vrema_Postuplenia,
Doctor.Name ORDER BY Karta.Vrema_Postuplenia;
```

Запрос 5: получить данные о загрузке (число пациентов, у которых врач в настоящее время является лечащим врачом). Запрос предоставлен в листинге 4.6

Листинг 4.6 – Запрос 5

```
SELECT
     Doctor.Name AS Доктор,
     Doctor.Specialization AS Специализация,
     Degree.Name AS Ученая_Степень,
     (SELECT COUNT(*) FROM Karta Where Doctor.id = Karta.IdDoctor)
     AS Загруженность
FROM Doctor
JOIN Degree ON Doctor.Degree = Degree.id
```

5 РЕАЛИЗАЦИЯ ЗАКОНЧЕННОГО ПРИЛОЖЕНИЯ

5.1 Проектирование интерфейса

Интерфейс взаимодействия пользователя с программой или устройством представляет собой метод, средство, с помощью которого пользователь осуществляет взаимодействие с программой или устройством. Включает в себя все доступные пользователю элементы, такие как кнопки, поля для ввода данных, меню и прочее. Главная задача данного интерфейса заключается в обеспечении наиболее простого и интуитивно понятного взаимодействия пользователя с программой.

Интерфейс пользователя приложения, которое оперирует базой данных, должен включать функциональность для просмотра, изменения, добавления и удаления данных из таблиц базы данных. Эти функции напрямую воздействуют на базу данных, выполняя соответствующие операции. Также предусмотрена возможность фильтрации данных в таблицах по значениям их столбцов. Переход между различными таблицами базы данных осуществляется с использованием специальных кнопок. Кроме того, пользовательский интерфейс предоставляет возможность выполнения запросов, указанных в задании.

5.2 Создание приложения, работающего с базой данных

Приложение, разработанное с использованием технологии Windows Presentation Foundation (WPF) и языка программирования C#, интегрируется с базой данных SQL Server с использованием библиотеки SQLClient.

WPF представляет собой фреймворк для создания приложений с графическим интерфейсом пользователя в .NET Framework. В основе приложения лежит модель окон, где каждое окно представляет визуальное представление данных для пользователя. Элементы интерфейса могут включать в себя текст, кнопки, выпадающие меню, таблицы данных и другие компоненты.

Язык программирования C# выбран для реализации приложения. C# предоставляет современный объектно-ориентированный и безопасный подход к программированию.

Для подключения к базе данных SQL Server используется библиотека SQLClient. Она обеспечивает возможность взаимодействия с сервером баз данных, выполняя операции чтения, записи и редактирования данных.

Вход программу начинается с авторизации. Появляется окно, предоставленное на рисунке 5.1.

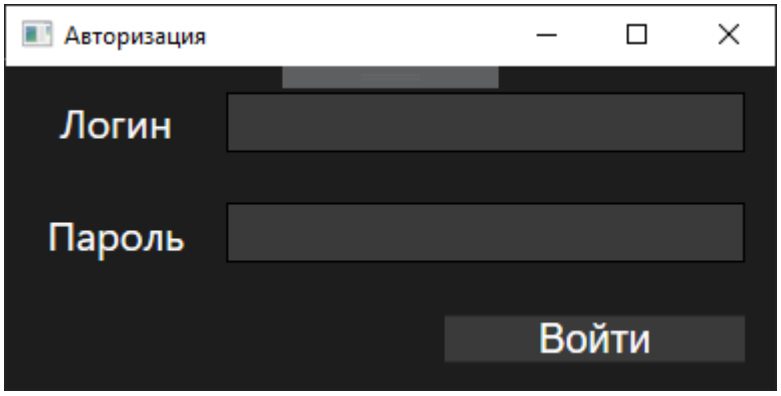

Рисунок 5.1 – Окно авторизации

Интерфейс главного меню приложения представлен на рисунке 5.2.

