Министерство образования и науки Российской Федерации

Муромский институт (филиал)

федерального государственного бюджетного образовательного учреждения высшего образования

«Владимирский государственный университет

Имени Александра Григорьевича и Николая Григорьевича Столетовых»

(МИ ВлГУ)

Факультет ИТР

Кафедра: ПИн

Курсовая работа

По: *Системы управления базами данных*

Тема: АИС система музыкального магазина

Руководитель

к. т. н., доц. каф. ПИН
(уч. степень, звание)

Колпаков А.А.
(фамилия, инициалы)

(Оценка)

(подпись) (дата)

Члены комиссии

Студент ПИН-122
(группа)

Хафизова М.М.
(фамилия, инициалы)

(Подпись Ф.И.О.)

(подпись) (дата)

(Подпись Ф.И.О.)

Муром 2024

Содержание

Введение 4
1. Анализ технического задания 4
2. Разработка моделей данных 7
3. Разработка и реализация АИС 9
3.1 Структура базы данных 10
3.2 Взаимодействие с базой данных 11
3.3 Структура программы 11
3.4 Инструкция пользователя 17
4. Тестирование АИС 19
4.1 Тестирование базы данных 19
4.2 Тестирование программы 26
Заключение 28
Список литературы 29
Приложения 30
Приложение А 30
Приложение Б 31
Приложение В 32

Введение

Современные информационные технологии стремительно проникают во все сферы нашей жизни, в том числе и в бизнес. Музыкальная индустрия не является исключением, и автоматизация процессов управления становится необходимым условием для успешного функционирования музыкальных магазинов. Автоматизированные информационные системы (АИС) играют ключевую роль в оптимизации бизнес-процессов, повышении эффективности работы и улучшении качества обслуживания клиентов.

Данная курсовая работа посвящена разработке АИС для музыкального магазина, которая позволит интегрировать различные аспекты его деятельности: управление запасами товаров, обработку заказов, учет клиентов, а также анализ продаж. В условиях растущей конкуренции на рынке музыкальных инструментов и аксессуаров, эффективная система управления может стать значительным конкурентным преимуществом.

Целью данной работы является создание концепции АИС, которая будет учитывать специфические потребности музыкального магазина и обеспечивать его устойчивое развитие. В ходе исследования будут рассмотрены ключевые функции системы, ее архитектура и технологии реализации, а также потенциальные выгоды от внедрения АИС в деятельность магазина.

Задачи:

Проанализировать существующие АИС в сфере розничной торговли музыкальными инструментами и выявить их преимущества и недостатки;

- Определить функциональные требования к разрабатываемой системе;
- Разработать архитектуру автоматизированной информационной системы для музыкального магазина;
- Описать основные алгоритмы работы системы.

Объект исследования: АИС музыкального магазина.

Предмет исследования: Проектирование АИС музыкального магазина.

Методы, используемые в работе:

- Анализ литературы по теме автоматизации розничной торговли;
- Сравнительный анализ существующих АИС;
- Метод проектирования для разработки архитектуры системы;

1. Анализ технического задания

Разработка автоматизированной информационной системы (АИС) для музыкального магазина направлена на оптимизацию процессов управления товарами, клиентами и продажами. В качестве платформы для реализации проекта выбраны WinForms для создания пользовательского интерфейса и PostgreSQL (через pgAdmin 4) для управления базой данных.

Выбор средств разработки

1. WinForms

Обоснование выбора:

- Простота использования: WinForms предоставляет удобные инструменты для быстрого создания графического интерфейса пользователя (GUI), что позволяет сосредоточиться на функциональности приложения.
- Поддержка C#: Язык C# является мощным и распространённым, что упрощает разработку и отладку.
- Быстрая разработка: WinForms позволяет быстро разрабатывать приложения с минимальными затратами времени на проектирование интерфейса.

Сравнение с альтернативами:

- WPF (Windows Presentation Foundation): Хотя WPF предоставляет более современные возможности для создания интерфейсов, его изучение требует

больше времени, а также может быть избыточным для простых приложений.

- UWP (Universal Windows Platform): UWP подходит для создания приложений, работающих на различных устройствах Windows, но имеет более сложные требования к развертыванию и совместимости.

2. PostgreSQL (pgAdmin 4)

Обоснование выбора:

- Надежность и производительность: PostgreSQL — это мощная реляционная

СУБД, известная своей надежностью и высокой производительностью при работе с большими объемами данных.

- Расширяемость: PostgreSQL поддерживает расширения, что позволяет адаптировать базу данных под конкретные нужды проекта.
- Поддержка сложных запросов: Возможности SQL в PostgreSQL позволяют эффективно обрабатывать сложные запросы и выполнять аналитические операции.

Сравнение с альтернативами:

− MySQL: Хотя MySQL также является популярной СУБД, PostgreSQL предлагает более широкие возможности для работы с данными и поддерживает больше типов данных.

− SQLite: SQLite подходит для небольших приложений, но не обеспечивает такой же уровень масштабируемости и многопользовательской поддержки, как PostgreSQL.

Выбор WinForms и PostgreSQL для разработки АИС музыкального магазина обусловлен необходимостью создания простого, но функционального интерфейса и надежной базы данных. Эти инструменты обеспечивают быструю разработку и высокую производительность, что делает их идеальными для реализации поставленных задач в рамках курсовой работы.

2. Разработка моделей данных

Разработка моделей данных является ключевым этапом в проектировании автоматизированной информационной системы (АИС) музыкального магазина. Модели данных определяют структуру и взаимосвязи между различными сущностями, которые будут использоваться в системе. В данном разделе будут рассмотрены основные сущности, их атрибуты и связи между ними.

Функциональные требований к системе описаны в формате UseCase (рисунок 1).

