

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования «Кубанский государственный университет»

Кафедра вычислительных технологий

ОТЧЕТ

о выполнении лабораторной работы № 9
по дисциплине «Конструирование алгоритмов и структур данных»

Выполнил: ст. гр. 26/1

Аванесов Р. А.

Проверил: доц. каф. ВТ

Полетайкин А.Н.

Краснодар

2024

Тема: Построение CRUD приложения для работы с базой данных PostgreSQL в MS Visual Studio 2019

Цель: Получить практические навыки разработки, тестирования, и построения CRUD приложения Windows Forms для работы с базой данных PostgreSQL.

Задание

1. В базу данных, разработанную в ходе лабораторной работы No5, добавить на

первичные ключи всех таблиц IDENTITY GENERATED ALWAYS, если ещё

не было установлено, для того чтобы ID генерировался автоматически.
2. Открыть в MS Visual Studio проект Windows Forms, разработанный в лабораторной работе No8. Создать универсальную форму, которая будет соответствовать модели CRUD (Create, Read, Update, Delete). Обеспечить открытие формы только в режиме R (Read), скрывая кнопки для других действий. 2.1. Read.

Обеспечить в форме чтение и отображение таблицы или представления из базы данных. Построить чтение данных таким образом, чтобы они возвращались с разделением на страницы (paginated view) с настраиваемым размером.

2.2. Delete.

Создать кнопку удаления, при нажатии на которую все выбранные записи удаляются из базы данных.

2.3. Для каждой таблицы создать свою форму добавления/изменения. Для ввода данных использовать TextBox. Для получения внешних

ключей обеспечить возможность открывать универсальную CRUD форму в режиме R для таблицы из которой должен быть получен id Ключа, и получать ID выбранной записи после её закрытия.

2.4. Create.

Создать кнопку добавления, при нажатии на которую открывается форма добавления/изменения записи с пустыми полями. 2.5. Update.

Создать кнопку изменения, при нажатии на которую открывается форма добавления/изменения записи с предзаполненными полями. Поля заполняются получением данных из родительской формы.

3. В главной форме изменить основное меню. В пунктах «Справочники» и «Данные» вызывать универсальную CRUD форму для таблиц. В пункте «Отчеты» вызывать универсальную CRUD форму в режиме R.
4. Выполнить описание разработанных форм приложения в виде табл. 9.1.

№ пп	Имя формы	Описание
---------	-----------	----------

Таблица 9.1. Перечень разработанных компонентов приложения

В отчет включить:

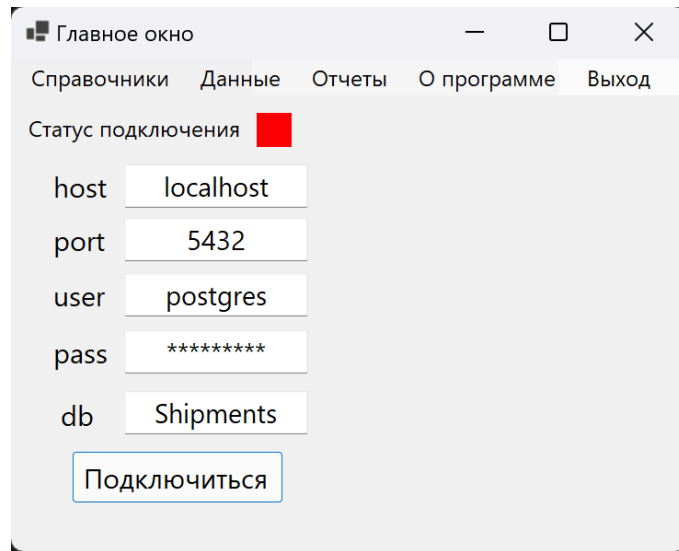
- перечень разработанных форм в виде таблицы 9.1;
- изображения разработанных форм приложения в режиме запуска с **полученными из базы данных тестовыми данными.**
- изображение обозревателя решений с развернутым проектом, листинги кода модуля Program.cs и всех разработанных форм.

Ход работы

Таблица 9.1

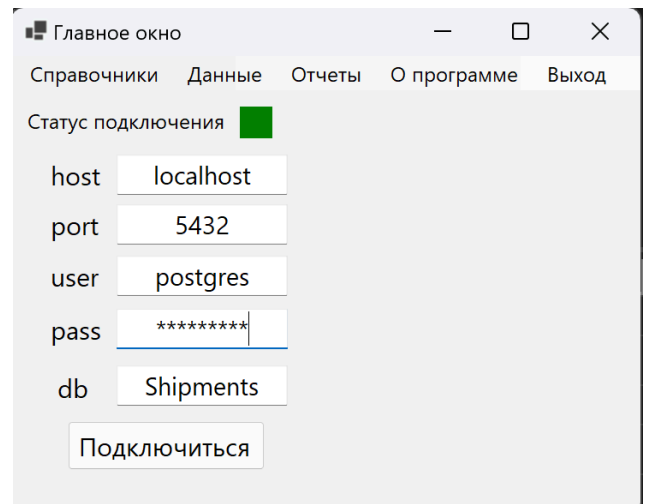
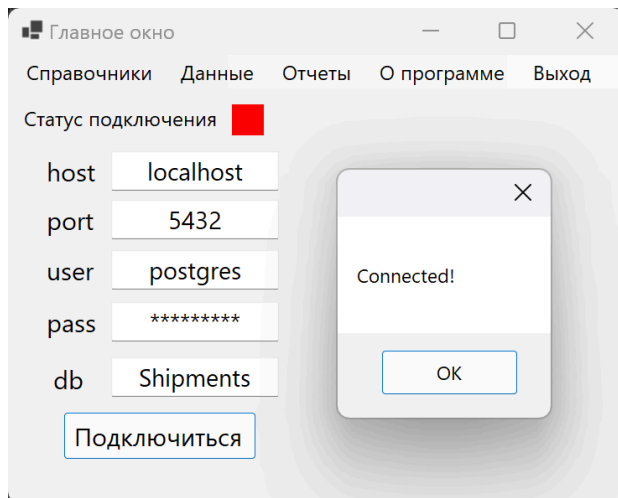
N	Имя формы	Описание
1	MainForm	Главная форма, содержит механизм подключения к базе данных, индикатор подключения и меню навигации.
2	CRUDForm	Основная форма, через которую происходит работа с данными из базы данных.
3	CreateUpdateForm	Форма, служащая для создания новых записей или изменения полей конкретной записи, выбранной в CRUDForm, соответствующих записям базы данных.
4	ChooseForm	Форма, которая является служебной, благодаря ей пользователь может наглядно выбрать запись для работы во время изменения или добавления данных. (Наглядный выбор записей, которые являются внешними ключами)
5	AboutForm	Форма, отображающая статическую информацию о программе.