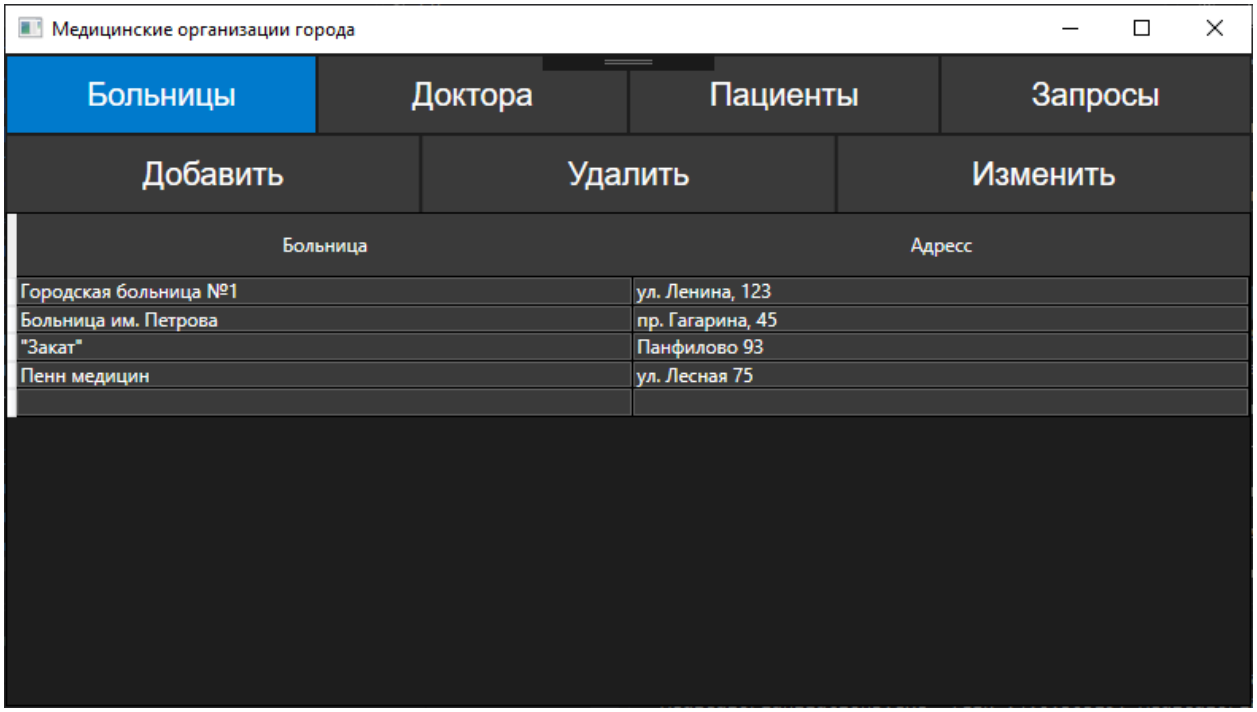

Рисунок 5.2 – Главное меню приложения

Данный интерфейс предоставляет ассортимент закладок, в которых будет осуществляться работа с базой данных. И три кнопки добавления, удаления и изменения, которые доступны только администратору. Обычному пользователю эти кнопки не будут показаны так как ему не разрешено изменять содержимое базы данных, а загромождать интерфейс недоступными кнопками не имеет смысла.

Для загрузки таблицы из базы данных используется функция Load() представленая в листинге 5.1.

Листинг 5.1 – Функция Load

```
private void LoadData()
{
     try
     {
          DoctorDataGrid.ItemsSource = dbManager.Query("SELECT
Doctor.id, Doctor.Name, Doctor.Specialization, Degree.Name AS
Degree FROM Doctor JOIN Degree ON Doctor.Degree =
Degree.id").DefaultView;
     }
     catch (Exception ex)
     {
          Console.WriteLine(ex.ToString());
          MessageBox.Show(ex.Message);
     }
}
```

Метод Query используемый объектом dbManager описан в листинге 5.2

Листинг 5.2 – метод Query

```
public DataTable Query(string Query)
{
     using (SqlConnection conn = new SqlConnection(connStr))
     {
          try
          {
               conn.Open();
               string selectSql = Query;
               SqlCommand  selectCmd  =  new  SqlCommand(selectSql,
conn);
               SqlDataAdapter da = new SqlDataAdapter(selectCmd);

               DataTable dt = new DataTable();
               da.Fill(dt);
               return dt;
          }



          catch (Exception ex)
          {
               Console.WriteLine(ex.ToString());
               throw new Exception($"Ошибка при выполнении запроса
{Query}: {ex.Message}");
          }
     }
}
```

6 ТЕСТИРОВАНИЕ

Тестирование программы начинается с отлова ошибок. Начнем с авторизации если указать неверный логин или пароль, то программа не только не пустит пользователя, но и выведет оповещение, предоставленное на рисунке 6.1.

