**Оглавление**

[Работа №1 2](#_Toc199434043)

[Работа № 2 17](#_Toc199434044)

[Работа № 3 32](#_Toc199434045)

[Работа № 4 48](#_Toc199434046)

[Работа № 5 68](#_Toc199434047)

[Работа № 6 84](#_Toc199434048)

[Работа № 7 102](#_Toc199434049)

[Работа № 8 119](#_Toc199434050)

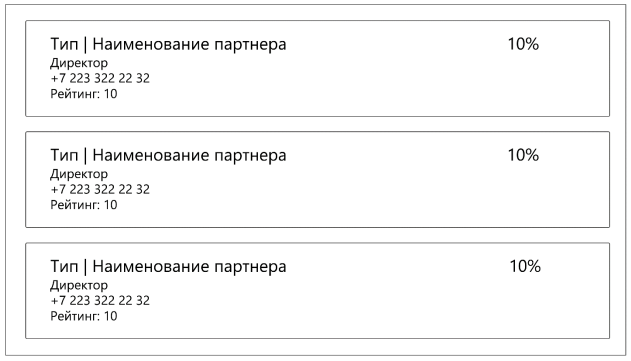
[Работа № 9 135](#_Toc199434051)

[Работа № 10 151](#_Toc199434052)

Работа №1

Задание: Разработать подсистему для работы с партнерами компании, обеспечивающую следующий функционал: – просмотр списка партнеров; – добавление/редактирование данных о партнере; – просмотр истории реализации продукции партнером. Получить ER-диаграмму средствами СУБД: ER-диаграмма должна быть представлена в формате PDF и содержать таблицы, связи между ними, атрибуты и ключи (типами данных на данном этапе можно пренебречь).

Разработать программный модуль для учета партнеров. Необходимо реализовать вывод списка партнеров, информация о которых хранятся в базе данных, согласно предоставленному макету:



Величина скидки для партнера рассчитывается на основании продажи продукции за весь период работы. Скидка зависит от общего количества реализованной партнером продукции и составляет: до 10000 – 0%, от 10000 – до 50000 – 5%, от 50000 – до 300000 – 10%, более 300000 – 15%. Созданную базу данных подключить к приложению работы с партнерами, реализующему необходимый функционал. Список партнеров на главной форме должен отображать информацию из базы данных.

Реализовать функции добавления/редактирования данных партнера в новом окне (странице) – форме для добавления/редактирования партнера. Переходы на эту форму должны быть реализованы из главной формы списка партнеров: для редактирования – при нажатии на конкретный элемент, для добавления – при нажатии кнопки

Необходимо осуществить вывод истории реализации продукции партнером с указанием наименования продукции, количества и даты продажи.

Разработать метод, который должен рассчитывать целое количество материала, необходимого для производства указанного количества продукции, учитывая возможный брак материала.

**Ход работы**

Подготовим данные для импорта. Для этого приведём все данные из предоставленных данных в 3НФ (Рисунок 1).

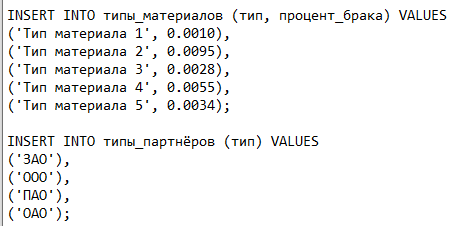


Рисунок 1 – Подготовка данных для импорта

Теперь необходимо создать базу данных (Рисунок 2)

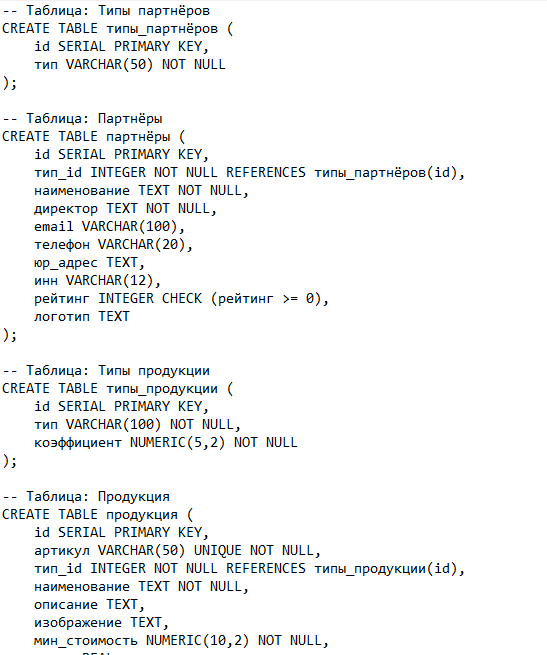


Рисунок 2 – Скрипт для создания БД

Теперь сделаем импорт данных (Рисунок 3)

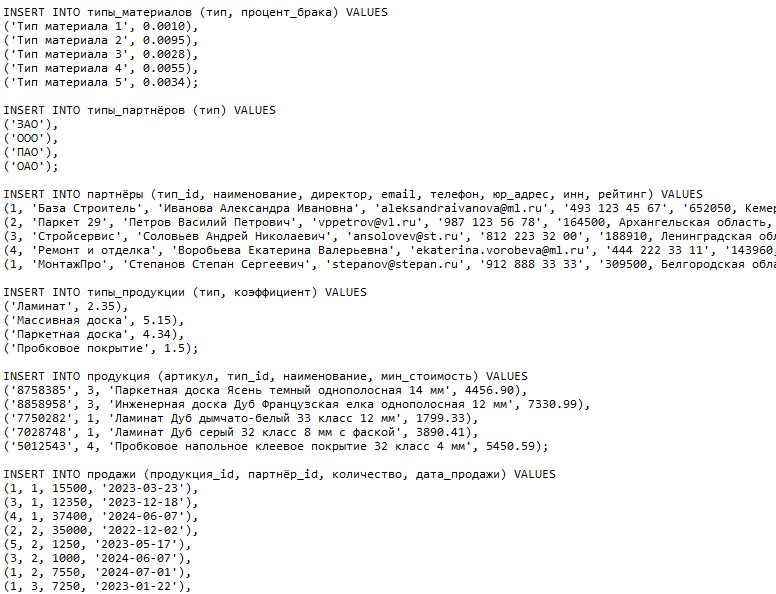


Рисунок 3 – Импорт данных в таблицу

Листинг кода формы авторизации:

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;

usingNpgsql; // ПодключениебиблиотекидляработысPostgreSQL

namespace YCHEBKA1

{

public partial class Form1 : Form

{

// Объявление объекта подключения к базе данных

private NpgsqlConnection connection;

// Строка подключения к базе данных PostgreSQL

private string connectionString = "Host=localhost;Username=postgres;Password=1111;Database=master\_pol";

publicForm1()

{

InitializeComponent(); // Инициализациякомпонентовформы

// Создание нового подключения (локальная переменная, не используется позже — можно исправить)

NpgsqlConnection connection = new NpgsqlConnection(connectionString);

connection.Open(); // Открытие соединения с базой данных

// Здесь нужно закрыть соединение или использовать его как глобальное, иначе может возникнуть утечка ресурсов

}

// Методдляаутентификациипользователя

private bool AuthenticateUser(string username, string password)

{

// Повторное объявление строки подключения (можно использовать поле класса вместо этого)

string connectionString = "Host=localhost;Username=postgres;Password=1111;Database=master\_pol";

// Создание подключения к базе данных в блоке using (автоматическое освобождение ресурсов)

using (NpgsqlConnection connection = new NpgsqlConnection(connectionString))

{

connection.Open(); // Открытие соединения

// SQL-запрос для проверки наличия пользователя с заданным логином и паролем

string query = "SELECT COUNT(\*) FROM db\_users WHERE user\_login = @Username AND user\_password = @Password";

using (NpgsqlCommand command = new NpgsqlCommand(query, connection))

{

// Параметры SQL-запроса, защищают от SQL-инъекций

command.Parameters.AddWithValue("@Username", username);

command.Parameters.AddWithValue("@Password", password);

// Получениеколичестванайденныхзаписей

int count = Convert.ToInt32(command.ExecuteScalar());

// Возвращаем true, если найден хотя бы один пользователь

return count > 0;

}

}

}

// Обработчик нажатия кнопки входа

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

{

// Получение логина и пароля из текстовых полей формы

string username = textBox1.Text;

string password = textBox2.Text;

// Вызовметодааутентификации

if (AuthenticateUser(username, password))

{

// Успешная аутентификация

MessageBox.Show("Успешная аутентификация! Добро пожаловать, " + username + "!");

Form f2 = new Form2(); // Создание второй формы

this.Hide(); // Скрытие текущей формы

f2.Show(); // Отображение второй формы

}

else

{

// Ошибка аутентификации

MessageBox.Show("Ошибка аутентификации. Пользователь с такими данными не найден.");

}

}

}

}

На рисунке4показана работа приведённого выше кода:

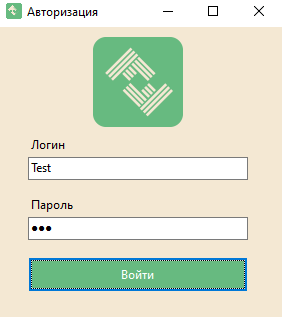


Рисунок 4 – Работа формы авторизации

Ниже листинг кода основной формы:

using Npgsql;

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Windows.Forms;

namespace YCHEBKA1

{

public partial class Form2 : Form

{

// Подключениекбазеданных PostgreSQL

private NpgsqlConnection conn;

// Адаптердляработыстаблицей

private NpgsqlDataAdapter adapter;

// Таблица данных для отображения в DataGridView

private DataTable dt;

// Словарь справочников: имя таблицы -> (ID, имя)

private Dictionary<string, Dictionary<int, string>> lookupData = new Dictionary<string, Dictionary<int, string>>();

// Словарь первичных ключей: имя таблицы -> имя колонки

private Dictionary<string, string> primaryKeys = new Dictionary<string, string>();

// Словарь внешних ключей: имя таблицы -> список (имя внешней колонки, имя связанной таблицы)

private Dictionary<string, List<Tuple<string, string>>> foreignKeys = new Dictionary<string, List<Tuple<string, string>>>();

public Form2()

{

InitializeComponent();

// Обработказакрытияформы

this.FormClosing += new FormClosingEventHandler(FormClosingevent);

// Строкаподключениякбазеданных

string connection = "Host=localhost;Username=postgres;Password=18273645;Database=master\_pol";

conn = new NpgsqlConnection(connection);

// Загрузкавсехтаблициключей

LoadSchema();

// Событие при выборе элемента в comboBox

comboBox1.SelectedIndexChanged += comboBox1\_SelectedIndexChanged;

}

// Метод для загрузки списка таблиц (кроме db\_users)

privatevoidLoadSchema()

{

conn.Open();

// Получениеимёнтаблиц

using (var cmd = new NpgsqlCommand(

"SELECT table\_name FROM information\_schema.tables WHERE table\_schema = 'public' AND table\_type='BASE TABLE' AND table\_name <> 'db\_users'", conn))

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

comboBox1.Items.Add(reader.GetString(0));

}

}

conn.Close();

// Загрузкаключейтаблиц

LoadPrimaryKeys();

LoadForeignKeys();

}

// Загрузкапервичныхключейтаблиц

private void LoadPrimaryKeys()

{

string query = @"

SELECT

tc.table\_name,

kcu.column\_name

FROM

information\_schema.table\_constraints tc

JOIN information\_schema.key\_column\_usage kcu

ON tc.constraint\_name = kcu.constraint\_name

WHERE

constraint\_type = 'PRIMARY KEY' AND

tc.table\_schema = 'public';";

conn.Open();

using (var cmd = new NpgsqlCommand(query, conn))

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

primaryKeys[reader.GetString(0)] = reader.GetString(1);

}

}

conn.Close();

}

// Загрузка внешних ключей таблиц

private void LoadForeignKeys()

{

string query = @"

SELECT

tc.table\_name,

kcu.column\_name,

ccu.table\_name AS foreign\_table\_name

FROM

information\_schema.table\_constraints AS tc

JOIN information\_schema.key\_column\_usage AS kcu

ON tc.constraint\_name = kcu.constraint\_name

JOIN information\_schema.constraint\_column\_usage AS ccu

ON ccu.constraint\_name = tc.constraint\_name

WHERE

constraint\_type = 'FOREIGN KEY' AND

tc.table\_schema = 'public';";

conn.Open();

using (var cmd = new NpgsqlCommand(query, conn))

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

string table = reader.GetString(0);

string column = reader.GetString(1);

string refTable = reader.GetString(2);

if (!foreignKeys.ContainsKey(table))

foreignKeys[table] = new List<Tuple<string, string>>();

foreignKeys[table].Add(new Tuple<string, string>(column, refTable));

}

}

conn.Close();

}

// Обработчик выбора таблицы из выпадающего списка

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

string table = comboBox1.Text;

string query = $"SELECT \* FROM \"{table}\"";

// Получениеданныхизвыбраннойтаблицы

adapter = new NpgsqlDataAdapter(query, conn);

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

dt = new DataTable();

adapter.Fill(dt);

// Очистка и установка источника данных для DataGridView

dataGridView1.Columns.Clear();

dataGridView1.DataSource = dt;

// Скрытиеколонкипервичногоключа

if (primaryKeys.TryGetValue(table, out string pk) && dataGridView1.Columns.Contains(pk))

{

dataGridView1.Columns[pk].Visible = false;

}

// Загрузка справочников и отображение внешних ключей как выпадающих списков

LoadLookupData(table);

ReplaceForeignKeysWithCombos(table);

// Установка обработчика для значений по умолчанию

dataGridView1.DefaultValuesNeeded -= dataGridView1\_DefaultValuesNeeded;

dataGridView1.DefaultValuesNeeded += dataGridView1\_DefaultValuesNeeded;

}

// Загрузка данных справочников для внешних ключей

private void LoadLookupData(string currentTable)

{

if (!foreignKeys.ContainsKey(currentTable)) return;

foreach (var tuple in foreignKeys[currentTable])

{

string column = tuple.Item1;

string refTable = tuple.Item2;

// Если данные справочника ещё не загружены

if (!lookupData.ContainsKey(refTable))

{

string nameColumn = GetFirstTextColumn(refTable); // Перваястроковаяколонка

if (nameColumn == null) continue;

var dict = new Dictionary<int, string>();

string query = $"SELECT id, \"{nameColumn}\" FROM \"{refTable}\"";

using (var cmd = new NpgsqlCommand(query, conn))

{

conn.Open();

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

int id = reader.GetInt32(0);

string value = reader.IsDBNull(1) ? string.Empty : reader.GetString(1);

dict[id] = value;

}

}

conn.Close();

}

lookupData[refTable] = dict;

}

}

}

// Получение первой текстовой колонки таблицы (кроме id)

private string GetFirstTextColumn(string table)

{

string query = $@"

SELECT column\_name

FROM information\_schema.columns

WHERE table\_schema = 'public'

AND table\_name = '{table}'

AND data\_type IN ('character varying', 'text')

AND column\_name <> 'id'

LIMIT 1;";

using (var cmd = new NpgsqlCommand(query, conn))

{

conn.Open();

var result = cmd.ExecuteScalar() as string;

conn.Close();

return result;

}

}

// Замена колонок с внешними ключами на выпадающие списки

private void ReplaceForeignKeysWithCombos(string table)

{

if (!foreignKeys.ContainsKey(table)) return;

foreach (var (column, refTable) in foreignKeys[table])

{

if (!dt.Columns.Contains(column)) continue;

var combo = new DataGridViewComboBoxColumn

{

Name = column,

DataPropertyName = column,

HeaderText = column.Replace("\_id", ""),

DataSource = new BindingSource(lookupData[refTable], null),

DisplayMember = "Value",

ValueMember = "Key",

FlatStyle = FlatStyle.Flat

};

// Заменяем оригинальную колонку на выпадающий список

int index = dataGridView1.Columns[column].Index;

dataGridView1.Columns.Remove(column);

dataGridView1.Columns.Insert(index, combo);

}

}

// Установка значений по умолчанию при добавлении новой строки

private void dataGridView1\_DefaultValuesNeeded(object sender, DataGridViewRowEventArgs e)

{

string table = comboBox1.Text;

if (!foreignKeys.ContainsKey(table)) return;

foreach (var (column, refTable) in foreignKeys[table])

{

if (lookupData.ContainsKey(refTable) && lookupData[refTable].Count > 0)

{

e.Row.Cells[column].Value = lookupData[refTable].First().Key;

}

}

}

// Сохранениеизменений (кнопка "Сохранить")

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

{

try

{

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

adapter.UpdateCommand = builder.GetUpdateCommand();

adapter.InsertCommand = builder.GetInsertCommand();

adapter.DeleteCommand = builder.GetDeleteCommand();

adapter.Update(dt);

MessageBox.Show("Изменениясохранены.");

}

catch (Exception ex)

{

MessageBox.Show("Ошибкаприсохранении: " + ex.Message);

}

}

// Удалениестроки (кнопка "Удалить")

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

{

if (dataGridView1.CurrentRow != null)

{

try

{

dataGridView1.Rows.RemoveAt(dataGridView1.CurrentRow.Index);

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

adapter.Update(dt);

MessageBox.Show("Записьудалена.");

}

catch (Exception ex)

{

MessageBox.Show("Ошибкаприудалении: " + ex.Message);

}

}

else

{

MessageBox.Show("Выберите строку для удаления.");

}

}

// Завершение приложения при закрытии формы

private void FormClosingevent(object sender, FormClosingEventArgs e)

{

Application.Exit();

}

// Переход к другой форме (кнопка "Назад")

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

{

this.Hide();

Form3 form3 = new Form3();

form3.Show();

}

}

}

Нарисунках5,6результатработыданногокода:

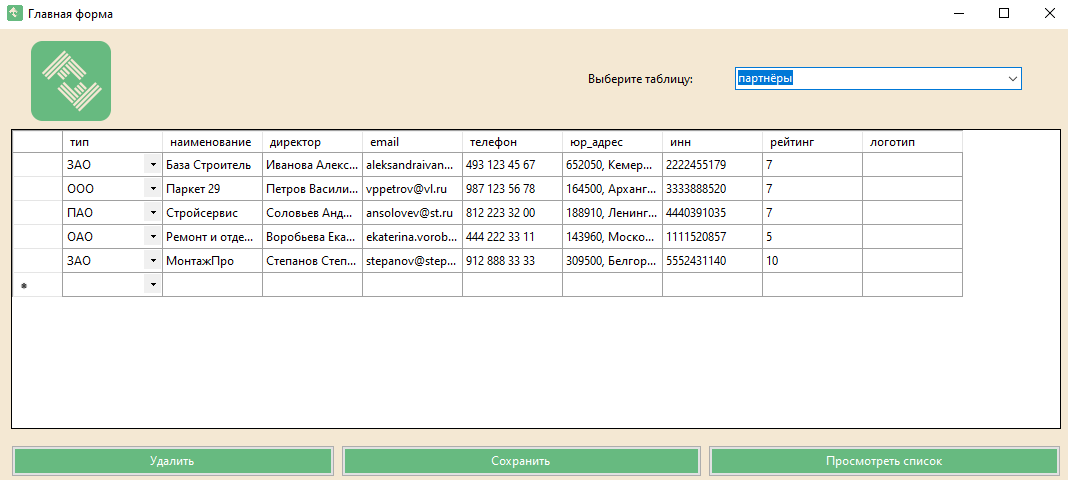


Рисунок 5 – Результат работы приложения

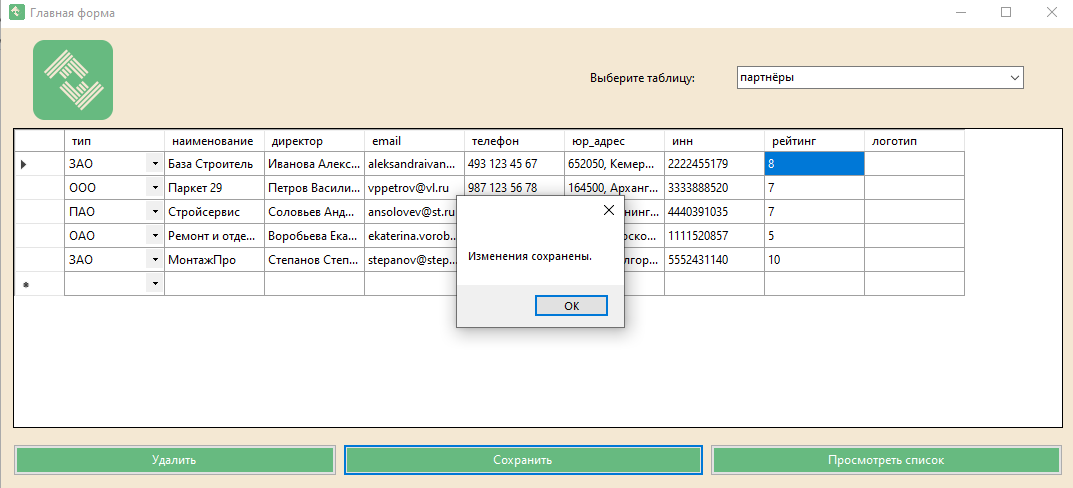


Рисунок 6 – Результат работы приложения

Листинг кода для списка:

using System;

using System.Data;

using System.Text;

using System.Windows.Forms;

using Npgsql;

using YCHEBKA1;

namespace YCHEBKA1

{

public partial class Form3 : Form

{

// Строка подключения к базе данных PostgreSQL

private const string CONNECTION\_STRING = "Host=localhost;Username=postgres;Password=18273645;Database=master\_pol";

public Form3()

{

InitializeComponent(); // Инициализациякомпонентовформы

LoadPartnerData(); // Загрузка и отображение информации о партнёрах

this.FormClosing += new FormClosingEventHandler(FormClosingevent); // Обработкасобытиязакрытияформы

}

// Метод для загрузки и отображения информации о партнёрах

private void LoadPartnerData()

{

// Создание и открытие соединения с базой данных

using (var conn = new NpgsqlConnection(CONNECTION\_STRING))

{

conn.Open();

// SQL-запрос для получения информации о партнёрах, их типах, директоре, телефоне, рейтинге и общей сумме продаж

string query = @"

SELECT

п.id,

тп.тип AS тип\_партнёра,

п.наименование,

п.директор,

п.телефон,

п.рейтинг,

COALESCE(SUM(пр.количество \* прд.мин\_стоимость), 0) AS общая\_сумма

FROM партнёры п

JOIN типы\_партнёров тп ON п.тип\_id = тп.id

LEFT JOIN продажи пр ON п.id = пр.партнёр\_id

LEFT JOIN продукция прд ON пр.продукция\_id = прд.id

GROUP BY п.id, тп.тип, п.наименование, п.директор, п.телефон, п.рейтинг

ORDER BY общая\_сумма DESC;

";

// Выполнение запроса и чтение результатов

using (var cmd = new NpgsqlCommand(query, conn))

using (var reader = cmd.ExecuteReader())

{

var sb = new StringBuilder(); // Используется для формирования текста для richTextBox

while (reader.Read())

{

// Чтение значений из результата запроса

string тип = reader["тип\_партнёра"].ToString();

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

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

string телефон = reader["телефон"] != DBNull.Value ? reader["телефон"].ToString() : "Не указан";

string рейтинг = reader["рейтинг"] != DBNull.Value ? reader["рейтинг"].ToString() : "Нет данных";

// Получение общей суммы покупок, либо 0, если значение NULL

decimal общаяСумма = reader["общая\_сумма"] != DBNull.Value ? Convert.ToDecimal(reader["общая\_сумма"]) : 0;

// Расчёт скидки в зависимости от суммы

int скидка = 0;

if (общаяСумма >= 300000)

скидка = 15;

else if (общаяСумма>= 50000)

скидка = 10;

else if (общаяСумма>= 10000)

скидка = 5;

// Формирование строки информации о партнёре

sb.AppendLine($"{тип} | {наименование} {скидка}%");

sb.AppendLine($"Директор: {директор}");

sb.AppendLine($"Телефон: {телефон}");

sb.AppendLine($"Рейтинг: {рейтинг}");

sb.AppendLine(new string(' ', 40)); // Добавление пустой строки для разделения блоков

}

// Отображение текста в richTextBox

richTextBox1.Text = sb.ToString();

}

}

}

// Обработка кнопки "Назад" — переход обратно на форму Form2

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

{

this.Hide(); // Скрытиетекущейформы

Form2 form2 = new Form2(); // Созданиеновойформы Form2

form2.Show(); // Показ новой формы

}

// Метод для обработки события закрытия формы

private void FormClosingevent(object sender, FormClosingEventArgs e)

{

Application.Exit(); // Завершение работы приложения

}

}

}

На рисунке 7работа представленного выше кода:

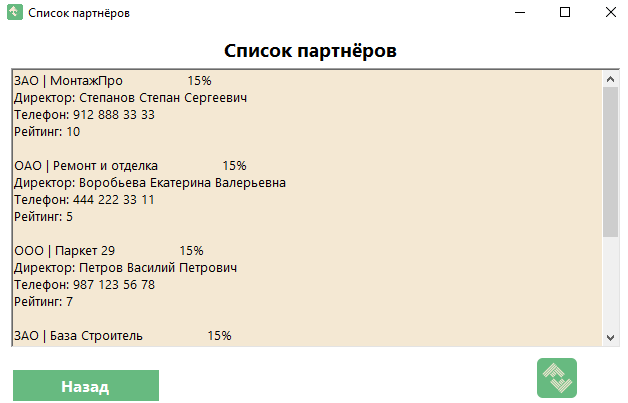


Рисунок 7 – Показ списка

Работа № 2

Задание:Разработать подсистему для работы с читателямибиблиотеки, обеспечивающую следующий функционал:

– оформление выдачи/возврата книг;

– добавление/редактирование данных о читателе;

– просмотр данных о читателях, не вернувших книги вовремя.

Получить ER-диаграмму средствами СУБД: ER-диаграмма должна быть представлена в формате PDF и содержать таблицы, связи между ними, атрибуты и ключи (типами данных на данном этапе можно пренебречь).

Разработать программный модуль для учета читателей. Необходимо реализовать вывод списка задолжников, информация о которых хранятся в базе данных, согласно предоставленному макету:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | ФИО читателя/телефон  Автор/Название книги/количество/дата получения/количество дней задолженности  Автор/Название книги/количество/дата получения/количество дней задолженности  Автор/Название книги/количество/дата получения/количество дней задолженности |  |
|  |  |  |
|  | ФИО читателя/телефон  Автор/Название книги/количество/дата получения/количество дней задолженности  Автор/Название книги/количество/дата получения/количество дней задолженности  Автор/Название книги/количество/дата получения/количество дней задолженности |  |
|  |  |  |
|  | ФИО читателя/телефон  Автор/Название книги/количество/дата получения/количество дней задолженности  Автор/Название книги/количество/дата получения/количество дней задолженности  Автор/Название книги/количество/дата получения/количество дней задолженности |  |
|  |  |  |

Реализовать функции добавления/редактирования данных читателя в новом окне (странице) – форме для добавления/редактирования читателя. Переходы на эту форму должны быть реализованы из формы списка читателей: для редактирования – при нажатии на конкретный элемент, для добавления – при нажатии кнопки.

На форме для добавления/редактирования читателя должны быть предусмотрены следующие поля: ФИО читателя, дату рождения, место работы/учебы, телефон, адреси email.

**Ход работы**

Подготовим данные для импорта. Для этого приведём все данные в предоставленных для импорта таблицах к 3НФ (Рисунок 8).