1. Далее приведу примеры работы с приложением: при запуске программы появляется главное окно

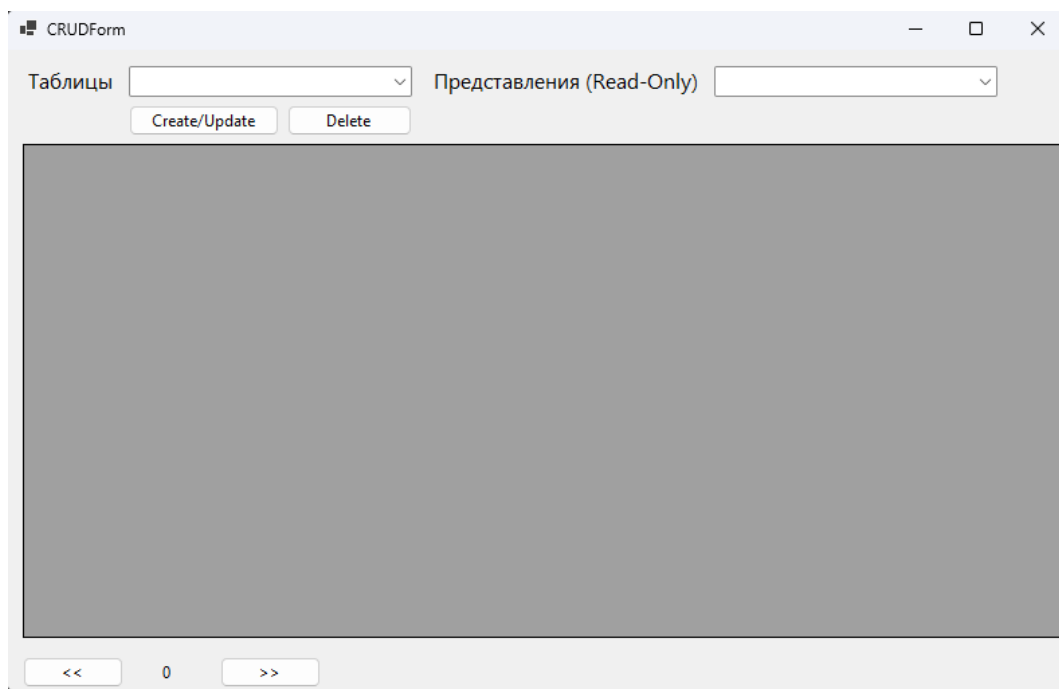


Затем

подключившись к базе данных, индикатор загорается зеленым цветом и появляется текстовое окно, информирующее о статусе подключения:

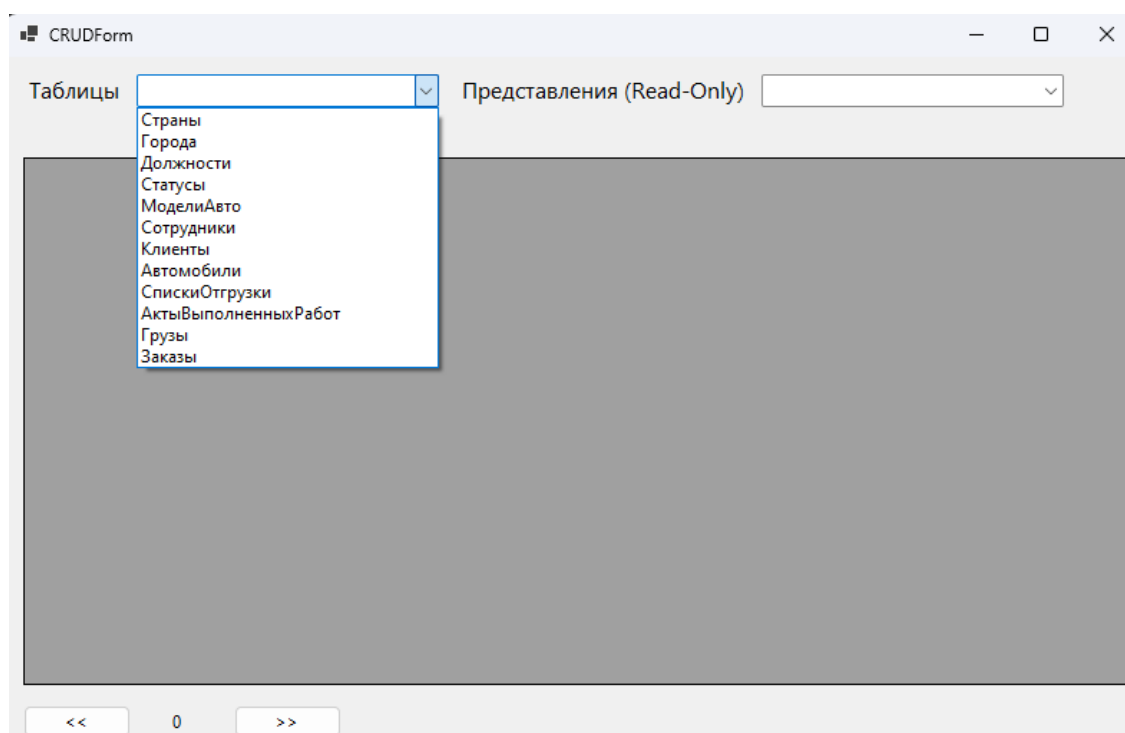


Выбрав в меню пункт “Справочники” или “Данные” появляется окно, отображающее CRUD форму, в котором можно выбрать таблицу из базы данных и посмотреть на ее содержимое, также в вкладке Представления содержится информация об Отчетах, она доступна только в режиме чтения:



The screenshot shows a window titled "CRUDForm". At the top, there are two dropdown menus: "Таблицы" (Tables) and "Представления (Read-Only)" (Views (Read-Only)). Below the "Таблицы" dropdown are two buttons: "Create/Update" and "Delete". The main area of the window is a large, empty gray rectangle. At the bottom, there are three buttons: "<<", "0", and ">>".

Далее нажав на поле Таблицы появляются все таблицы как оперативных, так и справочных данных.



The screenshot shows the same "CRUDForm" window, but the "Таблицы" dropdown menu is open, displaying a list of database tables. The list includes: Страны, Города, Должности, Статусы, МоделиАвто, Сотрудники, Клиенты, Автомобили, СпискиОтгрузки, АктыВыполненныхРабот, Грузы, and Заказы. The "Представления (Read-Only)" dropdown menu remains empty. The main area is still a large, empty gray rectangle. The bottom buttons "<<", "0", and ">>" are also visible.

Примеры выбранных таблиц:

CRUDForm

Таблицы

Города

Представления (Read-Only)

Create/Update

Delete

	ИдентификаторГорода	Название
▶	1	Москва
	2	Краснодар
	3	Нью-Йорк

<< 0 >>

CRUDForm

Таблицы

Сотрудники

Представления (Read-Only)

Create/Update

Delete

	ИдентификаторС	Имя	Фамилия	Отчество	_Должность	Телефон	НомерТрудовой	_Страна	_Город	Адрес	_Статус
▶	2	Мария	Петрова	Александровна	Администратор	89601234523	ТБ7654321	Россия	Краснодар	пр. Мира, д. 2	Неактивен
	3	Алексей	Сергеев	Петрович	Администратор	89601235353	ТБ1238901	Россия	Москва	ул. Березовая, ...	Активен
	1	Иван	Иванов	Иванович	Водитель	89601234553	ТБ123456	Россия	Москва	ул. Лесная, д. 1	Активен

<< 0 >>

CRUDForm

Таблицы

Заказы

Представления (Read-Only)

Create/Update

Delete

	ИдентификаторЗаказа	_ОтветственныйСотрудник	_Клиент	_ИдентификаторЕдиницыОтгрузки	ДатаОтгрузки	_Статус	_ИдентификаторАкта
▶	1	Иван Иванов	Алиса Жукова	1	01.04.2023	Выполнен	1
	2	Иван Иванов	Алиса Жукова	2	03.04.2023	Выполнен	2
	3	Мария Петрова	Борис Сидоров	2	11.11.2024	Выполнен	2

<< 0 >>

Далее продемонстрирую возможности удаления элементов из таблицы: удалю клиента, для этого выберу эту запись выделив все поля и нажму на кнопку удалить запись:

CRUDForm

Таблицы

Клиенты

Представления (Read-Only)

Create/Update

Delete

	ИдентификаторКлиента	Имя	Фамилия	Отчество	ДеталиКлиента	Телефон	_Страна	_Город	Адрес
	1	Алиса	Жукова		Номер счета: 123456	89231235353	США	Москва	ул. Садовая, д. 3
▶	6	Роман	Петров	Сергеевич	Номер счета: 562123	89701243222	США	Нью-Йорк	Колотушкина 12
	2	Борис	Сидоров		Номер счета: 153236	89121235353	США	Нью-Йорк	ул. Заречная, д...

