Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

СОЗДАНИЕ ПРОСТОГО ПРИЛОЖЕНИЯ

ДЛЯ РАБОТЫ С БАЗАМИ ДАННЫХ В MICROSOFT VISUAL STUDIO

Отчет по лабораторной работе по дисциплине «Основы разработки баз данных»

Студент гр. 573-3

\_\_\_\_\_\_\_\_\_\_\_\_ Е. А. Макаров

дата

\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Руководитель:

Преподаватель:

\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ Р. О. Остапенко

подпись

оценка

\_\_\_\_\_\_\_\_\_\_\_\_\_

дата

Томск, 2025

**Введение**

В современном мире базы данных играют ключевую роль в системах хранения и обработки информации. Они используются в самых разных областях — от небольших веб-приложений до крупных корпоративных систем. Эффективная организация данных позволяет оптимизировать работу программных продуктов, обеспечить целостность и безопасность информации, а также упростить её поиск и анализ.

Цель данной роботы – познакомиться с основными инструментами создания баз данных в среде Microsoft Visual studio, элементами библиотеки ADO.NET и научиться создавать приложения для работы с базой данных. При работе потребуется умение писать SQL запросы. В процессе выполнения будет создано приложение, которое позволит добавлять, редактировать, получать и удалять данные из базы данных с помощью пользовательского интерфейса.

**Задачи:**

Для своей БД, созданной в лабораторной работе №1, необходимо выполнить:

1. создать главную форму проекта с наличием главного меню, контекстного меню, панели инструментов;
2. настроить, чтобы главная форма сохраняла свое положение при повторном запуске программы;
3. провести подключение к БД в приложении;
4. сформировать конфигурационный файл для изменения пути к БД в приложении;
5. провести визуальное оформление данных БД в приложении.
   1. Обязательно:
   2. - наличие нескольких форм в проекте;
   3. - использование различных элементов управления Label, TextBox, ComboBox (и/или ListBox), CheckBox, PictureBox, DataGridView, BindingNavigator и др.
   4. Если в БД нет данных для таких объектов, то изменить структуру БД.
6. провести модификацию настроек DataGridView (типы колонок обязательно);
7. предоставить работу с графическим полем (в том числе загрузку фотографии из файла);
8. реализация шаблона проектирования «Одиночка» (англ. Singleton);
9. протестируйте работу полученного приложения
10. **ОСНОВНАЯ ЧАСТЬ**
    1. **Описание данных для реализации БД**

В ходе первой лабораторной работы мы создали свою базу данных на языке SQL, сейчас же нам предстоит реализовать доступ к ней через пользовательский интерфейс.

Для удобства дальнейшей работы, внесём несколько изменений в нашу базу данных:

* Уберём связь М:М между преподавателем и классом за её ненадобностью
* Изменим именование таблицы Четвертей

По итогам наших изменений, база данных будет иметь более подходящую форму для дальнейшей реализации (Рисунок 1.1).