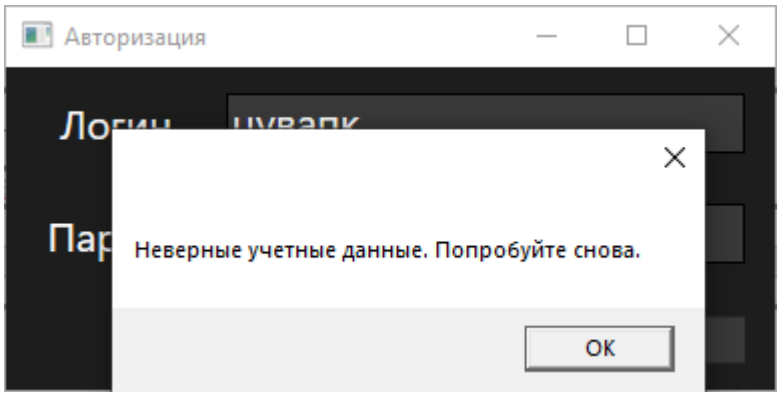

Рисунок 6.1 – Оповещение о неверной попытке входа.

При нажатии на кнопку добавить появляется окно добавления новой больницы. В этом окне вводятся значения новой больницы. Данное окно предоставлено на рисунке 6.2

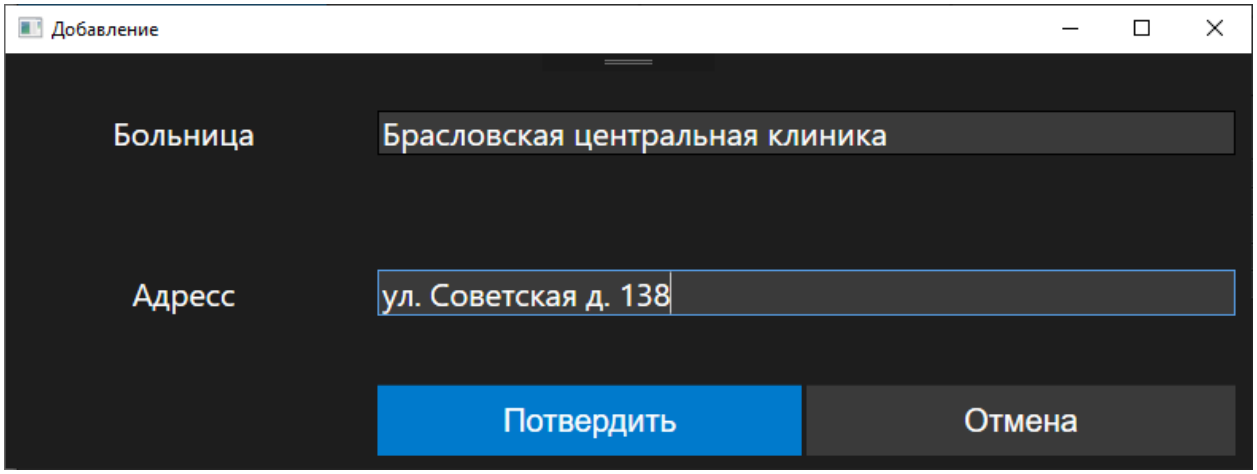

Рисунок 6.2 – Окно добавления больницы.