<< 0 >>

CRUDForm

Таблицы

Клиенты

Представления (Read-Only)

Create/Update

Delete

	ИдентификаторКлиента	Имя	Фамилия	Отчество	ДеталиКлиента	Телефон	_Страна	_Город	Адрес
	1	Алиса	Жукова		Номер счета: 123456	89231235353	США	Москва	ул. Садовая, д. 3
▶	6	Роман	Петров	Сергеевич	Номер счета: 562123	89701243222	США	Нью-Йорк	Колотушкина 12
	2	Борис	Сидоров		Номер счета: 153236	89121235353	США	Нью-Йорк	ул. Заречная, д...

Выбранные записи успешно удалены.
OK

<< 0 >>

CRUDForm

Таблицы

Клиенты

Представления (Read-Only)

Create/Update

Delete

	ИдентификаторКлиента	Имя	Фамилия	Отчество	ДеталиКлиента	Телефон	_Страна	_Город	Адрес
▶	1	Алиса	Жукова		Номер счета: 123456	89231235353	США	Москва	ул. Садовая, д. 3
	2	Борис	Сидоров		Номер счета: 153236	89121235353	США	Нью-Йорк	ул. Заречная, д. 4

<< 0 >>

Далее продемонстрирую добавление элементов в базу данных: добавлю новый автомобиль.

The screenshot shows the 'CRUDForm' application window. At the top, there are two dropdown menus: 'Таблицы' (Tables) set to 'Автомобили' and 'Представления (Read-Only)' (Views (Read-Only)). Below these are two buttons: 'Create/Update' and 'Delete'. The main area contains a table with the following data:

	VIN	Производитель	РегистрационныйНомер	_Модель	Пробег	_Статус
▶	VIN1234567AB12GV	ГАЗ	08880023	Калина	350000	Неактивен

Below the table, there are navigation buttons: '<<', '0', and '>>'.

Выбрав нужную таблицу, не выделяя записей нажму на кнопку Create/Update:

The screenshot shows the 'CRUDForm' application window with the 'CreateUpdateForm' dialog box open. The dialog box has a table with the following data:

	VIN	Производитель	РегистрационныйНо	_Модель	Пробег	_Статус
*						

Below the table, there are two buttons: 'Сохранить' (Save) and 'Отменить' (Cancel).

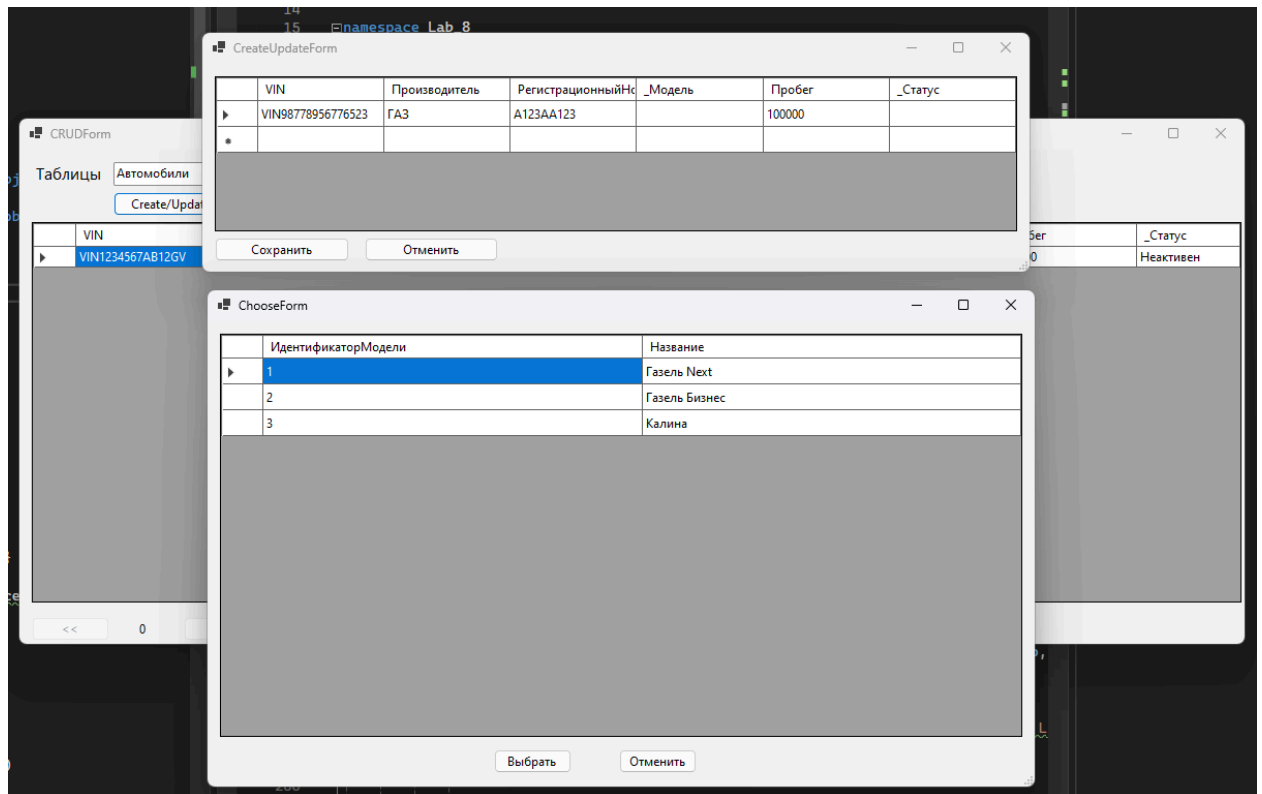
Появилась форма добавления записи, введу поля:

The screenshot shows the 'CRUDForm' application window with the 'CreateUpdateForm' dialog box open. The dialog box has a table with the following data:

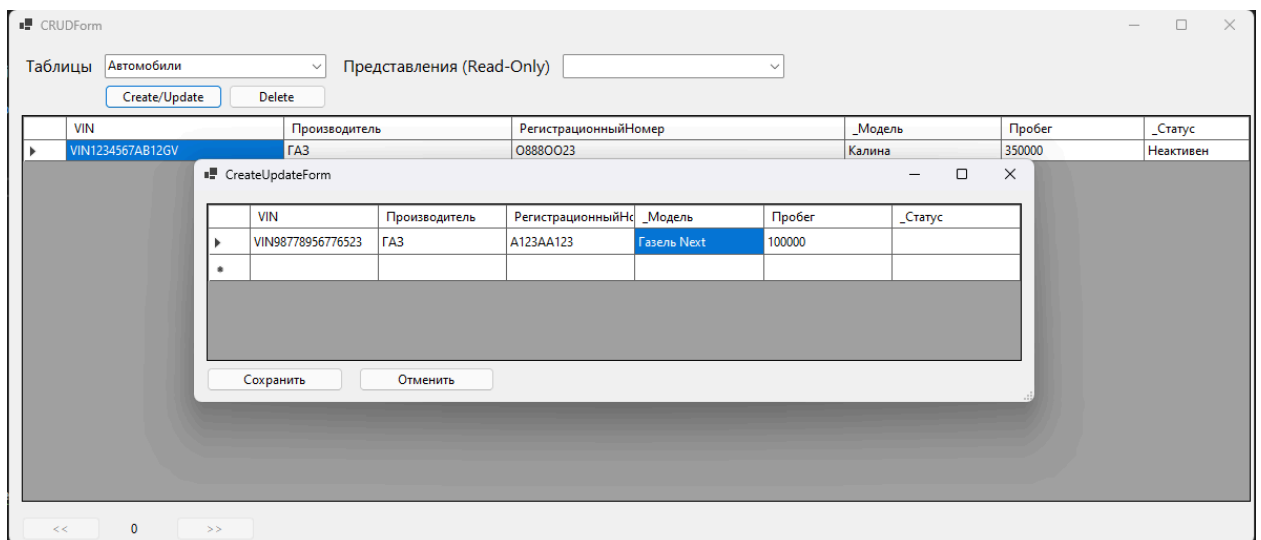
	VIN	Производитель	РегистрационныйНо	_Модель	Пробег	_Статус
✎	VIN98778956776523	ГАЗ	A123AA123		100000	
*						

Below the table, there are two buttons: 'Сохранить' (Save) and 'Отменить' (Cancel).