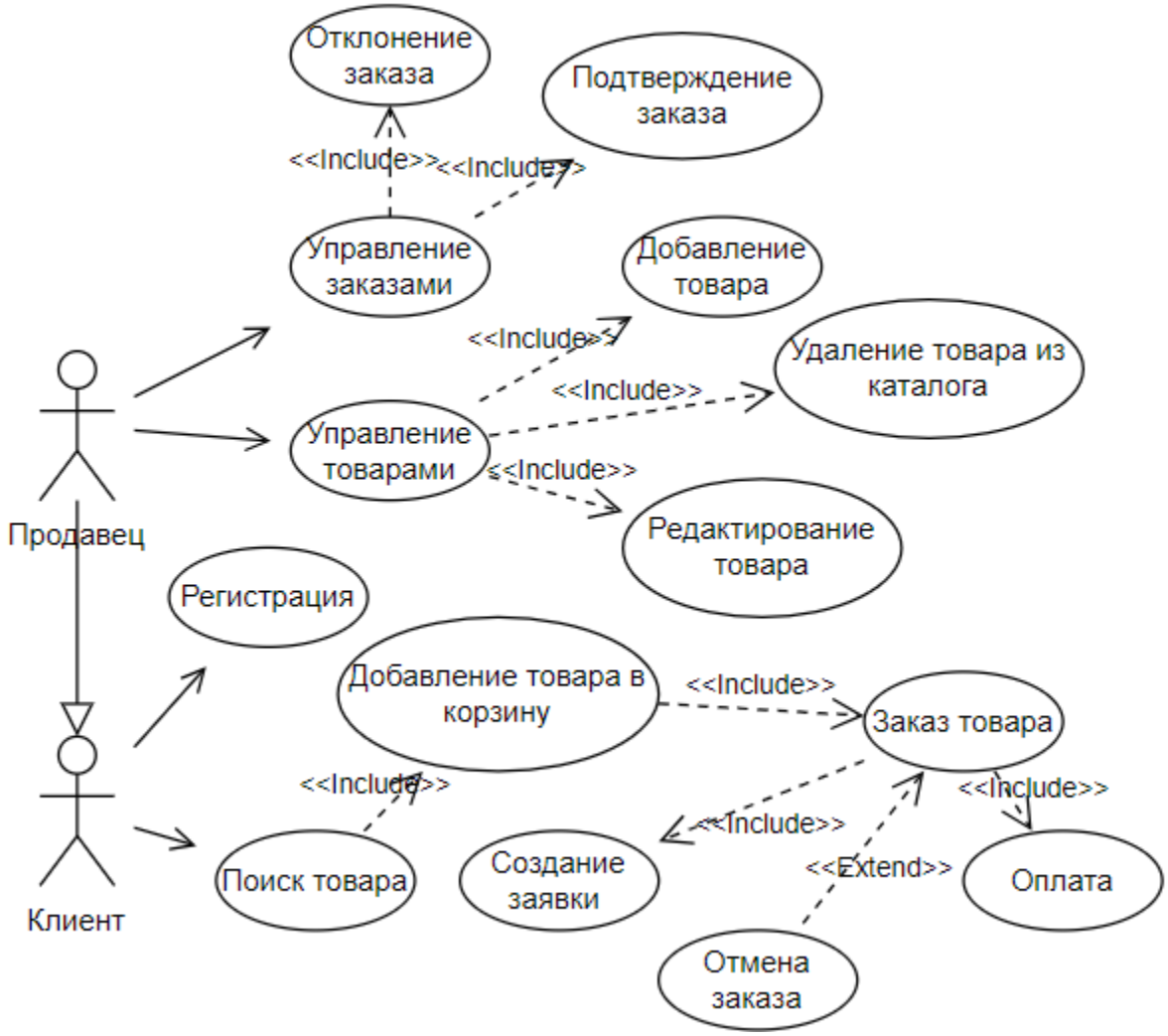

Рисунок 1 – UseCase диаграмма для АИС музыкального магазина.

В ходе проектирования логической структуры базы данных была создана ее логическая модель, которая представлена на рисунке А.1. На основе этой модели была разработана физическая реализация, изображенная на рисунке Б.1.

Доступ к базе данных имеет только продавец путём взаимодействия через приложение.

Вот несколько таблиц, которые демонстрируют структуру базы данных.

Таблица 1. Пояснение к таблице clients_catalog – каталог клиентов

Наименование столбца	Пояснение
id	Идентификатор записи. Первичный ключ
surname	Фамилия клиента
name	Имя клиента
birthday	Дата рождения клиента
phone_number	Телефонный номер клиента
email	Почта клиента

Таблица 2. Пояснение к таблице compositions_catalog – каталог композиций

Наименование столбца	Пояснение
id	Идентификатор композиции. Первичный ключ
name	Название композиции
musician_id	Идентификатор музыканта. Внешний ключ
ensemble_id	Идентификатор ансамбля. Внешний ключ
release_year	Год выпуска

Таблица 3. Пояснение к таблице vinyl_records – каталог виниловых пластинок

Наименование столбца	Пояснение
1	2
id	Идентификатор пластинки. Первичный ключ
label_number	Номер наклейки на пластинке
title	Название пластинки
release_year	Год выпуска пластинки
musician_id	Идентификатор музыканта. Внешний ключ
ensemble_id	Идентификатор ансамбля. Внешний ключ
genre	Жанр
wholesale_price	Оптовая цена
retail_price	Розничная цена
sold_last_year	Кол-во проданных экземпляров в прошлом году

1	2
sold_this_year	Кол-во проданных экземпляров в текущем году
stock	Остаточное кол-во экземпляров
image	Изображение пластинки

Таблица 4. Пояснение к таблице ensembles_catalog – каталог ансамблей

Наименование столбца	Пояснение
id	Идентификатор ансамбля. Первичный ключ
name	Название ансамбля
date_founded	Дата основания

Таблица 5. Пояснение к таблице ensemble_musicians

Наименование столбца	Пояснение
id	Идентификатор записи. Первичный ключ
ensemble_id	Идентификатор ансамбля. Внешний ключ
musician_id	Идентификатор музыканта. Внешний ключ

Таблица 6. Пояснение к таблице musician_composition

Наименование столбца	Пояснение
id	Идентификатор записи. Первичный ключ
composition_id	Идентификатор композиции. Внешний ключ
musician_id	Идентификатор музыканта. Внешний ключ

3. Разработка и реализация АИС

3.1 Структура базы данных

Структура базы данных для автоматизированной информационной системы музыкального магазина разработана на основе требований задания. В базе данных представлены, описанные в пункте «Разработка моделей данных».

Связи между таблицами

– Таблица VinylRecords связана с:

Musicians по полю MusicianID.

Ensembles по полю EnsembleID.

– Таблица EnsembleMusicians используется для связи многих музыкантов с одним ансамблем и наоборот.

Для обеспечения быстрого доступа к данным в базе реализована индексация следующих ключей:

– Первичные ключи всех таблиц.

– Внешние ключи для обеспечения ссылочной целостности.

Объём данных

– База данных содержит не менее 50 записей, включая данные о музыкантах, ансамблях, произведениях и пластинках.

Данная структура обеспечивает удобство хранения, обработки и анализа данных музыкального магазина.

3.2 Взаимодействие с базой данных

В рамках разработки программы для взаимодействия с базой данных используется библиотека Npgsql.

Подключение к базе данных осуществляется с помощью следующего кода, который задает строку подключения:

```
Private const string ConnectionString = "Host=localhost;Port=5432;Database=Music_store;Username=postgres;Password=1111;SearchPath=kursach";
```

Затем создается объект подключения к базе данных:

```
var connection = new NpgsqlConnection(ConnectionString)
```

Для работы с данными в базе применяются стандартные SQL-запросы. Например, для получения информации из таблицы используется команда SELECT, для изменения существующих записей — команда UPDATE, для добавления новых записей в таблицу — команда INSERT, а для удаления данных — команда DELETE.

3.3 Структура программы

В разработанной программе содержится 7 форм и 5 классов сущностей

Сначала рассмотрим структуру и предназначение форм.

Форма Menu содержит 5 кнопок и текстовое поле (рисунок 2).

- в текстовом поле содержится сообщение для пользователя;
- кнопка «Музыканты» открывает форму MusiciansForm;
- кнопка «Ансамбли» открывает форму EnsemblesForm;
- кнопка «Клиенты» открывает форму ClientsForm;
- кнопка «Композиции» открывает форму CompositionsForm;
- кнопка «Пластинки» открывает форму VinylRecordForm.

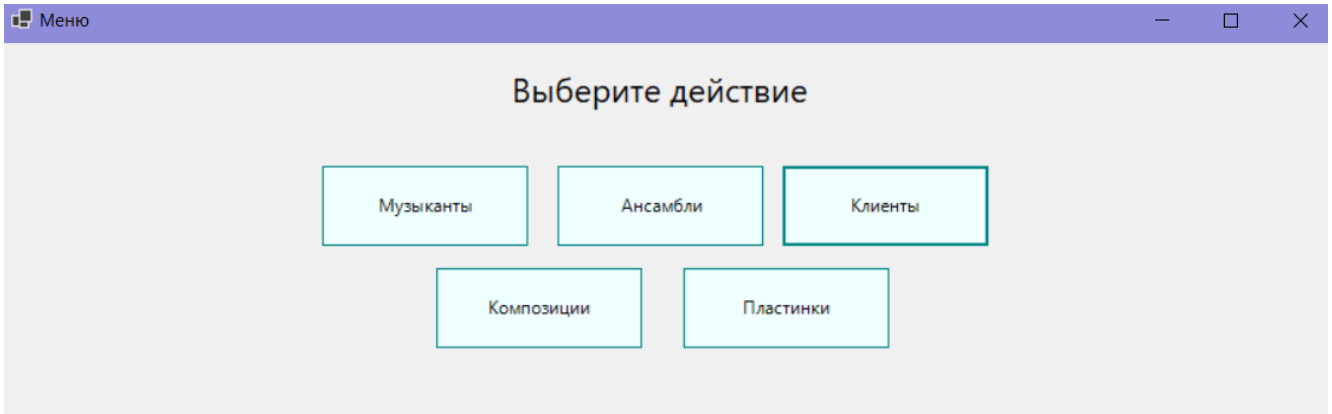

Рисунок 2 – Форма Menu

Форма Musicians_Form содержит элемент DataGridView и 3 кнопки (рисунок 3).