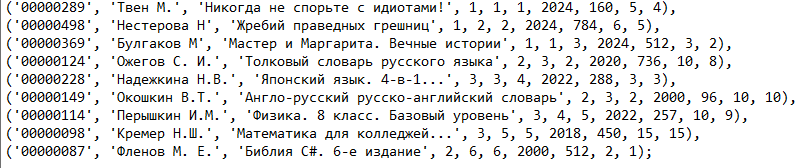


Рисунок 8 – Подготовка данных для импорта

Теперь необходимо создать базу данных (Рисунок 9)

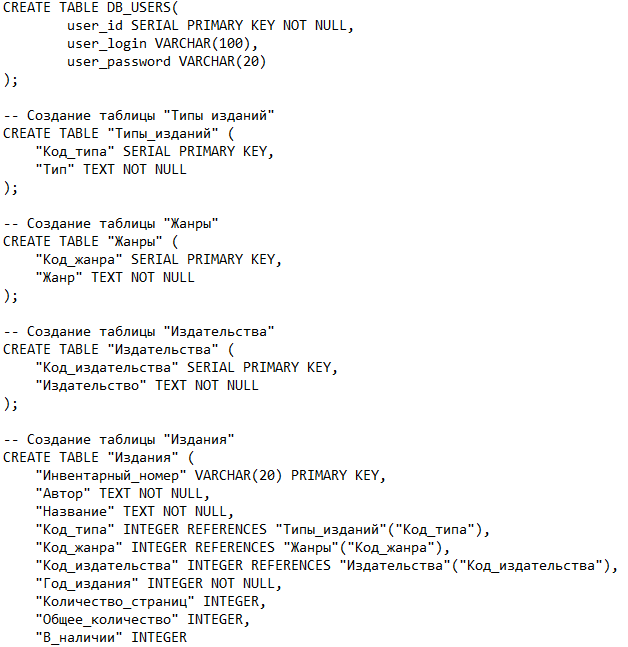


Рисунок 9 – Скрипт для создания БД

Теперь сделаем импорт данных (Рисунок 10)

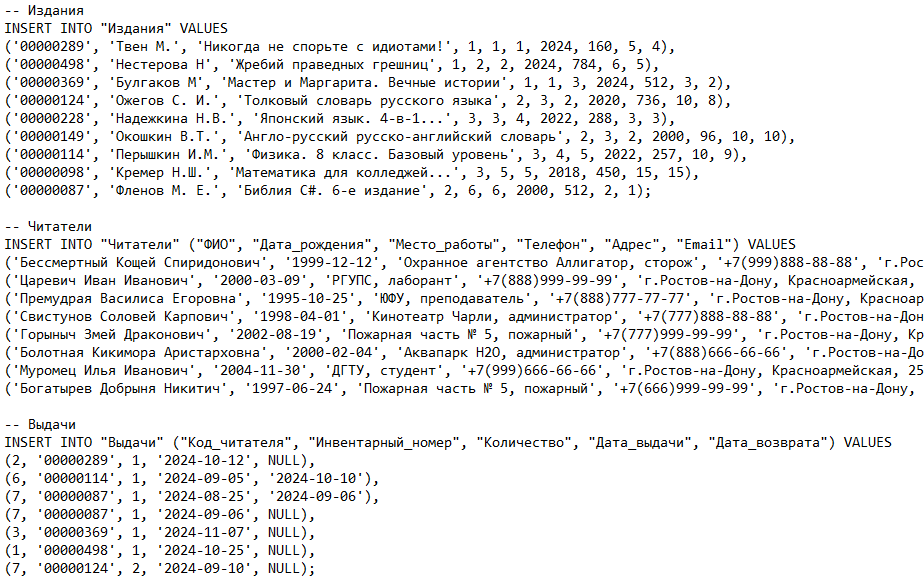


Рисунок 10 – Импорт данных в БД

Листинг кода формы авторизации:

// Подключение необходимых пространств имён

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;

// Подключение библиотеки для работы с PostgreSQL

using Npgsql;

namespace YCHEBKA2

{

// Основной класс формы авторизации

public partial class Form1 : Form

{

// Объектсоединениясбазойданных

private NpgsqlConnection connection;

// Строкаподключениякбазеданных PostgreSQL

private string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=BIBLIOTEKA";

// Конструктор формы

public Form1()

{

InitializeComponent(); // Инициализация компонентов формы

// Создание и открытие подключения к базе данных при запуске формы

NpgsqlConnection connection = new NpgsqlConnection(connectionString);

connection.Open(); // ВАЖНО: Это локальная переменная, которая перекрывает поле. Лучше использовать this.connection

}

// Обработчик события нажатия на кнопку "Войти"

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

{

// Получение введённых пользователем логина и пароля

string username = textBox1.Text;

string password = textBox2.Text;

// Проверка данных пользователя

if (AuthenticateUser(username, password))

{

// Если пользователь найден — сообщение об успехе и переход к следующей форме

MessageBox.Show("Успешная аутентификация! Добро пожаловать, " + username + "!");

Form f2 = new Form2(); // Создание экземпляра следующей формы

this.Hide(); // Скрытие текущей формы

f2.Show(); // Отображение новой формы

}

else

{

// Ошибка аутентификации — вывод сообщения

MessageBox.Show("Ошибка аутентификации. Пользователь с такими данными не найден.");

}

}

// Метод для проверки существования пользователя в базе данных

private bool AuthenticateUser(string username, string password)

{

// Повторное объявление строки подключения (можно убрать и использовать поле класса)

string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=BIBLIOTEKA";

// Создание и открытие подключения с использованием using (гарантирует автоматическое закрытие)

using (NpgsqlConnection connection = new NpgsqlConnection(connectionString))

{

connection.Open();

// SQL-запрос с параметрами для безопасного поиска пользователя

string query = "SELECT COUNT(\*) FROM db\_users WHERE user\_login = @Username AND user\_password = @Password";

using (NpgsqlCommand command = new NpgsqlCommand(query, connection))

{

// Передача параметров в запрос

command.Parameters.AddWithValue("@Username", username);

command.Parameters.AddWithValue("@Password", password);

// Выполнение запроса и получение количества совпадений

int count = Convert.ToInt32(command.ExecuteScalar());

// Если есть хотя бы одно совпадение — возвращаем true

return count > 0;

}

}

}

}

}

На рисунке 11 работа приведённого выше кода:

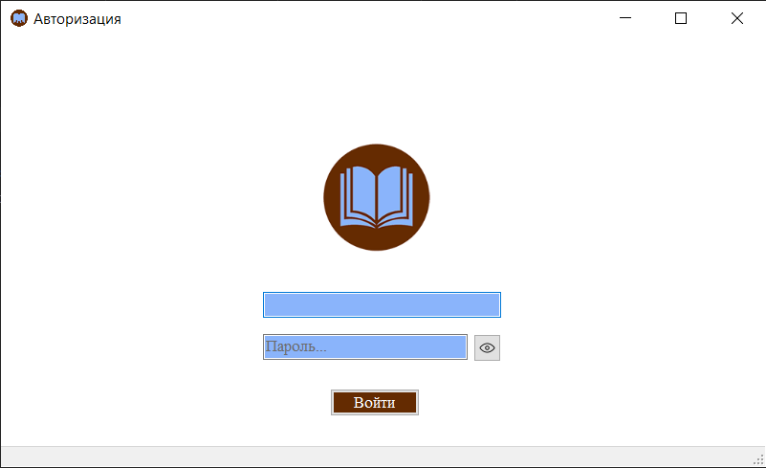


Рисунок 11 – Работа формы авторизации

Ниже листинг главной формы:

// Подключение необходимых пространств имён

using Npgsql;

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 YCHEBKA2

{

// Форма, отображающая данные и позволяющая управлять ими

public partial class Form2 : Form

{

// Подключениекбазеданных PostgreSQL

NpgsqlConnection conn;

// Адаптер данных для связи с БД

NpgsqlDataAdapter adapter;

// Таблица, в которую загружаются данные

DataTable dt;

// Словарь подстановок для отображения значений внешних ключей

Dictionary<string, Dictionary<object, string>> lookupData = new Dictionary<string, Dictionary<object, string>>();

// Конструкторформы

public Form2()

{

InitializeComponent();

// Обработка события закрытия формы (завершает приложение)

this.FormClosing += new FormClosingEventHandler(FormClosingevent);

// СтрокаподключениякБД

string connection = "Host=localhost;Username=postgres;Password=18273645;Database=BIBLIOTEKA";

conn = new NpgsqlConnection(connection);

// Заполнение выпадающего списка названиями таблиц

comboBox1.Items.AddRange(new object[] {

"Издания", "Читатели", "Выдачи", "Типы\_изданий", "Жанры", "Издательства"

});

// Назначение обработчика события выбора таблицы

comboBox1.SelectedIndexChanged += comboBox1\_SelectedIndexChanged;

}

// Обработчик выбора таблицы из выпадающего списка

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

string table = comboBox1.Text; // Получаемвыбраннуютаблицу

LoadLookupTables(); // Загружаемтаблицыподстановок

string query = GetSelectQuery(table); // Строим SELECT-запрос

adapter = new NpgsqlDataAdapter(query, conn); // Инициализируемадаптер

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter); // Автоматическоеформированиекоманд

dt = new DataTable();

adapter.Fill(dt); // ЗаполняемтаблицуданнымиизБД

dataGridView1.Columns.Clear(); // Очищаемпредыдущиеколонки

dataGridView1.DataSource = dt; // Назначаем новую таблицу источником данных

HidePrimaryKey(); // Скрываем ключевые поля

ReplaceForeignKeyColumns(table); // Подставляем выпадающие списки для внешних ключей

// Устанавливаем обработчик для установки значений по умолчанию

dataGridView1.DefaultValuesNeeded -= dataGridView1\_DefaultValuesNeeded;

dataGridView1.DefaultValuesNeeded += dataGridView1\_DefaultValuesNeeded;

}

// СохранениеизмененийвБД

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

{

try

{

// Создание команд для обновления, добавления и удаления

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

adapter.UpdateCommand = builder.GetUpdateCommand();

adapter.InsertCommand = builder.GetInsertCommand();

adapter.DeleteCommand = builder.GetDeleteCommand();

adapter.Update(dt); // Сохраняемизменения

MessageBox.Show("Изменениясохранены.");

}

catch (Exception ex)

{

MessageBox.Show("Ошибкаприсохранении: " + ex.Message);

}

}

// Удалениевыбраннойстроки

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

{

if (dataGridView1.CurrentRow != null)

{

try

{

dataGridView1.Rows.RemoveAt(dataGridView1.CurrentRow.Index); // Удаляемстроку

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

adapter.Update(dt); // Применяемизменения

MessageBox.Show("Записьудалена.");

}

catch (Exception ex)

{

MessageBox.Show("Ошибкаприудалении: " + ex.Message);

}

}

else

{

MessageBox.Show("Выберите строку для удаления.");

}

}

// Формирование SELECT-запроса для выбранной таблицы

private string GetSelectQuery(string table)

{

return $"SELECT \* FROM \"{table}\"";

}

// Скрытие столбцов с первичными ключами

private void HidePrimaryKey()

{

var hiddenKeys = new[] {

"Код\_выдачи", "Код\_читателя", "Код\_жанра", "Код\_издательства", "Код\_типа"

};

foreach (var key in hiddenKeys)

{

if (dataGridView1.Columns.Contains(key))

dataGridView1.Columns[key].Visible = false;

}

}

// Загрузкатаблицсправочников (подстановок)

private void LoadLookupTables()

{

lookupData["Типы\_изданий"] = LoadLookup("Типы\_изданий", "Код\_типа", "Тип");

lookupData["Жанры"] = LoadLookup("Жанры", "Код\_жанра", "Жанр");

lookupData["Издательства"] = LoadLookup("Издательства", "Код\_издательства", "Издательство");

lookupData["Читатели"] = LoadLookup("Читатели", "Код\_читателя", "ФИО");

lookupData["Издания"] = LoadLookup("Издания", "Инвентарный\_номер", "Название");

}

// Загрузка одной таблицы справочника и построение словаря ключ => значение

private Dictionary<object, string> LoadLookup(string table, string keyCol, string valCol)

{

Dictionary<object, string> dict = new Dictionary<object, string>();

string query = $"SELECT \"{keyCol}\", \"{valCol}\" FROM \"{table}\"";

using (var cmd = new NpgsqlCommand(query, conn))

{

conn.Open();

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

dict.Add(reader[0], reader[1].ToString());

}

}

conn.Close();

}

return dict;

}

// Замена колонок с внешними ключами на выпадающие списки

private void ReplaceForeignKeyColumns(string table)

{

Dictionary<string, string> fkMap = new Dictionary<string, string>();

// Карта внешних ключей для таблиц

if (table == "Издания")

{

fkMap.Add("Код\_типа", "Типы\_изданий");

fkMap.Add("Код\_жанра", "Жанры");

fkMap.Add("Код\_издательства", "Издательства");

}

else if (table == "Выдачи")

{

fkMap.Add("Код\_читателя", "Читатели");

fkMap.Add("Инвентарный\_номер", "Издания");

}

foreach (var kvp in fkMap)

{

string column = kvp.Key;

string lookupTable = kvp.Value;

if (!dt.Columns.Contains(column)) continue;

// Созданиеколонкисвыпадающимсписком

DataGridViewComboBoxColumn combo = new DataGridViewComboBoxColumn

{

Name = column,

DataPropertyName = column,

HeaderText = column,

DataSource = new BindingSource(lookupData[lookupTable], null),

DisplayMember = "Value", // Отображаемоезначение

ValueMember = "Key", // Значениевтаблице

FlatStyle = FlatStyle.Flat

};

int index = dataGridView1.Columns[column].Index;

dataGridView1.Columns.Remove(column);

dataGridView1.Columns.Insert(index, combo);

}

}

// Установка значений по умолчанию при добавлении новой строки

private void dataGridView1\_DefaultValuesNeeded(object sender, DataGridViewRowEventArgs e)

{

string table = comboBox1.Text;

if (table == "Выдачи")

{

if (lookupData["Читатели"].Count > 0)

e.Row.Cells["Код\_читателя"].Value = lookupData["Читатели"].First().Key;

if (lookupData["Издания"].Count > 0)

e.Row.Cells["Инвентарный\_номер"].Value = lookupData["Издания"].First().Key;

e.Row.Cells["Дата\_выдачи"].Value = DateTime.Today; // Установкадатывыдачи

}

}

// Кнопкапереходанатретьюформу

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

{

this.Hide();

Form3 form3 = new Form3();

form3.Show();

}

// Закрытие приложения при закрытии формы

private void FormClosingevent(object sender, FormClosingEventArgs e)

{

Application.Exit();

}

}

}

Нарисунках12, 13результатработыданногокода:

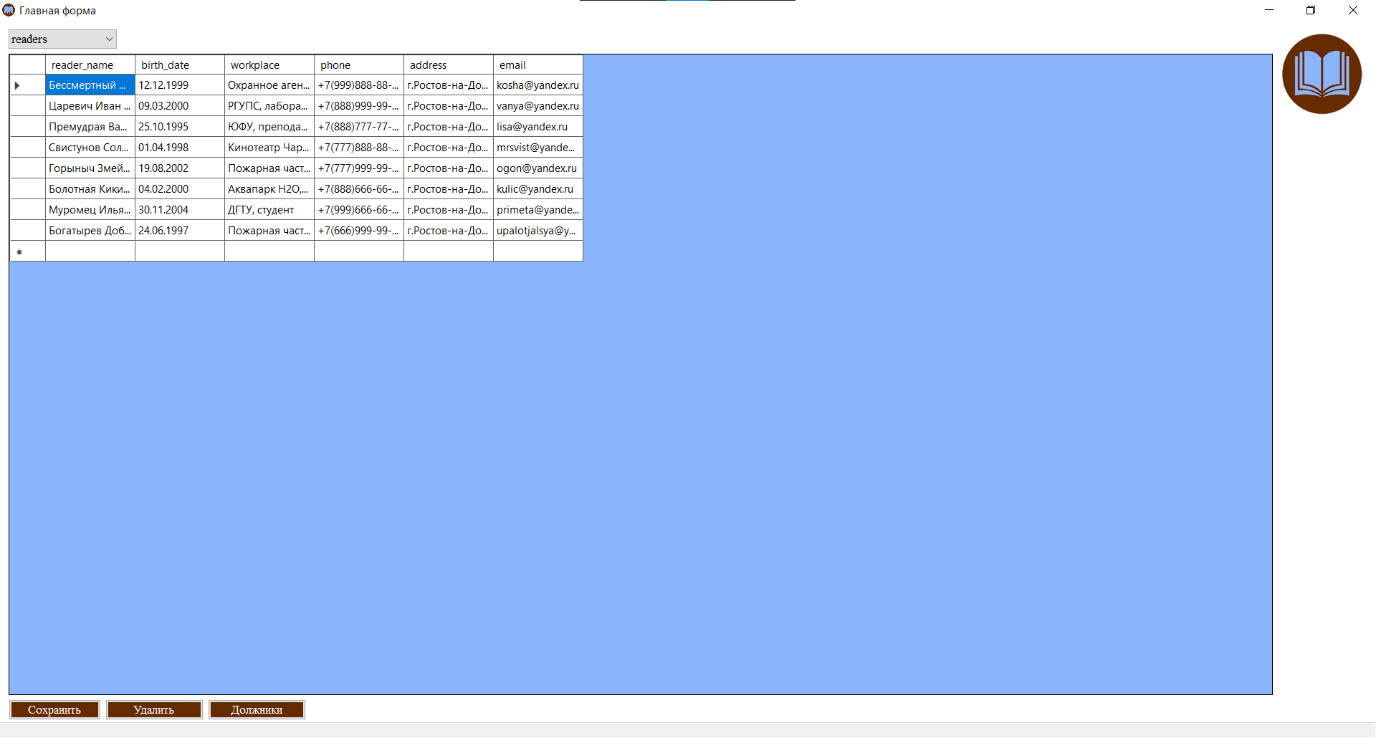


Рисунок 12 – Результат работы приложения

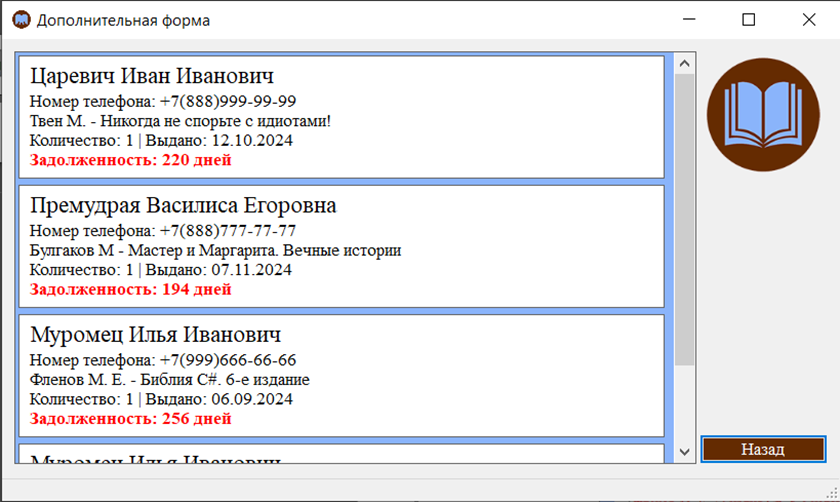


Рисунок 13 – Результат работы приложения

Листинг кода формы просмотра читателей:

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

using Npgsql;

namespace YCHEBKA2

{

public partial class Form3 : Form

{

// Строка подключения к базе данных PostgreSQL

private string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=BIBLIOTEKA";

public Form3()

{

InitializeComponent();

// Загружаемдолжниковсразупризапускеформы

LoadDebtors();

// Обработка закрытия формы — завершение приложения

this.FormClosing += new FormClosingEventHandler(FormClosingevent);

}

// Метод загрузки информации о должниках

private void LoadDebtors()

{

// Очищаем все элементы управления из панели перед обновлением

flowLayoutPanel1.Controls.Clear();

// Словарь для хранения должников и их задолженностей

var debtorList = new Dictionary<string, (string phone, List<(string author, string title, int count, DateTime date, int overdueDays)>)>();

using (var conn = new NpgsqlConnection(connectionString))

{

conn.Open();

// SQL-запрос для получения данных о должниках, у которых просрочены книги

string query = @"

SELECT

ч.""ФИО"",

ч.""Телефон"",

и.""Автор"",

и.""Название"",

в.""Количество"",

в.""Дата\_выдачи"",

EXTRACT(DAY FROM CURRENT\_DATE - (в.""Дата\_выдачи"" + INTERVAL '14 days')) AS days\_overdue

FROM ""Выдачи"" в

JOIN ""Читатели"" ч ON в.""Код\_читателя"" = ч.""Код\_читателя""

JOIN ""Издания"" и ON в.""Инвентарный\_номер"" = и.""Инвентарный\_номер""

WHERE в.""Дата\_возврата"" IS NULL

AND в.""Дата\_выдачи"" + INTERVAL '14 days' < CURRENT\_DATE

ORDER BY ч.""ФИО"";

";

using (var cmd = new NpgsqlCommand(query, conn))

using (var reader = cmd.ExecuteReader())

{

// Чтение строк из результата запроса

while (reader.Read())

{

string fio = reader.GetString(0); // ФИОчитателя

string phone = reader.IsDBNull(1) ? "—" : reader.GetString(1); // Телефонилипрочерк

string author = reader.GetString(2); // Авторкниги

string title = reader.GetString(3); // Названиекниги

int count = reader.GetInt32(4); // Количествоэкземпляров

DateTime borrowDate = reader.GetDateTime(5); // Датавыдачи

int overdueDays = Convert.ToInt32(reader["days\_overdue"]); // Кол-воднейпросрочки

// Если читатель ещё не добавлен — добавляем

if (!debtorList.ContainsKey(fio))

{

debtorList[fio] = (phone, new List<(string, string, int, DateTime, int)>());

}

// Добавляем информацию о книге к соответствующему читателю

debtorList[fio].Item2.Add((author, title, count, borrowDate, overdueDays));

}

}

}

// Отображаеминформациюнаформе

foreach (var debtor in debtorList)

{

// Панель для каждого должника

Panel panel = new Panel

{

BackColor = Color.White,

Width = flowLayoutPanel1.Width - 40,

AutoSize = true,

Padding = new Padding(10),

Margin = new Padding(5),

BorderStyle = BorderStyle.FixedSingle

};

// Заголовок: ФИОителефон

Label lblHeader = new Label

{

Text = $"{debtor.Key} / {debtor.Value.phone}",

Font = new Font("Times New Roman", 11, FontStyle.Bold),

AutoSize = true

};

panel.Controls.Add(lblHeader);

// Отступпередспискомкниг

Label lblSpacer = new Label

{

Text = "\n", // Пустой текст для создания отступа

AutoSize = true,

Margin = new Padding(0, 5, 0, 5)

};

panel.Controls.Add(lblSpacer);

// Списоккниг, невозвращённыхчитателем

foreach (var book in debtor.Value.Item2)

{

Label lblBook = new Label

{

Text = $"\n{book.author} / {book.title} / {book.count} / {book.date.ToShortDateString()} / {book.overdueDays} днейзадолженности",

Font = new Font("Times New Roman", 10),

AutoSize = true,

Margin = new Padding(0, 5, 0, 0),

ForeColor = book.overdueDays > 0 ? Color.Red : Color.Black // Красныйцветприпросрочке

};

panel.Controls.Add(lblBook);

// Отступмеждукнигами

Label lblBookSpacer = new Label

{

Text = "",

AutoSize = true,

Margin = new Padding(0, 5, 0, 10)

};

panel.Controls.Add(lblBookSpacer);

}

// Добавляем панель с данными о должнике в интерфейс

flowLayoutPanel1.Controls.Add(panel);

}

}

// Кнопка "Назад" — возвращение к предыдущей форме

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

{

this.Hide(); // Скрываем текущую форму

Form2 form2 = new Form2(); // Создаём и показываем форму 2

form2.Show();

}

// Обработчик закрытия формы — завершение приложения

private void FormClosingevent(object sender, FormClosingEventArgs e)

{

Application.Exit(); // Завершение всех форм

}

}

}

На рисунке 14 представлена работа формы:

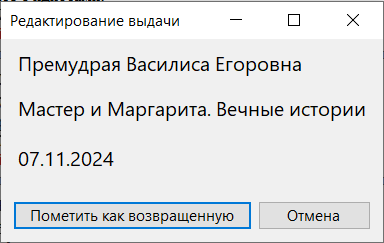


Рисунок 14 – Работа формы

Работа № 3

Задание: Разработать подсистему для работы с посетителямисалона красоты, обеспечивающую следующий функционал:

– оформление заявок на обслуживание;

– добавление/редактирование данных о посетителе;

– просмотр истории посещения салона клиентом.

Получить ER-диаграмму средствами СУБД: ER-диаграмма должна быть представлена в формате PDF и содержать таблицы, связи между ними, атрибуты и ключи (типами данных на данном этапе можно пренебречь).

Импортировать данные.

Разработать программный модуль для учета посетителей. Необходимо реализовать вывод списка клиентов, информация о которых хранятся в базе данных, согласно предоставленному макету:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | ФИО посетителя хх %  Телефон  Количество баллов |  |
|  |  |  |
|  | ФИО посетителя хх %  Телефон  Количество баллов |  |
|  |  |  |
|  | ФИО посетителя хх %  Телефон  Количество баллов |  |
|  |  |  |

В салоне предоставляется кешбэк 10 %, а также скидки на услуги в самое незагруженное время в салоне 20% на любую процедуру в будние дни с 12 до 14 часов.

Необходимо осуществить вывод истории реализации услуг клиенту с указанием наименования услуги, мастера и даты и времениуслуги.

Разработать интерфейс программного модуля для работы с читателями.

Реализовать функции добавления/редактирования данных клиента в новом окне (странице) – форме для добавления/редактирования клиента. Переходы на эту форму должны быть реализованы из формы списка читателей: для редактирования – при нажатии на конкретный элемент, для добавления – при нажатии кнопки.

На форме для добавления/редактирования клиента должны быть предусмотрены следующие поля: ФИО клиента, дату рождения, телефони email.

**Ход работы**

Подготовим данные для импорта. Для этого приведём все данные в предоставленных для импорта таблицах к 3НФ (Рисунок 15).

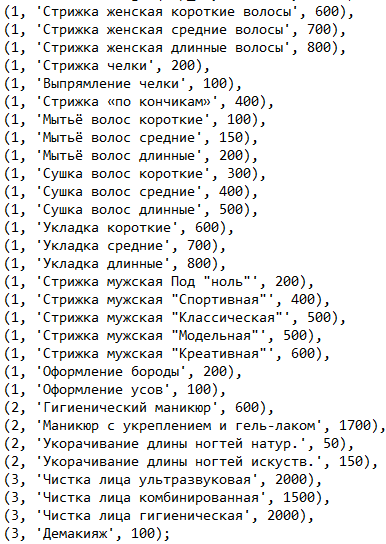


Рисунок 15 – Подготовка данных для импорта

Теперь необходимо создать базу данных (Рисунок 16)

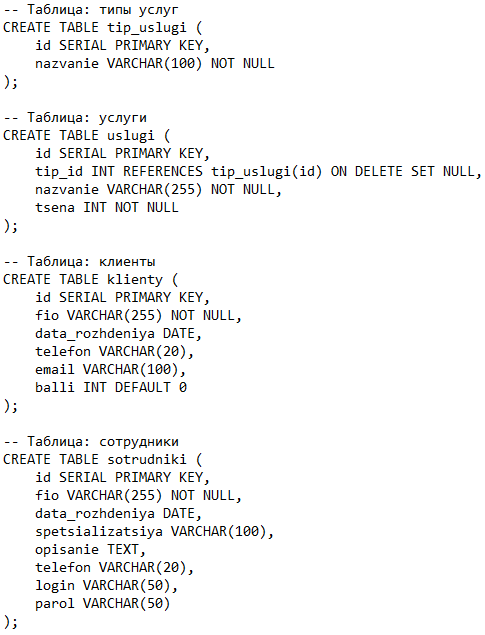


Рисунок 16 – Скрипт для создания БД

Теперь сделаем импорт данных (Рисунок 17)

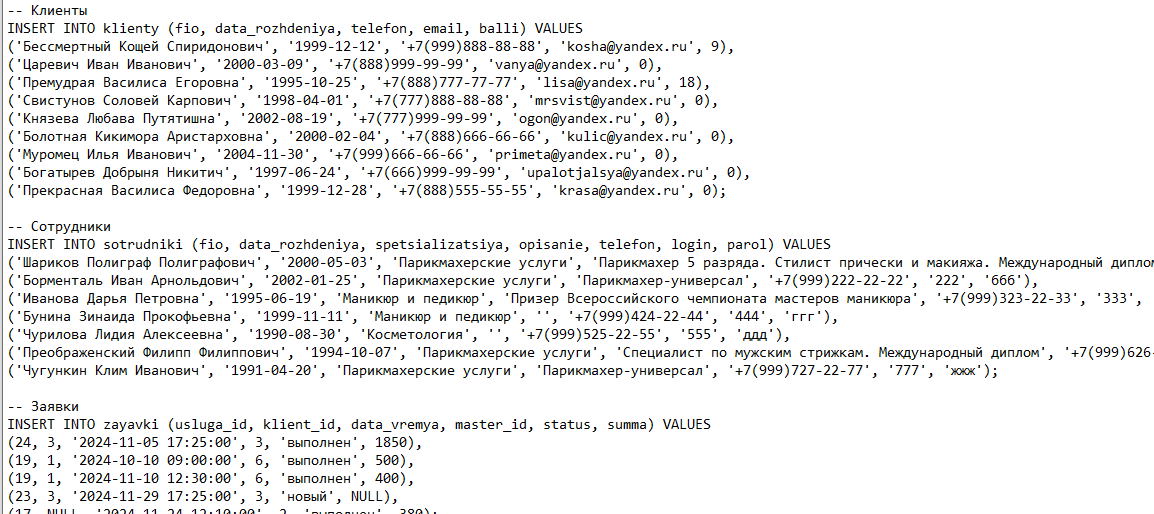


Рисунок 17 – Импорт данных в БД

Листинг кода формы авторизации:

// Подключение необходимых пространств имён

usingSystem;

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;

// ПодключениебиблиотекидляработысPostgreSQL

usingNpgsql;

namespaceYCHEBKA3

{

// Основнойклассформыавторизации

public partial class Form1 : Form

{

// Объектсоединениясбазойданных

private NpgsqlConnection connection;

// Строкаподключениякбазеданных PostgreSQL

private string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=salon";

// Конструктор формы

public Form1()

{

InitializeComponent(); // Инициализация компонентов формы

// Создание и открытие подключения к базе данных при запуске формы

NpgsqlConnection connection = new NpgsqlConnection(connectionString);

connection.Open(); // ВАЖНО: Это локальная переменная, которая перекрывает поле. Лучше использовать this.connection

}

// Обработчик события нажатия на кнопку "Войти"

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

{

// Получение введённых пользователем логина и пароля

string username = textBox1.Text;

string password = textBox2.Text;

// Проверка данных пользователя

if (AuthenticateUser(username, password))

{

// Если пользователь найден — сообщение об успехе и переход к следующей форме

MessageBox.Show("Успешная аутентификация! Добро пожаловать, " + username + "!");