Для того чтобы заполнить поля, которые являются внешними ключами (начинаются с нижнего подчеркивания), нажму двойным кликом на это поле:



Появляется форма, в которой пользователю предлагают выбрать запись, которая бы ему подходила в качестве внешнего ключа, для этого в появившейся форме нужно выделить запись и нажать кнопку Выбрать:



Аналогично и с полем _Статус:

The 'CreateUpdateForm' dialog box contains a table with the following data:

	VIN	Производитель	РегистрационныйНомер	_Модель	Пробег	_Статус
▶	VIN98778956776523	ГАЗ	A123AA123	Газель Next	100000	
*						

The 'ChooseForm' dialog box contains a table with the following data:

ИдентификаторСтатуса	Описание
1	Активен
2	Неактивен
3	Выполнен
11	Удален
12	Получен

The 'CreateUpdateForm' dialog box contains a table with the following data:

	VIN	Производитель	РегистрационныйНомер	_Модель	Пробег	_Статус
▶	VIN98778956776523	ГАЗ	A123AA123	Газель Next	100000	Неактивен
*						

Затем нажав кнопку Сохранить новая запись будет сохранена в базу данных:

The 'CreateUpdateForm' dialog box contains a table with the following data:

	VIN	Производитель	РегистрационныйНомер	_Модель	Пробег	_Статус
▶	VIN98778956776523	ГАЗ	A123AA123	Газель Next	100000	Неактивен
*						

A message box is displayed with the text: "Запись успешно добавлена" (Record successfully added).

CRUDForm

Таблицы: **Автомобили** Представления (Read-Only)

Create/Update Delete

VIN	Производитель	РегистрационныйНомер	Модель	Пробег	Статус
VIN98778956776523	GAZ	A123AA123	Газель Next	100000	Неактивен
VIN12345678912GV	GAZ	O888OO23	Калина	350000	Неактивен

<< 0 >>

Следующим шагом станет демонстрация изменения полей записи:

Выберу таблицу Клиенты и поменяю клиенту 1 Страну и Имя:

CRUDForm

Таблицы: **Клиенты** Представления (Read-Only)

Create/Update Delete

ИдентификаторКлиента	Имя	Фамилия	Отчество	ДеталиКлиента	Телефон	Страна	Город	Адрес
1	Алиса	Жукова		Номер счета: 123456	89231235353	США	Москва	ул. Садовая, д. 3
2	Борис	Сидоров		Номер счета: 153236	89121235353	США	Нью-Йорк	ул. Заречная, д...

<< 0 >>

Теперь чтобы изменить запись нужно выделить ее и нажать на кнопку Create/Update:

CRUDForm

Таблицы: **Клиенты** Представления (Read-Only)

Create/Update Delete

ИдентификаторКлиента	Имя	Фамилия	Отчество	ДеталиКлиента	Телефон	Страна	Город	Адрес
1	Алиса	Жукова		Номер счета: 123456	89231235353	США	Москва	ул. Садовая, д. 3
2	Борис	Сидоров		Номер счета: 153236	89121235353	США	Нью-Йорк	ул. Заречная, д...

<< 0 >>

CreateUpdateForm

Идентификат	Имя	Фамилия	Отчество	ДеталиКлиент	Телефон	Страна	Город	Адрес
1	Алиса	Жукова		Номер сче...	89231235353	США	Москва	ул. Садовая...
*								

Сохранить Отменить

Появляется форма, в которой уже заполнены поля, далее изменяем те, которые требуется.

Поле фамилия можно изменять как поле обычного текстовбкса:

The screenshot shows the 'CRUDForm' application window. At the top, there are dropdown menus for 'Таблицы' (Clients) and 'Представления (Read-Only)'. Below them are 'Create/Update' and 'Delete' buttons. A table lists client data with columns: ИдентификаторКлиента, Имя, Фамилия, Отчество, ДеталиКлиента, Телефон, _Страна, _Город, and Адрес. Two rows are visible: one for 'Алиса Жукова' and another for 'Борис Сидоров'. A modal window titled 'CreateUpdateForm' is open, displaying a form for editing the first client. The form has the same column structure as the main table, with input fields for each value. At the bottom of the modal are 'Сохранить' (Save) and 'Отменить' (Cancel) buttons.

ИдентификаторКлиента	Имя	Фамилия	Отчество	ДеталиКлиента	Телефон	_Страна	_Город	Адрес
1	Алиса	Жукова		Номер счета: 123456	89231235353	США	Москва	ул. Садовая, д. 3
2	Борис	Сидоров		Номер счета: 153236	89121235353	США	Нью-Йорк	ул. Заречная, д...

Идентификат	Имя	Фамилия	Отчество	ДеталиКлиен	Телефон	_Страна	_Город	Адрес
1	Алиса	Красная		Номер сче...	89231235353	США	Москва	ул. Садовая...
*								

Поле _Страна – поле внешний ключ, поэтому нажав на него двойным кликом появляется ChooseForm:

This screenshot shows the 'ChooseForm' modal window, which appears when a user double-clicks the '_Страна' field in the 'CreateUpdateForm'. The 'ChooseForm' window contains a table with two columns: 'ИдентификаторСтраны' and 'Название'. It lists two options: '1' for 'Россия' and '2' for 'США'. At the bottom are 'Выбрать' (Select) and 'Отменить' (Cancel) buttons. In the background, the 'CRUDForm' window and the 'CreateUpdateForm' modal are still visible, showing the context of the selection.

ИдентификаторСтраны	Название
1	Россия
2	США

CRUDForm

Таблицы

Клиенты

Представления (Read-Only)

Create/Update

Delete

	ИдентификаторКлиента	Имя	Фамилия	Отчество	ДеталиКлиента	Телефон	_Страна	_Город	Адрес
▶	1	Алиса	Жукова		Номер счета: 123456	89231235353	США	Москва	ул. Садовая, д. 3
	2	Борис	Сидоров		Номер счета: 153236	89121235353	США	Нью-Йорк	ул. Заречная, д...

CreateUpdateForm

	Идентификат	Имя	Фамилия	Отчество	ДеталиКлиен	Телефон	_Страна	_Город	Адрес
▶	1	Алиса	Красная		Номер сче...	89231235353	Россия	Москва	ул. Садовая...
*									

Сохранить

Отменить

CRUDForm

Таблицы

Клиенты

Представления (Read-Only)

Create/Update

Delete

	ИдентификаторКлиента	Имя	Фамилия	Отчество	ДеталиКлиента	Телефон	_Страна	_Город	Адрес
▶	1	Алиса	Жукова		Номер счета: 123456	89231235353	США	Москва	ул. Садовая, д. 3
	2	Борис	Сидоров		Номер счета: 153236	89121235353	США	Нью-Йорк	ул. Заречная, д...

CreateUpdateForm

	Идентификат	Имя	Фамилия	Отчество	ДеталиКлиен	Телефон	_Страна	_Город	Адрес
▶	1	Алиса	Красная		Номер сче...	89231235353	Россия	Москва	ул. Садовая...
*									

Сохранить

Отменить

Запись успешно обновлена

OK

CRUDForm

Таблицы

Клиенты

Представления (Read-Only)

Create/Update

Delete

	ИдентификаторКлиента	Имя	Фамилия	Отчество	ДеталиКлиента	Телефон	_Страна	_Город	Адрес
▶	1	Алиса	Красная		Номер счета: 123456	89231235353	Россия	Москва	ул. Садовая, д. 3
	2	Борис	Сидоров		Номер счета: 153236	89121235353	США	Нью-Йорк	ул. Заречная, д...

Для того чтобы открыть содержимое таблиц в режиме чтения, нужно в главном меню выбрать вкладку Отчеты, появится CRUD форма, в которой будут заблокированы кнопки редактирования и dataGridView, отображающий таблицу базы данных, будет находиться в режиме ReadOnly.