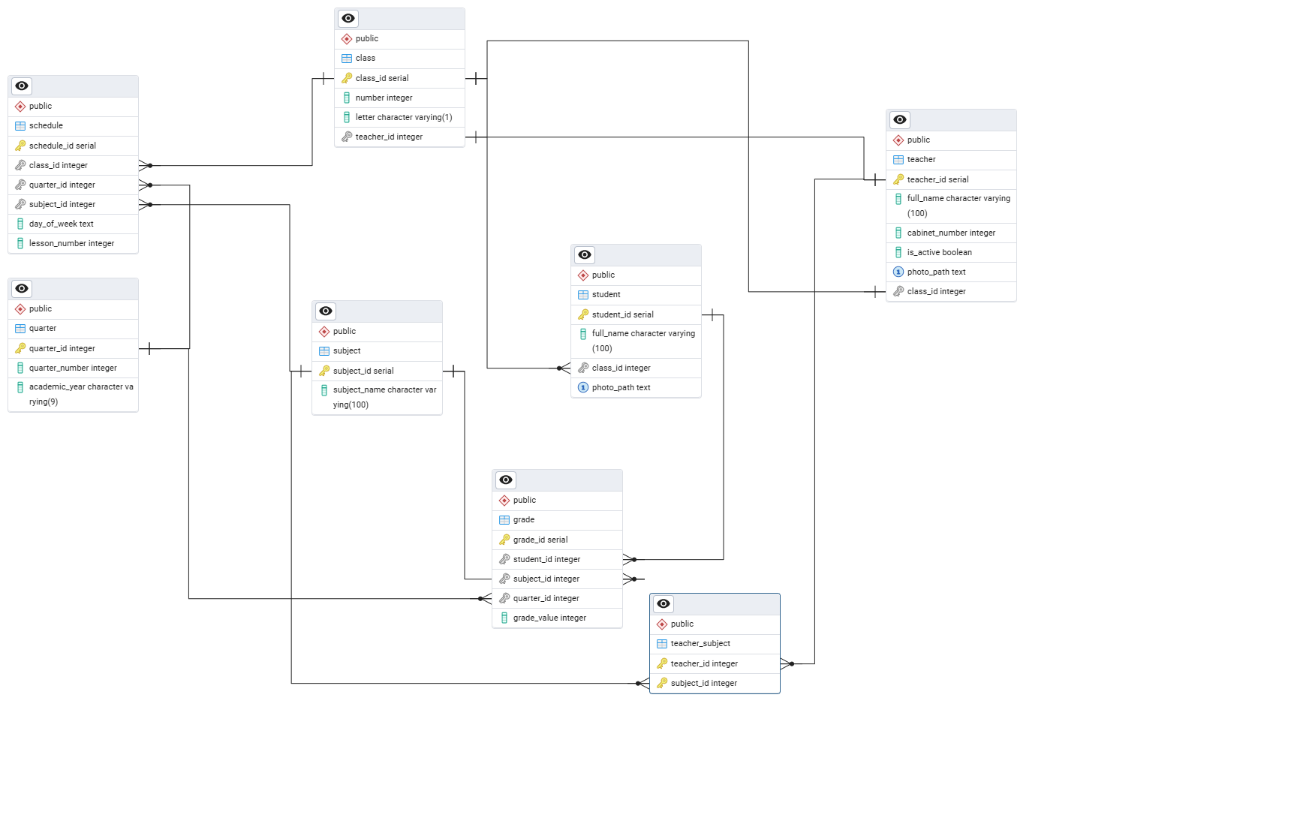


Рисунок 1.1 – Структура БД

* 1. **Выбор инстументов для работы**

Для создания приложение выбор упал на фреймворк WPF, так как он позволят создавать высокоэффективные приложения с пользовательским интерфейсом за короткий срок.

Для реализации доступа к БД под управлением СУБД PostgresSQL будет использоваться библиотека npgsql, и применён паттерн Репозиторий для доступа к каждой сущности без прямого написания SQL запросов напрямую в коде. Так же нам потребуются библиотеки MVVM Toolkit для связки интерфейса и его логики, и Dependency injection для удобного внедрения зависимостей в наши окна.

Структура решения будет разделена на 4 проекта:

* Domain – хранит основные сущности
* Application – хранит интерфейсы для реализации репозиториев
* Infrastructure – хранит репозитории для связи с БД
* Presentation – хранит пользовательский интерфейс и логику работы БД
  1. **Создание основ для работы с БД**

В первую очередь мы создадим основные сущности для работы с базой данных. На уровне Domen создадим классы (Рисунок 1.2), советующие сущностям из базы данных.

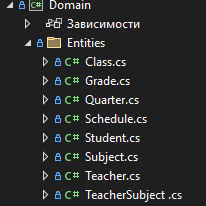


Рисунок 1.2 – Классы для работы с БД

На уровне Application создадим интерфейсы для репозиториев (Рисунок 1.3), которые будут нами использоваться в работе. Каждый интерфейс содержит базовые CRUD операции и операции специфичные для определённой сущности (Листинг 1.1).

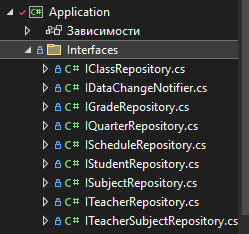


Рисунок 1.3 – Интерфейсы для репозиториев

Листинг 1.1 – Пример интерфейса со специфичной реализацией

using SchoolDB.Domain.Entities;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SchoolDB.Application.Interfaces

{

public interface IStudentRepository

{

void Create(Student entity);

Student GetById(int id);

IEnumerable<Student> GetAll();

void Update(Student entity);

void Delete(int id);

}

}

После чего создадим реализацию для наших интерфейсов (Рисунок 1.4). Каждый репозиторий будет содержать SQL запросы, которые будут обёрнуты в методы для более удобного использования в коде.