Form f2 = new Form2(); // Создание экземпляра следующей формы

this.Hide(); // Скрытие текущей формы

f2.Show(); // Отображение новой формы

}

else

{

// Ошибка аутентификации — вывод сообщения

MessageBox.Show("Ошибка аутентификации. Пользователь с такими данными не найден.");

}

}

// Метод для проверки существования пользователя в базе данных

private bool AuthenticateUser(string username, string password)

{

// Повторное объявление строки подключения (можно убрать и использовать поле класса)

string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=salon";

// Создание и открытие подключения с использованием using (гарантирует автоматическое закрытие)

using (NpgsqlConnection connection = new NpgsqlConnection(connectionString))

{

connection.Open();

// SQL-запрос с параметрами для безопасного поиска пользователя

string query = "SELECT COUNT(\*) FROM sotrudniki WHERE login = @Username AND parol = @Password";

using (NpgsqlCommand command = new NpgsqlCommand(query, connection))

{

// Передача параметров в запрос

command.Parameters.AddWithValue("@Username", username);

command.Parameters.AddWithValue("@Password", password);

// Выполнение запроса и получение количества совпадений

int count = Convert.ToInt32(command.ExecuteScalar());

// Если есть хотя бы одно совпадение — возвращаем true

return count > 0;

}

}

}

}

}

На рисунке 18 работа приведённого выше кода:



Рисунок 18 – Работа формы авторизации

Ниже листинг кода основной формы:

using Npgsql;

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Windows.Forms;

namespace YCHEBKA3

{

public partial class Form2 : Form

{

NpgsqlConnection conn; // Подключение к базе данных

NpgsqlDataAdapter adapter; // Адаптер для работы с данными

DataTable dt; // Таблица данных для привязки к Grid

Dictionary<string, Dictionary<object, string>> lookupData = new Dictionary<string, Dictionary<object, string>>(); // Словарьдляхраненияданныхизсправочников

// Конструктор формы

public Form2()

{

InitializeComponent(); // Инициализация компонентов формы

this.FormClosing += newFormClosingEventHandler(FormClosingevent); // Обработчикзакрытияформы

// Установка строки подключения к базе данных

string connection = "Host=localhost;Username=postgres;Password=18273645;Database=salon";

conn = new NpgsqlConnection(connection); // Инициализацияподключения

// Добавление значений в comboBox для выбора таблицы

comboBox1.Items.AddRange(new object[] {

"tip\_uslugi", "uslugi", "klienty", "sotrudniki", "zayavki", "raspisanie"

});

// Подключение обработчика изменения выбора в comboBox

comboBox1.SelectedIndexChanged += comboBox1\_SelectedIndexChanged;

}

// Обработчик изменения выбранного значения в comboBox

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

string table = comboBox1.Text; // Получаемвыбраннуютаблицуиз comboBox

LoadLookupTables(); // Загружаем справочники

// Формируем SQL запрос для получения всех данных из выбранной таблицы

string query = $"SELECT \* FROM \"{table}\"";

adapter = new NpgsqlDataAdapter(query, conn);

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

// Создаем новую таблицу и заполняем её данными из базы

dt = new DataTable();

adapter.Fill(dt);

// Очищаем текущие колонки и привязываем новые данные к DataGridView

dataGridView1.Columns.Clear();

dataGridView1.DataSource = dt;

// Скрываем колонку с первичным ключом и заменяем внешние ключи на комбобоксы

HidePrimaryKey();

ReplaceForeignKeyColumns(table);

// Подключаем обработчик для автоматического заполнения значений в новой строке

dataGridView1.DefaultValuesNeeded -= dataGridView1\_DefaultValuesNeeded;

dataGridView1.DefaultValuesNeeded += dataGridView1\_DefaultValuesNeeded;

}

// Обработчик для сохранения изменений в базе данных

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

{

try

{

// Обновляемданныевбазе

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

adapter.Update(dt);

MessageBox.Show("Изменениясохранены.");

}

catch (Exception ex)

{

// Обработка ошибок при сохранении

MessageBox.Show("Ошибка при сохранении: " + ex.Message);

}

}

// Обработчик для удаления выбранной строки

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

{

if (dataGridView1.CurrentRow != null) // Проверяем, что выбрана строка

{

try

{

// Удаляем выбранную строку

dataGridView1.Rows.RemoveAt(dataGridView1.CurrentRow.Index);

adapter.Update(dt);

MessageBox.Show("Записьудалена.");

}

catch (Exception ex)

{

// Обработка ошибок при удалении

MessageBox.Show("Ошибка при удалении: " + ex.Message);

}

}

else

{

// Сообщаем о том, что не выбрана строка

MessageBox.Show("Выберите строку для удаления.");

}

}

// Скрытие колонки с первичным ключом

private void HidePrimaryKey()

{

foreach (DataColumn col in dt.Columns)

{

if (col.ColumnName == "id" && dataGridView1.Columns.Contains(col.ColumnName))

{

dataGridView1.Columns[col.ColumnName].Visible = false; // Скрываемколонкус id

}

}

}

// Загрузкаданныхдлясправочников

private void LoadLookupTables()

{

// Загружаемсправочникидлякаждойтаблицы

lookupData["uslugi"] = LoadLookup("uslugi", "id", "nazvanie");

lookupData["klienty"] = LoadLookup("klienty", "id", "fio");

lookupData["sotrudniki"] = LoadLookup("sotrudniki", "id", "fio");

lookupData["tip\_uslugi"] = LoadLookup("tip\_uslugi", "id", "nazvanie");

}

// Метод для загрузки данных из справочника

private Dictionary<object, string> LoadLookup(string table, string keyCol, string valCol)

{

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

string query = $"SELECT \"{keyCol}\", \"{valCol}\" FROM \"{table}\"";

using (var cmd = new NpgsqlCommand(query, conn))

{

conn.Open();

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

dict[reader[0]] = reader[1].ToString(); // Заполняемсловарьзначениямиизбазы

}

conn.Close();

}

return dict;

}

// Метод для замены внешних ключей на комбобоксы

private void ReplaceForeignKeyColumns(string table)

{

var fkMap = new Dictionary<string, string>();

// Определяем, какие внешние ключи нужно заменить на комбобоксы

if (table == "uslugi")

fkMap.Add("tip\_id", "tip\_uslugi");

else if (table == "zayavki")

{

fkMap.Add("usluga\_id", "uslugi");

fkMap.Add("klient\_id", "klienty");

fkMap.Add("master\_id", "sotrudniki");

}

else if (table == "raspisanie")

fkMap.Add("sotrudnik\_id", "sotrudniki");

// Заменяем соответствующие колонки на комбобоксы

foreach (var kvp in fkMap)

{

string column = kvp.Key;

string lookupTable = kvp.Value;

if (!dt.Columns.Contains(column)) continue;

var combo = new DataGridViewComboBoxColumn

{

Name = column,

DataPropertyName = column,

HeaderText = column,

DataSource = new BindingSource(lookupData[lookupTable], null),

DisplayMember = "Value",

ValueMember = "Key",

FlatStyle = FlatStyle.Flat

};

int index = dataGridView1.Columns[column].Index;

dataGridView1.Columns.Remove(column);

dataGridView1.Columns.Insert(index, combo); // Вставляемкомбобокснаместоколонки

}

}

// Заполнениезначенийвновойстроке

private void dataGridView1\_DefaultValuesNeeded(object sender, DataGridViewRowEventArgs e)

{

string table = comboBox1.Text;

// Заполняем значения по умолчанию для различных таблиц

if (table == "zayavki")

{

if (lookupData["klienty"].Count > 0)

e.Row.Cells["klient\_id"].Value = lookupData["klienty"].First().Key;

if (lookupData["uslugi"].Count > 0)

e.Row.Cells["usluga\_id"].Value = lookupData["uslugi"].First().Key;

if (lookupData["sotrudniki"].Count > 0)

e.Row.Cells["master\_id"].Value = lookupData["sotrudniki"].First().Key;

e.Row.Cells["data\_vremya"].Value = DateTime.Now;

e.Row.Cells["status"].Value = "Новая";

}

if (table == "raspisanie")

{

if (lookupData["sotrudniki"].Count > 0)

e.Row.Cells["sotrudnik\_id"].Value = lookupData["sotrudniki"].First().Key;

}

}

// Обработчик для кнопки перехода на Form3

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

{

this.Hide();

Form3 form3 = new Form3();

form3.Show(); // Переходнадругуюформу

}

// Обработчикдлязакрытияформы

private void FormClosingevent(object sender, FormClosingEventArgs e)

{

Application.Exit(); // Завершаем приложение при закрытии формы

}

}

}

На рисунках 19, 20 результат работы данного кода:

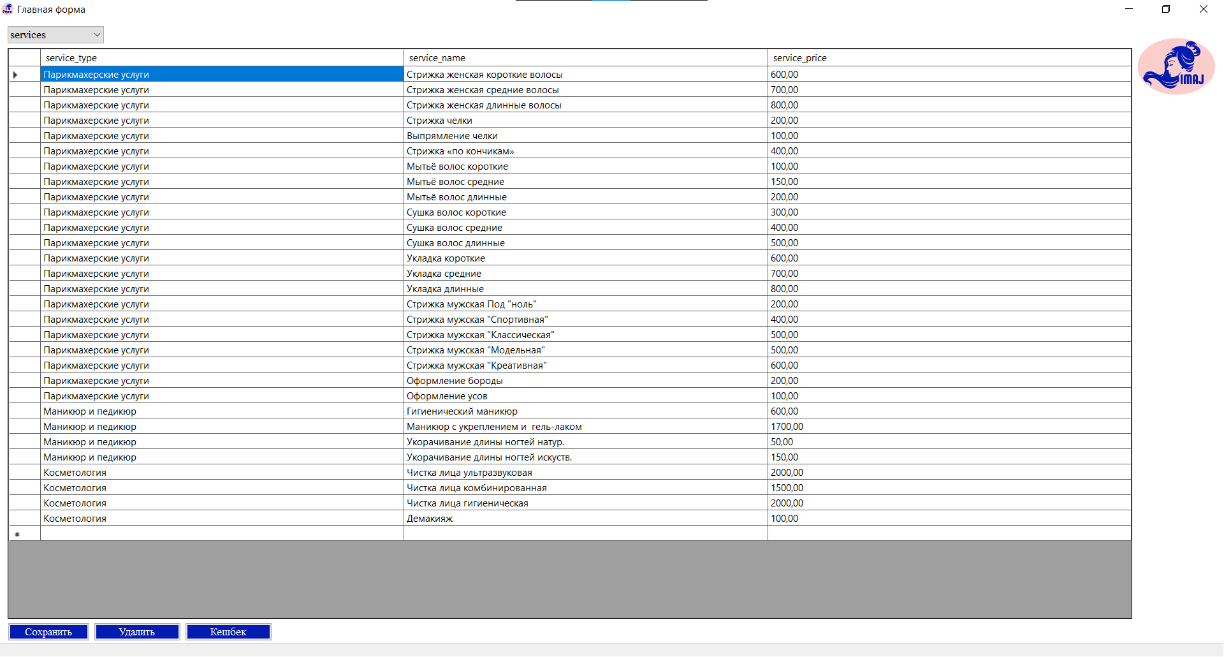


Рисунок 19 – Результат работы приложения

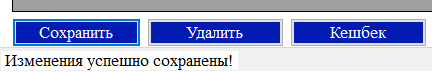


Рисунок 20 – Результат работы приложения

Листинг кода для списка и истории клиента:

using System;

using System.Windows.Forms;

using Npgsql;

namespace YCHEBKA3

{

public partial class Form3 : Form

{

bool showingHistory = false; // Флаг для переключения между списком клиентов и историей

string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=salon"; // Строкаподключениякбазеданных

// Конструктор формы

public Form3()

{

InitializeComponent(); // Инициализация компонентов формы

this.FormClosing += new FormClosingEventHandler(FormClosingevent); // Подключение обработчика закрытия формы

LoadClients(); // Загрузка списка клиентов при инициализации

}

// Обработчиксобытиязакрытияформы

private void FormClosingevent(object sender, FormClosingEventArgs e)

{

Application.Exit(); // Завершаем приложение при закрытии формы

}

// Обработчик кнопки для перехода на Form2

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

{

this.Hide(); // Скрываемтекущуюформу

Form2 form2 = new Form2(); // Создаёмэкземпляр Form2

form2.Show(); // Показываем Form2

}

// Обработчик кнопки для переключения между списком клиентов и историей

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

{

listBox1.Items.Clear(); // Очищаемтекущиеэлементысписка

// Переключаем видимость comboBox и текст на кнопке в зависимости от текущего состояния

comboBox1.Visible = !showingHistory;

if (showingHistory)

{

LoadClients(); // Загружаем список клиентов

button1.Text = "История клиентов"; // Меняем текст на кнопке

label1.Text = "Список клиентов"; // Меняем текст на лейбле

}

else

{

LoadClientNamesToComboBox(); // Загружаем имена клиентов в comboBox

button1.Text = "Список клиентов"; // Меняем текст на кнопке

label1.Text = "История клиентов"; // Меняем текст на лейбле

}

showingHistory = !showingHistory; // Переключаем флаг

}

// Метод для загрузки списка клиентов в listBox

private void LoadClients()

{

listBox1.Items.Clear(); // Очищаем текущие элементы списка

using (var conn = new NpgsqlConnection(connectionString)) // Создаёмподключениекбазеданных

{

conn.Open(); // Открываем соединение с базой

string query = "SELECT id, fio, telefon, balli FROM klienty"; // SQL-запрос для получения списка клиентов

using (var cmd = new NpgsqlCommand(query, conn)) // Создаём команду для выполнения запроса

using (var reader = cmd.ExecuteReader()) // Читаем результаты запроса

{

while (reader.Read()) // Проходим по результатам

{

string fullName = reader.GetString(1); // ПолучаемФИОклиента

string phone = reader.GetString(2); // Получаемтелефонклиента

int points = reader.GetInt32(3); // Получаемколичествобалловклиента

int discount = CalculateDiscount(points, DateTime.Now); // Рассчитываемскидку

// Добавляем информацию о клиенте в список

listBox1.Items.Add($"{fullName} {discount} %");

listBox1.Items.Add(phone);

listBox1.Items.Add("Количествобаллов: " + points);

listBox1.Items.Add(""); // Добавляем пустую строку для разделения

}

}

}

}

// Метод для расчёта скидки на основе баллов и текущего времени

private int CalculateDiscount(int points, DateTime dateTime)

{

int discount = points / 10; // Базовая скидка 1% за каждые 10 баллов

// Дополнительная скидка в зависимости от времени суток

if (dateTime.DayOfWeek >= DayOfWeek.Monday && dateTime.DayOfWeek <= DayOfWeek.Friday)

{

if (dateTime.Hour >= 12 && dateTime.Hour < 14) // В период с 12:00 до 14:00 в будние дни

discount += 20; // Добавляем дополнительную скидку 20%

}

return discount; // Возвращаем окончательную скидку

}

// Метод для загрузки имён клиентов в comboBox

private void LoadClientNamesToComboBox()

{

comboBox1.Items.Clear(); // Очищаем элементы в comboBox

comboBox1.Visible = true; // Делаем comboBox видимым

using (var conn = new NpgsqlConnection(connectionString)) // Создаёмподключениекбазеданных

{

conn.Open(); // Открываем соединение с базой

string query = "SELECT id, fio FROM klienty"; // SQL-запрос для получения списка имён клиентов

using (var cmd = new NpgsqlCommand(query, conn)) // Создаём команду для выполнения запроса

using (var reader = cmd.ExecuteReader()) // Читаем результаты запроса

{

while (reader.Read()) // Проходим по результатам

{

comboBox1.Items.Add(new ComboBoxItem

{

Id = reader.GetInt32(0), // Добавляем id клиента

Name = reader.GetString(1) // Добавляемимяклиента

});

}

}

}

// Подключаем обработчик для изменения выбранного элемента в comboBox

comboBox1.SelectedIndexChanged -= ComboBox1\_SelectedIndexChanged;

comboBox1.SelectedIndexChanged += ComboBox1\_SelectedIndexChanged;

}

// Обработчик события изменения выбранного элемента в comboBox

private void ComboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox1.SelectedItem is ComboBoxItem selectedClient) // Есливыбранклиент

{

LoadClientHistory(selectedClient.Id); // Загружаемисториюклиента

}

}

// Метод для загрузки истории выбранного клиента в listBox

private void LoadClientHistory(int clientId)

{

listBox1.Items.Clear(); // Очищаемтекущиеэлементысписка

using (var conn = new NpgsqlConnection(connectionString)) // Создаёмподключениекбазеданных

{

conn.Open(); // Открываем соединение с базой

string query = @"

SELECT u.nazvanie AS service, s.fio AS master, z.data\_vremya

FROM zayavki z

JOIN uslugi u ON z.usluga\_id = u.id

JOIN sotrudniki s ON z.master\_id = s.id

WHERE z.klient\_id = @clientId

ORDER BY z.data\_vremya DESC"; // SQL-запросдляполученияисторииклиента

using (var cmd = new NpgsqlCommand(query, conn)) // Создаёмкомандудлявыполнениязапроса

{

cmd.Parameters.AddWithValue("clientId", clientId); // Передаемпараметр clientId взапрос

using (var reader = cmd.ExecuteReader()) // Читаемрезультатызапроса

{

while (reader.Read()) // Проходим по результатам

{

string service = reader.GetString(0); // Названиеуслуги

string master = reader.GetString(1); // ФИОмастера

string date = reader.GetDateTime(2).ToString("dd.MM.yyyy HH:mm"); // Датаивремяоказанияуслуги

// Добавляем информацию о сервисе, мастере и дате в список

listBox1.Items.Add($"Услуга: {service}");

listBox1.Items.Add($"Мастер: {master}");

listBox1.Items.Add($"Дата: {date}");

listBox1.Items.Add(""); // Добавляем пустую строку для разделения

}

}

}

}

}

// Класс для представления элемента comboBox

private class ComboBoxItem

{

public int Id { get; set; } // ID клиента

public string Name { get; set; } // Имяклиента

// Переопределяем метод ToString для отображения имени клиента в comboBox

public override string ToString() => Name;

}

}

}

На рисунках21,22показано работа кода:

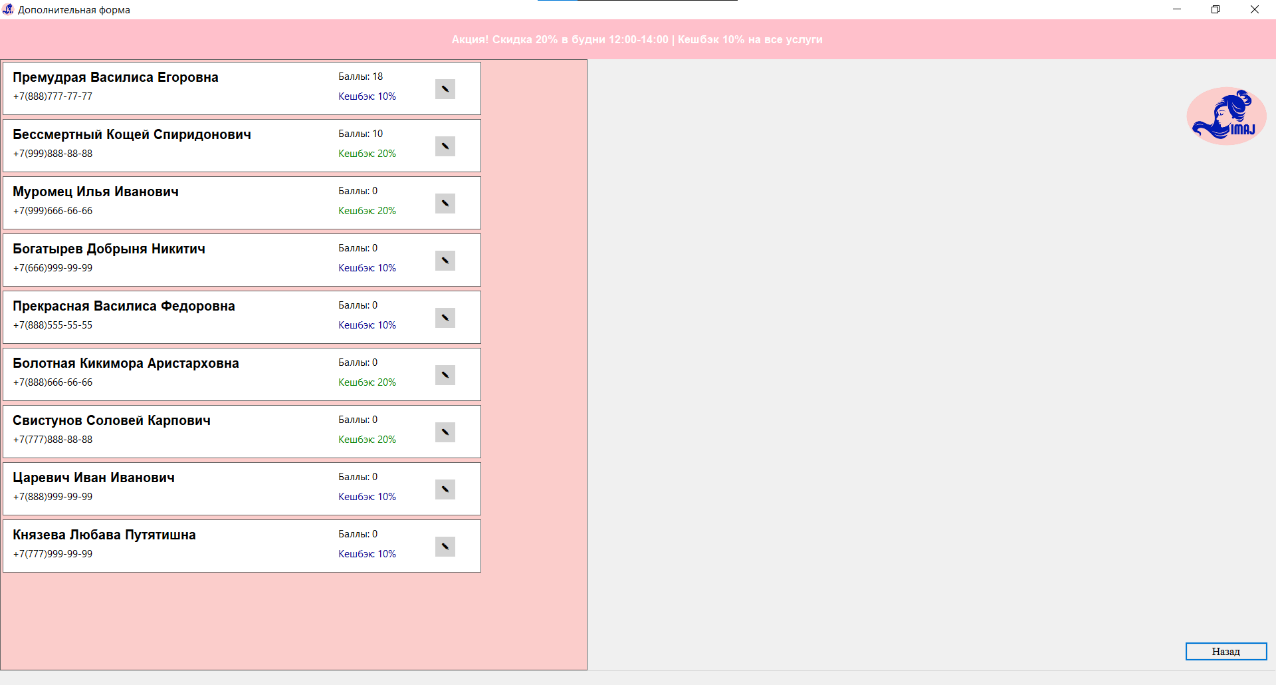


Рисунок 21 – Работа списка

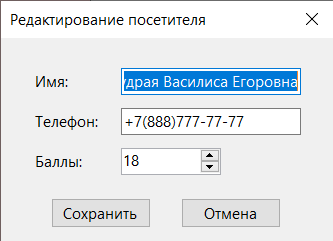


Рисунок 22 – Работа редактирования карточки

Работа № 4

Задание: Разработать подсистему для работы с товарами магазина «Заботливый фермер», обеспечивающую следующий функционал:

– оформление продаж;

– добавление/редактирование данных о товарах;

– просмотр товаров с истекающим сроком годности.

Получить ER-диаграмму средствами СУБД: ER-диаграмма должна быть представлена в формате PDF и содержать таблицы, связи между ними, атрибуты и ключи (типами данных на данном этапе можно пренебречь).

Разработать программный модуль для учета товаров. Необходимо реализовать вывод списка товаров с истекающим сроком годности, информация о которых хранятся в базе данных, согласно предоставленному макету:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | Наименование товара  Категория товара  Срок годности  Количество дней до окончания срока годности хх % скидки |  |
|  |  |  |
|  | Наименование товара  Категория товара  Срок годности  Количество дней до окончания срока годности хх % скидки |  |
|  |  |  |
|  | Наименование товара  Категория товара  Срок годности  Количество дней до окончания срока годности хх % скидки |  |
|  |  |  |

В магазине предоставляются скидки на товары за три дня до истечения срока годности 10 % и за один день до истечения 20%.

Необходимо осуществить оформление продажи товаров с выводом на экран списка товаров по категории, корзины покупателя и расчета суммы продажи, при этом учитывать срок годности (при истечении срока не допускать продажу товара).

Разработать интерфейс программного модуля для работы с товарами и продажами.

Реализовать функции добавления/редактирования данных товара в новом окне (странице) – форме для добавления/редактирования товара. Переходы на эту форму должны быть реализованы из формы списка товаров: для редактирования товаров из списка – при нажатии на конкретный элемент, для редактирования/добавления любого товара – при нажатии кнопки.

На форме для добавления/редактирования товара должны быть предусмотрены следующие поля: Название товара, Категория, Описание, размер упаковки, Дата изготовления, Дата срока годности, Количество.

**Ход работы**

Подготовим данные для импорта. Для этого приведём все данные в предоставленных для импорта таблицах к 3НФ (Рисунок 23).



Рисунок 23 – Подготовка данных для импорта

Теперь необходимо создать базу данных (Рисунок 24)

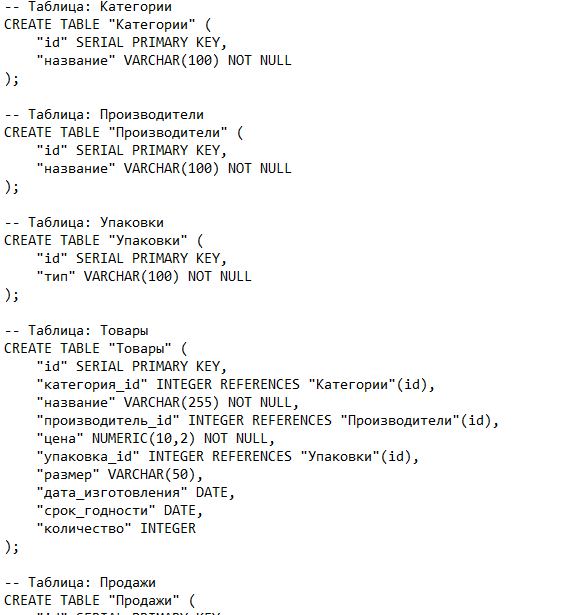


Рисунок 24 – Скрипт для создания БД

Теперь сделаем импорт данных (Рисунок 25)

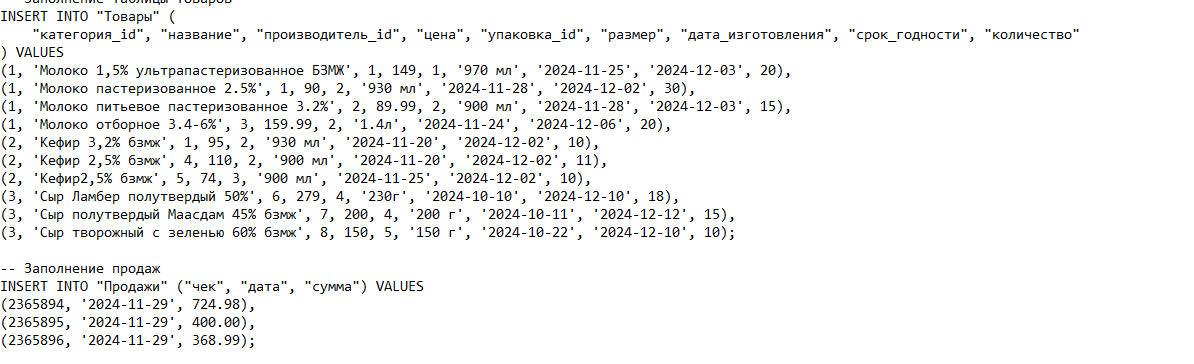


Рисунок 25 – Импорт данных в БД

Листинг кода формы авторизации:

;// Подключение необходимых пространств имён

using Npgsql;

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 YCHEBKA4

{

// Основной класс формы авторизации

public partial class Form1 : Form

{

// Объект соединения с базой данных

private NpgsqlConnection connection;

// Строка подключения к базе данных PostgreSQL

private string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=Ferma";

// Конструктор формы

public Form1()

{

InitializeComponent(); // Инициализация компонентов формы

// Создание и открытие подключения к базе данных при запуске формы

NpgsqlConnection connection = new NpgsqlConnection(connectionString);

connection.Open(); // ВАЖНО: Это локальная переменная, которая перекрывает поле. Лучше использовать this.connection

}

// Обработчик события нажатия на кнопку "Войти"

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

{

// Получение введённых пользователем логина и пароля

string username = textBox1.Text;

string password = textBox2.Text;

// Проверка данных пользователя

if (AuthenticateUser(username, password))

{

// Если пользователь найден — сообщение об успехе и переход к следующей форме

MessageBox.Show("Успешная аутентификация! Добро пожаловать, " + username + "!");

Form f2 = new Form2(); // Создание экземпляра следующей формы

this.Hide(); // Скрытие текущей формы

f2.Show(); // Отображение новой формы

}

else

{

// Ошибка аутентификации — вывод сообщения

MessageBox.Show("Ошибка аутентификации. Пользователь с такими данными не найден.");

}

}

// Метод для проверки существования пользователя в базе данных

private bool AuthenticateUser(string username, string password)

{

// Повторное объявление строки подключения (можно убрать и использовать поле класса)

string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=Ferma";

// Создание и открытие подключения с использованием using (гарантирует автоматическое закрытие)

using (NpgsqlConnection connection = new NpgsqlConnection(connectionString))

{

connection.Open();

// SQL-запрос с параметрами для безопасного поиска пользователя

string query = "SELECT COUNT(\*) FROM db\_users WHERE user\_login = @Username AND user\_password = @Password";

using (NpgsqlCommand command = new NpgsqlCommand(query, connection))

{

// Передача параметров в запрос

command.Parameters.AddWithValue("@Username", username);

command.Parameters.AddWithValue("@Password", password);

// Выполнение запроса и получение количества совпадений

int count = Convert.ToInt32(command.ExecuteScalar());

// Если есть хотя бы одно совпадение — возвращаем true

return count > 0;

}

}

}

}

}

На рисунке 26 работа приведённого выше кода:

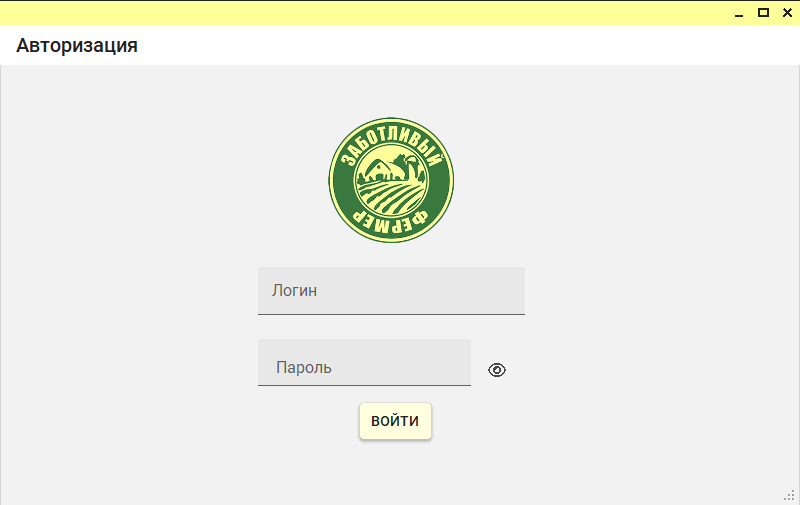


Рисунок 26 – Работа формы авторизации

Ниже листинг кода основной формы:

using Npgsql;