CRUDForm (Read-Only Mode)

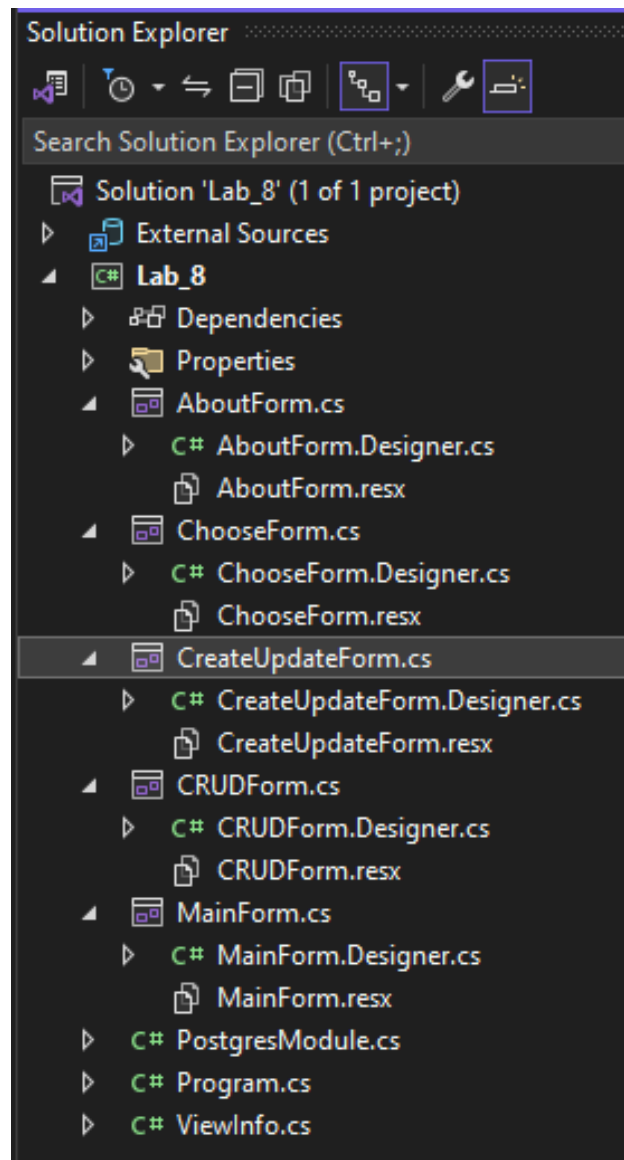
Таблицы: Должности Представления (Read-Only):

Create/Update Delete

ИдентификаторДолжности	Название
1	Водитель
2	Администратор

<< 0 >>

В результате проделанной работы проект выглядит следующим образом:



Вывод: в ходе лабораторной работы освоил возможности MS Visual Studio в создании приложений, отображающих содержимое базы данных.

Познакомился с возможностями платформы связывать создаваемые решения с внешними СУБД, в частности PostgreSQL. Разработал CRUD приложение, с помощью которого можно просматривать и редактировать содержимое базы данных.

Листинг программы.

Файл “Program.cs”

```
namespace Lab_8
{
    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 MainForm());
        }
    }
}
```

Файл “PostgresModule.cs”

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Web;
using System.Windows.Forms;
using Npgsql;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;
```

```

namespace Lab_8
{
    class PostgresModule
    {
        public static NpgsqlConnection _connection = null;

        public static void OpenConnection(string host, string port,
string user, string pass, string db)
        {
            if (_connection != null)
            {
                if (_connection.State ==
System.Data.ConnectionState.Open) _connection.Close();
                _connection.Dispose();
            }

            _connection = new NpgsqlConnection(@"Server=" + host +
";Port=" + port + ";User Id=" + user + ";Password=" + pass +
";DataBase=" + db);

            _connection.Open();

            if (_connection.State == System.Data.ConnectionState.Open)
MessageBox.Show("Connected!");
            else MessageBox.Show("Not connected!");
        }

        public static bool getConnectionStatus()
        {
            try
            {
                if (_connection != null)
                {
                    return _connection.State ==
System.Data.ConnectionState.Open;
                }
                throw new Exception("No connection!");
            }
        }
    }
}

```

```

        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            return false;
        }
    }

    public static DataTable? getSqlComResult(string sql_com)
    {
        DataTable dataTable = new DataTable();
        try
        {
            using (NpgsqlCommand command =
                _connection.CreateCommand())
            {
                command.CommandText = sql_com;

                using (NpgsqlDataReader reader =
                    command.ExecuteReader())
                {
                    dataTable.Load(reader);
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            return null;
        }

        return dataTable;
    }

    public static DataTable? getDataTable(string tableName)
    {
        try

```

```

        {
            string sql_com = "SELECT * FROM public." + tableName;
            return getSqlComResult(sql_com);
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error!\n" + ex.Message);
            return null;
        }
    }

    public static DataTable? getDataTable(string tableName, string
condition)
    {
        try
        {
            string sql_com = "SELECT * FROM public." + tableName +
condition;
            return getSqlComResult(sql_com);
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error!\n" + ex.Message);
            return null;
        }
    }

    public static void DeleteRows(DataGridView dataGridView,
string table)
    {
        try
        {
            using (NpgsqlCommand command =
_connection.CreateCommand())
            {

```

```

        // SQL шаблон для удаления
        DataTable dt = getDataTable(table, " LIMIT 0");

        string sqlTemplate = $"DELETE FROM {table} WHERE
{dt.Columns[0].ColumnName} = @Id";

        command.CommandText = sqlTemplate;

        NpgsqlParameter idParam = new
NpgsqlParameter("@Id", DbType.Object); // Используем DbType.Object для
общности

        command.Parameters.Add(idParam);

        // Удаление каждой выбранной строки
        foreach (DataGridViewRow row in
dataGridView.SelectedRows)
        {
            object id = row.Cells[0].Value;
            idParam.Value = id ?? DBNull.Value;
            command.ExecuteNonQuery();
        }

        MessageBox.Show("Выбранные записи успешно
удалены.");
    }
}
catch (Exception ex)
{
    MessageBox.Show("Ошибка удаления записей: " +
ex.Message);
}
}

public static void InsertRowsWithID(DataGridView dataGridView,
ViewInfo viewInfo, Dictionary<string, object> arguments)
{
    try
    {

```

```

        DataTable Table_template =
getDataTable(viewInfo.MainTableName, " LIMIT 0");

        string sql_com = $"INSERT INTO
{viewInfo.MainTableName} (";

        string valuesClause = "VALUES (";

        List<string> columnNames = new List<string>();

        foreach (DataColumn el in Table_template.Columns)
        {
            sql_com += (sql_com.EndsWith("(") ? "" : ", ") +
el.ColumnName;

            valuesClause += (valuesClause.EndsWith("(") ? "" :
", ") + "@" + el.ColumnName;

            columnNames.Add(el.ColumnName);
        }
        sql_com += ") " + valuesClause + ")";
        MessageBox.Show(sql_com);
        using (NpgsqlCommand command =
_connection.CreateCommand())
        {
            command.CommandText = sql_com;

            var resultDict = new Dictionary<string, object>();
            foreach (var kvp in arguments)
            {
                string key = kvp.Key;
                object value1 = kvp.Value;

                if (arguments.ContainsKey(key))
                {
                    string value2 =
viewInfo.TableViewCorrelationInfo[key];
                    resultDict[value2] = value1;
                }
            }

            for (int i = 0; i < columnNames.Count; ++i)

```

```

        {
            if (resultDict.Keys.Contains(columnNames[i]))
            {
                object value = resultDict[columnNames[i]];
                command.Parameters.AddWithValue("@ " +
columnNames[i], value ?? DBNull.Value);
            }
            else
            {
                var value =
dataGridView.Rows[0].Cells[i].Value;
                command.Parameters.AddWithValue("@ " +
columnNames[i], value ?? DBNull.Value);
            }
        }

        int res = command.ExecuteNonQuery();
        if (res > -1) MessageBox.Show("Запись успешно
добавлена");
    }
}
catch (Exception ex)
{
    MessageBox.Show("Ошибка добавления записи: " +
ex.Message);
}
}

public static void InsertRows(DataGridView dataGridView,
ViewInfo viewInfo, Dictionary<string, object> arguments)
{
    try
    {
        DataTable Table_template =
getDataTable(viewInfo.MainTableName, " LIMIT 0");
        string sql_com = $"INSERT INTO
{viewInfo.MainTableName} (";
        string valuesClause = "VALUES (";
    }
}

```