- элемент DataGridView выводит информацию о музыкантах, содержащуюся в базе;

– кнопки «Редактировать» и «Добавить» открывают форму Add_Or_Edit_Form;

– кнопка «Удалить» удаляет выбранную запись как на элементе, так и в таблице в базе данных;

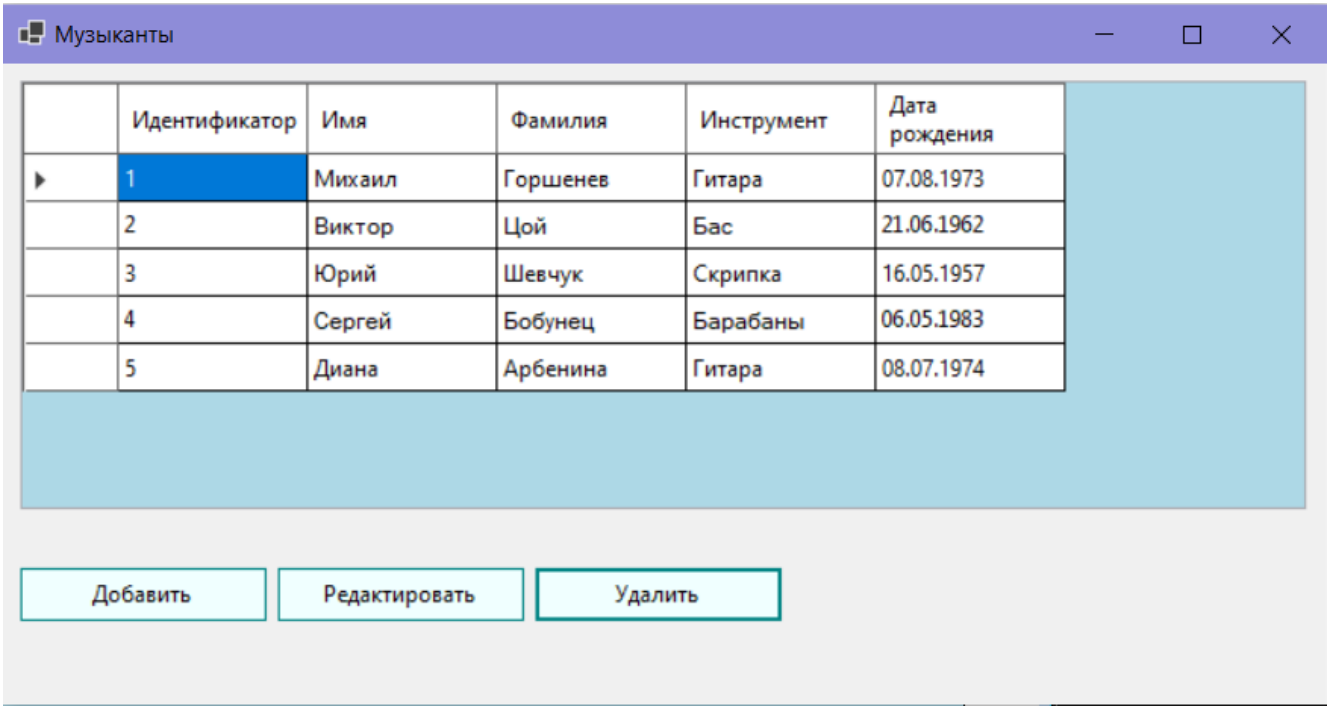

Рисунок 3 – Форма Musicians_Form

Форма Ensembles_Form содержит элемент DataGridView и 3 кнопки (рисунок 4).

– элемент DataGridView выводит информацию об ансамблях, содержащуюся в базе;

– кнопки «Редактировать» и «Добавить» открывают форму Add_Or_Edit_Form;

– кнопка «Удалить» удаляет выбранную запись как на элементе, так и в таблице в базе данных;

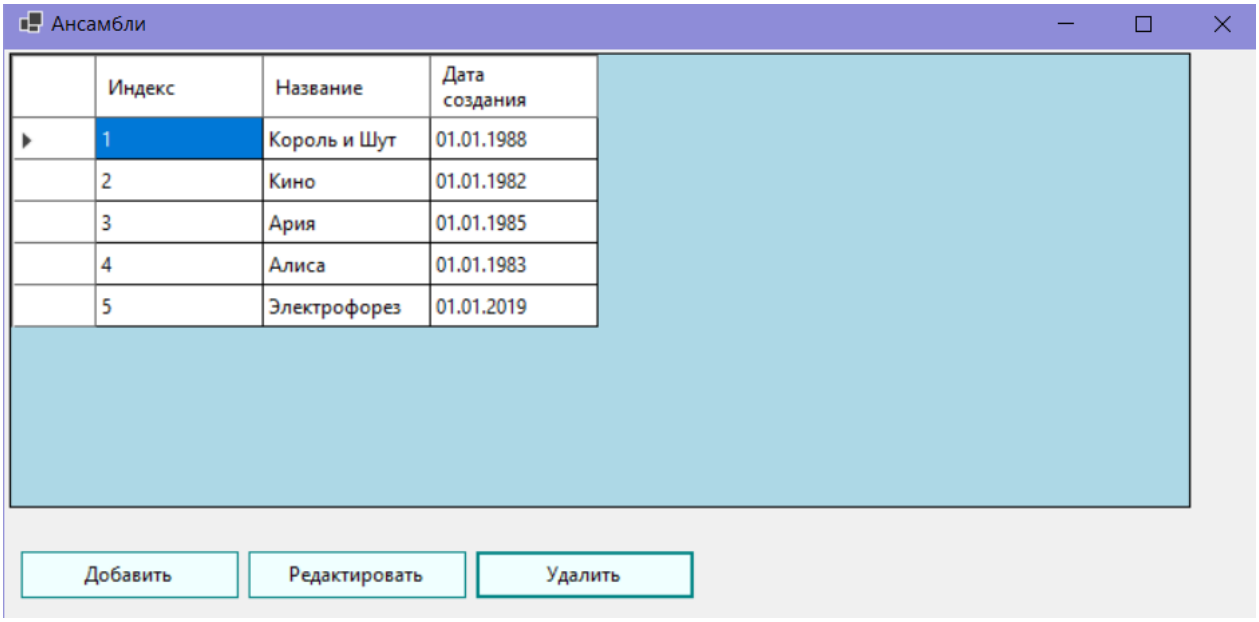

Рисунок 4 – Форма Ensembles_Form

Форма Clients_Form содержит элемент DataGridView и 3 кнопки у (рисунок 5).

– элемент DataGridView выводит информацию об ансамблях, содержащуюся в базе;

– кнопки «Редактировать» и «Добавить» открывают форму Add_Or_Edit_Form;

– кнопка «Удалить» удаляет выбранную запись как на элементе, так и в таблице в базе данных;