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Windows.Forms;

namespace YCHEBKA4

{

public partial class Form2 : Form

{

// Объявление переменных подключения и хранения данных

NpgsqlConnection conn;

NpgsqlDataAdapter adapter;

DataTable dt;

Dictionary<string, Dictionary<object, string>> lookupData = new Dictionary<string, Dictionary<object, string>>();

public Form2()

{

InitializeComponent();

this.FormClosing += new FormClosingEventHandler(FormClosingevent); // Обработка закрытия формы

// Строка подключения к базе данных PostgreSQL

string connection = "Host=localhost;Username=postgres;Password=18273645;Database=Ferma";

conn = new NpgsqlConnection(connection);

// Заполнение комбобокса названиями таблиц

comboBox1.Items.AddRange(new object[]

{

"Товары", "Продажи", "ПроданныеТовары", "Категории", "Производители", "Упаковки",

"Сотрудники", "Поставщики", "Поставки", "ПоставленныеТовары"

});

comboBox1.SelectedIndexChanged += comboBox1\_SelectedIndexChanged; // Подписка на событие выбора

}

// Обработчик выбора таблицы

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

string table = comboBox1.Text;

LoadLookupTables(); // Загрузка справочных таблиц для отображения связанных данных

try

{

// Получение всех записей из выбранной таблицы

string query = $"SELECT \* FROM \"{table}\"";

adapter = new NpgsqlDataAdapter(query, conn);

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

dt = new DataTable();

adapter.Fill(dt);

dataGridView1.Columns.Clear(); // Очистка колонок грида

dataGridView1.DataSource = dt; // Привязка таблицы к гриду

HidePrimaryKeys(); // Скрытие столбцов с первичными ключами

ReplaceForeignKeyColumns(table); // Замена внешних ключей на выпадающие списки

// Назначение обработчика по умолчанию для новых строк

dataGridView1.DefaultValuesNeeded -= dataGridView1\_DefaultValuesNeeded;

dataGridView1.DefaultValuesNeeded += dataGridView1\_DefaultValuesNeeded;

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке таблицы: " + ex.Message);

}

}

// Кнопка сохранения изменений

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

{

try

{

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

adapter.UpdateCommand = builder.GetUpdateCommand();

adapter.InsertCommand = builder.GetInsertCommand();

adapter.DeleteCommand = builder.GetDeleteCommand();

adapter.Update(dt); // Сохранение изменений в базе

MessageBox.Show("Изменения сохранены.");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при сохранении: " + ex.Message);

}

}

// Кнопка удаления записи

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

{

if (dataGridView1.CurrentRow != null)

{

try

{

dataGridView1.Rows.RemoveAt(dataGridView1.CurrentRow.Index); // Удаление строки

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

adapter.Update(dt); // Обновление базы

MessageBox.Show("Запись удалена.");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при удалении: " + ex.Message);

}

}

else

{

MessageBox.Show("Выберите строку для удаления.");

}

}

// Скрытие столбцов с первичными ключами

privatevoidHidePrimaryKeys()

{

foreach (DataGridViewColumn col in dataGridView1.Columns)

{

if (col.Name.ToLower() == "id" || col.Name.ToLower().Contains("код"))

{

col.Visible = false;

}

}

}

// Загрузка справочных таблиц в память

private void LoadLookupTables()

{

try

{

lookupData["Категории"] = LoadLookup("Категории", "id", "название");

lookupData["Производители"] = LoadLookup("Производители", "id", "название");

lookupData["Упаковки"] = LoadLookup("Упаковки", "id", "тип");

lookupData["Товары"] = LoadLookup("Товары", "id", "название");

lookupData["Сотрудники"] = LoadLookup("Сотрудники", "id", "ФИО");

lookupData["Поставщики"] = LoadLookup("Поставщики", "id", "название");

lookupData["Поставки"] = LoadLookup("Поставки", "id", "дата");

}

catch (Exceptionex)

{

MessageBox.Show("Ошибка при загрузке справочных таблиц: " + ex.Message);

}

}

// Загрузка данных справочной таблицы в словарь

private Dictionary<object, string> LoadLookup(string table, string keyCol, string valCol)

{

Dictionary<object, string> dict = new Dictionary<object, string>();

string query = $"SELECT \"{keyCol}\", \"{valCol}\" FROM \"{table}\"";

try

{

if (conn.State != ConnectionState.Open)

conn.Open();

using (var cmd = new NpgsqlCommand(query, conn))

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

dict[reader[0]] = reader[1].ToString(); // Запись пары ключ-значение

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке данных из таблицы {table}: {ex.Message}");

}

finally

{

if (conn.State == ConnectionState.Open)

conn.Close();

}

returndict;

}

// Замена внешних ключей на выпадающие списки

private void ReplaceForeignKeyColumns(string table)

{

var foreignKeys = new Dictionary<string, string>();

// Задание внешних ключей в зависимости от таблицы

if (table == "Товары")

{

foreignKeys.Add("категория\_id", "Категории");

foreignKeys.Add("производитель\_id", "Производители");

foreignKeys.Add("упаковка\_id", "Упаковки");

}

elseif (table == "ПроданныеТовары")

{

foreignKeys.Add("товар\_id", "Товары");

foreignKeys.Add("продажа\_id", "Продажи");

}

elseif (table == "Поставки")

{

foreignKeys.Add("поставщик\_id", "Поставщики");

foreignKeys.Add("сотрудник\_id", "Сотрудники");

}

elseif (table == "ПоставленныеТовары")

{

foreignKeys.Add("поставка\_id", "Поставки");

foreignKeys.Add("товар\_id", "Товары");

}

// Обход и замена колонок внешних ключей на комбобоксы

foreach (var fk in foreignKeys)

{

string column = fk.Key;

string refTable = fk.Value;

if (!dt.Columns.Contains(column) || !lookupData.ContainsKey(refTable))

continue;

DataGridViewComboBoxColumn combo = new DataGridViewComboBoxColumn

{

Name = column,

DataPropertyName = column,

HeaderText = column,

DataSource = new BindingSource(lookupData[refTable], null),

DisplayMember = "Value",

ValueMember = "Key",

FlatStyle = FlatStyle.Flat

};

int index = dataGridView1.Columns[column].Index;

dataGridView1.Columns.Remove(column);

dataGridView1.Columns.Insert(index, combo);

}

}

// Значения по умолчанию для новых строк

private void dataGridView1\_DefaultValuesNeeded(object sender, DataGridViewRowEventArgs e)

{

string table = comboBox1.Text;

if (table == "Продажи")

{

e.Row.Cells["дата"].Value = DateTime.Today;

}

else if (table == "ПроданныеТовары")

{

if (lookupData["Товары"].Count > 0)

e.Row.Cells["товар\_id"].Value = lookupData["Товары"].First().Key;

}

else if (table == "Товары")

{

if (lookupData["Категории"].Count > 0)

e.Row.Cells["категория\_id"].Value = lookupData["Категории"].First().Key;

if (lookupData["Производители"].Count > 0)

e.Row.Cells["производитель\_id"].Value = lookupData["Производители"].First().Key;

if (lookupData["Упаковки"].Count > 0)

e.Row.Cells["упаковка\_id"].Value = lookupData["Упаковки"].First().Key;

e.Row.Cells["дата\_изготовления"].Value = DateTime.Today;

e.Row.Cells["срок\_годности"].Value = DateTime.Today.AddMonths(1);

e.Row.Cells["количество"].Value = 0;

}

}

// Кнопка перехода на Form3

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

{

this.Hide();

Form3 form3 = new Form3();

form3.Show();

}

// Обработка закрытия формы — завершение приложения

private void FormClosingevent(object sender, FormClosingEventArgs e)

{

Application.Exit();

}

// Кнопка перехода на Form4

private void button4\_Click(object sender, EventArgs e)

{

this.Hide();

Form4 form4 = new Form4();

form4.Show();

}

}

}

На рисунках 27, 28 результат работы данного кода:

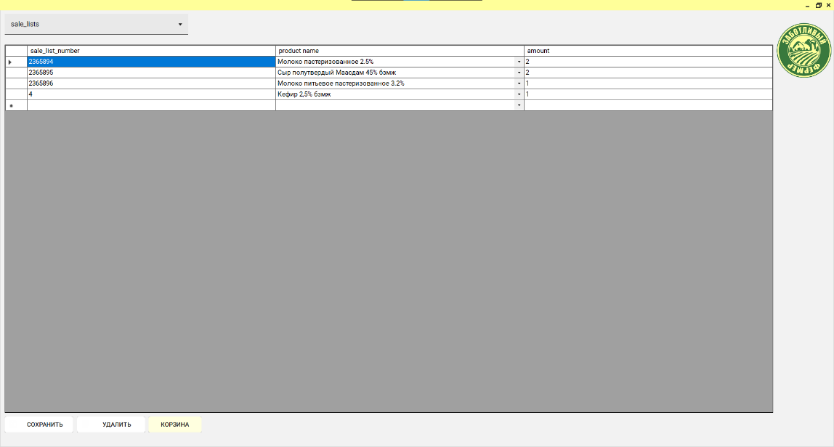


Рисунок 27 – Результат работы приложения

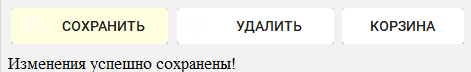


Рисунок 28 – Результат работы приложения

На рисунке 29 представлена работа списка товаров:

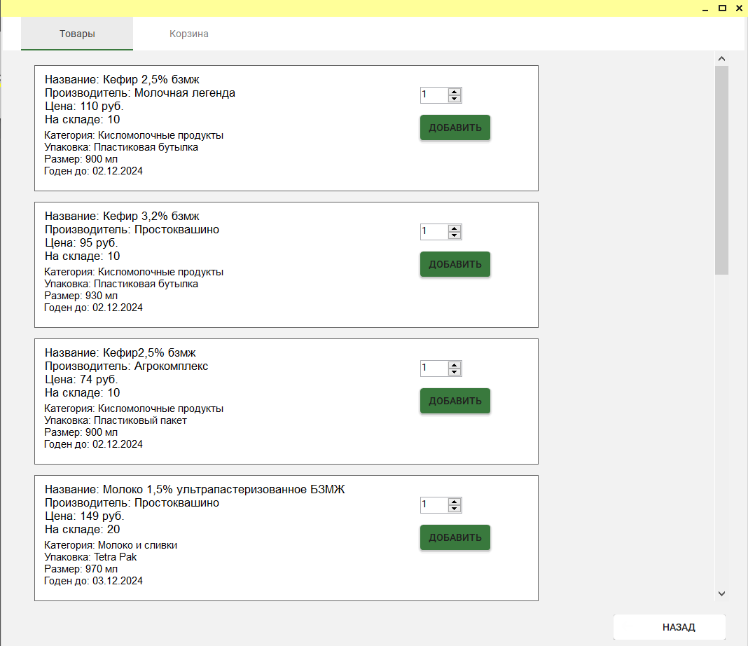


Рисунок 29 – Список товаров

Листинг кода списка товаров:

using System;

using System.Data;

using System.Windows.Forms;

using Npgsql;

namespace YCHEBKA4

{

public partial class Form3 : Form

{

// Объявление соединения с базой данных PostgreSQL

private NpgsqlConnection connection;

// Конструкторформы

public Form3()

{

InitializeComponent();

// Строка подключения к базе данных

string connString = "Host=localhost;Username=postgres;Password=18273645;Database=Ferma";

connection = new NpgsqlConnection(connString);

// Обработкасобытиязакрытияформы

this.FormClosing += new FormClosingEventHandler(FormClosingevent);

}

// Событиезагрузкиформы

private void Form3\_Load(object sender, EventArgs e)

{

// Открытие соединения с базой данных

connection.Open();

// Загрузка данных о товарах с истекающим сроком годности

LoadProducts();

}

// Метод загрузки информации о товарах, срок годности которых истекает в ближайшие 3 дня

private void LoadProducts()

{

// SQL-запрос для выборки нужных данных

string query = @"

SELECT

t.название,

c.название AS категория,

t.срок\_годности,

EXTRACT(DAY FROM (t.срок\_годности::timestamp - CURRENT\_DATE::timestamp))::int AS дни\_до\_истечения

FROM Товары t

JOIN Категории c ON t.категория\_id = c.id

WHERE t.срок\_годности> CURRENT\_DATE

AND t.срок\_годности<= CURRENT\_DATE + INTERVAL '3 days'";

// Выполнениезапросаичтениерезультатов

using (NpgsqlCommand command = new NpgsqlCommand(query, connection))

using (NpgsqlDataReader reader = command.ExecuteReader())

{

// Очистка списка перед добавлением новых данных

listBox1.Items.Clear();

// Обработка каждой строки результата запроса

while (reader.Read())

{

// Получение значений из результата запроса

string name = reader.GetString(0);

string category = reader.GetString(1);

DateTime expiry = reader.GetDateTime(2);

int daysLeft = reader.GetInt32(3);

// Расчет скидки в зависимости от количества оставшихся дней

double discount = CalculateDiscount(daysLeft);

// Добавление информации о товаре в список

listBox1.Items.Add($"Наименование товара: {name}");

listBox1.Items.Add($"Категория товара: {category}");

listBox1.Items.Add($"Срок годности до: {expiry.ToShortDateString()}");

listBox1.Items.Add($"Количество дней до окончания срока годности: {daysLeft} дней\t\t\t\t{discount}% скидки");

listBox1.Items.Add(""); // Пустая строка для разделения

}

// Если подходящих товаров не найдено

if (listBox1.Items.Count == 0)

{

listBox1.Items.Add("Нет товаров, срок годности которых истекает в ближайшие 3 дня.");

}

}

}

// Метод для расчета скидки в зависимости от количества оставшихся дней до окончания срока годности

private double CalculateDiscount(int daysLeft)

{

if (daysLeft <= 1)

return 20; // 20% скидка, если остался 1 день или меньше

else if (daysLeft <= 3)

return 10; // 10% скидка, если осталось до 3 дней

return 0; // Без скидки

}

// Обработка кнопки "Назад"

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

{

// Скрытие текущей формы и отображение Form2

this.Hide();

Form2 form2 = new Form2();

form2.Show();

}

// Обработка события закрытия формы — завершение приложения

private void FormClosingevent(object sender, FormClosingEventArgs e)

{

Application.Exit();

}

// Обработчик нажатия на изображение (можно использовать для переходов или информации)

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

{

}

// Обработчик изменения выбранного элемента в списке

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

}

На рисунке 30показана форма покупки:

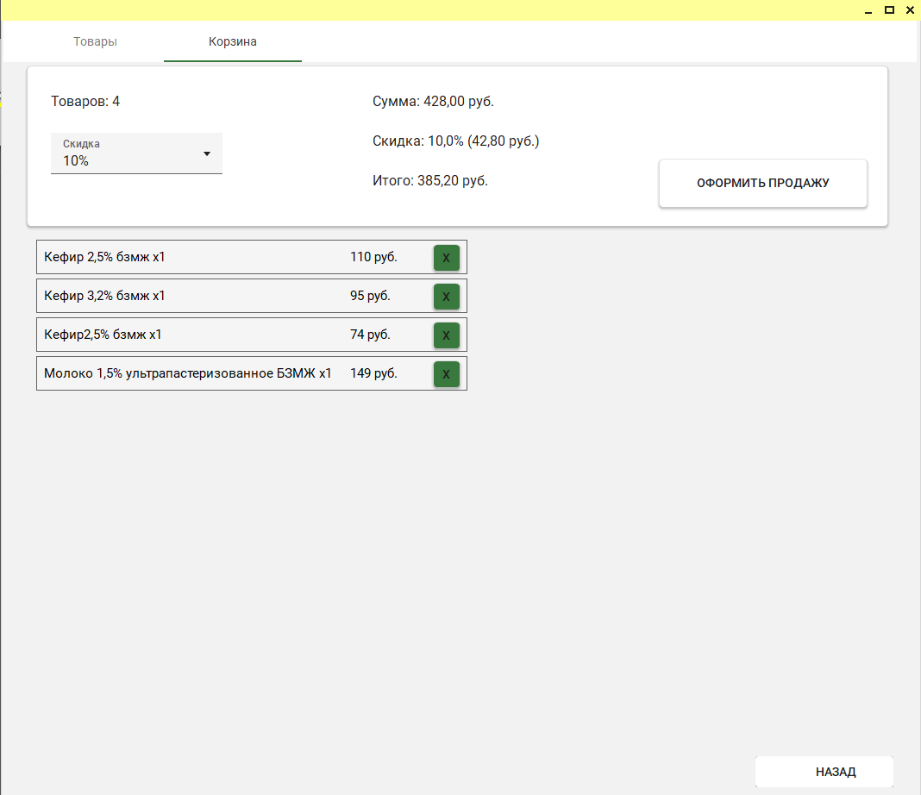


Рисунок 30 – Форма покупки

Листинг кода формы покупки:

using System;

using System.Collections.Generic;

using System.Data;

using System.Windows.Forms;

using Npgsql;

namespace YCHEBKA4

{

public partial class Form4 : Form

{

// Соединение с базой данных

private NpgsqlConnection conn;

// Общая сумма покупки

private decimal totalAmount = 0;

// Списоктовароввкорзине

private List<CartItem> cartItems = new List<CartItem>();

public Form4()

{

InitializeComponent();

// Обработчиксобытиязакрытияформы

this.FormClosing += new FormClosingEventHandler(FormClosingevent);

// Инициализациясоединениясбазойданных

conn = new NpgsqlConnection("Host=localhost;Username=postgres;Password=18273645;Database=Ferma");

// Привязкаобработчиковсобытий

comboBoxCategories.SelectedIndexChanged += ComboBoxCategories\_SelectedIndexChanged;

buttonAddToCart.Click += ButtonAddToCart\_Click;

buttonRemoveFromCart.Click += ButtonRemoveFromCart\_Click;

buttonCheckout.Click += ButtonCheckout\_Click;

// Загрузкакатегорийтоваров

LoadCategories();

}

// Обработка закрытия формы — выход из приложения

private void FormClosingevent(object sender, FormClosingEventArgs e)

{

Application.Exit();

}

// Загрузка категорий из базы данных в выпадающий список

private void LoadCategories()

{

conn.Open();

NpgsqlCommand cmd = new NpgsqlCommand("SELECT id, название FROM Категории", conn);

NpgsqlDataReader reader = cmd.ExecuteReader();

Dictionary<int, string> dict = new Dictionary<int, string>();

while (reader.Read())

{

dict.Add(reader.GetInt32(0), reader.GetString(1));

}

conn.Close();

comboBoxCategories.DataSource = new BindingSource(dict, null);

comboBoxCategories.DisplayMember = "Value";

comboBoxCategories.ValueMember = "Key";

}

// Обработка выбора категории — загрузка товаров данной категории

private void ComboBoxCategories\_SelectedIndexChanged(object sender, EventArgs e)

{

int categoryId = ((KeyValuePair<int, string>)comboBoxCategories.SelectedItem).Key;

listBoxProducts.Items.Clear();

conn.Open();

string query = @"

SELECT id, название, срок\_годности, цена

FROM Товары

WHERE категория\_id = @cat AND срок\_годности> CURRENT\_DATE";

NpgsqlCommand cmd = new NpgsqlCommand(query, conn);

cmd.Parameters.AddWithValue("cat", categoryId);

NpgsqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

int id = reader.GetInt32(0);

string name = reader.GetString(1);

DateTime expiry = reader.GetDateTime(2);

decimal price = reader.GetDecimal(3);

int daysLeft = (expiry - DateTime.Today).Days;

double discount = CalculateDiscount(daysLeft);

decimal finalPrice = price \* (decimal)(1 - discount / 100);

listBoxProducts.Items.Add(id + "|" + name + "|" + finalPrice.ToString("0.00") + "₽|Скидка " + discount + "%");

}

conn.Close();

}

// Добавление товара в корзину

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

{

if (listBoxProducts.SelectedItem == null) return;

string item = listBoxProducts.SelectedItem.ToString();

string[] parts = item.Split('|');

int productId = int.Parse(parts[0]);

string name = parts[1];

decimal price = decimal.Parse(parts[2].Replace("₽", ""));

cartItems.Add(new CartItem(productId, name, price, 1));

listBoxCart.Items.Add(name + " - " + price.ToString("0.00") + "₽");

totalAmount += price;

labelTotal.Text = "Итого: " + totalAmount.ToString("0.00") + "₽";

}

// Удалениевыбранноготовараизкорзины

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

{

int index = listBoxCart.SelectedIndex;

if (index >= 0)

{

totalAmount -= cartItems[index].price;

cartItems.RemoveAt(index);

listBoxCart.Items.RemoveAt(index);

labelTotal.Text = "Итого: " + totalAmount.ToString("0.00") + "₽";

}

}

// Оформлениепокупки

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

{

if (cartItems.Count == 0)

{

MessageBox.Show("Корзинапуста!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

conn.Open();

NpgsqlTransaction tran = conn.BeginTransaction();

try

{

// Генерация случайного номера чека

Random rnd = new Random();

long receiptNumber = rnd.Next(100000000, 999999999);

// Подсчет суммы с учетом скидок

decimal total = 0;

foreach (CartItem item in cartItems)

{

int daysLeft = GetDaysLeft(item.productId);

double discount = CalculateDiscount(daysLeft);

decimal finalPrice = item.price \* (decimal)(1 - discount / 100);

total += finalPrice \* item.quantity;

}

// Вставка записи о продаже

NpgsqlCommand cmdSale = new NpgsqlCommand("INSERT INTO Продажи(чек, дата, сумма) VALUES(@ch, @d, @s) RETURNING id", conn);

cmdSale.Parameters.AddWithValue("ch", receiptNumber);

cmdSale.Parameters.AddWithValue("d", DateTime.Today);

cmdSale.Parameters.AddWithValue("s", total);

int saleId = Convert.ToInt32(cmdSale.ExecuteScalar());

// Вставкапроданныхтоваров

foreach (CartItem item in cartItems)

{

NpgsqlCommand cmdSold = new NpgsqlCommand("INSERT INTO ПроданныеТовары(продажа\_id, товар\_id, количество) VALUES(@pid, @tid, @qty)", conn);

cmdSold.Parameters.AddWithValue("pid", saleId);

cmdSold.Parameters.AddWithValue("tid", item.productId);

cmdSold.Parameters.AddWithValue("qty", item.quantity);

cmdSold.ExecuteNonQuery();

}

// Подтверждениетранзакции

tran.Commit();

MessageBox.Show("Покупкаоформлена!", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

// Очисткакорзины

cartItems.Clear();

listBoxCart.Items.Clear();

totalAmount = 0;

labelTotal.Text = "Итого: 0₽";

}

catch (Exception ex)

{

// Откат транзакции при ошибке

tran.Rollback();

MessageBox.Show("Ошибка при покупке: " + ex.Message);

}

conn.Close();

}

// Получение количества дней до истечения срока годности товара

private int GetDaysLeft(int productId)

{

int days = 365; // значениепоумолчанию

NpgsqlCommand cmd = new NpgsqlCommand("SELECT срок\_годности FROM Товары WHERE id = @id", conn);

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

object result = cmd.ExecuteScalar();

if (result != null && result != DBNull.Value)

{

DateTime expiry = (DateTime)result;

days = (expiry - DateTime.Today).Days;

}

return days;

}

// Расчет скидки в зависимости от оставшихся дней

private double CalculateDiscount(int daysLeft)

{

if (daysLeft <= 1)

return 20;

else if (daysLeft <= 3)

return 10;

return 0;

}

// Класс для представления товара в корзине

private class CartItem

{

public int productId;

public string name;

public decimal price;

public int quantity;

public CartItem(int productId, string name, decimal price, int quantity)

{

this.productId = productId;

this.name = name;

this.price = price;

this.quantity = quantity;

}

}

// Кнопкавозвратакпредыдущейформе

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

{

this.Hide();

Form2 form2 = new Form2();

form2.Show();

}

// Дополнительный обработчик кнопки добавления (не используется)

private void buttonAddToCart\_Click\_1(object sender, EventArgs e)

{

}

}

}

Работа № 5

Задание: Разработать подсистему для работы с клиентами агентства недвижимости «Терем», обеспечивающую следующий функционал:

– оформление заявки на приобретение/продажу недвижимости;

– добавление/редактирование данных о недвижимости;

– просмотр недвижимости по условию.

Получить ER-диаграмму средствами СУБД: ER-диаграмма должна быть представлена в формате PDF и содержать таблицы, связи между ними, атрибуты и ключи (типами данных на данном этапе можно пренебречь).

Разработать программный модуль для учета недвижимости. Необходимо реализовать вывод списка объектов, информация о которых хранятся в базе данных, согласно предоставленному макету:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | |  |
|  | Тип жилья  Район  Количество комнат  Площадь  Год постройки  Адрес  Цена  Статус |  | Тип жилья  Район  Количество комнат  Площадь  Год постройки  Адрес  Цена  Статус |  |
|  |  | | |  |
|  | Тип жилья  Район  Количество комнат  Площадь  Год постройки  Адрес  Цена  Статус |  | Тип жилья  Район  Количество комнат  Площадь  Год постройки  Адрес  Цена  Статус |  |
|  |  | | |  |

Необходимо осуществить оформление заявки с возможностью добавления клиента в окне оформления заявки.

Разработать интерфейс программного модуля для работы с товарами и продажами.

Реализовать функции добавления/редактирования данных недвижимости в новом окне (странице) – форме для добавления/редактирования недвижимости. Переходы на эту форму должны быть реализованы из формы списка объектов: для редактирования товаров из списка – при нажатии на конкретный элемент, для редактирования/добавления любого объекта – при нажатии кнопки.

На форме для добавления/редактирования объекта недвижимости должны быть предусмотрены следующие поля: Тип недвижимости, Район, Адрес, Площадь, Количество комнат, Год постройки, Цена, Статус.

**Ход работы**

Подготовим данные для импорта. Для этого приведём все данные в предоставленных для импорта таблицах к 3НФ (Рисунок 31).