```

bool first = true;
List<string> columnNames = new List<string>();

foreach (DataColumn el in Table_template.Columns)
{
    if (first)
    {
        first = false; // Пропускаем первый столбец,
        так как предполагается, что он serial типа
        continue;
    }
    sql_com += (sql_com.EndsWith("(") ? "" : ", ") +
    el.ColumnName;

    valuesClause += (valuesClause.EndsWith("(") ? "" :
    ", ") + "@" + el.ColumnName;
    columnNames.Add(el.ColumnName);
}
sql_com += ") " + valuesClause + ")";

using (NpgsqlCommand command =
_connection.CreateCommand())
{
    command.CommandText = sql_com;

    var resultDict = new Dictionary<string, object>();

    foreach (var kvp in arguments)
    {
        string key = kvp.Key;
        object value1 = kvp.Value;

        if (arguments.ContainsKey(key))
        {
            string value2 =
viewInfo.TableViewCorrelationInfo[key];
            resultDict[value2] = value1;

```



```

        }
    }

    for (int i = 0; i < columnNames.Count; ++i)
    {
        if (resultDict.Keys.Contains(columnNames[i]))
        {
            object value = resultDict[columnNames[i]];
            command.Parameters.AddWithValue("@ " +
columnNames[i], value ?? DBNull.Value);
        }
        else
        {
            var value = dataGridView.Rows[0].Cells[i +
1].Value;

            command.Parameters.AddWithValue("@ " +
columnNames[i], value ?? DBNull.Value);
        }
    }

    int res = command.ExecuteNonQuery();
    if (res > -1) MessageBox.Show("Запись успешно
добавлена");
    }
}

catch (Exception ex)
{
    MessageBox.Show("Ошибка добавления записи: " +
ex.Message);
}
}

public static TKey FindKeyByValue<TKey,
TValue>(Dictionary<TKey, TValue> dictionary, TValue value)
{
    foreach (var pair in dictionary)

```

```

        {
            if
(EqualityComparer<TValue>.Default.Equals(pair.Value, value))
                return pair.Key;
        }
        return default(TKey);
    }

```

```

        public static void UpdateRows(DataGridView dataGridView,
ViewInfo viewInfo, Dictionary<string, object> arguments)

```

```

    {
        try
        {
            DataTable Table_template =
getDataTable(viewInfo.MainTableName, " LIMIT 0");

            string sql_com = $"UPDATE {viewInfo.MainTableName} SET
";

            List<string> columnNames = new List<string>();

            // Собираем имена столбцов для SQL запроса
            foreach (DataColumn el in Table_template.Columns)
            {
                columnNames.Add(el.ColumnName);
            }

            string keyColumn = columnNames[0];

            bool first = true;
            for (int i = 1; i < columnNames.Count; i++)
            {
                if (!first) sql_com += ", ";
                sql_com += columnNames[i] + " = @" +
columnNames[i];

                first = false;
            }

```

```

        sql_com += " WHERE " + keyColumn + " = @" + keyColumn;

        string this_table_name = viewInfo.MainTableName;
        string sql_temp = $"SELECT * FROM {this_table_name}
WHERE
{viewInfo.TableViewCorrelationInfo[dataGridView.Columns[0].Name]} =
{dataGridView.Rows[0].Cells[0].Value}";

        if (this_table_name == "cars")
        {
            sql_temp = $"SELECT * FROM {this_table_name} WHERE
\"{viewInfo.TableViewCorrelationInfo[dataGridView.Columns[0].Name]}\"
= '{dataGridView.Rows[0].Cells[0].Value}'";
        }

        DataTable temp_dt = getSqlComResult(sql_temp);

        bool skip1 = true;
        foreach(DataColumn col in temp_dt.Columns)
        {
            if (skip1) { skip1 = false; continue; }

            if (col.ColumnName.Contains("id"))
            {
                object currentValue;
                string field_name = FindKeyByValue<string,
string>(viewInfo.TableViewCorrelationInfo, col.ColumnName);
                if (arguments.TryGetValue(field_name, out
currentValue))
                {
                    arguments[field_name] = currentValue;
                }
                else
                {
                    arguments.Add(field_name,
temp_dt.Rows[0][col]);
                }
            }
        }
    }

```

```

string debug = "";

foreach(var pair in arguments)
{
    debug += $"{pair.Key} : {pair.Value}\n";
}
MessageBox.Show(debug);

var resultDict = new Dictionary<string, object>();

foreach (var kvp in arguments)
{
    string key = kvp.Key;
    object value1 = kvp.Value;

    if (arguments.ContainsKey(key))
    {
        string value2 =
viewInfo.TableViewCorrelationInfo[key];
        resultDict[value2] = value1;
    }
}

using (NpgsqlCommand command =
_connection.CreateCommand())
{
    command.CommandText = sql_com;

    for (int i = 0; i < columnNames.Count; ++i)
    {

        if (resultDict.Keys.Contains(columnNames[i]))
        {

```

```

        object value = resultDict[columnNames[i]];
        command.Parameters.AddWithValue("@ " +
columnNames[i], value ?? DBNull.Value);
    }
    else
    {
        var value =
dataGridView.Rows[0].Cells[i].Value;
        command.Parameters.AddWithValue("@ " +
columnNames[i], value ?? DBNull.Value);
    }
}
int res = command.ExecuteNonQuery();
if (res > 0) MessageBox.Show("Запись успешно
обновлена");
else MessageBox.Show("Обновление не выполнено.
Проверьте данные.");
}
}
catch (Exception ex)
{
    MessageBox.Show("Ошибка обновления записи: " +
ex.Message);
}
}

```

```

public static void setGridView(DataTable dataTable,
DataGridView dataGridView)
{
    try
    {
        dataGridView.Columns.Clear();

        if (dataTable == null)
            throw new Exception("Ошибка чтения таблицы с
данными.");

        foreach (DataColumn column in dataTable.Columns)
        {

```

```

        column.ReadOnly = false;
    }

    dataGridView.DataSource = dataTable;

    dataGridView.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.Fill;
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error!\n" + ex.Message);
    }
}

public static void setGridViewREAD(DataTable dataTable,
DataGridView dataGridView)
{
    try
    {
        dataGridView.Columns.Clear();

        if (dataTable == null)
            throw new Exception("Ошибка чтения таблицы с
данными.");

        dataGridView.DataSource = dataTable;

        dataGridView.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.Fill;
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error!\n" + ex.Message);
    }
}

public static string getViewByField(string field)

```

```

{
    switch (field) {
        case "_Страна":
            return "Страны";
        case "_Город":
            return "Города";
        case "_Должность":
            return "Должности";
        case "_Статус":
            return "Статусы";
        case "_Модель":
            return "МоделиАвто";
        case "_ОтветственныйСотрудник":
            return "Сотрудники";
        case "_ГосНомерАвтомобиля":
            return "Автомобили";
        case "_ИдентификаторСпискаОтгрузки":
            return "СпискиОтгрузки";
        case "_Клиент":
            return "Клиенты";
        case "_ИдентификаторАкта":
            return "АктыВыполненныхРабот";
        case "_ИдентификаторЕдиницыОтгрузки":
            return "Грузы";
        default:
            return "";
    }
}

```