После чего данная больница добавляется в таблицу и базу данных это отлично видно на рисунке 6.3

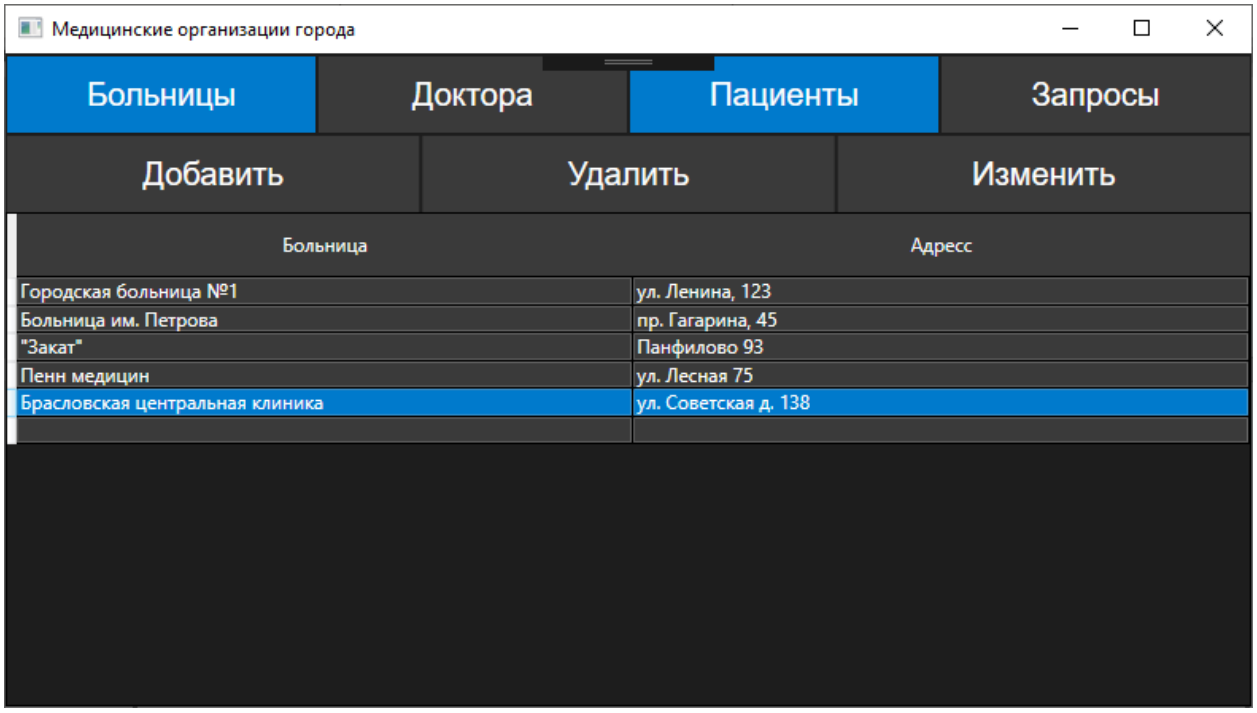

Рисунок 6.3 – Успешное добавление больницы

Если нажать на кнопку изменения появляется все тоже окно только с уже заполненными данными. Окно предоставлено на рисунке 6.4