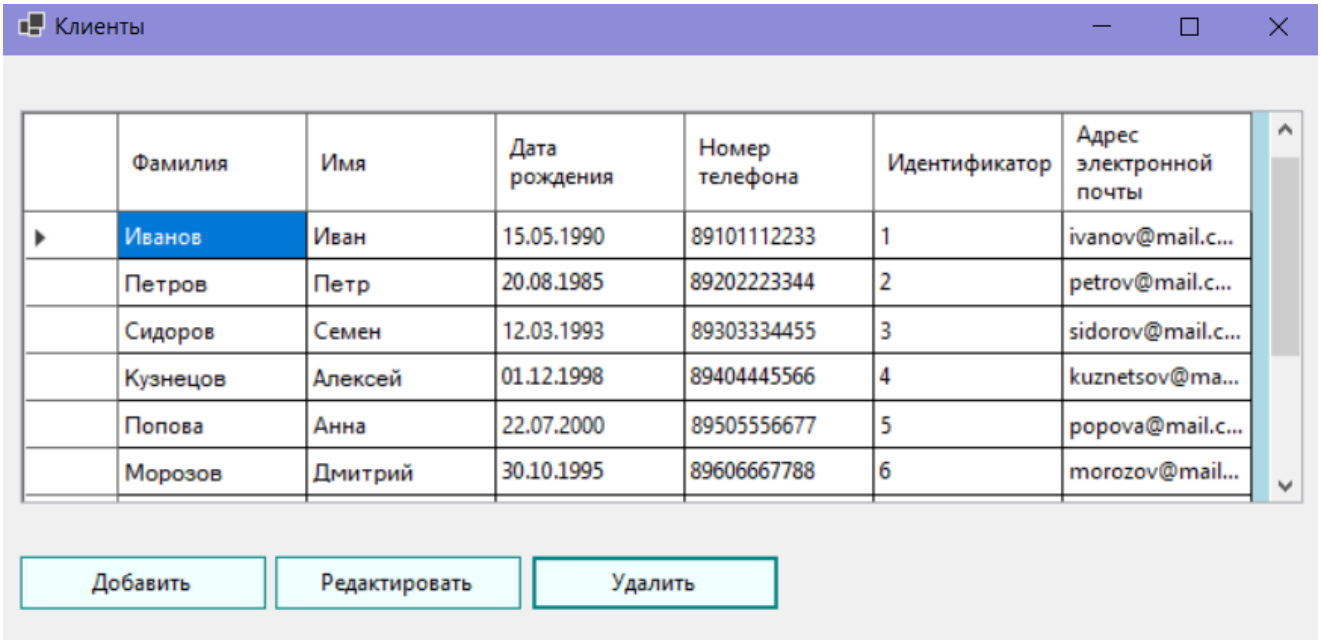

Рисунок 5 – Форма Clients_Form

Форма Compositions_Form содержит элемент DataGridView и 3 кнопки (рисунок 6)

– элемент DataGridView выводит информацию об ансамблях, содержащуюся в базе;

– кнопки «Редактировать» и «Добавить» открывают форму Add_Or_Edit_Form;

– кнопка «Удалить» удаляет выбранную запись как на элементе, так и в таблице в базе данных;

Рисунок 6 – Форма Compositions_Form

Форма Vinyl_Record_Form содержит элемент DataGridView и 4 кнопки (рисунок 7).

– элемент DataGridView выводит информацию о пластинках, содержащуюся в базе;
– кнопки «Редактировать» и «Добавить» открывают форму Add_Or_Edit_Form;
– кнопка «Удалить» удаляет выбранную запись как на элементе, так и в таблице в базе данных;
– кнопка «Сохранить» сохраняет изменения в форме Vinyl_Record_Form;

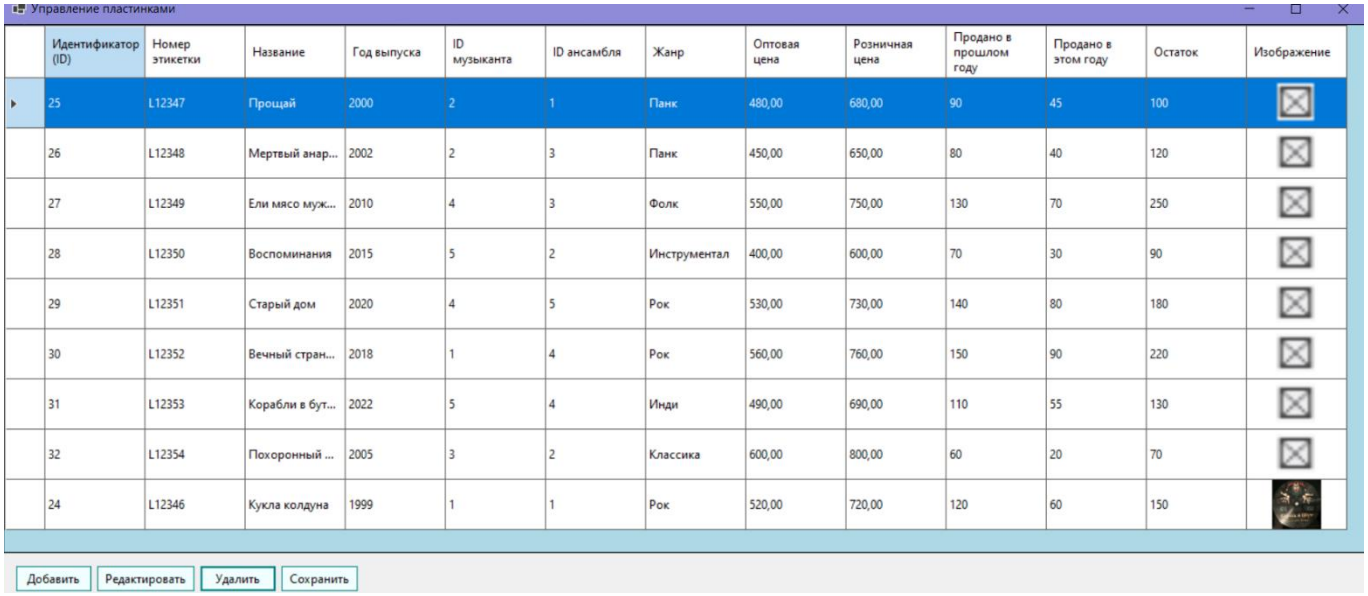

Рисунок 7 – Форма Vinyl_Record_Form

Форма Add_Or_Edit_Form универсальная и открывается при нажатии кнопок «Добавить» и «Редактировать», подстраивая поля для каждой формы. Она содержит 2 кнопки «Сохранить» и «Отмена», количество остальных элементов, таких как: ComboBox, TextBox, NumericUpDown, DateTimePicker, Label, FlowLayoutPanel зависит от сущности которую редактируют или добавляют в нее новое поле.

Для примера представлю вид формы «Редактирования/Добавления» в форме музыкантов (рисунок 8).

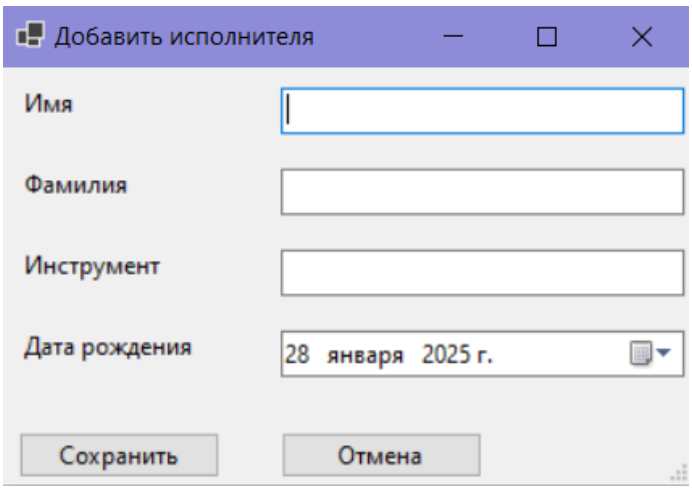

Рисунок 8 – Форма Add_Or_Edit_Form на примере музыкантов

Далее рассмотрим классы реализованные в программе

Класс Database

`public static class Database`

Поля

`ConnectionString`

Строка подключения к базе данных PostgreSQL.

Методы

`ExecuteNonQuery`

Выполняет SQL-команды, не возвращая данные.

`ExecuteReader`

Выполняет SQL-запрос и возвращает список объектов типа T.

`ClassToTableMap`

Словарь, который сопоставляет названия классов с именами таблиц в базе данных.

`GetTableName`

Возвращает имя таблицы по названию класса.

`FindMaxId`

Находит максимальный идентификатор в указанной таблице и возвращает значение, увеличенное на 1.

`GetMusicians`

Получает список всех музыкантов из таблицы musicians_catalog.

`AddMusician`

Добавляет нового музыканта в таблицу musicians_catalog.

`UpdateMusician`

Обновляет данные музыканта в таблице musicians_catalog.

`DeleteMusician`

Удаляет музыканта из таблицы musicians_catalog по его идентификатору.