```

public static string getRetFieldBySrcField(string field)
{
    switch (field)
    {
        case "_Страна":
            return " (Название) ";
    }
}

```

```

        case "_Город":
            return " (Название) ";
        case "_Модель":
            return " (Название) ";
        case "_Должность":
            return " (Название) ";
        case "_Статус":
            return " (Описание) ";
        case "_ОтветственныйСотрудник":
            return " (Имя, Фамилия) ";
        case "_Клиент":
            return " (Имя, Фамилия) ";
        case "_ГосНомерАвтомобиля":
            return " (РегистрационныйНомер) ";
        case "_ИдентификаторЕдиницыОтгрузки":
            return " (ИдентификаторЕдиницыОтгрузки) ";
        case "_ИдентификаторСпискаОтгрузки":
            return " (ИдентификаторСпискаОтгрузки) ";
        case "_ИдентификаторАкта":
            return " (ИдентификаторАкта) ";
        default:
            return "";
    }
}

}

```

Файл “MainForm.cs”

```

namespace Lab_8
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
        }
    }
}

```



```

    }

    private void postgresConnectionButton_Click(object sender,
EventArgs e)
    {
        try
        {
            string host = hostTextBox.Text;
            string port = portTextBox.Text;
            string user = userTextBox.Text;
            string pass = passTextBox.Text;
            string dbName = dbNameTextBox.Text;

            PostgresModule.OpenConnection(host, port, user, pass,
dbName);

            if (PostgresModule.getConnectionStatus() == true)
            {
                connectionStatusIndicator.BackColor = Color.Green;
            }

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

    private void guidesStripMenuButton_Click(object sender,
EventArgs e)
    {
        try
        {
            GuidesForm guidesForm = new GuidesForm();
            guidesForm.Show();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

```

```

e)         private void dataStripMenuData_Click(object sender, EventArgs
{
            try
            {
                DataForm dataForm = new DataForm();
                dataForm.Show();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void creditsStipMenuButton_Click(object sender,
EventArgs e)
        {
            try
            {
                CreditsForm creditsForm = new CreditsForm();
                creditsForm.Show();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void aboutStripMenuButton_Click(object sender,
EventArgs e)
        {
            AboutForm aboutForm = new AboutForm();
            aboutForm.Show();
        }

        private void closeStripMenuButton_Click(object sender,
EventArgs e)
        {
            Close();
        }
    }
}

```

Файл “ViewInfo.cs”

```
using Npgsql;
using System;
using System.Collections.Generic;
using System.Data.Common;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab_8
{
    public class ViewInfo
    {
        public string ViewName { get; set; }
        public string MainTableName { get; set; }
        public Dictionary<string, string> TableViewCorrelationInfo {
get; set; }
        // view_field <-> table_field

        public ViewInfo() { }

        public ViewInfo(string mainTableName , string viewName)
        {
            ViewName = viewName;
            MainTableName = mainTableName;
            TableViewCorrelationInfo = new Dictionary<string,
string>();

            string TABLE_sql_com = $"SELECT * FROM {MainTableName}
LIMIT 0";

            string VIEW_sql_com = $"SELECT * FROM {ViewName} LIMIT
0";

            var tableSchema =
PostgresModule.getSqlComResult(TABLE_sql_com);
```

```

        var viewSchema =
PostgresModule.getSqlComResult(VIEW_sql_com);

        if (tableSchema != null && viewSchema != null)
        {
            for (int i = 0; i < tableSchema.Columns.Count;
++i)
            {
                string tn = tableSchema.Columns[i].ColumnName;
                string vn = viewSchema.Columns[i].ColumnName;
                TableViewCorrelationInfo.Add(vn, tn);
            }
        }

    }

    public string getTableFieldByViewField(string viewName)
    {
        return TableViewCorrelationInfo[viewName];
    }

    public bool ifViewContainsField(string field)
    {
        foreach (var el in TableViewCorrelationInfo.Keys)
        {
            if (el == field) return true;
        }
        return false;
    }

}

}

```

Файл "CRUDForm.cs"

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Linq.Expressions;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Lab_8
{
    public partial class CRUDForm : Form
    {

        List<ViewInfo> viewInfos = new List<ViewInfo>();

        private int currentPage = 1;
        private int totalRows = 0;
        private int rowsPerPage = 10;

        public CRUDForm()
        {
            InitializeComponent();
            setFullAccessMode();

            dataGridView1.Anchor = AnchorStyles.Top |
AnchorStyles.Bottom | AnchorStyles.Left | AnchorStyles.Right;

            viewInfos.Add(new ViewInfo("cars", "Автомобили"));
            viewInfos.Add(new ViewInfo("acts_of_work_performed",
"АктыВыполненныхРабот"));
        }
    }
}
```

```

        viewInfos.Add(new ViewInfo("auto_models", "МоделиАвто"));
        viewInfos.Add(new ViewInfo("cities", "Города"));
        viewInfos.Add(new ViewInfo("clients", "Клиенты"));
        viewInfos.Add(new ViewInfo("countries", "Страны"));
        viewInfos.Add(new ViewInfo("employee_positions",
"Должности"));
        viewInfos.Add(new ViewInfo("employees", "Сотрудники"));
        viewInfos.Add(new ViewInfo("orders", "Заказы"));
        viewInfos.Add(new ViewInfo("shipment_lists",
"СпискиОтгрузки"));
        viewInfos.Add(new ViewInfo("shipment_units", "Грузы"));
        viewInfos.Add(new ViewInfo("statuses", "Статусы"));
    }

```

```

public void setFullAccessMode()
{
    dataGridView1.AllowUserToAddRows = false;
    dataGridView1.AllowUserToDeleteRows = false;
    dataGridView1.ReadOnly = true;
}

```

```

public void setReadMode()
{
    dataGridView1.AllowUserToDeleteRows = false;
    dataGridView1.AllowUserToAddRows = false;
    CreateUpdateButton.Enabled = false;
    DeleteButton.Enabled = false;
    this.Text += " (Read-Only Mode)";
}

```

```

        private void comboBox1_SelectedIndexChanged(object sender,
EventArgs e)
        {

```

```

        PostgresModule.setGridViewREAD(PostgresModule.getDataTable(comboBox1.T
ext), dataGridView1);

```

```

        comboBox2.Text = "";
        currentPage = 1;
        LoadPage();
    }

    private void comboBox2_SelectedIndexChanged(object sender,
EventArgs e)
    {

PostgresModule.setGridViewREAD(PostgresModule.getDataTable(comboBox2.T
ext), dataGridView1);

        CreateUpdateButton.Enabled = false;
        DeleteButton.Enabled = false;
        comboBox1.Text = "";
        currentPage = 1;
        LoadPage();
    }

    private void dataGridView1_KeyUp(object sender, KeyEventArgs
e)
    {

    }

    private void CreateUpdateButton_Click(object sender, EventArgs
e)
    {
        try
        {
            object selected_id =
dataGridView1.SelectedRows[0].Cells[0].Value;

            CreateUpdateForm createUpdateForm = new
CreateUpdateForm(PostgresModule.getDataTable(comboBox1.Text),
comboBox1.Text, selected_id, viewInfos);

            createUpdateForm.ShowDialog();
        }

        catch (ArgumentOutOfRangeException ex)

```

```

        {
            CreateUpdateForm createUpdateForm = new
CreateUpdateForm(PostgresModule.getDataTable(comboBox1.Text),
comboBox1.Text, null, viewInfos);

            createUpdateForm.ShowDialog();

        }

        catch (Exception ex)

        {

            MessageBox.Show(ex.Message + "\n" +
ex.GetType().ToString());

        }

    }

    private void DeleteButton_Click(object sender, EventArgs e)

    {

        try

        {

            ViewInfo viewInfo = viewInfos.Find(view =>
view.ViewName == comboBox1.Text);

            PostgresModule.DeleteRows(dataGridView1,
viewInfo.MainTableName);

            PostgresModule.setGridView(PostgresModule.getDataTable(comboBox1.Text)
, dataGridView1);

        }

        catch (Exception ex)

        {

            MessageBox.Show(ex.Message);

        }

    }

    private void LoadPage()

    {

        try

        {