Рисунок 31 – Подготовка данных для импорта

Теперь необходимо создать базу данных (Рисунок 32)

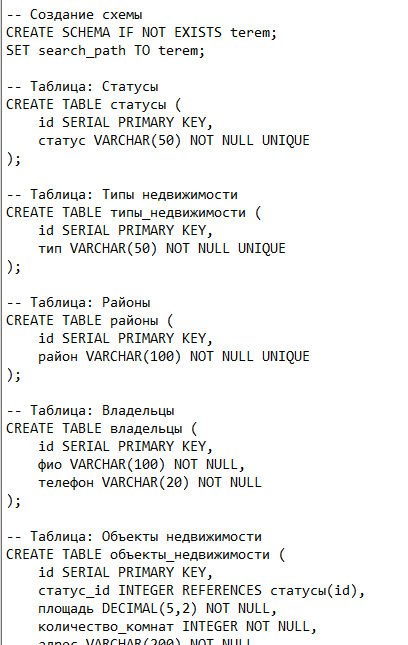


Рисунок 32 – Скрипт для создания БД

Теперь сделаем импорт данных (Рисунок 33)

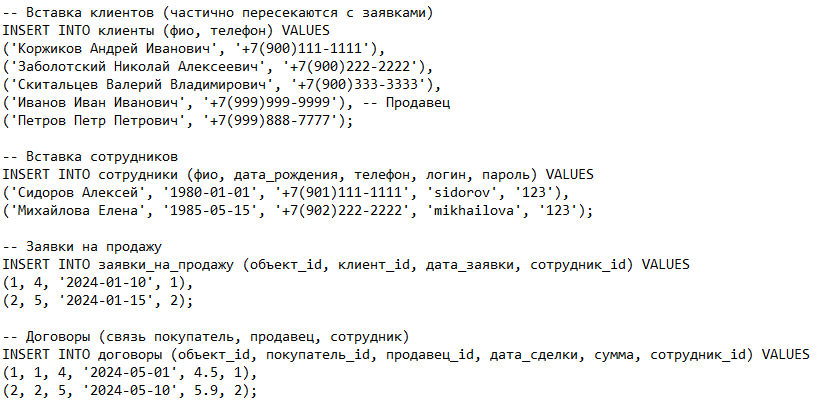


Рисунок 33 – Импорт данных в БД

Листинг кода формы авторизации:

;// Подключение необходимых пространств имён

usingSystem;

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;

// ПодключениебиблиотекидляработысPostgreSQL

usingNpgsql;

namespaceYCHEBKA5

{

// Основнойклассформыавторизации

public partial class Form1 : Form

{

// Объектсоединениясбазойданных

private NpgsqlConnection connection;

// Строкаподключениякбазеданных PostgreSQL

private string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=Terem;Search Path=terem";

// Конструктор формы

public Form1()

{

InitializeComponent(); // Инициализация компонентов формы

// Создание и открытие подключения к базе данных при запуске формы

NpgsqlConnection connection = new NpgsqlConnection(connectionString);

connection.Open(); // ВАЖНО: Это локальная переменная, которая перекрывает поле. Лучше использовать this.connection

}

// Обработчик события нажатия на кнопку "Войти"

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

{

// Получение введённых пользователем логина и пароля

string username = textBox1.Text;

string password = textBox2.Text;

// Проверка данных пользователя

if (AuthenticateUser(username, password))

{

// Если пользователь найден — сообщение об успехе и переход к следующей форме

MessageBox.Show("Успешная аутентификация! Добро пожаловать, " + username + "!");

Form f2 = new Form2(); // Создание экземпляра следующей формы

this.Hide(); // Скрытие текущей формы

f2.Show(); // Отображение новой формы

}

else

{

// Ошибка аутентификации — вывод сообщения

MessageBox.Show("Ошибка аутентификации. Пользователь с такими данными не найден.");

}

}

// Метод для проверки существования пользователя в базе данных

private bool AuthenticateUser(string username, string password)

{

// Повторное объявление строки подключения (можно убрать и использовать поле класса)

string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=Terem;Search Path=terem";

// Создание и открытие подключения с использованием using (гарантирует автоматическое закрытие)

using (NpgsqlConnection connection = new NpgsqlConnection(connectionString))

{

connection.Open();

// SQL-запрос с параметрами для безопасного поиска пользователя

string query = "SELECT COUNT(\*) FROM db\_users WHERE user\_login = @Username AND user\_password = @Password";

using (NpgsqlCommand command = new NpgsqlCommand(query, connection))

{

// Передача параметров в запрос

command.Parameters.AddWithValue("@Username", username);

command.Parameters.AddWithValue("@Password", password);

// Выполнение запроса и получение количества совпадений

int count = Convert.ToInt32(command.ExecuteScalar());

// Если есть хотя бы одно совпадение — возвращаем true

return count > 0;

}

}

}

}

}

На рисунке 34показана работа приведённого выше кода:

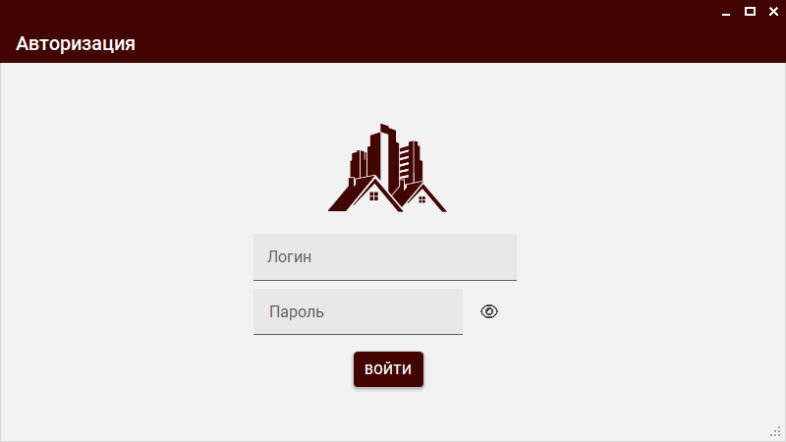


Рисунок 34 – Работа формы авторизации

Ниже листинг кода основной формы:

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;

usingNpgsql; // Подключение библиотеки для работы с PostgreSQL

using System.Windows.Forms;

namespace YCHEBKA5

{

// Класс для работы с формой, где отображаются данные и выполняются операции

public partial class Form2 : Form

{

NpgsqlConnectionconn; // Объект для подключения к базе данных

NpgsqlDataAdapteradapter; // Адаптер данных для работы с базой

DataTabledt; // Таблица, в которой будут храниться данные

Dictionary<string, Dictionary<object, string>> lookupData = new Dictionary<string, Dictionary<object, string>>(); // Словарь для хранения связанных данных (например, для внешних ключей)

// Конструктор формы

publicForm2()

{

InitializeComponent(); // Инициализация компонентов формы

this.FormClosing += FormClosingevent; // Подключение обработчика события закрытия формы

// Строка подключения к базе данных PostgreSQL с указанием схемы 'terem'

string connection = "Host=localhost;Username=postgres;Password=18273645;Database=Terem;Search Path=terem";

// Создание соединения с базой данных

conn = new NpgsqlConnection(connection);

// Заполнение выпадающего списка доступными таблицами

comboBox1.Items.AddRange(newobject[] {

"объекты\_недвижимости", "заявки\_на\_покупку", "владельцы", "районы",

"типы\_недвижимости", "статусы", "заявки\_на\_продажу", "договоры", "клиенты", "сотрудники"

});

// Подключение обработчика для события изменения выбора в комбобоксе

comboBox1.SelectedIndexChanged += comboBox1\_SelectedIndexChanged;

}

// Обработчик изменения выбора таблицы из выпадающего списка

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

string table = comboBox1.Text; // Получение выбранной таблицы

LoadLookupTables(); // Загрузка данных для внешних ключей

stringquery = GetSelectQuery(table); // Получаем SQL запрос для выбора всех данных из выбранной таблицы

adapter = newNpgsqlDataAdapter(query, conn); // Создаём адаптер для выполнения запроса

NpgsqlCommandBuilderbuilder = newNpgsqlCommandBuilder(adapter); // Строим команды для обновления, вставки и удаления

dt = newDataTable(); // Создаём новый DataTable для хранения данных

adapter.Fill(dt); // Заполняем DataTable результатами запроса

dataGridView1.Columns.Clear(); // Очищаем старые столбцы в DataGridView

dataGridView1.DataSource = dt; // Привязываем новые данные к DataGridView

HidePrimaryKey(); // Прячем столбцы с первичными ключами

ReplaceForeignKeyColumns(table); // Заменяем внешние ключи на комбинированные списки

dataGridView1.DefaultValuesNeeded -= dataGridView1\_DefaultValuesNeeded; // Убираем старый обработчик событий

dataGridView1.DefaultValuesNeeded += dataGridView1\_DefaultValuesNeeded; // Добавляем новый обработчик

}

// Обработчик кнопки сохранения данных

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

{

try

{

// Создаём команды для обновления, вставки и удаления

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

adapter.UpdateCommand = builder.GetUpdateCommand();

adapter.InsertCommand = builder.GetInsertCommand();

adapter.DeleteCommand = builder.GetDeleteCommand();

// Применяем изменения к базе данных

adapter.Update(dt);

MessageBox.Show("Изменения сохранены."); // Информируем пользователя о успешном сохранении

}

catch (Exceptionex)

{

MessageBox.Show("Ошибка при сохранении: " + ex.Message); // В случае ошибки показываем сообщение

}

}

// Обработчик кнопки удаления записи

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

{

if (dataGridView1.CurrentRow != null) // Проверка, выбрана ли строка для удаления

{

try

{

// Удаляем выбранную строку

dataGridView1.Rows.RemoveAt(dataGridView1.CurrentRow.Index);

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

adapter.Update(dt); // Применяем изменения в базе данных

MessageBox.Show("Запись удалена."); // Информируем пользователя об успешном удалении

}

catch (Exceptionex)

{

MessageBox.Show("Ошибка при удалении: " + ex.Message); // В случае ошибки показываем сообщение

}

}

else

{

MessageBox.Show("Выберите строку для удаления."); // Если строка не выбрана, показываем сообщение

}

}

// Метод для получения SQL запроса для выборки всех данных из таблицы

private string GetSelectQuery(string table)

{

return $"SELECT \* FROM \"{table}\""; // Формируем запрос для получения всех строк из выбранной таблицы

}

// Метод для скрытия столбцов с первичными ключами

privatevoidHidePrimaryKey()

{

string[] hiddenKeys = newstring[] { "id" }; // Перечень столбцов для скрытия (например, первичные ключи)

foreach (string key in hiddenKeys)

{

if (dataGridView1.Columns.Contains(key)) // Если столбец существует

dataGridView1.Columns[key].Visible = false; // Скрываем столбец

}

}

// Метод для загрузки данных внешних ключей для всех таблиц

privatevoidLoadLookupTables()

{

try

{

lookupData["статусы"] = LoadLookup("статусы", "id", "статус");

lookupData["районы"] = LoadLookup("районы", "id", "район");

lookupData["типы\_недвижимости"] = LoadLookup("типы\_недвижимости", "id", "тип");

lookupData["владельцы"] = LoadLookup("владельцы", "id", "фио");

lookupData["клиенты"] = LoadLookup("клиенты", "id", "фио");

lookupData["сотрудники"] = LoadLookup("сотрудники", "id", "фио");

lookupData["объекты\_недвижимости"] = LoadLookup("объекты\_недвижимости", "id", "адрес");

}

catch (Exceptionex)

{

MessageBox.Show("Ошибка при загрузке справочников: " + ex.Message);

}

}

// Метод для загрузки данных для внешнего ключа

private Dictionary<object, string> LoadLookup(string table, string keyCol, string valCol)

{

Dictionary<object, string> dict = new Dictionary<object, string>(); // Словарь для хранения данных

string query = $"SELECT \"{keyCol}\", \"{valCol}\" FROM \"{table}\""; // Формируем запрос для выборки данных

using (var cmd = new NpgsqlCommand(query, conn)) // Создаём команду для выполнения запроса

{

conn.Open(); // Открываем соединение с базой данных

using (var reader = cmd.ExecuteReader()) // Выполняем чтение данных

{

while (reader.Read()) // Читаем строки из результата

{

dict.Add(reader[0], reader[1].ToString()); // Добавляем данные в словарь

}

}

conn.Close(); // Закрываем соединение

}

returndict; // Возвращаем словарь с данными внешнего ключа

}

// Метод для замены внешних ключей на комбинированные списки

private void ReplaceForeignKeyColumns(string table)

{

Dictionary<string, string> fkMap = new Dictionary<string, string>();

if (table == "объекты\_недвижимости")

{

fkMap.Add("статус\_id", "статусы");

fkMap.Add("район\_id", "районы");

fkMap.Add("тип\_id", "типы\_недвижимости");

fkMap.Add("владелец\_id", "владельцы");

}

elseif (table == "заявки\_на\_покупку")

{

fkMap.Add("район\_id", "районы");

fkMap.Add("тип\_id", "типы\_недвижимости");

}

elseif (table == "заявки\_на\_продажу")

{

fkMap.Add("объект\_id", "объекты\_недвижимости");

fkMap.Add("клиент\_id", "клиенты");

fkMap.Add("сотрудник\_id", "сотрудники");

}

elseif (table == "договоры")

{

fkMap.Add("объект\_id", "объекты\_недвижимости");

fkMap.Add("покупатель\_id", "клиенты");

fkMap.Add("продавец\_id", "клиенты");

fkMap.Add("сотрудник\_id", "сотрудники");

}

foreach (KeyValuePair<string, string> kvp in fkMap)

{

if (!dt.Columns.Contains(kvp.Key)) continue;

DataGridViewComboBoxColumn combo = new DataGridViewComboBoxColumn();

combo.Name = kvp.Key;

combo.DataPropertyName = kvp.Key;

combo.HeaderText = kvp.Key;

combo.DataSource = new BindingSource(lookupData[kvp.Value], null);

combo.DisplayMember = "Value";

combo.ValueMember = "Key";

int index = dataGridView1.Columns[kvp.Key].Index;

dataGridView1.Columns.Remove(kvp.Key);

dataGridView1.Columns.Insert(index, combo);

}

}

// Метод для установки значений по умолчанию для новых строк

private void dataGridView1\_DefaultValuesNeeded(object sender, DataGridViewRowEventArgs e)

{

string table = comboBox1.Text;

if (table == "объекты\_недвижимости")

{

e.Row.Cells["дата\_регистрации"].Value = DateTime.Today;

if (lookupData["статусы"].Count > 0)

e.Row.Cells["статус\_id"].Value = lookupData["статусы"].First().Key;

if (lookupData["районы"].Count > 0)

e.Row.Cells["район\_id"].Value = lookupData["районы"].First().Key;

if (lookupData["типы\_недвижимости"].Count > 0)

e.Row.Cells["тип\_id"].Value = lookupData["типы\_недвижимости"].First().Key;

if (lookupData["владельцы"].Count > 0)

e.Row.Cells["владелец\_id"].Value = lookupData["владельцы"].First().Key;

}

elseif (table == "заявки\_на\_продажу")

{

e.Row.Cells["дата\_заявки"].Value = DateTime.Today;

}

elseif (table == "договоры")

{

e.Row.Cells["дата\_сделки"].Value = DateTime.Today;

}

}

// Обработчик кнопки перехода к другой форме

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

{

this.Hide(); // Скрываем текущую форму

Form3 form3 = new Form3(); // Создаём экземпляр новой формы

form3.Show(); // Отображаем новую форму

}

// Обработчик события закрытия формы

private void FormClosingevent(object sender, FormClosingEventArgs e)

{

Application.Exit(); // Закрываем приложение

}

}

}

Нарисунках 35, 36результатработыданногокода:

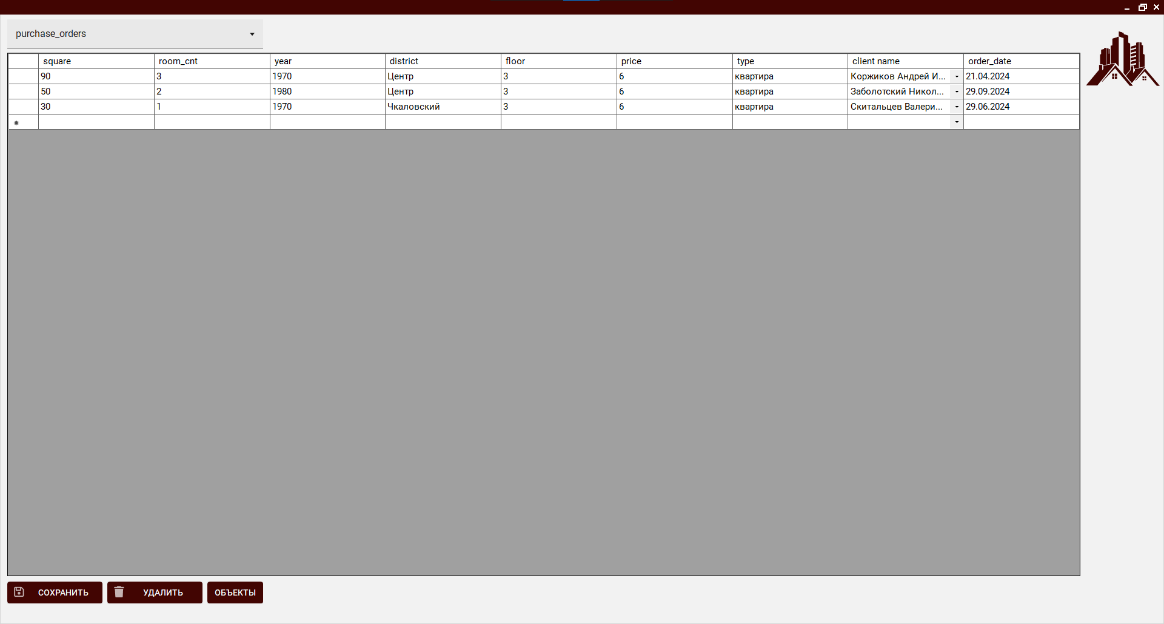


Рисунок 35 – Результат работы приложения

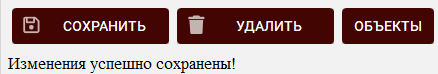


Рисунок 36 – Результат работы приложения

Листинг кода списка:

using System;

using System.Collections.Generic;

using System.Data;

using System.Drawing;

using System.Windows.Forms;

using Npgsql;

namespace YCHEBKA5

{

public partial class Form3 : Form

{

// Строкаподключениякбазеданных

private string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=Terem;Search Path=terem";

// Конструкторформы

public Form3()

{

InitializeComponent();

LoadData(); // Загружаемданныеизбазы

this.FormClosing += FormClosingevent; // Обработчиксобытиязакрытияформы

}

// Метод для загрузки данных об объектах недвижимости из базы

private void LoadData()

{

string query = @"

SELECT

o.площадь,

o.количество\_комнат,

o.год\_постройки,

o.адрес,

o.цена,

st.статус,

t.тип,

r.район

FROM объекты\_недвижимости o

LEFT JOIN статусы st ON o.статус\_id = st.id

LEFT JOIN типы\_недвижимости t ON o.тип\_id = t.id

LEFT JOIN районы r ON o.район\_id = r.id

";

// Открываем подключение к базе данных и выполняем запрос

using (var conn = new NpgsqlConnection(connectionString))

using (var cmd = new NpgsqlCommand(query, conn))

{

conn.Open();

using (var reader = cmd.ExecuteReader())

{

// Для каждой строки результата запроса создаем карточку объекта недвижимости

while (reader.Read())

{

AddPropertyCard(

reader["тип"].ToString(),

reader["район"].ToString(),

reader["количество\_комнат"].ToString(),

reader["площадь"].ToString(),

reader["год\_постройки"].ToString(),

reader["адрес"].ToString(),

reader["цена"].ToString(),

reader["статус"].ToString()

);

}

}

}

}

// Метод для создания карточки объекта недвижимости на форме

private void AddPropertyCard(string тип, string район, string комнаты, string площадь, string год, string адрес, string цена, string статус)

{

// Создаем панель для карточки объекта

Panel card = new Panel

{

Width = 300,

Height = 260,

Margin = new Padding(10),

BorderStyle = BorderStyle.FixedSingle,

BackColor = Color.White

};

// Список меток для отображения данных

var labels = new List<Label>

{

CreateLabel("Тип жилья: " + тип),

CreateLabel("Район: " + район),

CreateLabel("Комнат: " + комнаты),

CreateLabel("Площадь: " + площадь + " м²"),

CreateLabel("Год: " + год),

CreateLabel("Адрес: " + адрес),

CreateLabel("Цена: " + цена + " млн ₽"),

CreateLabel("Статус: " + статус)

};

// Размещаем метки на панели

int y = 5;

foreach (var lbl in labels)

{

lbl.Location = new Point(5, y);

card.Controls.Add(lbl);

y += lbl.Height + 3;

}

// Кнопка "Оформитьзаявку"

Button btnOrder = new Button

{

Text = "Оформитьзаявку",

Location = new Point(5, y + 5),

Size = new Size(130, 25)

};

// КомбоБокс с клиентами (по умолчанию скрыт)

ComboBox comboClients = new ComboBox

{

Location = new Point(5, y + 35),

Size = new Size(200, 25),

Visible = false

};

// Кнопка подтверждения заявки (по умолчанию скрыта)

Button btnConfirm = new Button

{

Text = "Подтвердить",

Location = new Point(210, y + 35),

Size = new Size(85, 25),

Visible = false

};

// Обработчик клика по кнопке "Оформить заявку"

btnOrder.Click += (s, e) =>

{

comboClients.Visible = true; // Показываем выпадающий список клиентов

btnConfirm.Visible = true; // Показываем кнопку "Подтвердить"

comboClients.Items.Clear(); // Очищаем список клиентов

using (var conn = new NpgsqlConnection(connectionString))

{

conn.Open();

using (var cmd = new NpgsqlCommand("SELECT фио FROM владельцы", conn))

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

comboClients.Items.Add(reader.GetString(0)); // Добавляемклиентоввсписок

}

}

}

if (comboClients.Items.Count > 0)

comboClients.SelectedIndex = 0; // Выбираемпервогоклиента

};

// Обработчик клика по кнопке "Подтвердить"

btnConfirm.Click += (s, e) =>

{

// Проверяем, что выбран клиент

if (comboClients.SelectedItem == null)

{

MessageBox.Show("Выберитеклиента");

return;

}

// Вставляем заявку на покупку в базу данных

using (var conn = new NpgsqlConnection(connectionString))

{

conn.Open();

var cmd = new NpgsqlCommand(@"

INSERT INTO заявки\_на\_покупку

(площадь, количество\_комнат, год\_постройки, район\_id, этаж, цена, тип\_id, клиент, дата\_заявки)

VALUES

(@площадь, @комнаты, @год,

(SELECT id FROM районы WHERE район = @район),

1, @цена,

(SELECT id FROM типы\_недвижимости WHERE тип = @тип),

@клиент, CURRENT\_DATE)", conn);

// Заполняем параметры для запроса

cmd.Parameters.AddWithValue("площадь", Convert.ToDecimal(площадь));

cmd.Parameters.AddWithValue("комнаты", Convert.ToInt32(комнаты));

cmd.Parameters.AddWithValue("год", Convert.ToInt32(год));

cmd.Parameters.AddWithValue("район", район);

cmd.Parameters.AddWithValue("цена", Convert.ToDecimal(цена));

cmd.Parameters.AddWithValue("тип", тип);

cmd.Parameters.AddWithValue("клиент", comboClients.SelectedItem.ToString());

cmd.ExecuteNonQuery(); // Выполняем запрос

}

MessageBox.Show("Заявка оформлена!"); // Уведомляем пользователя о успешной заявке

flowLayoutPanel1.Controls.Remove(card); // Удаляемкарточкусформы

card.Dispose(); // Освобождаем ресурсы

// Обновляем статус объекта в базе данных

using (var updateConn = new NpgsqlConnection(connectionString))

{

updateConn.Open();

var updateCmd = new NpgsqlCommand(@"

UPDATE объекты\_недвижимости

SET статус\_id = (SELECT id FROM статусы WHERE статус = 'наоформлении')

WHERE адрес = @адрес", updateConn);

updateCmd.Parameters.AddWithValue("адрес", адрес);

updateCmd.ExecuteNonQuery(); // Обновляем статус

}

};

// Добавляем элементы на карточку

card.Controls.Add(btnOrder);

card.Controls.Add(comboClients);

card.Controls.Add(btnConfirm);

// Добавляем карточку на форму

flowLayoutPanel1.Controls.Add(card);

}

// Метод для создания метки

private Label CreateLabel(string text)

{

return new Label

{

Text = text,

AutoSize = true,

Font = new Font("Segoe UI", 9, FontStyle.Regular),

ForeColor = Color.Black

};

}

// Обработчиксобытиякнопки "Назад"

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

{

this.Hide();

Form2 form2 = new Form2();

form2.Show();

}

// Обработчик события закрытия формы

private void FormClosingevent(object sender, FormClosingEventArgs e)

{

Application.Exit(); // Завершаем приложение

}

}

}

На рисунке 37список недвижимости:

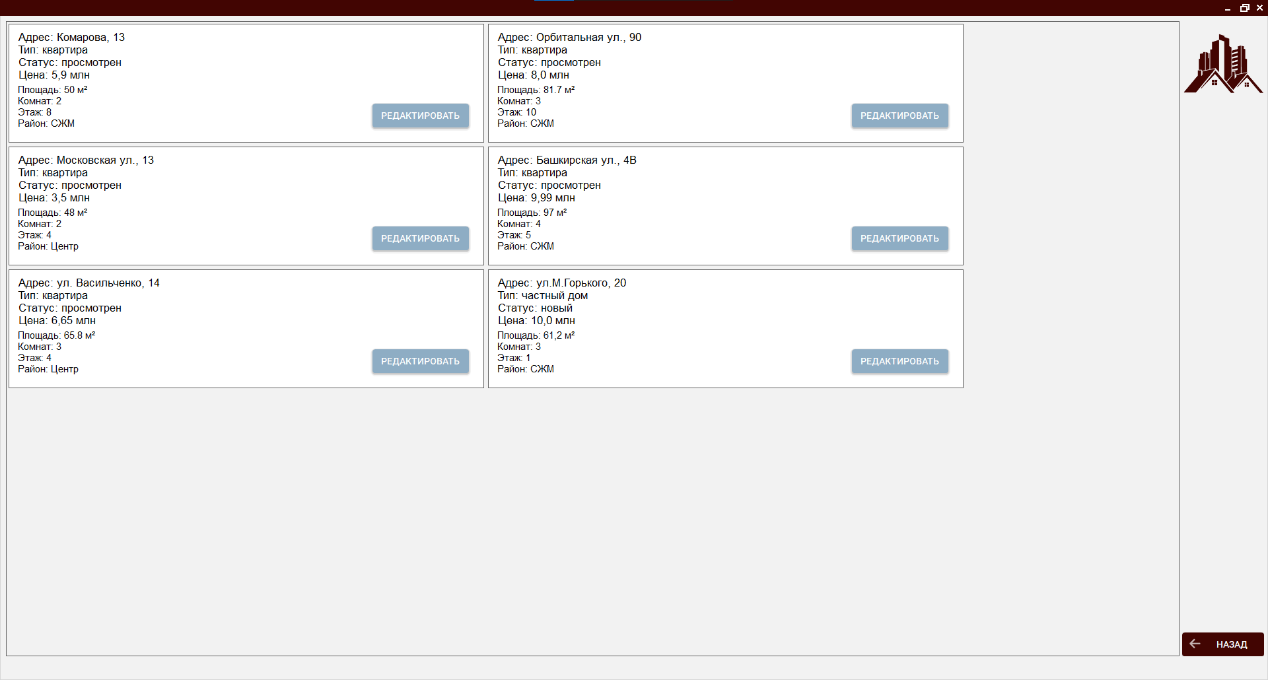


Рисунок 37 – Список недвижимости

Работа № 6

Задание: Разработать подсистему для работы с партнерами компании, обеспечивающую следующий функционал:

– просмотр списка партнеров;

– добавление/редактирование данных о партнере;

– просмотр истории реализации продукции партнером.

Получить ER-диаграмму средствами СУБД: ER-диаграмма должна быть представлена в формате PDF и содержать таблицы, связи между ними, атрибуты и ключи (типами данных на данном этапе можно пренебречь).

Разработать программный модуль для учета партнеров. Необходимо реализовать вывод списка партнеров, информация о которых хранятся в базе данных, согласно предоставленному макету:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | Тип/Наименование партнера хх %  ФИО контактного лица  Телефон  Рейтинг |  |
|  |  |  |
|  | Тип/Наименование партнера хх %  ФИО контактного лица  Телефон  Рейтинг |  |
|  |  |  |
|  | Тип/Наименование партнера хх %  ФИО контактного лица  Телефон  Рейтинг |  |
|  |  |  |

Величина скидки для партнера рассчитывается на основании продажи продукции за весь период работы. Скидка зависит от общего количества приобретенной партнером продукции и составляет: до 10000 – 0%, от 10000 – до 50000 – 5%, от 50000 – до 300000 – 10%, более 300000 – 15%.

Выполнить интеграцию модуля вывода истории реализации продукции партнером: обеспечить соответствие стилю приложения, единый для системы согласованный внешний вид. Реализовать переход на данное окно (страницу), получить историю реализации продукции для конкретного партнера.

Разработать интерфейс программного модуля для работы с партнерами.

Реализовать функции добавления/редактирования данных партнера в новом окне (странице) – форме для добавления/редактирования партнера. Переходы на эту форму должны быть реализованы из главной формы списка партнеров: для редактирования – при нажатии на конкретный элемент, для добавления – при нажатии кнопки.

На форме для добавления/редактирования партнера должны быть предусмотрены следующие поля: Наименование, Тип партнера (выпадающий список), Рейтинг, Адрес, ФИО контактного лица, Телефон и Email компании.

Рейтинг партнера должен быть целыми неотрицательным числом.

**Ход работы**

Подготовим данные для импорта. Для этого приведём все данные в предоставленных для импорта таблицах к 3НФ (Рисунок 38).