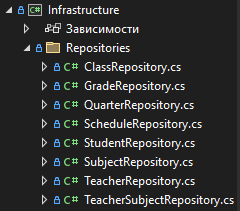


Рисунок 1.4 – Созданные репозитории

Для доступа базе данных вынесем строку подключения в отдельный JSON файл, который будет читать программа при запуске. Создадим файл для дальнейшей инъекции зависимостей и сервис для уведомления об изменениях в БД (Листинг 1.2).

Листинг 1.2 – Сервис для уведомления об изменениях

using Application.Interfaces;

using SchoolDB.Application.Interfaces;

using System;

namespace Infrastructure.Services

{

public class DataChangeNotifier : IDataChangeNotifier

{

public event Action ClassChanged;

public event Action StudentChanged;

public event Action TeacherChanged;

public event Action SubjectChanged;

public event Action GradeChanged;

public event Action QuarterChanged;

public event Action ScheduleChanged;

public event Action TeacherSubjectChanged;

public void NotifyClassChanged() => ClassChanged?.Invoke();

public void NotifyStudentChanged() => StudentChanged?.Invoke();

public void NotifyTeacherChanged() => TeacherChanged?.Invoke();

public void NotifySubjectChanged() => SubjectChanged?.Invoke();

public void NotifyGradeChanged() => GradeChanged?.Invoke();

public void NotifyQuarterChanged() => QuarterChanged?.Invoke();

public void NotifyScheduleChanged() => ScheduleChanged?.Invoke();

public void NotifyTeacherSubjectChanged() => TeacherSubjectChanged?.Invoke();

}

}

* 1. **Создание пользовательского интерфейса**

Для удобной навигации по нашему приложению мы создадим главное окно (Рисунок 1.5), которое будет давать доступ к остальным. Все окна будут реализовывать паттерн Одиночка, чтобы мы могли открывать сразу несколько окон, но каждое будет уникальным.

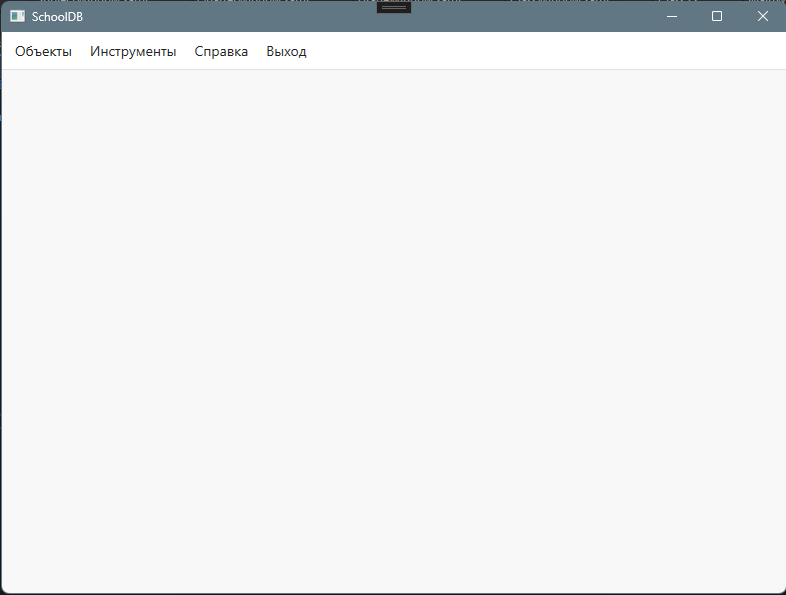


Рисунок 1.5 – Вид главного окна

Для остальных сущностей мы создадим по отдельной странице, чтобы можно было работать сразу с несколькими одновременно.

Начнём с окна для класса (Рисунок 1.6). В него мы добавим возможность создавать классы и назначать в них школьников.