```



```

        int startIndex = (currentPage - 1) * rowsPerPage;

        var dataTable =
PostgresModule.getDataTable(comboBox1.Text);

        totalRows = dataTable.Rows.Count;

        var pageTable = dataTable.AsEnumerable()
                                .Skip(startIndex)
                                .Take(rowsPerPage)
                                .CopyToDataTable();

        dataGridView1.DataSource = pageTable;

        UpdatePaginationButtons();

dataGridView1.AutoSizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);

        AdjustFormSizeToGrid();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

private void AdjustFormSizeToGrid()
{
    foreach (DataGridViewColumn column in
dataGridView1.Columns)
    {
        column.MinimumWidth = 100;
    }

    int width = dataGridView1.PreferredSize.Width;
    int height = dataGridView1.PreferredSize.Height;

```

```

        this.MinimumSize = new Size(Math.Max(width,
this.MinimumSize.Width), Math.Max(height, this.MinimumSize.Height));

        this.Size = this.MinimumSize;
    }

    private void UpdatePaginationButtons()
    {
        prev_page_button.Enabled = currentPage > 1;
        next_page_button.Enabled = (currentPage * rowsPerPage) <
totalRows;
    }

    private void next_page_button_Click(object sender, EventArgs
e)
    {
        currentPage++;
        LoadPage();
    }

    private void prev_page_button_Click(object sender, EventArgs
e)
    {
        if (currentPage > 1)
        {
            currentPage--;
            LoadPage();
        }
    }
}
}

```

Файл “CreateUpdateForm.cs”

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics.Eventing.Reader;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Web;
using System.Windows.Forms;
using static Npgsql.Replication.PgOutput.Messages.RelationMessage;

namespace Lab_8
{
    public partial class CreateUpdateForm : Form
    {
        List<ViewInfo> viewInfos = new List<ViewInfo>();

        DataTable dataTable1 = new DataTable();
        string SourceView = "";
        bool update = false;
        Dictionary<string, object> arguments = new Dictionary<string,
object>();

        public CreateUpdateForm(DataTable dataTable, string TableName,
object? selected_id, List<ViewInfo> viewInfos)
        {

            InitializeComponent();

            SourceView = TableName;
```

```

        this.viewInfos = viewInfos;

        dataGridView1.EditMode = DataGridViewEditMode.EditOnEnter;
        dataGridView1.ReadOnly = false;
        dataGridView1.Columns.Clear();

        dataGridView1.AllowUserToAddRows = true;

        string condition = " LIMIT 0";
        if (selected_id != null)
        {
            update = true;
            condition = $" WHERE (
{dataTable.Columns[0].ColumnName} = {selected_id} )";
        }

PostgresModule.setGridView(PostgresModule.getDataTable(SourceView,
condition), dataGridView1);

        dataGridView1.Rows[0].Cells[0].ReadOnly = true;

    }

    private void AdjustFormSizeToGrid()
    {
        foreach (DataGridViewColumn column in
dataGridView1.Columns)
        {
            column.MinimumWidth = 100;
        }

        int width = dataGridView1.PreferredSize.Width;
        int height = dataGridView1.PreferredSize.Height;

        this.MinimumSize = new Size(Math.Max(width,
this.MinimumSize.Width), Math.Max(height, 200));
    }

```

```

        this.Size = this.MinimumSize;
    }

    private void button1_Click(object sender, EventArgs e)
    {
        //отменить
        Close();
        Dispose();
    }

    private void button2_Click(object sender, EventArgs e)
    {
        //сохранить

        try
        {
            ViewInfo viewInfo = viewInfos.Find(view =>
view.ViewName == SourceView);

            if (update)
            {
                PostgresModule.UpdateRows(dataGridView1, viewInfo,
arguments);
            }
            else
            {
                if (viewInfo.MainTableName == "cars")
                {
                    PostgresModule.InsertRowsWithID(dataGridView1,
viewInfo, arguments);
                }
                else
                {

```

```

        PostgresModule.InsertRows(dataGridView1,
viewInfo, arguments);
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message + "\nОшибка записи");
}

Close();
Dispose();
}

```

```

private void dataGridView1_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0 && e.ColumnIndex >= 0)
    {
        if (dataGridView1.Columns[e.ColumnIndex].Name[0] ==
'_' )
        {
            ChooseForm chooseForm = new
ChooseForm(dataGridView1.Columns[e.ColumnIndex].Name, e);
            chooseForm.OnRowSelected +=
ChooseForm_OnRowSelected;
            chooseForm.ShowDialog();
        }
    }
}

```

```

private void ChooseForm_OnRowSelected(object chosen_field_id,
string src_field, object ret_field_val, DataGridViewCellEventArgs e)
{

```

```

        dataGridView1.Rows[e.RowIndex].Cells[e.ColumnIndex].Value
= ret_field_val;
        arguments.Add(src_field, chosen_field_id);

dataGridView1.Rows[e.RowIndex].Cells[e.ColumnIndex].ReadOnly = true;
    }

}
}

```

Файл “ChooseForm.cs”

```

using Npgsql;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Common;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Runtime.InteropServices.JavaScript.JSType;

namespace Lab_8
{
    public partial class ChooseForm : Form
    {
        public delegate void SelectedRowHandler(object chosen_field,
string src_field, object ret_field_val, DataGridViewCellEventArgs e);
        public event SelectedRowHandler OnRowSelected;
        public DataGridViewCellEventArgs src_e;
        private string src_field;

        public ChooseForm(string field_name, DataGridViewCellEventArgs
src_e)

```

```

    {
        InitializeComponent();
        this.src_e = src_e;
        src_field = field_name;

PostgresModule.setGridViewREAD(PostgresModule.getDataTable(PostgresModule.getViewByField(field_name)), dataGridView1);

        dataGridView1.AllowUserToAddRows = false;
        dataGridView1.ReadOnly = true;
        dataGridView1.AllowUserToDeleteRows = false;

    }

private void button2_Click(object sender, EventArgs e)
{
    Close();
    Dispose();
}

private void button1_Click(object sender, EventArgs e)
{
    try
    {
        if (dataGridView1.SelectedRows.Count > 0)
        {
            DataGridViewRow row =
dataGridView1.SelectedRows[0];

            object chosen_field_id = row.Cells[0].Value;

            string ret_field =
PostgresModule.getRetFieldBySrcField(src_field);

```



```

        string sql_com = $"SELECT {ret_field} FROM
{PostgresModule.getViewByField(src_field)}" +

        $" WHERE
({PostgresModule.getDataTable(PostgresModule.getViewByField(src_field)
).Columns[0].ColumnName} = {chosen_field_id})";

        if (src_field == "_ГосНомерАвтомобиля")
        {
            sql_com = $"SELECT {ret_field} FROM
{PostgresModule.getViewByField(src_field)}" +

            $" WHERE
(\"{PostgresModule.getDataTable(PostgresModule.getViewByField(src_fiel
d)).Columns[0].ColumnName}\" = '{chosen_field_id}')";
        }

        DataTable valueDt =
PostgresModule.getSqlComResult(sql_com);

        object ret_field_val = valueDt.Rows[0][0];
        if (ret_field.Contains(','))
        {
            object[] array = valueDt.Rows[0][0] as
object[];

            ret_field_val = string.Join(" ",
array.Select(x => x.ToString()));
        }

        OnRowSelected?.Invoke(chosen_field_id, src_field,
ret_field_val, src_e);

        Close();

        Dispose();
    }

}

catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}

```

```
    }  
}  
}
```

Файл “AboutForm.cs”

```
using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Drawing;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.Windows.Forms;  
  
namespace Lab_8  
{  
    public partial class AboutForm : Form  
    {  
        public AboutForm()  
        {  
            InitializeComponent();  
        }  
    }  
}
```