`GetClients`

Получает список всех клиентов из таблицы clients_catalog

`AddClient`

Добавляет нового клиента в таблицу clients_catalog

`UpdateClient`

Обновляет данные клиента в таблице clients_catalog

`DeleteClient`

Удаляет клиента из таблицы clients_catalog по его идентификатору

`GetCompositions`

Получает список всех композиций из таблицы compositions_catalog

`AddComposition`

Добавляет новую композицию в таблицу compositions_catalog

`UpdateComposition`

Обновляет данные о композиции в таблице compositions_catalog

`DeleteComposition`

Удаляет композицию из таблицы compositions_catalog по её идентификатору

Класс Client

`public class Client`

Класс сущности клиент

Поля

`public string Surname`

Хранит фамилию клиента

`public string Name`

Хранит имя клиента

`public DateTime Birthday`

Хранит дату рождения клиента

`public string Phone_number`

Хранит телефонный номер клиента

`public int Id`

Хранит идентификатор клиента

`public string Email`

Хранит адрес электронной почты клиента

Класс Musician

`public class Musician`

Поля

`public int Id`

Хранит идентификатор музыканта

`public string Name`

Хранит имя музыканта

`public string Surname`

Хранит фамилию музыканта

`public string Instrument`

Хранит инструмент музыканта

`public DateTime Date_of_birth`

Хранит дату рождения музыканта

3.4 Инструкция пользователя

Программа предназначена для управления базой данных музыкального магазина, содержащей информацию о пластинках, ансамблях, музыкантах и музыкальных произведениях. С её помощью можно добавлять, редактировать, удалять данные, а также просматривать и анализировать информацию.

Основные возможности программы:

Управление пластинками

–Просмотр списка пластинок с детальной информацией (название, год выпуска, жанр, цены, остаток на складе и изображение).

– Добавление новой пластинки.

– Редактирование данных о пластинке.

– Удаление пластинки.

Работа с ансамблями и музыкантами

– Просмотр и редактирование списка ансамблей и музыкантов.

– Добавление и удаление ансамблей и музыкантов.

Главная таблица

– Расположена в центре формы.

– Отображает список записей (например, пластинок).

Кнопки управления

– «Добавить» создаёт новую запись.

– «Редактировать» открывает форму редактирования выбранной записи.

– «Удалить» удаляет выбранную запись.

– «Сохранить» сохраняет изменения.

Форма добавления/редактирования

Содержит поля для ввода данных:

– TextBox: Для ввода текста.

– NumericUpDown: Для чисел (цены, год выпуска, остатки).

– ComboBox: Для выбора связанного значения (например, ансамбля или музыканта).

– DateTimePicker: Для выбора дат.

– Кнопка загрузки изображения: Позволяет добавить обложку пластинки.

После ввода данных нажмите кнопку Сохранить для записи изменений.

Как пользоваться программой:

Просмотр данных

– После запуска программы откроется основная таблица, где вы видите текущий список пластинок.

Добавление новой записи

– Нажмите кнопку «Добавить».

– Заполните все необходимые поля в открывшейся форме.

– Нажмите «Сохранить», чтобы добавить запись в базу данных.

Редактирование записи

– Выберите нужную запись в таблице.

– Нажмите кнопку «Редактировать».

– Измените данные в форме и нажмите «Сохранить».

Удаление записи

– Выберите запись в таблице.

– Нажмите «Удалить». Подтвердите действие.

4. Тестирование АИС

4.1 Тестирование базы данных

Получение данных из базы данных происходит с помощью SELECT запросов. Чтобы проверить правильность загрузки данных в таблицу приложения сравним результаты запросов.

Пример запроса в программе:

```
public static List<Client> GetClients()
{
    try
    {
        return ExecuteReader("SELECT * FROM clients_catalog", reader => new Client
        {
            Surname = reader.GetString(0),
            Name = reader.GetString(1),
            Birthday = reader.GetDateTime(2),
            Phone_number = reader.GetString(3),
            Id = reader.GetInt32(4),
            Email = reader.GetString(5),

        });
    }
    catch (InvalidCastException ex)
    {
        Console.WriteLine($"Ошибка при чтении данных: {ex.Message}");
        throw;
    }
}
```

	Фамилия	Имя	Дата рождения	Номер телефона	Идентификатор	Адрес электронной почты
▶	Иванов	Иван	15.05.1990	89101112233	1	ivanov@mail.c...
	Петров	Петр	20.08.1985	89202223344	2	petrov@mail.c...
	Сидоров	Семен	12.03.1993	89303334455	3	sidorov@mail.c...
	Кузнецов	Алексей	01.12.1998	89404445566	4	kuznetsov@ma...
	Попова	Анна	22.07.2000	89505556677	5	popova@mail.c...
	Морозов	Дмитрий	30.10.1995	89606667788	6	morozov@mail...

Рисунок 9 – Результат запроса в программе

Пример запроса в используемом редакторе:

```
select*from kursach.clients_catalog
```

	surname text	name text	birthday date	phone_number character varying (15)	id [PK] integer	email text
1	Иванов	Иван	1990-05-15	89101112233	1	ivanov@mail.com
2	Петров	Петр	1985-08-20	89202223344	2	petrov@mail.com
3	Сидоров	Семен	1993-03-12	89303334455	3	sidorov@mail.com
4	Кузнецов	Алексей	1998-12-01	89404445566	4	kuznetsov@mail.com
5	Попова	Анна	2000-07-22	89505556677	5	popova@mail.com
6	Морозов	Дмитрий	1995-10-30	89606667788	6	morozov@mail.com
7	Соколова	Мария	1987-04-18	89707778899	7	sokolova@mail.com
8	Васильев	Евгений	1992-11-09	89808889900	8	vasiliev@mail.com
9	Федорова	Ольга	1989-06-25	89909990011	9	fedorova@mail.com
10	Николаев	Николай	1997-01-05	89010001122	10	nikolaev@mail.com

Рисунок 10 – Результат SQL запроса в редакторе

Сравнив рисунки 9 и 10 можно сделать вывод что запрос в программе работает правильно.

Необходимо также проанализировать запросы INSERT по тому же принципу, чтобы удостовериться, что программа корректно добавляет новые записи.

Пример запроса из программы:

```
public static void AddClient(Client client)
{
    ExecuteNonQuery(
        "INSERT INTO clients_catalog (id, name, surname, birthday, email, phone_number) VALUES (@id, @name, @surname, @birthday, @email, @phone_number)",
        command =>
        {
            command.Parameters.AddWithValue("@id", client.Id);
            command.Parameters.AddWithValue("@name", client.Name);
            command.Parameters.AddWithValue("@surname", client.Surname);
            command.Parameters.AddWithValue("@birthday", client.Birthday);
            command.Parameters.AddWithValue("@email", client.Email);
            command.Parameters.AddWithValue("@phone_number", client.Phone_number);
        }
    );
}
```

В рамках данной программы мы создадим тестового клиента и проверим, как он отображается в приложении. Результаты можно увидеть на рисунках 11 и 12.