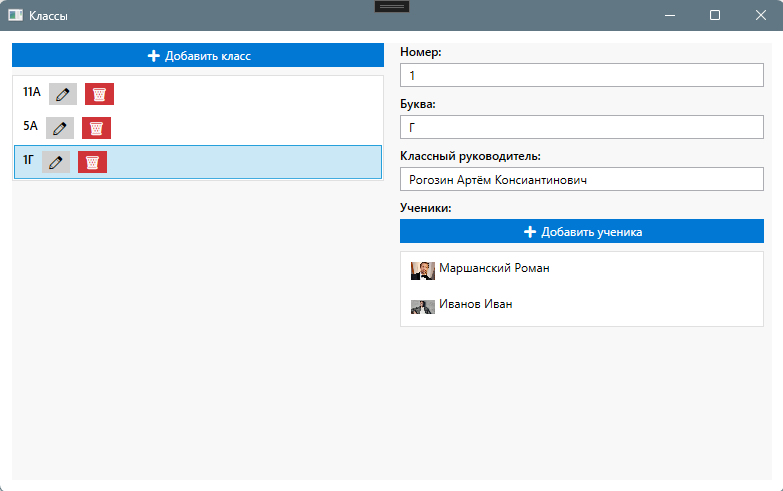


Рисунок 1.6 – Окно для классов

После мы создадим окно для учеников (Рисунок 1.7). В нём мы сможем регистрировать новых учеников, просматривать их оценки, добавлять им фотографии и переводить в классы.

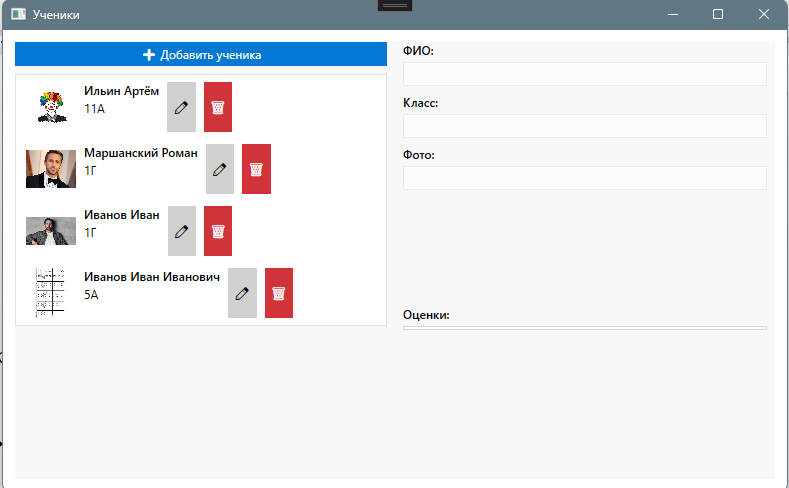


Рисунок 1.7 – Окно для учеников

После мы сделаем окно для преподавателей (Рисунок 1.8), в котором сможем регистрировать новых преподавателей, добавлять им классы и кабинеты, просматривать данные. Также, если преподаватель сейчас не активен, то его имя будет автоматически краситься в красный.

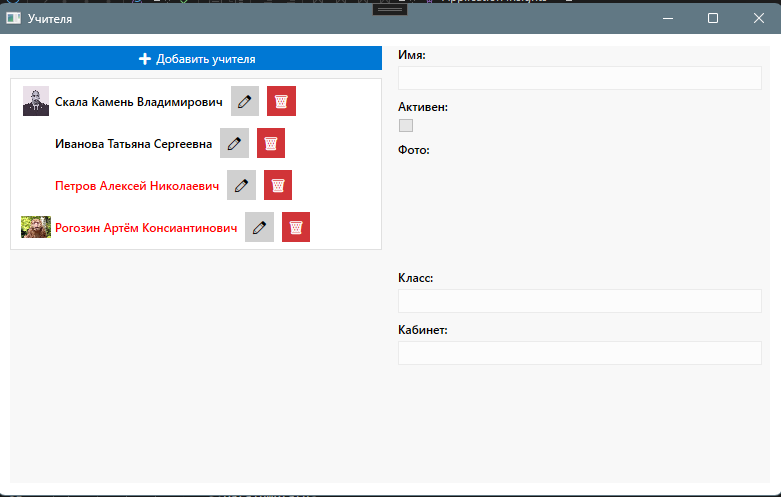


Рисунок 1.8 – Окно для преподавателей

Следующим окном будет окно для предметов (Рисунок 1.9). В нём можно будет добавлять новые предметы и назначать преподавателей.