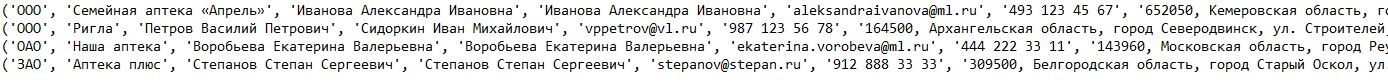


Рисунок 38 – Подготовка данных для импорта

Теперь необходимо создать базу данных (Рисунок 39)

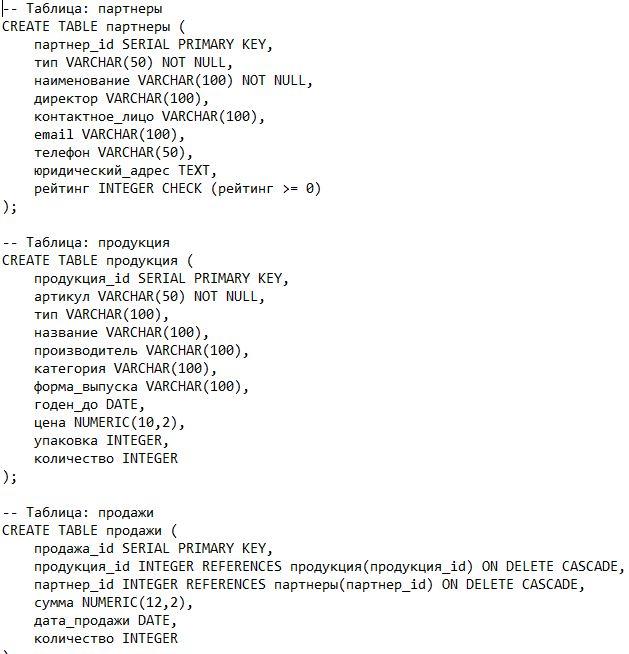


Рисунок 39 – Скрипт на создание бд и таблиц

Теперь сделаем импорт данных (Рисунок 40)

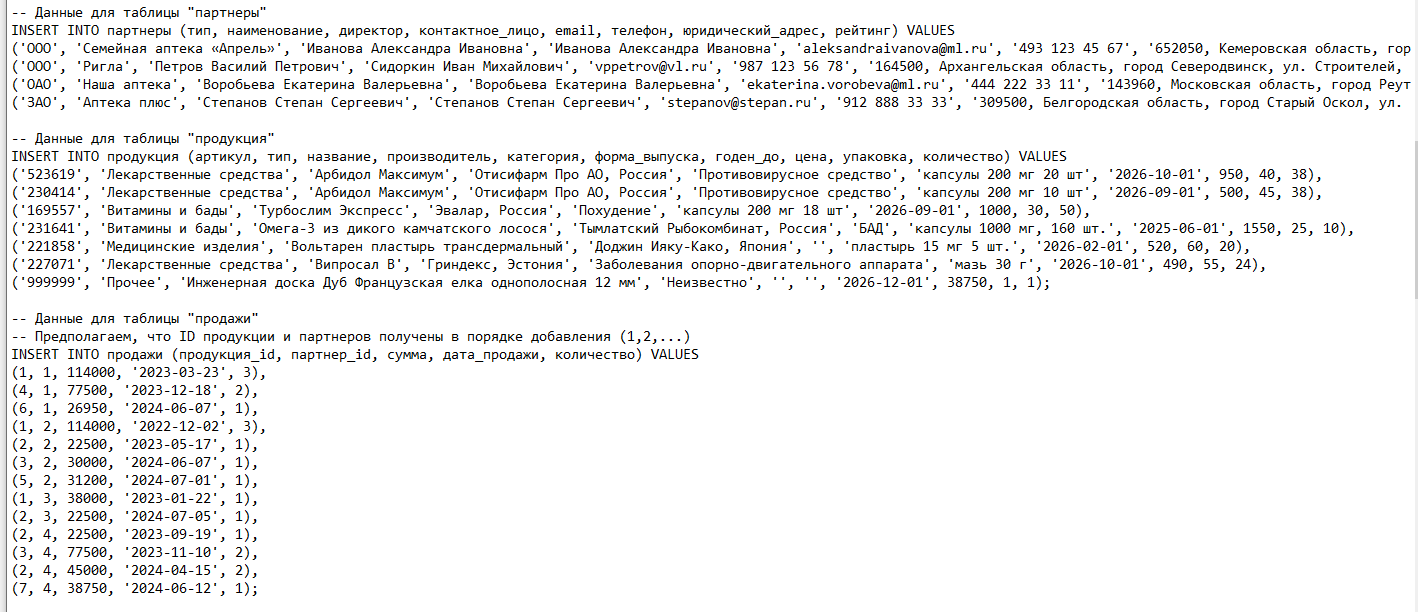


Рисунок 40 – Импорт данных в БД

Листинг кода формы авторизации:

// Подключение необходимых пространств имён

usingSystem;

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;

// ПодключениебиблиотекидляработысPostgreSQL

using Npgsql;

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

namespace YCHEBKA6

{

// Основной класс формы авторизации

public partial class Form1 : Form

{

// Объектсоединениясбазойданных

private NpgsqlConnection connection;

// Строкаподключениякбазеданных PostgreSQL

private string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=Aptechka";

// Конструктор формы

public Form1()

{

InitializeComponent(); // Инициализация компонентов формы

// Создание и открытие подключения к базе данных при запуске формы

NpgsqlConnection connection = new NpgsqlConnection(connectionString);

connection.Open(); // ВАЖНО: Это локальная переменная, которая перекрывает поле. Лучше использовать this.connection

}

// Обработчик события нажатия на кнопку "Войти"

private void button1\_Click\_1(object sender, EventArgs e)

{

// Получение введённых пользователем логина и пароля

string username = textBox1.Text;

string password = textBox2.Text;

// Проверка данных пользователя

if (AuthenticateUser(username, password))

{

// Если пользователь найден — сообщение об успехе и переход к следующей форме

MessageBox.Show("Успешная аутентификация! Добро пожаловать, " + username + "!");

Form f2 = new Form2(); // Создание экземпляра следующей формы

this.Hide(); // Скрытие текущей формы

f2.Show(); // Отображение новой формы

}

else

{

// Ошибка аутентификации — вывод сообщения

MessageBox.Show("Ошибка аутентификации. Пользователь с такими данными не найден.");

}

}

// Метод для проверки существования пользователя в базе данных

private bool AuthenticateUser(string username, string password)

{

// Повторное объявление строки подключения (можно убрать и использовать поле класса)

string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=Aptechka";

// Создание и открытие подключения с использованием using (гарантирует автоматическое закрытие)

using (NpgsqlConnection connection = new NpgsqlConnection(connectionString))

{

connection.Open();

// SQL-запрос с параметрами для безопасного поиска пользователя

string query = "SELECT COUNT(\*) FROM db\_users WHERE user\_login = @Username AND user\_password = @Password";

using (NpgsqlCommand command = new NpgsqlCommand(query, connection))

{

// Передача параметров в запрос

command.Parameters.AddWithValue("@Username", username);

command.Parameters.AddWithValue("@Password", password);

// Выполнение запроса и получение количества совпадений

int count = Convert.ToInt32(command.ExecuteScalar());

// Если есть хотя бы одно совпадение — возвращаем true

return count > 0;

}

}

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

}

На рисунке 41 работа приведённого выше кода:

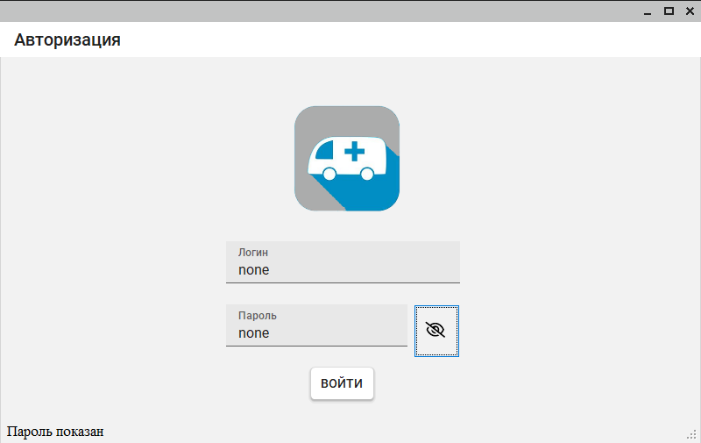


Рисунок 41 – Работа формы авторизации

Ниже листинг кода основной формы:

using Npgsql;

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Windows.Forms;

namespace YCHEBKA6

{

public partial class Form2 : Form

{

// Подключениек PostgreSQL

private NpgsqlConnection conn;

private NpgsqlDataAdapter adapter;

private DataTable dt;

// Словарь для хранения данных подстановки по внешним ключам

private Dictionary<string, Dictionary<object, string>> lookupData = new Dictionary<string, Dictionary<object, string>>();

public Form2()

{

InitializeComponent();

// Обработказакрытияформы

this.FormClosing += Form2\_FormClosing;

// Строкаподключениякбазеданных PostgreSQL

string connStr = "Host=localhost;Username=postgres;Password=18273645;Database=Aptechka";

conn = new NpgsqlConnection(connStr);

// Заполнение выпадающего списка таблицами

comboBox1.Items.AddRange(new object[] { "партнеры", "продукция", "продажи", "типы\_партнеров", "история\_рейтингов" });

comboBox1.SelectedIndexChanged += comboBox1\_SelectedIndexChanged;

}

// Обработкавыборатаблицы

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

string table = comboBox1.Text;

try

{

LoadLookupTables();

string query = $"SELECT \* FROM \"{table}\"";

adapter = new NpgsqlDataAdapter(query, conn);

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

dt = new DataTable();

adapter.Fill(dt);

dataGridView1.Columns.Clear();

dataGridView1.DataSource = dt;

HidePrimaryKey(table);

ReplaceForeignKeyColumns(table);

dataGridView1.DefaultValuesNeeded -= DataGridView1\_DefaultValuesNeeded;

dataGridView1.DefaultValuesNeeded += DataGridView1\_DefaultValuesNeeded;

}

catch (Exception ex)

{

MessageBox.Show($"Ошибкапризагрузкетаблицы \"{table}\": {ex.Message}");

}

}

// Кнопка "Сохранитьизменения"

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

{

try

{

// Генерациякоманд Insert/Update/Delete

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

adapter.UpdateCommand = builder.GetUpdateCommand();

adapter.InsertCommand = builder.GetInsertCommand();

adapter.DeleteCommand = builder.GetDeleteCommand();

// Сохраняемизменениявбазу

adapter.Update(dt);

MessageBox.Show("Изменениясохранены.");

}

catch (Exception ex)

{

MessageBox.Show("Ошибкаприсохранении: " + ex.Message);

}

}

// Кнопка "Удалитьзапись"

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

{

if (dataGridView1.CurrentRow != null)

{

try

{

// Удаление текущей строки и сохранение изменений

dataGridView1.Rows.RemoveAt(dataGridView1.CurrentRow.Index);

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

adapter.Update(dt);

MessageBox.Show("Записьудалена.");

}

catch (Exception ex)

{

MessageBox.Show("Ошибкаприудалении: " + ex.Message);

}

}

}

// Методдляскрытияпервичныхключей

private void HidePrimaryKey(string table)

{

string[] keys = { "партнер\_id", "продукция\_id", "продажа\_id", "user\_id" };

foreach (string key in keys)

{

if (dataGridView1.Columns.Contains(key))

dataGridView1.Columns[key].Visible = false;

}

}

// Загрузка данных для подстановки (выпадающих списков)

private void LoadLookupTables()

{

lookupData["продукция"] = LoadLookup("продукция", "продукция\_id", "название");

lookupData["партнеры"] = LoadLookup("партнеры", "партнер\_id", "наименование");

lookupData["типы\_партнеров"] = LoadLookup("типы\_партнеров", "тип\_id", "наименование");

}

// Метод для загрузки словаря "ID - Название" из таблицы

private Dictionary<object, string> LoadLookup(string table, string keyCol, string valCol)

{

Dictionary<object, string> dict = new Dictionary<object, string>();

string query = $"SELECT \"{keyCol}\", \"{valCol}\" FROM \"{table}\"";

using (var cmd = new NpgsqlCommand(query, conn))

{

conn.Open();

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

dict[reader[0]] = reader[1].ToString();

}

}

conn.Close();

}

return dict;

}

// Замена колонок внешнего ключа на выпадающие списки

private void ReplaceForeignKeyColumns(string table)

{

Dictionary<string, string> fkMap = new Dictionary<string, string>();

if (table == "продажи")

{

fkMap["продукция\_id"] = "продукция";

fkMap["партнер\_id"] = "партнеры";

}

else if (table == "партнеры")

{

fkMap["тип\_id"] = "типы\_партнеров";

}

foreach (var kvp in fkMap)

{

string column = kvp.Key;

string lookupTable = kvp.Value;

if (!dt.Columns.Contains(column)) continue;

DataGridViewComboBoxColumn combo = new DataGridViewComboBoxColumn()

{

Name = column,

DataPropertyName = column,

HeaderText = column,

DataSource = new BindingSource(lookupData[lookupTable], null),

DisplayMember = "Value",

ValueMember = "Key",

FlatStyle = FlatStyle.Flat

};

int index = dataGridView1.Columns[column].Index;

dataGridView1.Columns.Remove(column);

dataGridView1.Columns.Insert(index, combo);

}

}

// Значения по умолчанию при добавлении новой строки

private void DataGridView1\_DefaultValuesNeeded(object sender, DataGridViewRowEventArgs e)

{

string table = comboBox1.Text;

if (table == "продажи")

{

// Значения по умолчанию для внешних ключей

if (lookupData["партнеры"].Count > 0)

e.Row.Cells["партнер\_id"].Value = lookupData["партнеры"].First().Key;

if (lookupData["продукция"].Count > 0)

e.Row.Cells["продукция\_id"].Value = lookupData["продукция"].First().Key;

// Установка текущей даты

e.Row.Cells["дата\_продажи"].Value = DateTime.Today;

}

}

// Кнопка "Перейти к истории продаж"

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

{

this.Hide(); // Скрываемтекущуюформу

Form3 form3 = new Form3();

form3.Show(); // Показываем Form3

}

// При закрытии формы — завершить приложение

private void Form2\_FormClosing(object sender, FormClosingEventArgs e)

{

Application.Exit();

}

}

}

Нарисунках42, 43результатработыданногокода:

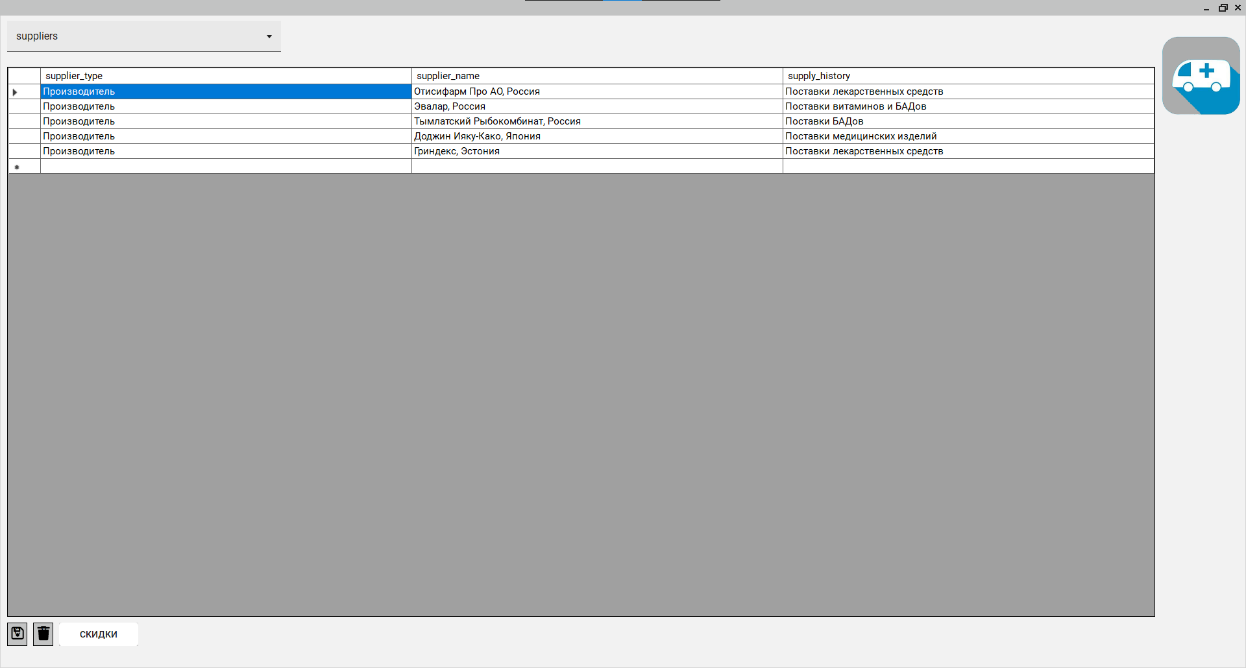


Рисунок 42 – Результат работы приложения

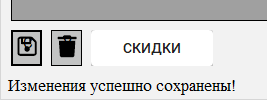


Рисунок 43 – Результат работы приложения

Листинг кода списка:

usingSystem;

using System.Windows.Forms;

using Npgsql;

using System.Drawing;

namespace YCHEBKA6

{

public partial class Form3 : Form

{

bool showingSalesHistory = false;

string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=Aptechka";

private int currentPartnerId = -1;

public Form3()

{

InitializeComponent();

// Настройка FlowLayoutPanel

flowLayoutPanel1.AutoScroll = true;

flowLayoutPanel1.WrapContents = false;

flowLayoutPanel1.FlowDirection = FlowDirection.TopDown;

this.FormClosing += FormClosingevent;

LoadPartners();

// Инициализация кнопки "Добавить партнера"

btnAddNewPartner.Click += (s, e) =>

{

currentPartnerId = -1;

AddPartnerCard(-1, 1, "", "", "", "", "", "", "", 1, 0);

flowLayoutPanel1.ScrollControlIntoView(flowLayoutPanel1.Controls[flowLayoutPanel1.Controls.Count - 1]);

};

// Инициализация кнопки "Назад к партнерам"

btnBackToPartners.Click += (s, e) =>

{

showingSalesHistory = false;

LoadPartners();

button1.Text = "История продаж";

label1.Text = "Список партнеров";

};

}

private void FormClosingevent(object sender, FormClosingEventArgs e)

{

Application.Exit();

}

private void LoadPartners()

{

flowLayoutPanel1.Controls.Clear();

flowLayoutPanel1.Controls.Add(btnAddNewPartner);

using (var conn = new NpgsqlConnection(connectionString))

{

conn.Open();

string query = @"

SELECT p.партнер\_id, t.тип\_id, t.наименование AS тип, p.наименование, p.директор,

p.контактное\_лицо, p.email, p.телефон, p.юридический\_адрес, p.рейтинг

FROM партнеры p

JOIN типы\_партнеров t ON p.тип\_id = t.тип\_id";

using (var cmd = new NpgsqlCommand(query, conn))

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

int partnerId = reader.GetInt32(0);

int typeId = reader.GetInt32(1);

string partnerType = reader.GetString(2);

string partnerName = reader.GetString(3);

string director = reader.GetString(4);

string contactPerson = reader.GetString(5);

string email = reader.GetString(6);

string phone = reader.GetString(7);

string address = reader.GetString(8);

int rating = reader.GetInt32(9);

decimal totalAmount = GetTotalSalesAmount(partnerId);

decimal discount = CalculateDiscount(totalAmount);

AddPartnerCard(partnerId, typeId, partnerType, partnerName, director,

contactPerson, email, phone, address, rating, discount);

}

}

}

}

private void AddPartnerCard(int partnerId, int typeId, string partnerType, string partnerName,

string director, string contactPerson, string email,

string phone, string address, int rating, decimal discount)

{

GroupBox partnerGroup = new GroupBox();

partnerGroup.Text = partnerName;

partnerGroup.Width = flowLayoutPanel1.ClientSize.Width - 30;

partnerGroup.Tag = partnerId;

partnerGroup.AutoSize = true;

partnerGroup.Margin = new Padding(5);

TableLayoutPanel table = new TableLayoutPanel();

table.ColumnCount = 2;

table.ColumnStyles.Add(new ColumnStyle(SizeType.Absolute, 150));

table.ColumnStyles.Add(new ColumnStyle(SizeType.Percent, 100));

table.AutoSize = true;

table.Dock = DockStyle.Fill;

// Добавляем поля для редактирования

ComboBox cmbType = AddComboBoxToTable(table, "Тип:", GetPartnerTypes(), typeId);

TextBox txtName = AddTextBoxToTable(table, "Наименование:", partnerName);

AddLabelToTable(table, "Скидка:", $"{discount}%");

TextBox txtDirector = AddTextBoxToTable(table, "Директор:", director);

TextBox txtContact = AddTextBoxToTable(table, "Контактное лицо:", contactPerson);

TextBox txtEmail = AddTextBoxToTable(table, "Email:", email);

TextBox txtPhone = AddTextBoxToTable(table, "Телефон:", phone);

TextBox txtAddress = AddTextBoxToTable(table, "Юр. адрес:", address);

NumericUpDown numRating = AddNumericToTable(table, "Рейтинг:", rating, 1, 10);

// Кнопки управления

FlowLayoutPanel buttonsPanel = new FlowLayoutPanel();

buttonsPanel.AutoSize = true;

buttonsPanel.FlowDirection = FlowDirection.RightToLeft;

buttonsPanel.Dock = DockStyle.Bottom;

Button btnSave = new Button { Text = "Сохранить", Tag = partnerId };

btnSave.Click += SavePartner\_Click;

buttonsPanel.Controls.Add(btnSave);

Button btnDelete = new Button { Text = "Удалить", Tag = partnerId };

btnDelete.Click += DeletePartner\_Click;

buttonsPanel.Controls.Add(btnDelete);

Button btnSales = new Button { Text = "Продажи", Tag = partnerId };

btnSales.Click += ShowSales\_Click;

buttonsPanel.Controls.Add(btnSales);

table.RowCount++;

table.RowStyles.Add(new RowStyle(SizeType.AutoSize));

table.Controls.Add(buttonsPanel, 0, table.RowCount - 1);

table.SetColumnSpan(buttonsPanel, 2);

partnerGroup.Controls.Add(table);

flowLayoutPanel1.Controls.Add(partnerGroup);

}

private ComboBox GetPartnerTypes()

{

ComboBox cmb = new ComboBox();

cmb.DropDownStyle = ComboBoxStyle.DropDownList;

using (var conn = new NpgsqlConnection(connectionString))

{

conn.Open();

string query = "SELECT тип\_id, наименование FROM типы\_партнеров";

using (var cmd = new NpgsqlCommand(query, conn))

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

cmb.Items.Add(new ComboBoxItem

{

Id = reader.GetInt32(0),

Name = reader.GetString(1)

});

}

}

}

return cmb;

}

private ComboBox AddComboBoxToTable(TableLayoutPanel table, string labelText, ComboBox comboBox, int selectedId)

{

Label label = new Label

{

Text = labelText,

Font = new Font(DefaultFont, FontStyle.Bold),

AutoSize = true,

Dock = DockStyle.Fill,

TextAlign = ContentAlignment.MiddleLeft

};

comboBox.Dock = DockStyle.Fill;

comboBox.SelectedIndexChanged += (s, e) => { /\* Можно добавить обработчик при необходимости \*/ };

// Устанавливаем выбранное значение

foreach (ComboBoxItemitemincomboBox.Items)

{

if (item.Id == selectedId)

{

comboBox.SelectedItem = item;

break;

}

}

table.RowCount++;

table.RowStyles.Add(new RowStyle(SizeType.AutoSize));

table.Controls.Add(label, 0, table.RowCount - 1);

table.Controls.Add(comboBox, 1, table.RowCount - 1);

return comboBox;

}

private TextBox AddTextBoxToTable(TableLayoutPanel table, string labelText, string valueText)

{

Label label = new Label

{

Text = labelText,

Font = new Font(DefaultFont, FontStyle.Bold),

AutoSize = true,

Dock = DockStyle.Fill,

TextAlign = ContentAlignment.MiddleLeft

};

TextBox textBox = new TextBox { Text = valueText, Dock = DockStyle.Fill };

table.RowCount++;

table.RowStyles.Add(new RowStyle(SizeType.AutoSize));

table.Controls.Add(label, 0, table.RowCount - 1);

table.Controls.Add(textBox, 1, table.RowCount - 1);

return textBox;

}

private NumericUpDown AddNumericToTable(TableLayoutPanel table, string labelText, int value, int min, int max)

{

Label label = new Label

{

Text = labelText,

Font = new Font(DefaultFont, FontStyle.Bold),

AutoSize = true,

Dock = DockStyle.Fill,

TextAlign = ContentAlignment.MiddleLeft

};

NumericUpDown numeric = new NumericUpDown { Value = value, Minimum = min, Maximum = max, Dock = DockStyle.Fill };

table.RowCount++;

table.RowStyles.Add(new RowStyle(SizeType.AutoSize));

table.Controls.Add(label, 0, table.RowCount - 1);

table.Controls.Add(numeric, 1, table.RowCount - 1);

return numeric;

}

private void AddLabelToTable(TableLayoutPanel table, string labelText, string valueText)

{

Label label = new Label

{

Text = labelText,

Font = new Font(DefaultFont, FontStyle.Bold),

AutoSize = true,

Dock = DockStyle.Fill,

TextAlign = ContentAlignment.MiddleLeft

};

Label value = new Label

{

Text = valueText,

AutoSize = true,

Dock = DockStyle.Fill,

TextAlign = ContentAlignment.MiddleLeft

};

table.RowCount++;

table.RowStyles.Add(new RowStyle(SizeType.AutoSize));

table.Controls.Add(label, 0, table.RowCount - 1);

table.Controls.Add(value, 1, table.RowCount - 1);

}

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

{

Button btn = (Button)sender;

int partnerId = (int)btn.Tag;

// Попытка безопасно найти GroupBox

Control current = btn.Parent;

while (current != null && !(current is GroupBox))

current = current.Parent;

var partnerGroup = current as GroupBox;

if (partnerGroup == null) return;

TableLayoutPanel table = (TableLayoutPanel)partnerGroup.Controls[0];

int typeId = ((ComboBoxItem)((ComboBox)table.GetControlFromPosition(1, 0)).SelectedItem).Id;

string name = ((TextBox)table.GetControlFromPosition(1, 1)).Text;

string director = ((TextBox)table.GetControlFromPosition(1, 3)).Text;

string contact = ((TextBox)table.GetControlFromPosition(1, 4)).Text;

string email = ((TextBox)table.GetControlFromPosition(1, 5)).Text;

string phone = ((TextBox)table.GetControlFromPosition(1, 6)).Text;

string address = ((TextBox)table.GetControlFromPosition(1, 7)).Text;

int rating = (int)((NumericUpDown)table.GetControlFromPosition(1, 8)).Value;

using (var conn = new NpgsqlConnection(connectionString))

{

conn.Open();

string query = partnerId == -1 ?

@"INSERTINTO партнеры (тип\_id, наименование, директор, контактное\_лицо, email, телефон, юридический\_адрес, рейтинг)

VALUES (@typeId, @name, @director, @contact, @email, @phone, @address, @rating)

RETURNING партнер\_id" :

@"UPDATE партнеры SET тип\_id = @typeId, наименование = @name, директор = @director,

контактное\_лицо = @contact, email = @email, телефон = @phone,

юридический\_адрес = @address, рейтинг = @rating

WHERE партнер\_id = @partnerId";

using (var cmd = new NpgsqlCommand(query, conn))

{

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

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

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

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

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

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

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

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

if (partnerId != -1)

{

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

cmd.ExecuteNonQuery();

}

else

{

partnerId = (int)cmd.ExecuteScalar();

}

}

}

LoadPartners(); // Обновляем список

}

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

{

if (MessageBox.Show("Удалить этого партнера?", "Подтверждение",

MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

Button btn = (Button)sender;

int partnerId = (int)btn.Tag;

using (var conn = new NpgsqlConnection(connectionString))

{

conn.Open();

string query = "DELETE FROM партнеры WHERE партнер\_id = @partnerId";

using (var cmd = new NpgsqlCommand(query, conn))

{

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

cmd.ExecuteNonQuery();

}

}

LoadPartners(); // Обновляем список

}

}

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

{

Button btn = (Button)sender;

currentPartnerId = (int)btn.Tag;

showingSalesHistory = true;

LoadPartnerSalesHistory(currentPartnerId);

button1.Text = "Список партнеров";

label1.Text = "История продаж";

comboBox1.Visible = false;

}

private Button btnAddNewPartner = new Button

{

Text = "Добавить нового партнера",

AutoSize = true,

Margin = new Padding(5)

};

private decimal CalculateDiscount(decimal totalAmount)

{

if (totalAmount <= 10000) return 0;

else if (totalAmount <= 50000) return 5;

else if (totalAmount <= 300000) return 10;

else return 15;

}

private decimal GetTotalSalesAmount(int partnerId)

{

decimal totalAmount = 0;

using (var conn = new NpgsqlConnection(connectionString))

{

conn.Open();

string query = @"SELECT SUM(s.сумма) FROM продажи s WHERE s.партнер\_id = @partnerId";

using (var cmd = new NpgsqlCommand(query, conn))

{

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

var result = cmd.ExecuteScalar();

if (result != DBNull.Value) totalAmount = Convert.ToDecimal(result);

}

}

return totalAmount;

}

private void LoadPartnerSalesHistory(int partnerId)

{

flowLayoutPanel1.Controls.Clear();

using (var conn = new NpgsqlConnection(connectionString))

{

conn.Open();

string query = @"

SELECT p.название AS product\_name, s.сумма, s.дата\_продажи, s.количество

FROM продажи s

JOIN продукция p ON s.продукция\_id = p.продукция\_id

WHERE s.партнер\_id = @partnerId

ORDER BY s.дата\_продажи DESC";

using (var cmd = new NpgsqlCommand(query, conn))

{

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

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

string productName = reader.GetString(0);

decimal saleAmount = reader.GetDecimal(1);

DateTime saleDate = reader.GetDateTime(2);

int quantity = reader.GetInt32(3);

AddSalesViewCard(productName, saleAmount, saleDate, quantity);

}

}

}

}

}

private void AddSalesViewCard(string productName, decimal saleAmount, DateTime saleDate, int quantity)

{

var saleGroup = new GroupBox();

saleGroup.Text = $"Продажа от {saleDate:dd.MM.yyyy}";

saleGroup.AutoSize = true;

saleGroup.AutoSizeMode = AutoSizeMode.GrowAndShrink;

saleGroup.Margin = new Padding(5);

saleGroup.Padding = new Padding(10);

var table = new TableLayoutPanel();

table.ColumnCount = 2;

table.ColumnStyles.Add(new ColumnStyle(SizeType.Absolute, 150));

table.ColumnStyles.Add(new ColumnStyle(SizeType.Percent, 100));

table.AutoSize = true;

table.Dock = DockStyle.Fill;

AddLabelToTable(table, "Продукция:", productName);

AddLabelToTable(table, "Сумма:", $"{saleAmount:C}");

AddLabelToTable(table, "Дата:", saleDate.ToString("dd.MM.yyyy"));

AddLabelToTable(table, "Количество:", quantity.ToString());

saleGroup.Controls.Add(table);

flowLayoutPanel1.Controls.Add(saleGroup);

Console.WriteLine("Загружено карточек: " + flowLayoutPanel1.Controls.Count);

}

private Button btnBackToPartners = new Button

{

Text = "Назад к партнерам",

AutoSize = true,

Margin = new Padding(5)

};

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

{

if (showingSalesHistory)

{

showingSalesHistory = false;

LoadPartners();

button1.Text = "История продаж";

label1.Text = "Список партнеров";

}

else

{

showingSalesHistory = true;

if (currentPartnerId == -1 && flowLayoutPanel1.Controls.Count > 0)

{

// Пытаемся получить ID первого партнера

foreach (Control control in flowLayoutPanel1.Controls)

{

if (control is GroupBox groupBox && groupBox.Tag is int id && id != -1)

{

currentPartnerId = id;

break;

}

}

}

LoadPartnerSalesHistory(currentPartnerId);

button1.Text = "Список партнеров";

label1.Text = "История продаж";

}

}

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

{

this.Hide();

Form2 form2 = new Form2();

form2.Show();

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox1.SelectedItem is ComboBoxItem selectedPartner)

{

currentPartnerId = selectedPartner.Id;

LoadPartnerSalesHistory(currentPartnerId);

}

}

private class ComboBoxItem

{

public int Id { get; set; }

public string Name { get; set; }

public override string ToString() => Name;

}

}

}

На рисунке 44список партнеров и история продаж:

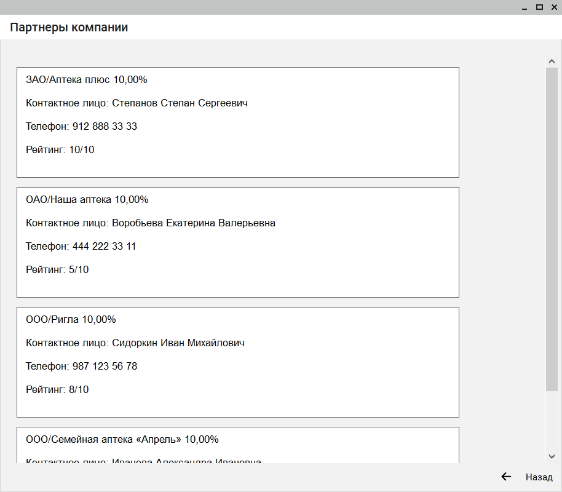


Рисунок 44 – Список партнеров

На рисунке 45история продаж:

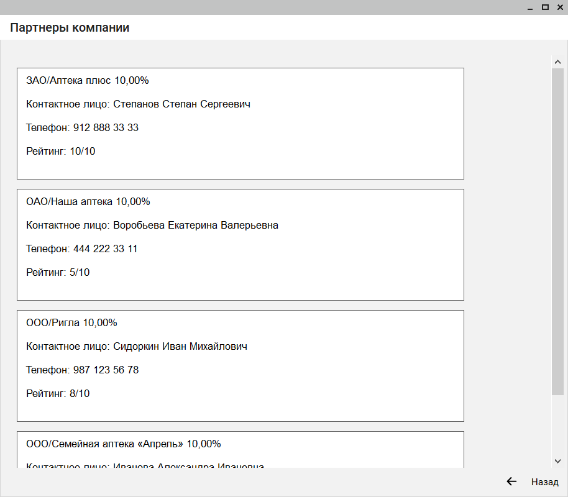


Рисунок 45 – История продаж

Работа № 7

Задание: Разработать подсистему для работы с товарами магазина «Технопомощник», обеспечивающую следующий функционал:

– оформление продаж и доставки;

– добавление/редактирование данных покупателей и доставки;

– просмотр доставки товаров.