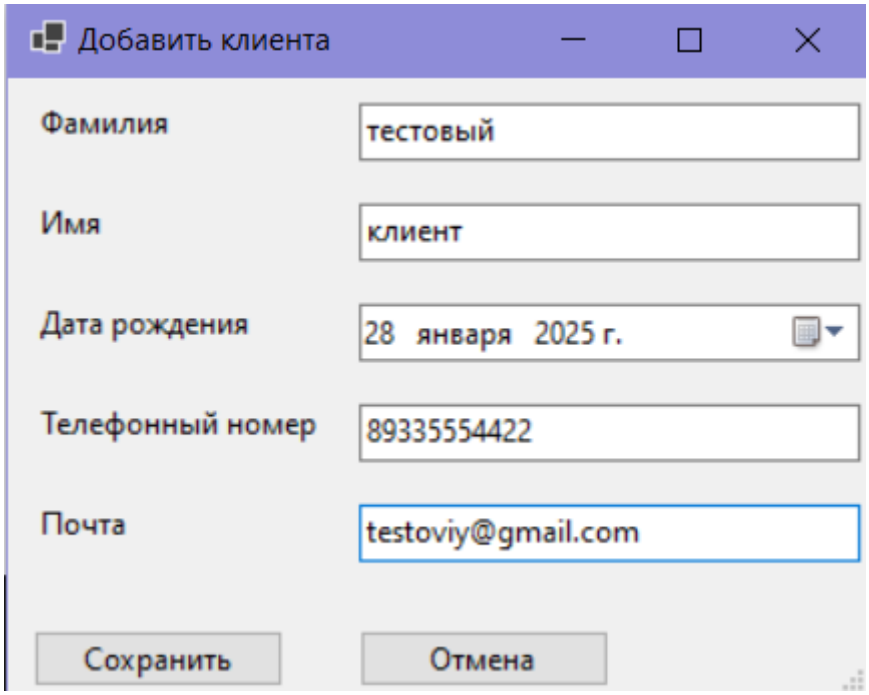

Рисунок 11 – Добавление клиента

	Фамилия	Имя	Дата рождения	Номер телефона	Идентификатор	Адрес электронной почты
	Морозов	Дмитрий	30.10.1995	89606667788	6	morozov@mail...
	Соколова	Мария	18.04.1987	89707778899	7	sokolova@mail...
	Васильев	Евгений	09.11.1992	89808889900	8	vasiliev@mail.c...
	Федорова	Ольга	25.06.1989	89909990011	9	fedorova@mail...
	Николаев	Николай	05.01.1997	89010001122	10	nikolaev@mail....
	тестовый	клиент	28.01.2025	89335554422	11	testoviy@gmail...

Рисунок 12 – Результат добавления тестового клиента

Тестовый клиент успешно добавился в таблицу в приложении. Теперь нужно убедиться, что он так же появился в базе данных.

	surname text	name text	birthday date	phone_number character varying (15)	id [PK] integer	email text
1	Иванов	Иван	1990-05-15	89101112233	1	ivanov@mail.com
2	Петров	Петр	1985-08-20	89202223344	2	petrov@mail.com
3	Сидоров	Семен	1993-03-12	89303334455	3	sidorov@mail.com
4	Кузнецов	Алексей	1998-12-01	89404445566	4	kuznetsov@mail.co...
5	Попова	Анна	2000-07-22	89505556677	5	popova@mail.com
6	Морозов	Дмитрий	1995-10-30	89606667788	6	morozov@mail.com
7	Соколова	Мария	1987-04-18	89707778899	7	sokolova@mail.com
8	Васильев	Евгений	1992-11-09	89808889900	8	vasiliev@mail.com
9	Федорова	Ольга	1989-06-25	89909990011	9	fedorova@mail.com
10	Николаев	Николай	1997-01-05	89010001122	10	nikolaev@mail.com
11	тестовый	клиент	2025-01-28	89335554422	11	testoviy@gmail.com

Рисунок 13 – Результат добавления тестового клиента

Тестовый клиент так же добавился в базу данных.

Теперь проверим DELETE запрос.

Пример запроса из программы:

```
public static void DeleteClient(int clientId)
{
    ExecuteNonQuery(
        "DELETE FROM clients_catalog WHERE id = @id",
        command => command.Parameters.AddWithValue("@id", clientId)
    );
}
```

При удалении сущности необходимо подтвердить действие. Предупреждение представлено на рисунке 14.

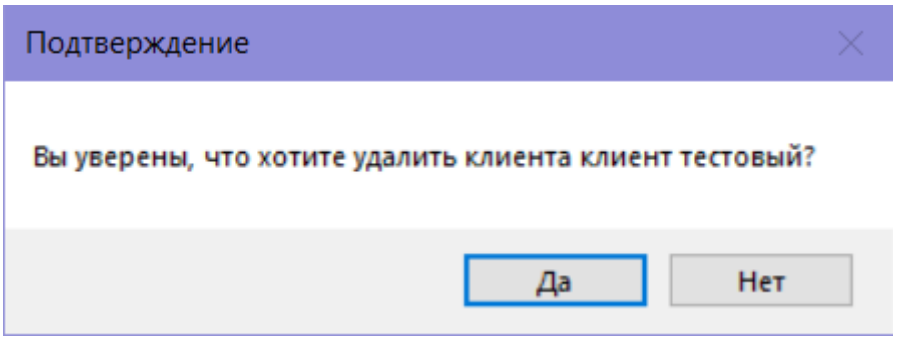

Рисунок 14.

Результаты удаления тестового клиента представлены на рисунках 15 и 16.

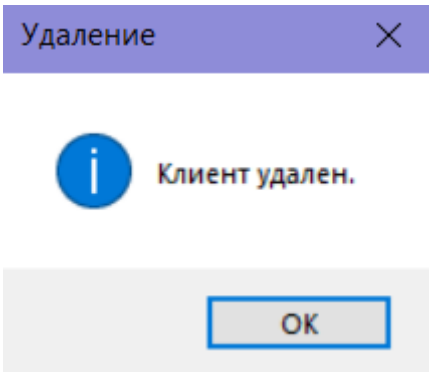

Рисунок 15.

	Фамилия	Имя	Дата рождения	Номер телефона	Идентификатор	Адрес электронной почты
	Кузнецов	Алексей	01.12.1998	89404445566	4	kuznetsov@ma...
	Попова	Анна	22.07.2000	89505556677	5	popova@mail.c...
	Морозов	Дмитрий	30.10.1995	89606667788	6	morozov@mail...
	Соколова	Мария	18.04.1987	89707778899	7	sokolova@mail...
	Васильев	Евгений	09.11.1992	89808889900	8	vasiliev@mail.c...
	Федорова	Ольга	25.06.1989	89909990011	9	fedorova@mail...

Рисунок 16.

Тестовый клиент успешно удалился из таблицы приложения. Теперь так же нужно проверить удалился ли он из базы.

	surname text	name text	birthday date	phone_number character varying (15)	id [PK] integer	email text
1	Иванов	Иван	1990-05-15	89101112233	1	ivanov@mail.com
2	Петров	Петр	1985-08-20	89202223344	2	petrov@mail.com
3	Сидоров	Семен	1993-03-12	89303334455	3	sidorov@mail.com
4	Кузнецов	Алексей	1998-12-01	89404445566	4	kuznetsov@mail.com
5	Попова	Анна	2000-07-22	89505556677	5	popova@mail.com
6	Морозов	Дмитрий	1995-10-30	89606667788	6	morozov@mail.com
7	Соколова	Мария	1987-04-18	89707778899	7	sokolova@mail.com
8	Васильев	Евгений	1992-11-09	89808889900	8	vasiliev@mail.com
9	Федорова	Ольга	1989-06-25	89909990011	9	fedorova@mail.com

Рисунок 17 – Результат удаления клиента

Из базы тестовый клиент удалился так же успешно, следовательно DELETE запросы работают правильно.

Еще один немаловажный запрос UPDATE, его тоже нужно проверить.

Пример запроса в программе:

```
        public static void UpdateComposition(Composition composition)
          {
              ExecuteNonQuery(
                  "UPDATE compositions_catalog SET name = @name, musician_id =
@musician_id, ensemble_id = @ensemble_id, release_year = @release_year WHERE id =
@id",
                  command =>
                  {
                      command.Parameters.AddWithValue("@id", composition.Id);
                      command.Parameters.AddWithValue("@name",
composition.Name);
                      command.Parameters.AddWithValue("@musician_id",
composition.MusicianId);
                      command.Parameters.AddWithValue("@ensemble_id",
composition.EnsembleId);
                      command.Parameters.AddWithValue("@release_year",
composition.ReleaseYear);
                  }
              );
          }
```

Точно так же выбираем клиента с которым хотим работать и нажимаем кнопку «Редактировать». Изменяем фамилию и почту клиента. Изначальный вариант представлен на рисунке 18, а результат запроса на рисунке 19.