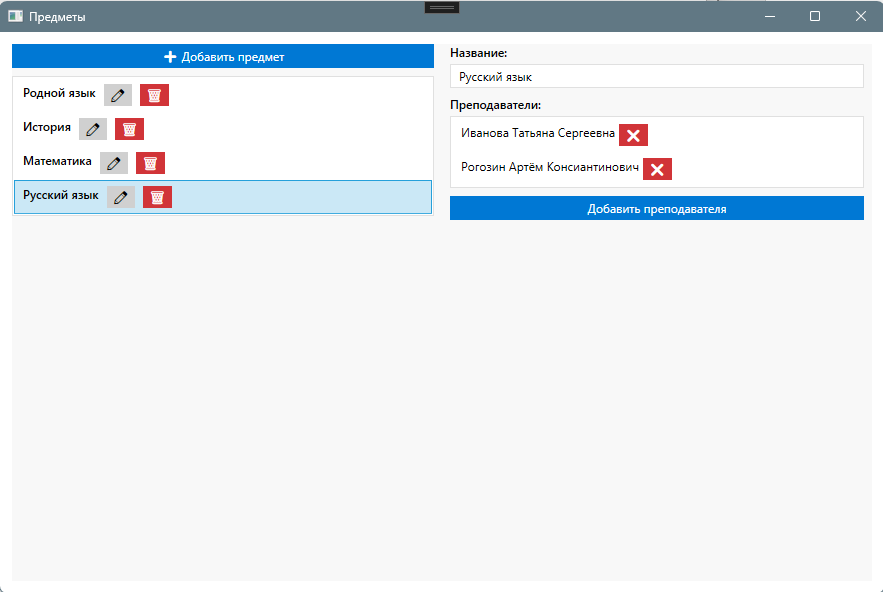


Рисунок 1.9 – Окно для предметов

И последним окном из сущностей будет окно четвертей (Рисунок 1.10). Оно будет позволять добавлять и редактировать новые четверти.

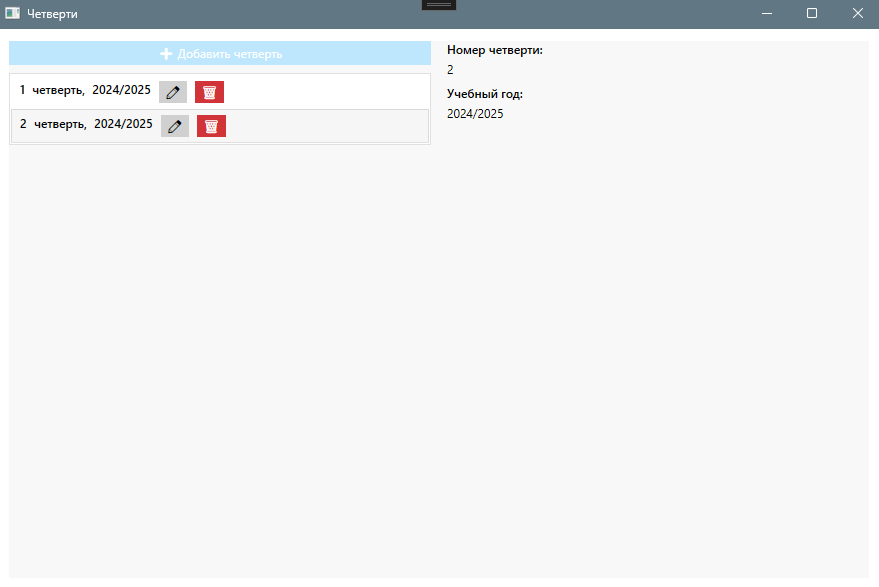


Рисунок 1.10 – Окно для четвертей

Теперь перейдём к окнам инструментов для выставления оценок (Рисунок 1.11) и редактирования расписания (Рисунок 1.12). Они нужны для выставления отметок и редактирования расписания для учеников.