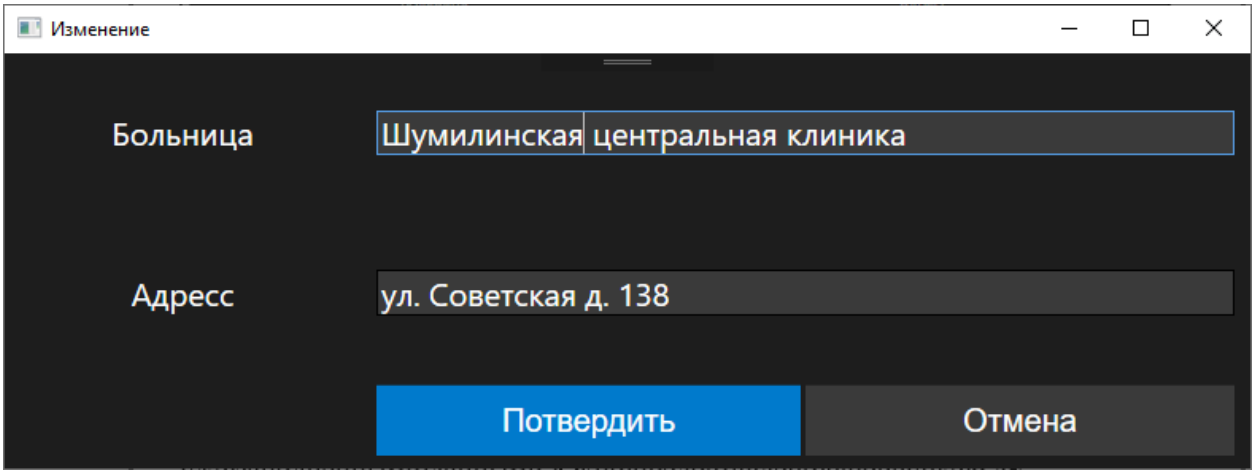

Рисунок 6.4 – Изменение данных о больнице

Результат изменения предоставлен на рисунке 6.5

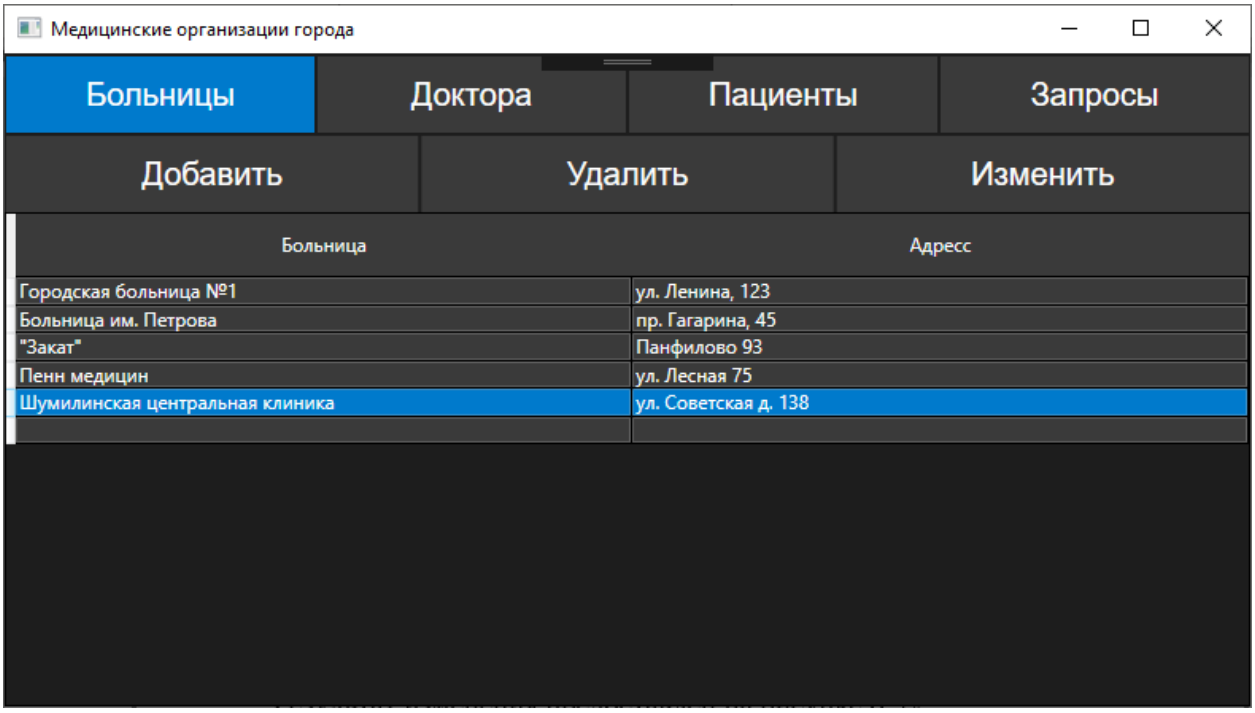

Рисунок 6.5 – Успешное изменение

При нажатии на кнопку удаления появляется окно запрашивающее разрешение на удаление записи/записей. Окно предоставлено на рисунке 6.6.