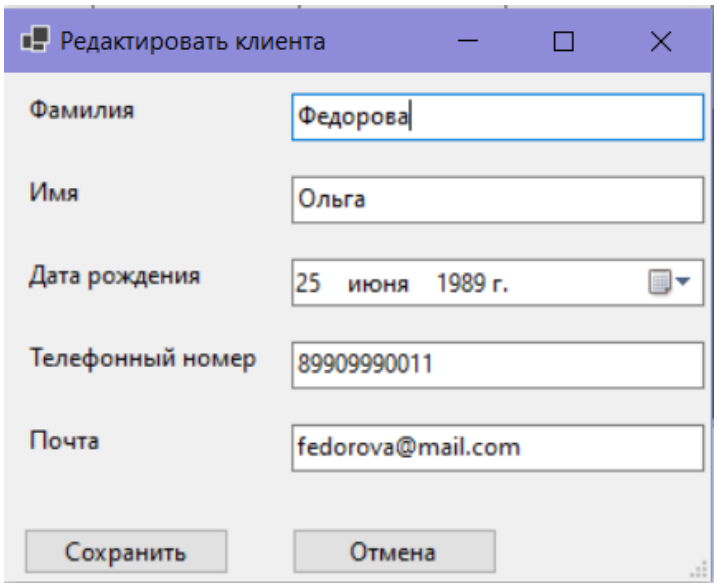

Рисунок 18.

	Фамилия	Имя	Дата рождения	Номер телефона	Идентификатор	Адрес электронной почты
	Кузнецов	Алексей	01.12.1998	89404445566	4	kuznetsov@ma...
	Попова	Анна	22.07.2000	89505556677	5	popova@mail.c...
	Морозов	Дмитрий	30.10.1995	89606667788	6	morozov@mail...
	Соколова	Мария	18.04.1987	89707778899	7	sokolova@mail...
	Васильев	Евгений	09.11.1992	89808889900	8	vasiliev@mail.c...
	Князева	Ольга	25.06.1989	89909990011	9	knyazeva@mail...

Рисунок 19.

Фамилия и почта клиента успешно изменились в таблице программы. Теперь проверим как это выглядит в базе.

	surname text	name text	birthday date	phone_number character varying (15)	id [PK] integer	email text
1	Иванов	Иван	1990-05-15	89101112233	1	ivanov@mail.com
2	Петров	Петр	1985-08-20	89202223344	2	petrov@mail.com
3	Сидоров	Семен	1993-03-12	89303334455	3	sidorov@mail.com
4	Кузнецов	Алексей	1998-12-01	89404445566	4	kuznetsov@mail.com
5	Попова	Анна	2000-07-22	89505556677	5	popova@mail.com
6	Морозов	Дмитрий	1995-10-30	89606667788	6	morozov@mail.com
7	Соколова	Мария	1987-04-18	89707778899	7	sokolova@mail.com
8	Васильев	Евгений	1992-11-09	89808889900	8	vasiliev@mail.com
9	Князева	Ольга	1989-06-25	89909990011	9	knyazeva@mail.com

Рисунок 20 – Результат редактирования клиента

В базе все тоже выглядит отлично.

Исходя из проделанных тестов делаем вывод что все запросы работают корректно как с базой, так и с программой.

4.2 Тестирование программы

Приложение должно быть устойчиво к разного рода проблемам, связанным с неправильным заполнением таблиц.

Для тестирования в таблице «Музыканты» добавим музыканта с незаполненным полем «инструмент» и посмотрим, как программа отреагирует на это.

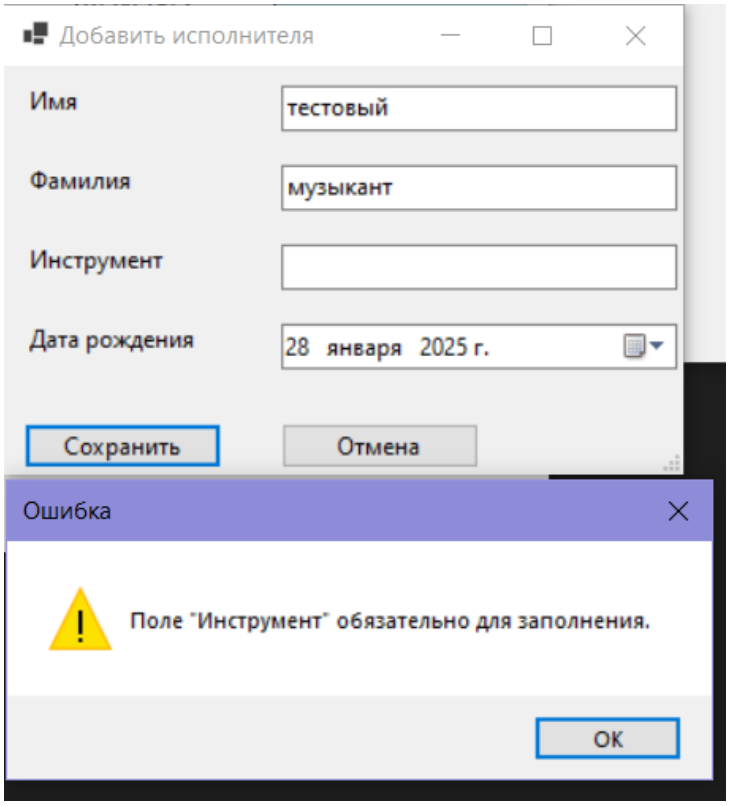

Рисунок 21 – Результат добавления музыканта с незаполненным полем

Программа не добавляет такого музыканта и выводит ошибку, говорящую о том, что мы не заполнили определенное поле.

При попытке редактирования или удаления программа просит выбрать строку, с которой мы хотим совершить действие (рисунки 22 и 23).

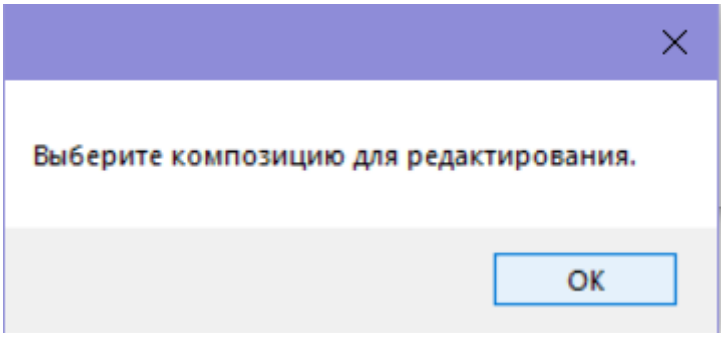

Рисунок 22.

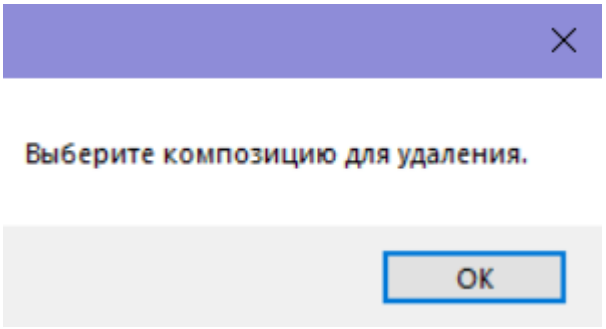

Рисунок 23.

Заключение

В ходе курсовой работы была разработана структура базы данных с использованием PostgreSQL и создан интерфейс на WinForms. Основное внимание уделялось нормализации данных, обеспечению целостности и удобству взаимодействия пользователя с системой.

Использование PostgreSQL обеспечило надежность и гибкость, а WinForms позволил создать интуитивно понятный интерфейс. Результаты работы подтверждают важность качественного проектирования баз данных и удобного пользовательского интерфейса для эффективного управления информацией.

Данная работа открывает возможности для дальнейшего изучения современных технологий в области баз данных и их применения в различных сферах.

Список литературы

1. Наместников, А.М. Базы данных. Практический курс. В 2ч. Ч. 1. Объектно-реляционные базы данных на примере PostgreSQL 9.5: учебное пособие / А.М. Наместников, А.А. Филиппов. – Ульяновск: УлГТУ, 2017. – 113с.