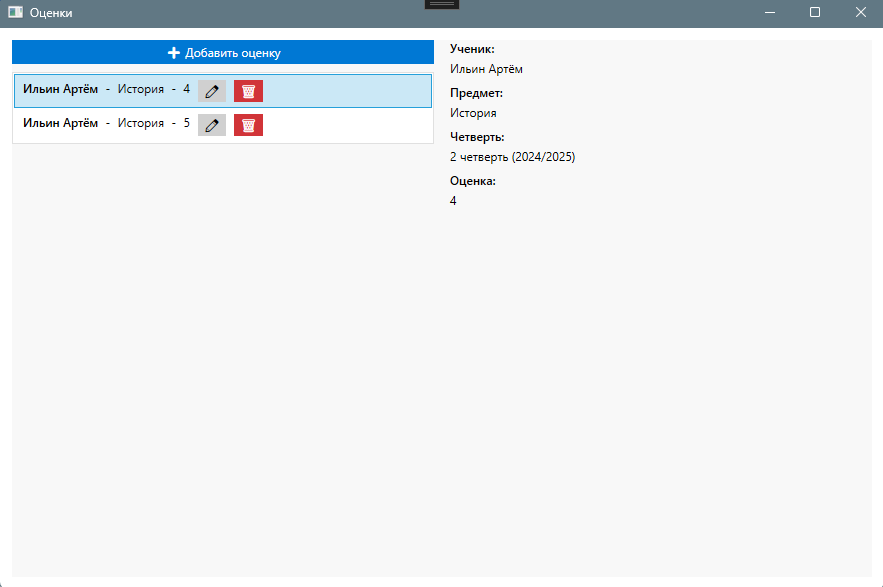


Рисунок 1.11 – Окно для выставления оценок

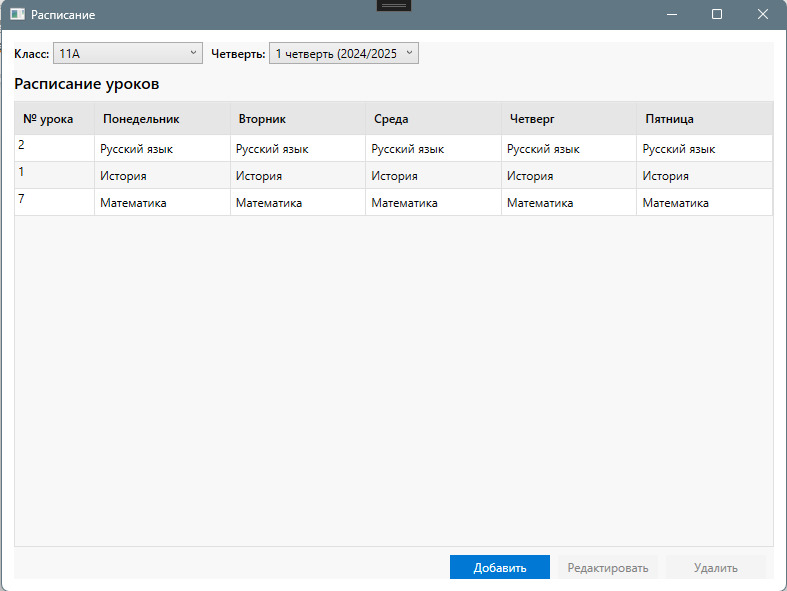


Рисунок 1.12 – Окно для редактирования расписания

Для каждого окна была написана ModelView часть (Рисунок 1.13), которая содержит логику работы окна.

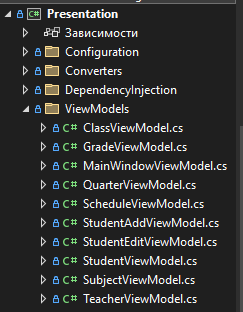


Рисунок 1.13 – Вид ModelView части

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения лабораторной работы было спроектировано и реализовано приложение для работы с базой данных. Были созданы окна для работы со сущностями, а также реализована логика добавления, редактирования, получения и удаления из базы данных.

Были изучены инструменты библиотеки ADO.NET для работы с базами данных и элементы управления в WPF, для работы с запросами и отображения данных.

Таким образом, цель лабораторной работы была достигнута: приобретены теоретические знания и практические умения по проектированию, созданию и тестированию реляционных баз данных.