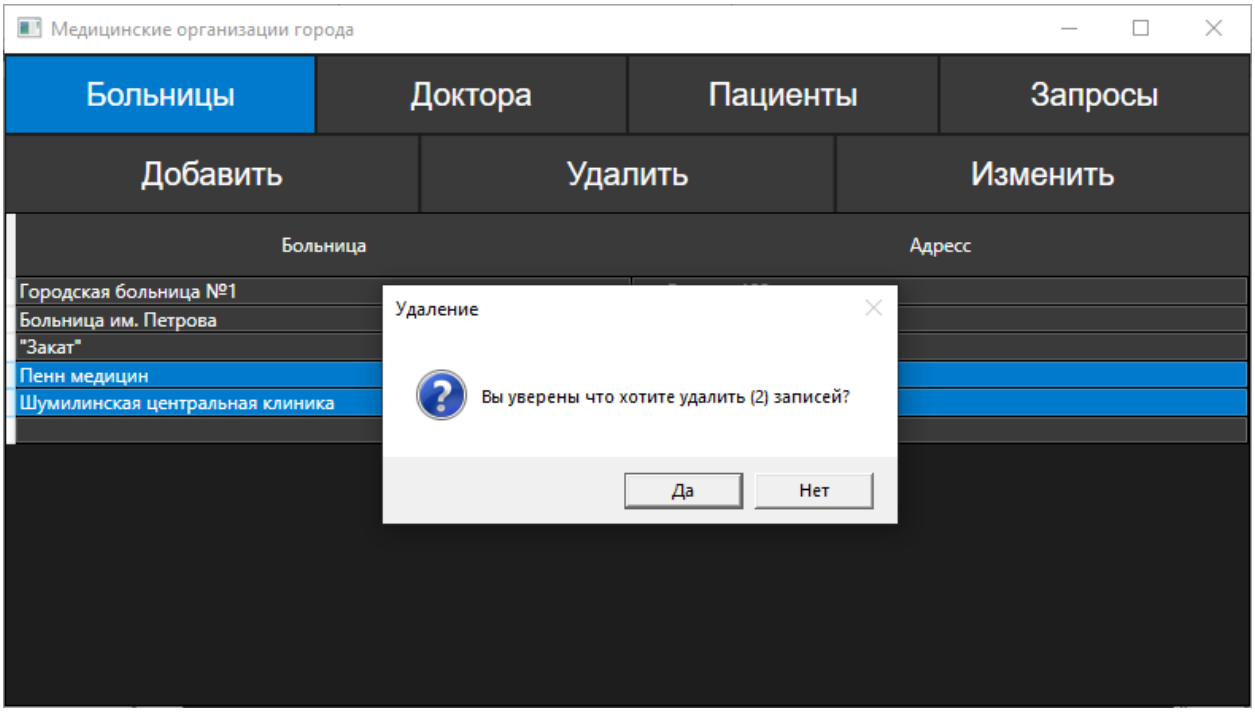

Рисунок 6.6 – Удаление

Результат удаления предоставлен на рисунке 6.7.

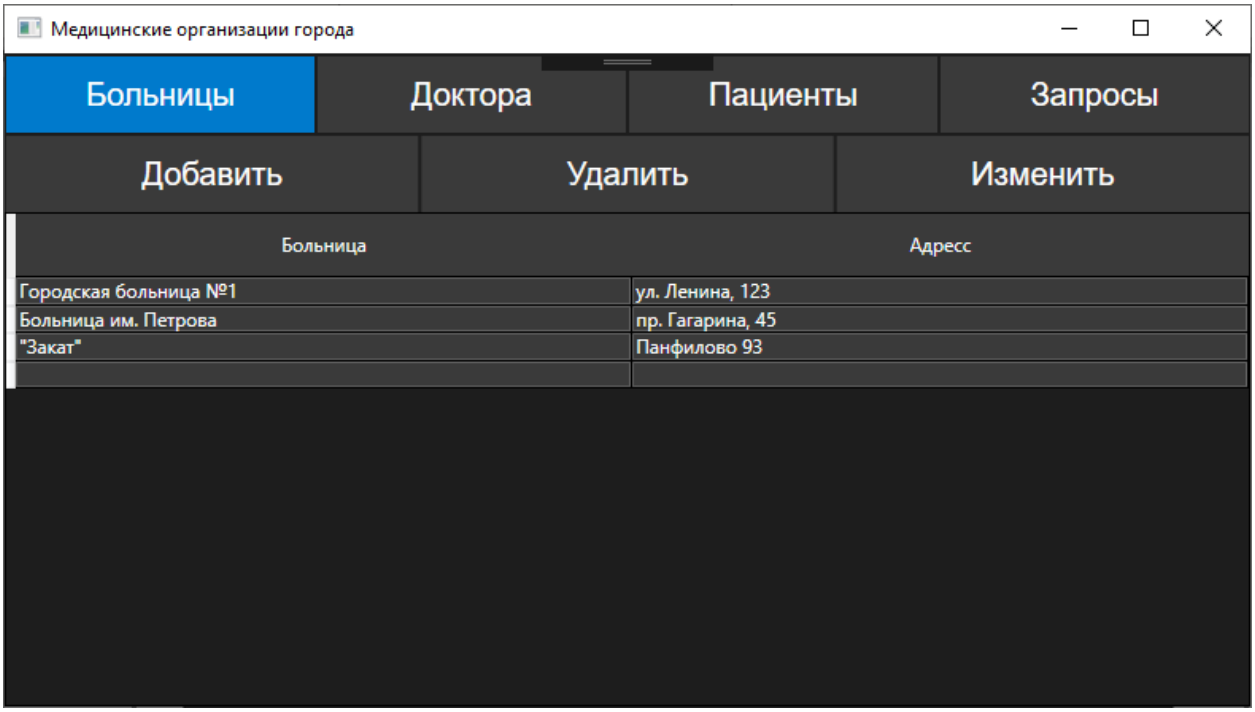

Рисунок 6.7 – Успешное удаление

После татуировки программы можно подвести итоги. Программа работает с базой данных и способна самостоятельно отлавливать ошибки пользователя и оповещать его об этом.

ЗАКЛЮЧЕНИЕ

В ходе этой курсовой работы освоены умения и навыки, необходимые для разработки программного продукта, взаимодействующего с базой данных. В частности, был осуществлен тщательный процесс создания программы с применением технологии Windows Presentation Foundation (WPF) и языка программирования C#.

Экспериментирование с WPF позволило овладеть техникой построения графического интерфейса пользователя, что важно для обеспечения удобства взаимодействия конечного пользователя с программой. Также были приобретены навыки работы с элементами интерфейса, такими как текстовые поля, кнопки, выпадающие меню и таблицы данных, что способствует созданию более интуитивного и привлекательного пользовательского опыта.

Более того, в рамках выполнения проекта были освоены технологии работы с базой данных, а именно язык SQL. Этот опыт не только расширил понимание организации данных, но и позволил разработать собственную базу данных, что является значимым шагом в понимании внутренних механизмов хранения и обработки информации.

Таким образом, результатом данной курсовой работы стало усвоение ключевых аспектов разработки программ, работающих с базой данных, включая использование современных технологий для создания интуитивно понятного пользовательского интерфейса и освоение языка SQL для эффективного управления данными.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Википедия [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org. Дата обращения: 10.12.2023

2. Skillbox [Электронный ресурс] – Режим доступа: https://skillbox.ru/media/code/sistema-upravleniya-bazami-dannykh-chto-eto-takoe-i-zachem-ona-nuzhna/. Дата обращения: 10.12.2023

3. Microsoft Azure [Электронный ресурс] – Режим доступа: https://azure.microsoft.com/ru-ru/resources/cloud-computing-dictionary/what-is-a-relational-database. Дата обращения: 10.12.2023

4. Microsoft Support [Электронный ресурс] – Режим доступа: https://support.microsoft.com/ru-ru/topic/основные-сведения-о-создании-баз-данных-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5#bmterms. Дата обращения: 11.12.2023

5. 42buketa.ru [Электронный ресурс] – Режим доступа: https://42buketa.ru/faq/analiz-predmetnoi-oblasti-sut-i-znacenie. Дата обращения: 11.12.2023

ПРИЛОЖЕНИЕ А
(обязательное)
Схема реляционной базы данных