2. Самохвалов, Э. Н. Введение в проектирование и разработку приложений на языке программирования C#: учебное пособие / Э. Н. Самохвалов, Г. И. Ревунков, Ю. Е. Гапанюк. – М.: Издательство МГТУ им. Н. Э. Баумана, 2018. – 244 с.

3. О персональных данных: Федеральный закон от 27 июля 2006 года № 152-ФЗ // Собрание законодательства Российской Федерации. – 2006. - № 31. – С. 3451-3457.

4. Васюткина, И. А. Разработка приложений на C# с использованием СУБД PostgreSQL: учебное пособие : [Текст]/ И. А. Васюткина, Г. В. Трошина, М. И. Бычков, С. А. Менжулин . – Новосибирск: Новосибирский государственный технический университет, 2015. – 143 с.

Приложения

Приложение А

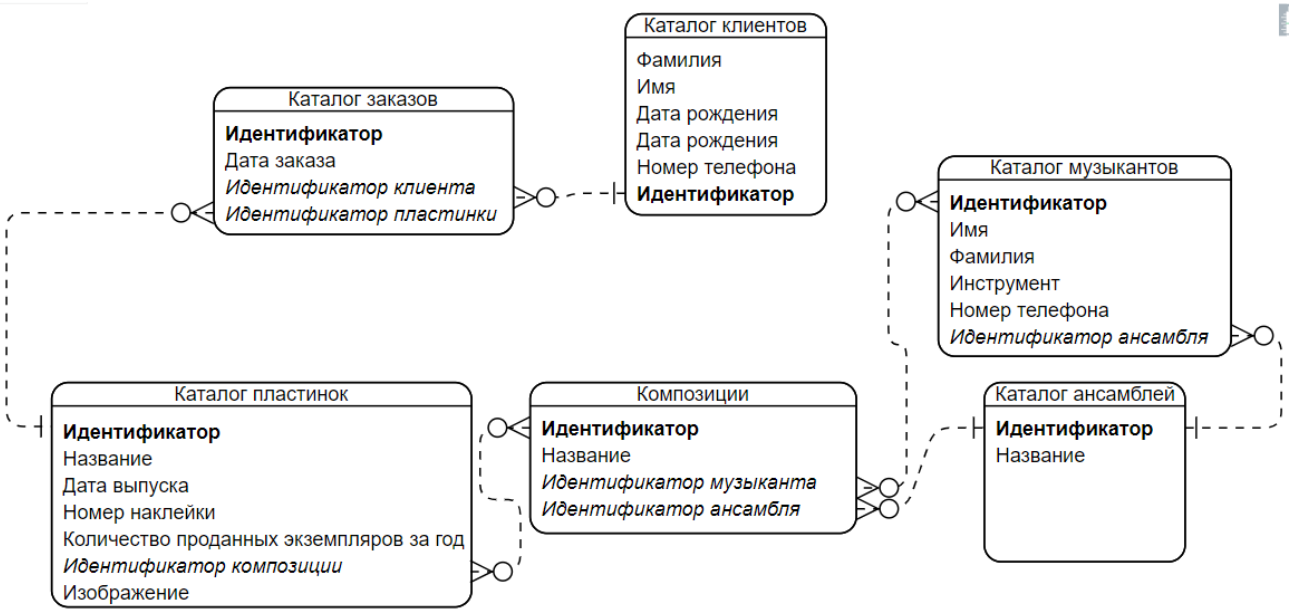

Рисунок А.1 – Логическая модель базы данных.

Приложение Б

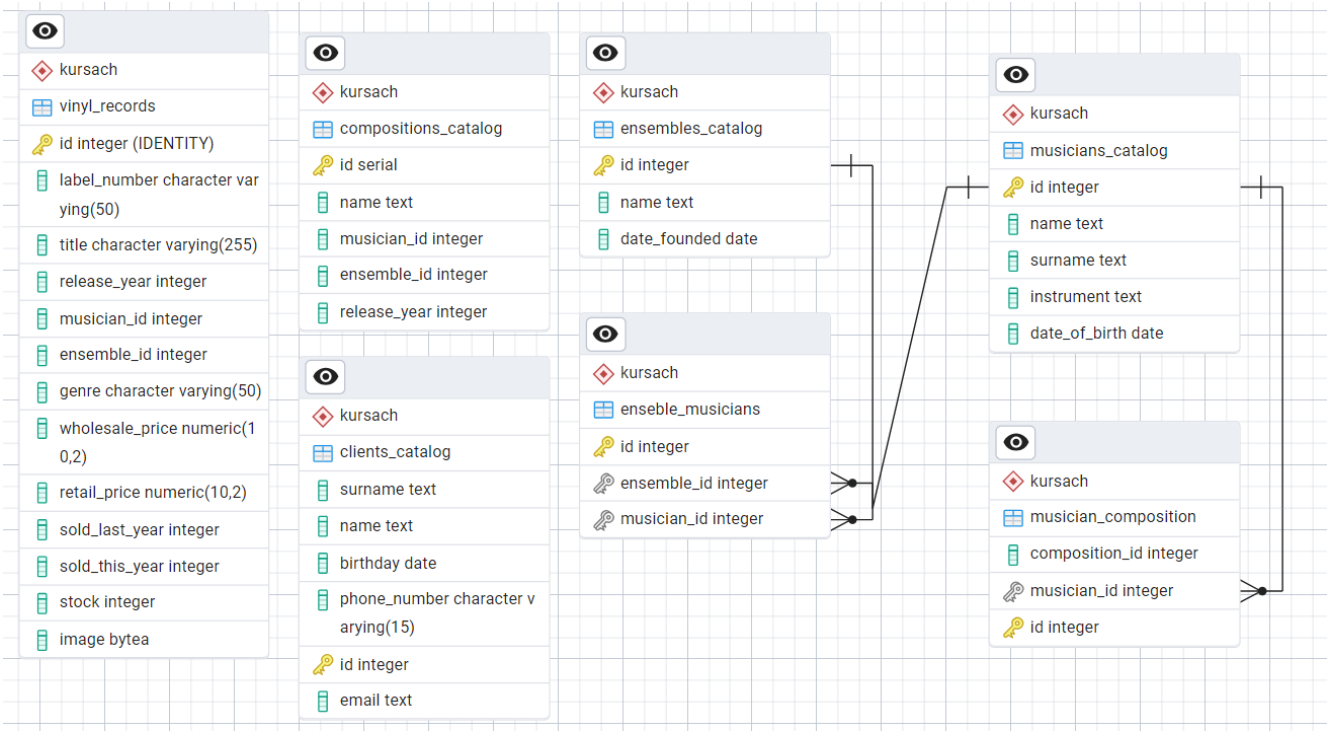

Рисунок Б.1 – Физическая модель базы данных

Приложение В

Ссылка на репозиторий с кодом:

https://github.com/BajiKeiske/SYBD

В данной курсовой работе разработана автоматизированная информационная система для музыкального магазина, целью которой является управление ассортиментом виниловых пластинок, а также учет музыкантов, ансамблей и музыкальных произведений. В работе рассматривается проектирование базы данных, включающей таблицы для хранения информации о музыкантах, ансамблях, произведениях и пластинках, с поддержкой связей между сущностями и функционалом для выполнения различных операций, таких как добавление, редактирование и удаление записей. Система реализует интерфейс для взаимодействия с данными через графический интерфейс, обеспечивая простоту работы с приложением для продавца.

Программная часть работы включает создание и использование различных форм для редактирования данных. Работа направлена на обеспечение эффективного управления запасами товара, учета продаж и формирования аналитической отчетности, что является важной частью современного музыкального магазина.

In this coursework, an automated information system for a music store is developed, aimed at managing the inventory of vinyl records, as well as keeping track of musicians, ensembles, and musical works. The work focuses on the design of a database that includes tables for storing information about musicians, ensembles, works, and records, with support for relationships between entities and functionality to perform various operations such as adding, editing, and deleting records. The system implements an interface for interacting with the data through a graphical interface, ensuring ease of use for the store's seller.

The software part of the work includes the creation and use of various forms for data editing. The project is aimed at ensuring effective inventory management, sales tracking, and generating analytical reports, which are key components of a modern music store.