Получить ER-диаграмму средствами СУБД: ER-диаграмма должна быть представлена в формате PDF и содержать таблицы, связи между ними, атрибуты и ключи (типами данных на данном этапе можно пренебречь).

Разработать программный модуль для учета покупателей. Необходимо реализовать вывод списка доставки, информация о которых хранятся в базе данных, согласно предоставленному макету:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | ФИО покупателя  Адрес  Телефон  Наименование товара  Категория товара  Дата/время доставки  Статус |  |
|  |  |  |
|  | ФИО покупателя  Адрес  Телефон  Наименование товара  Категория товара  Дата/время доставки  Статус |  |
|  |  |  |
|  | ФИО покупателя  Адрес  Телефон  Наименование товара  Категория товара  Дата/время доставки  Статус |  |
|  |  |  |

Необходимо осуществить оформление продажи товаров с выводом на экран списка товаров по категории, корзины покупателя и расчета суммы продажи, в случае необходимости доставки обеспечить оформление и расчет суммы с учетом доставки. Учитывать количество товаров на складах, чтобы нельзя было продавать товар в минус для организации со склада.

Разработать интерфейс программного модуля для работы с покупателями, доставкой и продажами.

Реализовать функции добавления/редактирования данных покупателя в новом окне (странице) – форме для добавления/редактирования покупателя и доставки. Переходы на эту форму должны быть реализованы из формы списка доставки: для редактирования доставки из списка – при нажатии на конкретный элемент, для редактирования/добавления любого покупателя – при нажатии кнопки.

На форме для добавления/редактирования доставки должны быть предусмотрены следующие поля: ФИО клиента, Адрес, Телефон, Дата доставки, Наименование товара, Статус.

**Ход работы**

Подготовим данные для импорта. Для этого приведём все данные в предоставленных для импорта таблицах к 3НФ (Рисунок 46).

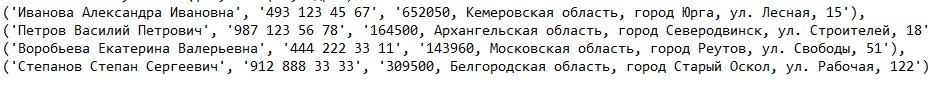


Рисунок 46 – Подготовка данных для импорта

Теперь необходимо создать базу данных (Рисунок 47)

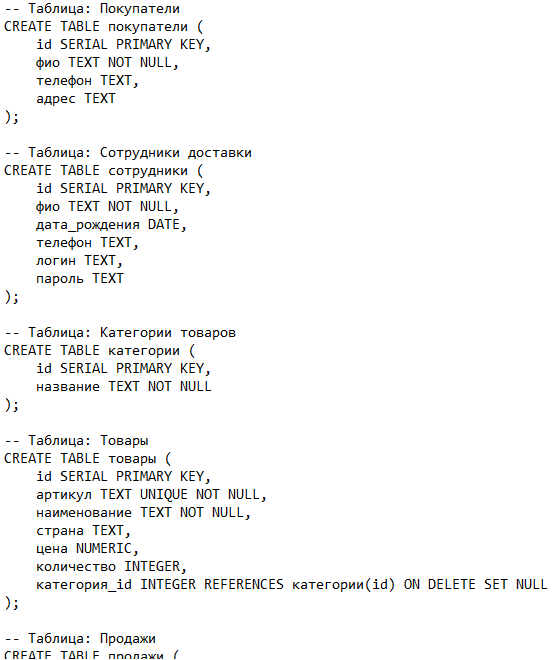


Рисунок 47 – Скрипт для создания БД

Теперь сделаем импорт данных (Рисунок 48)

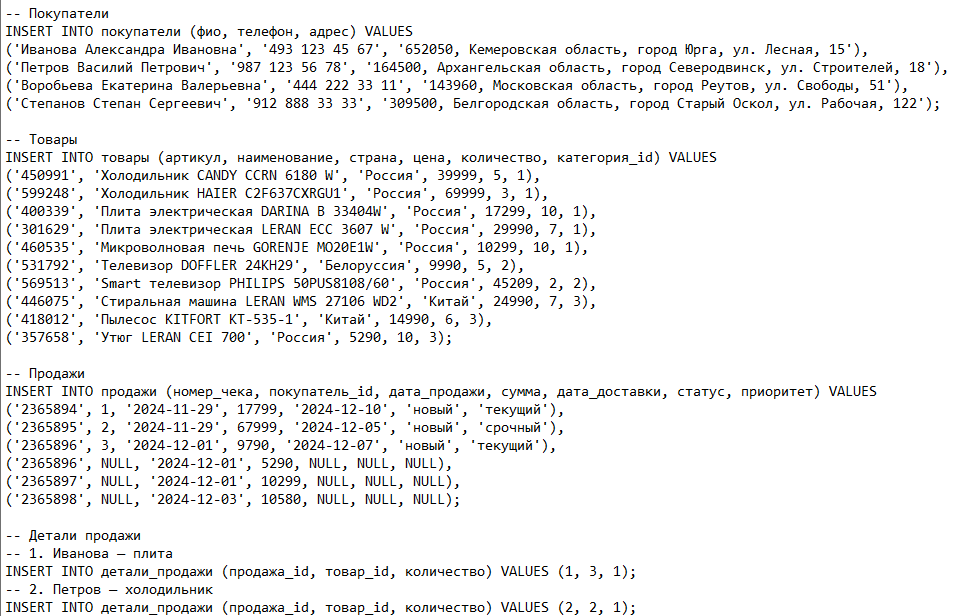


Рисунок 48 – Импорт данных в БД

Листинг кода формы авторизации:

// Подключение необходимых пространств имён

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;

// Подключениебиблиотекидляработыс PostgreSQL

using Npgsql;

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

namespace YCHEBKA7

{

// Основной класс формы авторизации

public partial class Form1 : Form

{

// Объектсоединениясбазойданных

private NpgsqlConnection connection;

// Строкаподключениякбазеданных PostgreSQL

private string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=Techno";

// Конструктор формы

public Form1()

{

InitializeComponent(); // Инициализация компонентов формы

// Создание и открытие подключения к базе данных при запуске формы

NpgsqlConnection connection = new NpgsqlConnection(connectionString);

connection.Open(); // ВАЖНО: Это локальная переменная, которая перекрывает поле. Лучше использовать this.connection

}

// Обработчик события нажатия на кнопку "Войти"

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

{

// Получение введённых пользователем логина и пароля

string username = textBox1.Text;

string password = textBox2.Text;

// Проверка данных пользователя

if (AuthenticateUser(username, password))

{

// Если пользователь найден — сообщение об успехе и переход к следующей форме

MessageBox.Show("Успешная аутентификация! Добро пожаловать, " + username + "!");

Form f2 = new Form2(); // Создание экземпляра следующей формы

this.Hide(); // Скрытие текущей формы

f2.Show(); // Отображение новой формы

}

else

{

// Ошибка аутентификации — вывод сообщения

MessageBox.Show("Ошибка аутентификации. Пользователь с такими данными не найден.");

}

}

// Метод для проверки существования пользователя в базе данных

private bool AuthenticateUser(string username, string password)

{

// Повторное объявление строки подключения (можно убрать и использовать поле класса)

string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=Techno";

// Создание и открытие подключения с использованием using (гарантирует автоматическое закрытие)

using (NpgsqlConnection connection = new NpgsqlConnection(connectionString))

{

connection.Open();

// SQL-запрос с параметрами для безопасного поиска пользователя

string query = "SELECT COUNT(\*) FROM сотрудники WHERE логин = @Username AND пароль = @Password";

using (NpgsqlCommand command = new NpgsqlCommand(query, connection))

{

// Передача параметров в запрос

command.Parameters.AddWithValue("@Username", username);

command.Parameters.AddWithValue("@Password", password);

// Выполнение запроса и получение количества совпадений

int count = Convert.ToInt32(command.ExecuteScalar());

// Если есть хотя бы одно совпадение — возвращаем true

return count > 0;

}

}

}

}

}

На рисунке 49 работа приведённого выше кода:

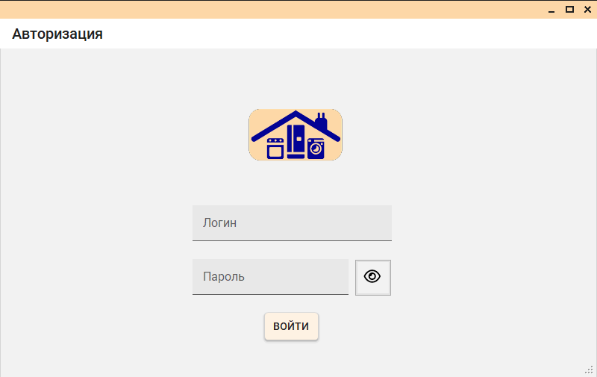


Рисунок 49 – Работа формы авторизации

Ниже листинг кода основной формы:

using Npgsql;

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Windows.Forms;

namespace YCHEBKA7

{

public partial class Form2 : Form

{

// Подключение к базе данных

NpgsqlConnectionconn;

// Адаптер для связи между DataTable и базой данных

NpgsqlDataAdapteradapter;

// Таблица данных для отображения в DataGridView

DataTabledt;

// Словарь для хранения отображения внешних ключей

Dictionary<string, Dictionary<object, string>> lookupData = new Dictionary<string, Dictionary<object, string>>();

public Form2()

{

InitializeComponent();

this.FormClosing += new FormClosingEventHandler(FormClosingevent);

// Строка подключения к PostgreSQL

string connection = "Host=localhost;Username=postgres;Password=18273645;Database=Techno";

conn = new NpgsqlConnection(connection);

// Заполнение comboBox1 названиями таблиц

comboBox1.Items.AddRange(new object[] {

"покупатели", "сотрудники", "категории", "товары", "продажи", "детали\_продажи"

});

// Подключение обработчика выбора таблицы

comboBox1.SelectedIndexChanged += comboBox1\_SelectedIndexChanged;

}

// Обработчик выбора таблицы

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

string table = comboBox1.Text;

// Загрузка справочных данных

LoadLookupTables();

// Формирование SQL-запроса

string query = $"SELECT \* FROM \"{table}\"";

adapter = new NpgsqlDataAdapter(query, conn);

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

dt = new DataTable();

adapter.Fill(dt); // Заполнение DataTable из базы данных

// Получение списка первичных ключей

List<string> primaryKeys = GetPrimaryKeys(table);

// Очистка и привязка таблицы к DataGridView

dataGridView1.Columns.Clear();

dataGridView1.DataSource = dt;

// Скрытие первичных ключей в таблице

foreach (string pk in primaryKeys)

{

if (dataGridView1.Columns.Contains(pk))

{

dataGridView1.Columns[pk].Visible = false;

}

}

// Замена внешних ключей на выпадающие списки

ReplaceForeignKeyColumns(table);

// Добавление значений по умолчанию при создании новых строк

dataGridView1.DefaultValuesNeeded -= dataGridView1\_DefaultValuesNeeded;

dataGridView1.DefaultValuesNeeded += dataGridView1\_DefaultValuesNeeded;

}

// Получение списка первичных ключей таблицы

private List<string> GetPrimaryKeys(string tableName)

{

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

string query = @"

SELECT a.attname

FROM pg\_index i

JOIN pg\_attribute a ON a.attrelid = i.indrelid AND a.attnum = ANY(i.indkey)

WHERE i.indrelid = @tableName::regclass AND i.indisprimary;

";

using (var cmd = new NpgsqlCommand(query, conn))

{

cmd.Parameters.AddWithValue("tableName", tableName);

conn.Open();

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

primaryKeys.Add(reader.GetString(0));

}

}

conn.Close();

}

return primaryKeys;

}

// Обработчик кнопки "Сохранить"

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

{

try

{

// Создание SQL-команд для операций обновления

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

adapter.UpdateCommand = builder.GetUpdateCommand();

adapter.InsertCommand = builder.GetInsertCommand();

adapter.DeleteCommand = builder.GetDeleteCommand();

// Сохранение изменений в базе данных

adapter.Update(dt);

MessageBox.Show("Изменения сохранены.");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при сохранении: " + ex.Message);

}

}

// Обработчик кнопки "Удалить"

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

{

if (dataGridView1.CurrentRow != null)

{

try

{

// Удаление выбранной строки

dataGridView1.Rows.RemoveAt(dataGridView1.CurrentRow.Index);

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

adapter.Update(dt);

MessageBox.Show("Запись удалена.");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при удалении: " + ex.Message);

}

}

else

{

MessageBox.Show("Выберите строку для удаления.");

}

}

// Загрузка справочных таблиц для отображения вместо ID

privatevoidLoadLookupTables()

{

lookupData["категории"] = LoadLookup("категории", "id", "название");

lookupData["покупатели"] = LoadLookup("покупатели", "id", "фио");

lookupData["сотрудники"] = LoadLookup("сотрудники", "id", "фио");

lookupData["товары"] = LoadLookup("товары", "id", "наименование");

lookupData["продажи"] = LoadLookup("продажи", "id", "номер\_чека");

}

// Загрузка справочной таблицы и формирование словаря

private Dictionary<object, string> LoadLookup(string table, string keyCol, string valCol)

{

Dictionary<object, string> dict = new Dictionary<object, string>();

string query = $"SELECT \"{keyCol}\", \"{valCol}\" FROM \"{table}\"";

using (var cmd = new NpgsqlCommand(query, conn))

{

conn.Open();

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

dict.Add(reader[0], reader[1].ToString());

}

}

conn.Close();

}

returndict;

}

// Замена столбцов внешних ключей на выпадающие списки

private void ReplaceForeignKeyColumns(string table)

{

Dictionary<string, string> fkMap = new Dictionary<string, string>();

// Определение внешних ключей по таблице

if (table == "товары")

{

fkMap.Add("категория\_id", "категории");

}

elseif (table == "продажи")

{

fkMap.Add("покупатель\_id", "покупатели");

fkMap.Add("исполнитель\_id", "сотрудники");

}

elseif (table == "детали\_продажи")

{

fkMap.Add("продажа\_id", "продажи");

fkMap.Add("товар\_id", "товары");

}

// Замена каждого столбца внешнего ключа на ComboBox

foreach (var kvp in fkMap)

{

string column = kvp.Key;

string lookupTable = kvp.Value;

if (!dt.Columns.Contains(column)) continue;

DataGridViewComboBoxColumn combo = new DataGridViewComboBoxColumn

{

Name = column,

DataPropertyName = column,

HeaderText = column,

DataSource = new BindingSource(lookupData[lookupTable], null),

DisplayMember = "Value",

ValueMember = "Key",

FlatStyle = FlatStyle.Flat

};

int index = dataGridView1.Columns[column].Index;

dataGridView1.Columns.Remove(column);

dataGridView1.Columns.Insert(index, combo);

}

}

// Установка значений по умолчанию при добавлении новой строки

private void dataGridView1\_DefaultValuesNeeded(object sender, DataGridViewRowEventArgs e)

{

string table = comboBox1.Text;

if (table == "продажи")

{

e.Row.Cells["дата\_продажи"].Value = DateTime.Today;

e.Row.Cells["статус"].Value = "новый";

e.Row.Cells["приоритет"].Value = "текущий";

if (lookupData["покупатели"].Count > 0)

e.Row.Cells["покупатель\_id"].Value = lookupData["покупатели"].First().Key;

if (lookupData["сотрудники"].Count > 0)

e.Row.Cells["исполнитель\_id"].Value = lookupData["сотрудники"].First().Key;

}

}

// Переход к форме Form3

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

{

this.Hide();

Form3 form3 = new Form3();

form3.Show();

}

// Завершение работы приложения при закрытии формы

private void FormClosingevent(object sender, FormClosingEventArgs e)

{

Application.Exit();

}

// Переход к форме Form4

private void button4\_Click(object sender, EventArgs e)

{

this.Hide();

Form4 form4 = new Form4();

form4.Show();

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

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

{

}

}

}

Нарисунках50, 51результатработыданногокода:

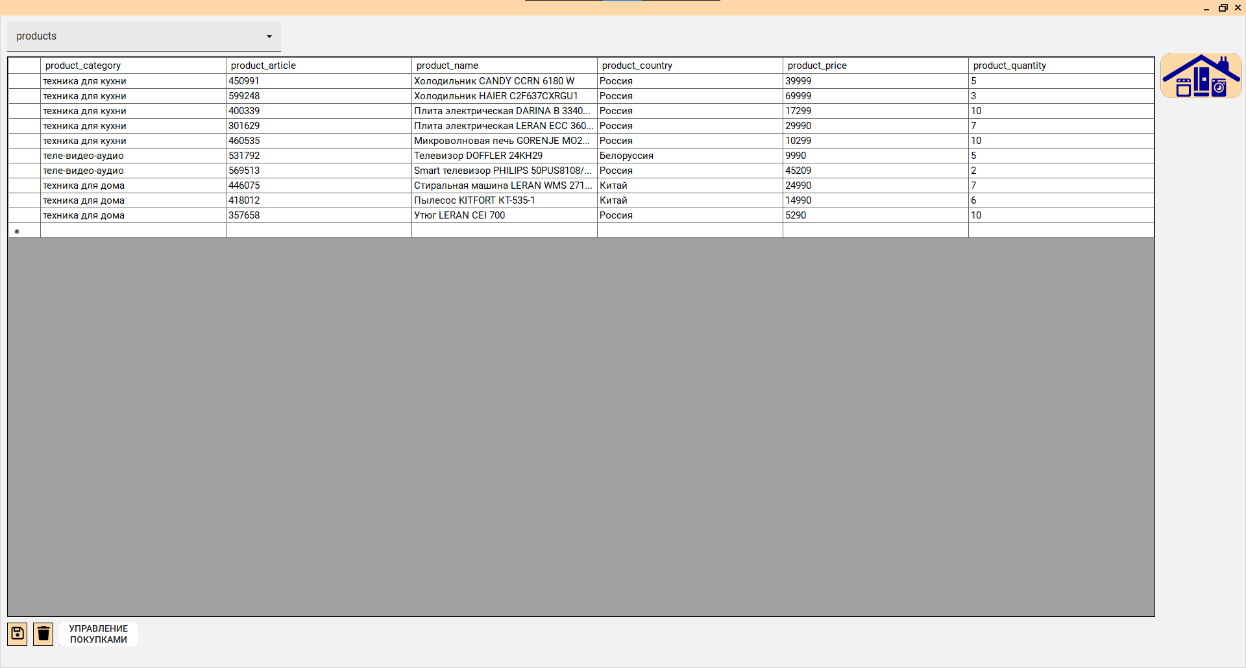


Рисунок 50 – Результат работы приложения

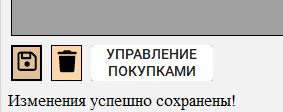


Рисунок 51 – Результат работы приложения

На рисунке 52 представлена работа формы списка доставок:

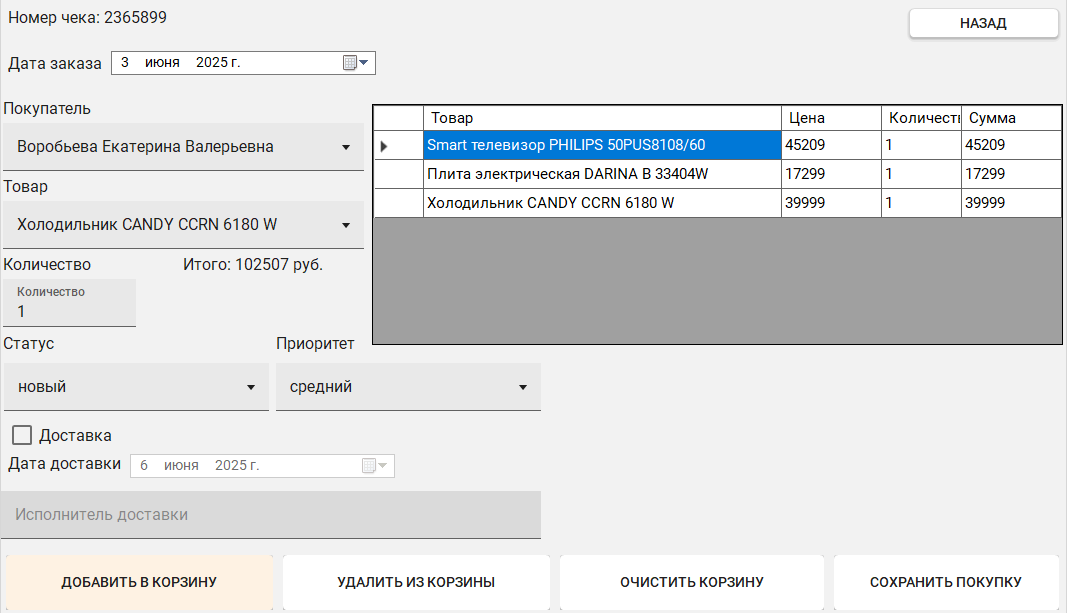


Рисунок 52 – Управление покупками

Листинг кода для управления покупками:

;using System;

using System.Data;

using System.Text;

using System.Windows.Forms;

using Npgsql; // Библиотека для работы с PostgreSQL

namespace YCHEBKA7

{

public partial class Form3 : Form

{

// Строка подключения к базе данных PostgreSQL

private const string CONNECTION\_STRING = "Host=localhost;Username=postgres;Password=18273645;Database=Techno";

// Конструктор формы

public Form3()

{

InitializeComponent(); // Инициализация компонентов формы

LoadData(); // Загрузка данных при открытии формы

this.FormClosing += new FormClosingEventHandler(FormClosingevent); // Обработчикзакрытияформы

}

/// Метод загрузки данных из базы данных и отображения их в richTextBox1

private void LoadData()

{

// Создание соединения с базой данных

using (var conn = new NpgsqlConnection(CONNECTION\_STRING))

{

conn.Open(); // Открытие соединения

// SQL-запрос для получения информации о покупателях, товарах и продажах

string query = @"

SELECT

п.фио AS фио,

п.адрес AS адрес,

п.телефон AS телефон,

т.наименование AS товар,

к.название AS категория,

пр.дата\_доставки AS доставка,

пр.статус AS статус

FROM продажи пр

JOIN покупатели п ON пр.покупатель\_id = п.id

JOIN детали\_продажи дп ON пр.id = дп.продажа\_id

JOIN товары т ON дп.товар\_id = т.id

LEFT JOIN категории к ON т.категория\_id = к.id;

";

// Создание команды и выполнение запроса

using (var cmd = new NpgsqlCommand(query, conn))

using (var reader = cmd.ExecuteReader()) // Чтение данных из результата запроса

{

var sb = new StringBuilder(); // Буфер для накопления текста

// Чтение всех строк результата

while (reader.Read())

{

// Добавление информации о каждом поле с проверкой на NULL

sb.AppendLine($"ФИО покупателя: {(reader["фио"] != DBNull.Value ? reader["фио"].ToString() : "Не указано")}");

sb.AppendLine($"Адрес: {(reader["адрес"] != DBNull.Value ? reader["адрес"].ToString() : "Адрес не указан")}");

sb.AppendLine($"Телефон: {(reader["телефон"] != DBNull.Value ? reader["телефон"].ToString() : "Телефон не указан")}");

sb.AppendLine($"Наименование товара: {(reader["товар"] != DBNull.Value ? reader["товар"].ToString() : "Нет наименования")}");

sb.AppendLine($"Категория товара: {(reader["категория"] != DBNull.Value ? reader["категория"].ToString() : "Категория не задана")}");

// Преобразование даты доставки

string дата = reader["доставка"] != DBNull.Value

? Convert.ToDateTime(reader["доставка"]).ToString("g") // "g" — краткий формат даты и времени

: "Дата доставки не указана";

sb.AppendLine($"Дата/время доставки: {дата}");

sb.AppendLine($"Статус: {(reader["статус"] != DBNull.Value ? reader["статус"].ToString() : "Статус неизвестен")}");

// Добавление отступа между записями

sb.AppendLine(new string(' ', 40));

}

// Выводрезультатав richTextBox1

richTextBox1.Text = sb.ToString();

}

}

}

/// Обработчик кнопки "Назад" — возвращение на предыдущую форму

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

{

this.Hide(); // Скрытие текущей формы

Form2 form2 = new Form2(); // Создание экземпляра второй формы

form2.Show(); // Отображение второй формы

}

/// Обработчик события закрытия формы — завершение работы приложения

private void FormClosingevent(object sender, FormClosingEventArgs e)

{

Application.Exit(); // Полное завершение приложения

}

}

}

На рисунке 53показано форма корзины:

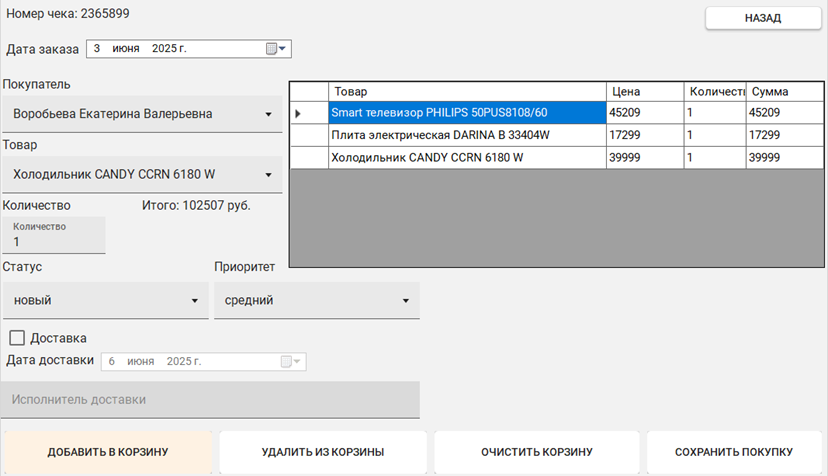


Рисунок 53 – Корзина

Листинг кода для корзины:

// Подключение необходимых пространств имен

using System;

using System.Collections.Generic;

using System.Data;

using System.Windows.Forms;

using Npgsql; // Библиотека для работы с PostgreSQL

namespace YCHEBKA7

{

// Классформы Form4

public partial class Form4 : Form

{

private NpgsqlConnection conn; // ПодключениекБД

private decimal totalAmount = 0; // Общаясуммапокупок

private List<CartItem> cartItems = new List<CartItem>(); // Списоктовароввкорзине

// Конструктор формы

public Form4()

{

InitializeComponent(); // Инициализация компонентов формы

conn = new NpgsqlConnection("Host=localhost;Username=postgres;Password=18273645;Database=Techno");

// Закрытие приложения при закрытии формы

this.FormClosing += (s, e) => Application.Exit();

// Подписканасобытияинтерфейса

comboBoxCategories.SelectedIndexChanged += ComboBoxCategories\_SelectedIndexChanged;

buttonAddToCart.Click += ButtonAddToCart\_Click;

buttonRemoveFromCart.Click += ButtonRemoveFromCart\_Click;

buttonCheckout.Click += ButtonCheckout\_Click;

// Загрузкаданных

LoadCategories();

LoadBuyers();

LoadCouriers();

}

// Загрузкаспискапокупателейввыпадающийсписок

private void LoadBuyers()

{

conn.Open();

NpgsqlCommand cmd = new NpgsqlCommand("SELECT id, фио FROM покупатели", conn);

var reader = cmd.ExecuteReader();

Dictionary<int, string> buyers = new Dictionary<int, string>();

while (reader.Read())

buyers.Add(reader.GetInt32(0), reader.GetString(1));

conn.Close();

comboBoxBuyer.DataSource = new BindingSource(buyers, null);

comboBoxBuyer.DisplayMember = "Value";

comboBoxBuyer.ValueMember = "Key";

}

// Загрузка списка курьеров в выпадающий список

private void LoadCouriers()

{

conn.Open();

NpgsqlCommand cmd = new NpgsqlCommand("SELECT id, фио FROM сотрудники", conn);

var reader = cmd.ExecuteReader();

Dictionary<int, string> couriers = new Dictionary<int, string>();

while (reader.Read())

couriers.Add(reader.GetInt32(0), reader.GetString(1));

conn.Close();

comboBoxCourier.DataSource = new BindingSource(couriers, null);

comboBoxCourier.DisplayMember = "Value";

comboBoxCourier.ValueMember = "Key";

}

// Загрузкакатегорийтоваров

private void LoadCategories()

{

conn.Open();

NpgsqlCommand cmd = new NpgsqlCommand("SELECT id, название FROM категории", conn);

var reader = cmd.ExecuteReader();

Dictionary<int, string> dict = new Dictionary<int, string>();

while (reader.Read())

{

dict.Add(reader.GetInt32(0), reader.GetString(1));

}

conn.Close();

comboBoxCategories.DataSource = new BindingSource(dict, null);

comboBoxCategories.DisplayMember = "Value";

comboBoxCategories.ValueMember = "Key";

}

// Загрузкатоваровповыбраннойкатегории

private void ComboBoxCategories\_SelectedIndexChanged(object sender, EventArgs e)

{

int categoryId = ((KeyValuePair<int, string>)comboBoxCategories.SelectedItem).Key;

listBoxProducts.Items.Clear();

conn.Open();

string query = @"SELECT id, наименование, цена FROM товары WHERE категория\_id = @cat AND количество> 0";

var cmd = new NpgsqlCommand(query, conn);

cmd.Parameters.AddWithValue("cat", categoryId);

var reader = cmd.ExecuteReader();

while (reader.Read())

{

int id = reader.GetInt32(0);

string name = reader.GetString(1);

decimal price = reader.IsDBNull(2) ? 0 : reader.GetDecimal(2);

// Форматированиестрокидляотображения

listBoxProducts.Items.Add(id + "|" + name + "|" + price.ToString("0.00") + "₽");

}

conn.Close();

}

// Добавление товара в корзину

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

{

if (listBoxProducts.SelectedItem == null) return;

string[] parts = listBoxProducts.SelectedItem.ToString().Split('|');

int productId = int.Parse(parts[0]);

string name = parts[1];

decimal price = decimal.Parse(parts[2].Replace("₽", ""));

// Добавлениевсписок

cartItems.Add(new CartItem(productId, name, price, 1));

listBoxCart.Items.Add(name + " - " + price.ToString("0.00") + "₽");

totalAmount += price;

labelTotal.Text = "Итого: " + totalAmount.ToString("0.00") + "₽";

}

// Удалениетовараизкорзины

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

{

int index = listBoxCart.SelectedIndex;

if (index >= 0)

{

totalAmount -= cartItems[index].price;

cartItems.RemoveAt(index);

listBoxCart.Items.RemoveAt(index);

labelTotal.Text = "Итого: " + totalAmount.ToString("0.00") + "₽";

}

}

// Оформлениепродажи

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

{

if (cartItems.Count == 0)

{

MessageBox.Show("Корзинапуста!");

return;

}

if (comboBoxBuyer.SelectedItem == null || comboBoxCourier.SelectedItem == null || comboBoxPriority.SelectedItem == null)

{

MessageBox.Show("Выберите покупателя, исполнителя и приоритет!");

return;

}

conn.Open();

var tran = conn.BeginTransaction(); // Началотранзакции

try

{

// Генерацияномерачека

string checkNumber = new Random().Next(100000000, 999999999).ToString();

int buyerId = ((KeyValuePair<int, string>)comboBoxBuyer.SelectedItem).Key;

int courierId = ((KeyValuePair<int, string>)comboBoxCourier.SelectedItem).Key;

string priority = comboBoxPriority.SelectedItem.ToString();

DateTime deliveryDate = dateTimePickerDelivery.Value.Date;

// Вставка записи в таблицу продажи

NpgsqlCommand insertSale = new NpgsqlCommand(@"

INSERT INTO продажи

(номер\_чека, покупатель\_id, дата\_продажи, сумма, дата\_доставки, исполнитель\_id, статус, приоритет)

VALUES (@номер, @покупатель, @дата, @сумма, @доставка, @исполнитель, @статус, @приоритет)

RETURNING id", conn);

insertSale.Parameters.AddWithValue("номер", checkNumber);

insertSale.Parameters.AddWithValue("покупатель", buyerId);

insertSale.Parameters.AddWithValue("дата", DateTime.Today);

insertSale.Parameters.AddWithValue("сумма", totalAmount);

insertSale.Parameters.AddWithValue("доставка", deliveryDate);

insertSale.Parameters.AddWithValue("исполнитель", courierId);

insertSale.Parameters.AddWithValue("статус", "Оформлено");

insertSale.Parameters.AddWithValue("приоритет", priority);

int saleId = Convert.ToInt32(insertSale.ExecuteScalar());

// Вставка записей в таблицу деталей продажи

foreach (var item in cartItems)

{

var cmd = new NpgsqlCommand(

"INSERT INTO детали\_продажи (продажа\_id, товар\_id, количество) VALUES (@pid, @tid, @qty)", conn);

cmd.Parameters.AddWithValue("pid", saleId);

cmd.Parameters.AddWithValue("tid", item.productId);

cmd.Parameters.AddWithValue("qty", item.quantity);

cmd.ExecuteNonQuery();

}

tran.Commit(); // Подтверждениетранзакции

MessageBox.Show("Продажа успешно оформлена!");

// Очистка корзины

cartItems.Clear();

listBoxCart.Items.Clear();

totalAmount = 0;

labelTotal.Text = "Итого: 0₽";

}

catch (Exception ex)

{

tran.Rollback(); // Откат транзакции при ошибке

MessageBox.Show("Ошибка: " + ex.Message);

}

conn.Close();

}

// Класс для хранения данных о товаре в корзине

private class CartItem

{

public int productId;

public string name;

public decimal price;

public int quantity;

public CartItem(int productId, string name, decimal price, int quantity)

{

this.productId = productId;

this.name = name;

this.price = price;

this.quantity = quantity;

}

}

// Переходнадругуюформу

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

{

this.Hide();

Form2 form2 = new Form2();

form2.Show();

}

}

}

Работа № 8

Задание:Разработать подсистему для работы центра занятости «Рекрутер», обеспечивающую следующий функционал:

– оформление вакансий;

– добавление/редактирование данных кандидатов на рабочие места;

– просмотр кандидатов по вакансии.

Получить ER-диаграмму средствами СУБД: ER-диаграмма должна быть представлена в формате PDF и содержать таблицы, связи между ними, атрибуты и ключи (типами данных на данном этапе можно пренебречь).

Разработать программный модуль для учета соискателей. Необходимо реализовать вывод списка претендентов на должность вакансии, информация о которых хранятся в базе данных, согласно предоставленному макету:

|  |  |  |
| --- | --- | --- |
|  | Предприятие <выбор из списка> Адрес  Должность <выбор из списка> Образование Квалификация  Вод.права Иностранные языки Др.информация  Должностные обязанности  Условия труда |  |
|  | ФИО претендента  Возраст  Пол  Должность  Образование  Квалификация  Предполагаемая з/пл.  Адрес  Телефон  Дата заполнения  Статус |  |
|  |  |  |
|  | ФИО претендента  Возраст  Пол  Должность  Образование  Квалификация  Предполагаемая з/пл.  Адрес  Телефон  Дата заполнения  Статус |  |
|  |  |  |
|  | ФИО претендента  Возраст  Пол  Должность  Образование  Квалификация  Предполагаемая з/пл.  Адрес  Телефон  Дата заполнения  Статус |  |
|  |  |  |

Необходимо осуществить оформление вакансий, предприятий и претендентов.

Разработать интерфейс программного модуля для работы с организациями, вакансиями и претендентами на должность.

Реализовать функции добавления/редактирования данных соискателя в новом окне (странице) – форме для добавления/редактирования данных кандидатов на должность. Переходы на эту форму должны быть реализованы из формы списка соискателей должности: для редактирования клиентов из списка – при нажатии на конкретный элемент, для редактирования/добавления любого клиента центра занятости – при нажатии кнопки.

На форме для добавления/редактирования клиентов должны быть предусмотрены следующие поля: ФИО клиента, дата рождения, пол, должность, образование, квалификация, предполагаемая з/пл., адрес, телефон, дата заполнения, статус.

**Ход работы**

Подготовим данные для импорта. Для этого приведём все данные в предоставленных для импорта таблицах к 3НФ (Рисунок 54).

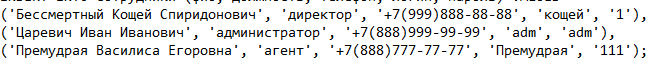


Рисунок 54 – Подготовка данных для импорта

Теперь необходимо создать базу данных (Рисунок 55)

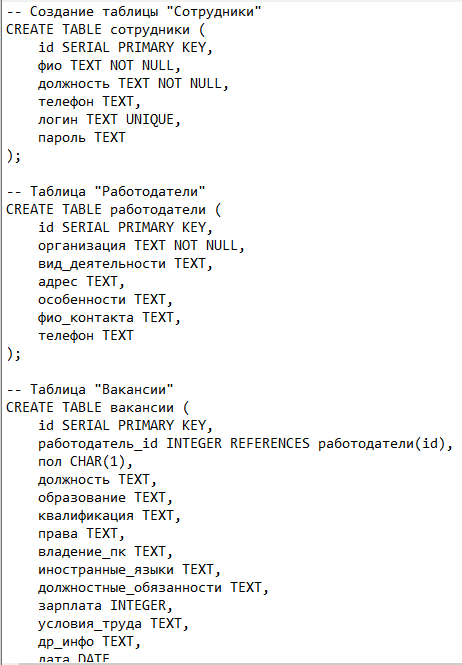


Рисунок 55 – Скрипт для создания БД

Теперь сделаем импорт данных (Рисунок 56)



Рисунок 56 – Импорт данных в БД

Листинг кода авторизации:

// Подключение необходимых пространств имён

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;

// Подключениебиблиотекидляработыс PostgreSQL

using Npgsql;

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

namespace YCHEBKA8

{

// Основной класс формы авторизации

public partial class Form1 : Form

{

// Объектсоединениясбазойданных

private NpgsqlConnection connection;

// Строкаподключениякбазеданных PostgreSQL

private string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=Rekryt";

// Конструктор формы

public Form1()

{

InitializeComponent(); // Инициализация компонентов формы

// Создание и открытие подключения к базе данных при запуске формы

NpgsqlConnection connection = new NpgsqlConnection(connectionString);

connection.Open(); // ВАЖНО: Это локальная переменная, которая перекрывает поле. Лучше использовать this.connection

}

// Обработчик события нажатия на кнопку "Войти"

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

{

// Получение введённых пользователем логина и пароля

string username = textBox1.Text;

string password = textBox2.Text;

// Проверка данных пользователя

if (AuthenticateUser(username, password))

{

// Если пользователь найден — сообщение об успехе и переход к следующей форме

MessageBox.Show("Успешная аутентификация! Добро пожаловать, " + username + "!");

Form f2 = new Form2(); // Создание экземпляра следующей формы

this.Hide(); // Скрытие текущей формы

f2.Show(); // Отображение новой формы

}

else

{

// Ошибка аутентификации — вывод сообщения

MessageBox.Show("Ошибка аутентификации. Пользователь с такими данными не найден.");

}

}

// Метод для проверки существования пользователя в базе данных

private bool AuthenticateUser(string username, string password)

{

// Повторное объявление строки подключения (можно убрать и использовать поле класса)

string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=Rekryt";

// Создание и открытие подключения с использованием using (гарантирует автоматическое закрытие)

using (NpgsqlConnection connection = new NpgsqlConnection(connectionString))

{

connection.Open();

// SQL-запрос с параметрами для безопасного поиска пользователя

string query = "SELECT COUNT(\*) FROM сотрудники WHERE логин = @Username AND пароль = @Password";

using (NpgsqlCommand command = new NpgsqlCommand(query, connection))

{

// Передача параметров в запрос

command.Parameters.AddWithValue("@Username", username);

command.Parameters.AddWithValue("@Password", password);

// Выполнение запроса и получение количества совпадений

int count = Convert.ToInt32(command.ExecuteScalar());

// Если есть хотя бы одно совпадение — возвращаем true

return count > 0;

}

}

}

}

На рисунке 57 результат данного кода:

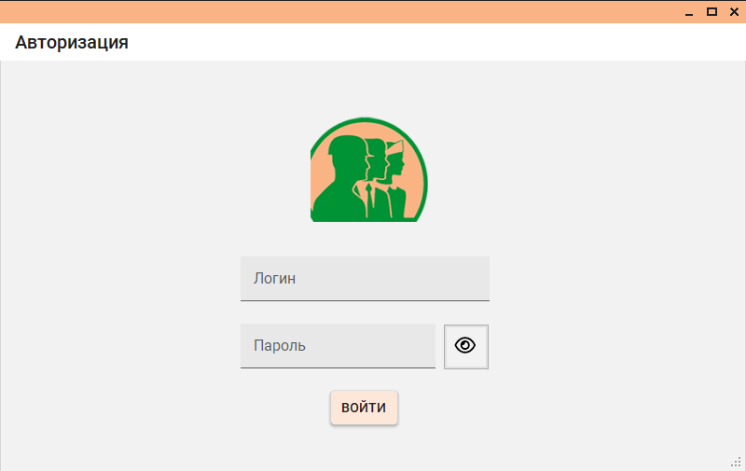


Рисунок 57 – Авторизация

Листинг кода основной формы:

using Npgsql;

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Windows.Forms;

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

namespace YCHEBKA8

{

public partial class Form2 : Form

{

// Объявление переменных для работы с базой данных

NpgsqlConnection conn; // Соединение с PostgreSQL

NpgsqlDataAdapter adapter; // Адаптер для работы с данными

DataTable dt; // Таблица для хранения данных

Dictionary<string, Dictionary<object, string>> lookupData = new Dictionary<string, Dictionary<object, string>>(); // Словарьдлясправочныхданных

public Form2()

{

InitializeComponent();

this.FormClosing += new FormClosingEventHandler(FormClosingevent);

// Строкаподключениякбазеданных

string connection = "Host=localhost;Username=postgres;Password=18273645;Database=Rekryt";

conn = new NpgsqlConnection(connection);

// Заполнение ComboBox доступнымитаблицами

comboBox1.Items.AddRange(new object[] {

"сотрудники", "работодатели", "вакансии", "соискатели",

"сделки", "статусы", "пол", "должности"

});

// Подписка на событие изменения выбранного элемента в ComboBox

comboBox1.SelectedIndexChanged += comboBox1\_SelectedIndexChanged;

}

// Обработчик изменения выбранной таблицы в ComboBox

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

string table = comboBox1.Text;

LoadLookupTables(); // Загрузкасправочныхданных

// SQL-запрос для выбора данных из таблицы

string query = $"SELECT \* FROM \"{table}\"";

adapter = new NpgsqlDataAdapter(query, conn);

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

dt = new DataTable();

adapter.Fill(dt); // Заполнение DataTable данными

// Получениепервичныхключейтаблицы

List<string> primaryKeys = GetPrimaryKeys(table);

// Настройка DataGridView

dataGridView1.Columns.Clear();

dataGridView1.DataSource = dt;

// Скрытие столбцов с первичными ключами

foreach (string pk in primaryKeys)

if (dataGridView1.Columns.Contains(pk))

dataGridView1.Columns[pk].Visible = false;

// Замена внешних ключей на выпадающие списки

ReplaceForeignKeyColumns(table);

// Настройка значений по умолчанию для новых строк

dataGridView1.DefaultValuesNeeded -= dataGridView1\_DefaultValuesNeeded;

dataGridView1.DefaultValuesNeeded += dataGridView1\_DefaultValuesNeeded;

}

// Метод для получения первичных ключей таблицы

private List<string> GetPrimaryKeys(string tableName)

{

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

// SQL-запрос для получения первичных ключей

string query = @"SELECT a.attname FROM pg\_index i JOIN pg\_attribute a ON a.attrelid = i.indrelid AND a.attnum = ANY(i.indkey) WHERE i.indrelid = @tableName::regclass AND i.indisprimary;";

using (var cmd = new NpgsqlCommand(query, conn))

{

cmd.Parameters.AddWithValue("tableName", tableName);

conn.Open();

using (var reader = cmd.ExecuteReader())

while (reader.Read())

primaryKeys.Add(reader.GetString(0));

conn.Close();

}

return primaryKeys;

}

// Обработчиккнопки "Изменитьданные"

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

{

try

{

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

adapter.Update(dt); // Сохранение изменений в базе данных

MessageBox.Show("Изменениясохранены.");

}

catch (Exception ex)

{

MessageBox.Show("Ошибкаприсохранении: " + ex.Message);

}

}

// Обработчиккнопки "Удалитьданные"

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

{

if (dataGridView1.CurrentRow != null)

{

try

{

dataGridView1.Rows.RemoveAt(dataGridView1.CurrentRow.Index);

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

adapter.Update(dt); // Удаление записи из базы данных

MessageBox.Show("Записьудалена.");

}

catch (Exception ex)

{

MessageBox.Show("Ошибкаприудалении: " + ex.Message);

}

}

else

{

MessageBox.Show("Выберите строку для удаления.");

}

}

// Метод для загрузки справочных данных

private void LoadLookupTables()

{

lookupData["работодатели"] = LoadLookup("работодатели", "id", "организация");

lookupData["соискатели"] = LoadLookup("соискатели", "id", "фио");

lookupData["сотрудники"] = LoadLookup("сотрудники", "id", "фио");

lookupData["вакансии"] = LoadLookup("вакансии", "id", "должность");

lookupData["статусы"] = LoadLookup("статусы", "id", "название");

lookupData["пол"] = LoadLookup("пол", "код", "описание");

}

// Методдлязагрузкиданныхсправочника

private Dictionary<object, string> LoadLookup(string table, string keyCol, string valCol)

{

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

string query = $"SELECT \"{keyCol}\", \"{valCol}\" FROM \"{table}\"";

using (var cmd = new NpgsqlCommand(query, conn))

{

conn.Open();

using (var reader = cmd.ExecuteReader())

while (reader.Read())

dict[reader[0]] = reader[1].ToString();

conn.Close();

}

return dict;

}

// Метод для замены столбцов с внешними ключами на выпадающие списки

private void ReplaceForeignKeyColumns(string table)

{

var fkMap = new Dictionary<string, string>();

// Определение соответствий между столбцами и справочными таблицами

if (table == "вакансии")

fkMap.Add("работодатель\_id", "работодатели");

else if (table == "соискатели")

fkMap.Add("пол", "пол");

else if (table == "вакансии")

fkMap.Add("пол", "пол");

else if (table == "сделки")

{

fkMap.Add("соискатель\_id", "соискатели");

fkMap.Add("вакансия\_id", "вакансии");

fkMap.Add("сотрудник\_id", "сотрудники");

fkMap.Add("статус\_id", "статусы");

}

// Замена столбцов на выпадающие списки

foreach (var kvp in fkMap)

{

string column = kvp.Key;

string lookupTable = kvp.Value;

if (!dt.Columns.Contains(column) || !lookupData.ContainsKey(lookupTable)) continue;

DataGridViewComboBoxColumn combo = new DataGridViewComboBoxColumn

{

Name = column,

DataPropertyName = column,

HeaderText = column,

DataSource = new BindingSource(lookupData[lookupTable], null),

DisplayMember = "Value",

ValueMember = "Key",

FlatStyle = FlatStyle.Flat

};

int index = dataGridView1.Columns[column].Index;

dataGridView1.Columns.Remove(column);

dataGridView1.Columns.Insert(index, combo);

}

}

// Обработчик события установки значений по умолчанию для новой строки

private void dataGridView1\_DefaultValuesNeeded(object sender, DataGridViewRowEventArgs e)

{

string table = comboBox1.Text;

if (table == "сделки")

{

e.Row.Cells["дата"].Value = DateTime.Today;

if (lookupData["статусы"].Count > 0)

e.Row.Cells["статус\_id"].Value = lookupData["статусы"].First().Key;

}

}

// Обработчик кнопки "Открыть список претендентов"

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

{

this.Hide();

Form3 form3 = new Form3();

form3.Show();

}

// Обработчик кнопки (закомментирован)

private void button4\_Click(object sender, EventArgs e)

{

/\*this.Hide();

Form4 form4 = new Form4();

form4.Show();\*/

}

// Обработчикзакрытияформы

private void FormClosingevent(object sender, FormClosingEventArgs e)

{

Application.Exit();

}

}

}

Нарисунках58, 59результатработыданногокода:

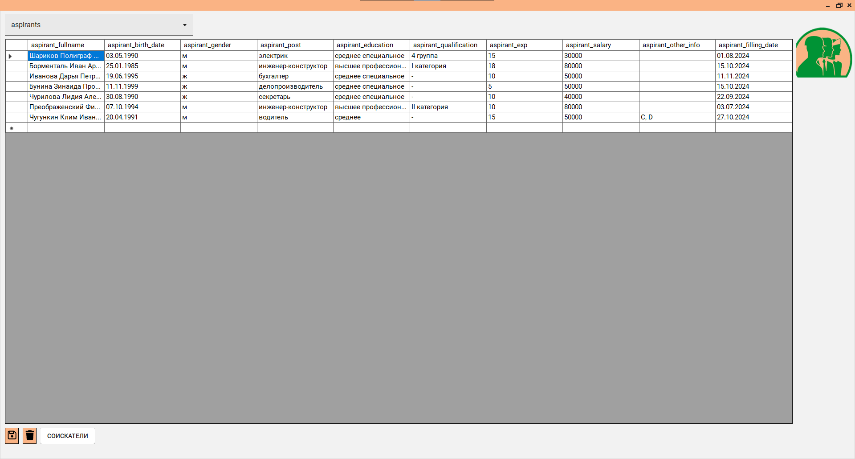


Рисунок 58 – Результат работы приложения

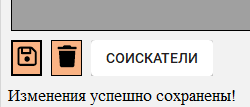


Рисунок 59 – Результат работы приложения

Листинг кода для списка претендентов:

using System;

using System.Data;

using System.Windows.Forms;

using Npgsql;

namespace YCHEBKA8

{

public partial class Form3 : Form

{

// Строка подключения к базе данных PostgreSQL

private string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=Rekryt";

publicForm3()

{

InitializeComponent();

// Подписка на событие закрытия формы

this.FormClosing += new FormClosingEventHandler(FormClosingevent);

// Загрузка данных при инициализации формы

LoadEnterprises();

LoadJobTitles();

}

// Метод загрузки предприятий из базы данных

private void LoadEnterprises()

{

comboBoxEnterprise.Items.Clear();

using (var conn = new NpgsqlConnection(connectionString))

{

conn.Open();

// SQL-запрос для получения списка работодателей

varcmd = newNpgsqlCommand("SELECTid, организация, адрес FROM работодатели", conn);

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

// Добавление предприятий в ComboBox с сохранением ID и адреса

comboBoxEnterprise.Items.Add(new ComboBoxItem(

reader.GetString(1),

reader.GetInt32(0),

reader.IsDBNull(2) ? "Адрес не указан" : reader.GetString(2))); // Обработка NULL для адреса

}

}

}

}

// Метод загрузки должностей из базы данных

private void LoadJobTitles()

{

comboBoxPosition.Items.Clear();

using (var conn = new NpgsqlConnection(connectionString))

{

conn.Open();

// SQL-запрос для получения списка должностей

var cmd = new NpgsqlCommand("SELECT id, название FROM должности", conn);

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

// Добавление должностей в ComboBox

comboBoxPosition.Items.Add(reader.GetString(1));

}

}

}

}

// Обработчик изменения выбранного предприятия

private void comboBoxEnterprise\_SelectedIndexChanged(object sender, EventArgs e)

{

var selected = comboBoxEnterprise.SelectedItem as ComboBoxItem;

if (selected != null)

{

// Отображение адреса выбранного предприятия

textBoxAddress.Text = selected.Metadata;

}

}

// Метод для безопасного получения строки из DataReader с обработкой NULL

private string GetSafeString(NpgsqlDataReader reader, string columnName)

{

return reader.IsDBNull(reader.GetOrdinal(columnName)) ?

"Данные отсутствуют" :

reader[columnName].ToString();

}

// Метод для безопасного получения числового значения из DataReader с обработкой NULL

private string GetSafeInt(NpgsqlDataReader reader, string columnName)

{

return reader.IsDBNull(reader.GetOrdinal(columnName)) ?

"Не указано" :

reader[columnName].ToString();

}

// Метод для безопасного получения даты из DataReader с обработкой NULL

private string GetSafeDate(NpgsqlDataReader reader, string columnName)

{

return reader.IsDBNull(reader.GetOrdinal(columnName)) ?

"Дата не указана" :

Convert.ToDateTime(reader[columnName]).ToShortDateString();

}

// Обработчик кнопки "Показать претендентов"

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

{

// Очистка панели с результатами

flowLayoutPanelApplicants.Controls.Clear();

var selectedEnterprise = comboBoxEnterprise.SelectedItem as ComboBoxItem;

var selectedPosition = comboBoxPosition.SelectedItem;

// Проверка выбора предприятия

if (selectedEnterprise == null)

{

MessageBox.Show("Пожалуйста, выберите предприятие.");

return;

}

// Проверка выбора должности

if (selectedPosition == null)

{

MessageBox.Show("Пожалуйста, выберите должность.");

return;

}

try

{

using (var conn = new NpgsqlConnection(connectionString))

{

conn.Open();

// SQL-запрос для получения соискателей по выбранным критериям

string query = @"

SELECT с.фио,

EXTRACT(YEAR FROM age(с.дата\_рождения)) AS возраст,

п.описание AS пол,

с.должность,

с.образование,

с.квалификация,

с.зарплата,

с.доп\_информация,

с.дата\_заполнения,

ст.название AS статус,

с.id,

р.адрес,

COALESCE(с.телефон, 'Телефон не указан') AS телефон,

в.должность AS вакансия\_должность

FROM сделки d

JOIN соискатели с ONd.соискатель\_id = с.id

JOIN вакансии в ONd.вакансия\_id = в.id

JOIN работодатели р ON в.работодатель\_id = р.id

JOIN пол п ON с.пол = п.код

JOIN статусы ст ONd.статус\_id = ст.id

WHERE р.id = @enterpriseId AND в.должность = @position";

using (var cmd = new NpgsqlCommand(query, conn))

{

// Параметры запроса

cmd.Parameters.AddWithValue("enterpriseId", selectedEnterprise.Value);

cmd.Parameters.AddWithValue("position", selectedPosition.ToString());

using (var reader = cmd.ExecuteReader())

{

// Проверка наличия результатов

if (!reader.HasRows)

{

MessageBox.Show("Не найдено ни одного соискателя по выбранным критериям.");

return;

}

// Обработка результатов запроса

while (reader.Read())

{

// Создание панели для отображения информации о соискателе

var panel = new Panel

{

Width = 750,

Height = 120,

BorderStyle = BorderStyle.FixedSingle,

Margin = new Padding(5) // Добавляем отступ между панелями

};

// Формирование текста с проверкой на NULL для всех полей

string infoText = $"ФИО: {GetSafeString(reader, "фио")}\n" +

$"Возраст: {GetSafeInt(reader, "возраст")}, Пол: {GetSafeString(reader, "пол")}\n" +

$"Должность соискателя: {GetSafeString(reader, "должность")}\n" +

$"Вакансия: {GetSafeString(reader, "вакансия\_должность")}\n" +

$"Образование: {GetSafeString(reader, "образование")}, Квалификация: {GetSafeString(reader, "квалификация")}\n" +

$"Желаемая зарплата: {GetSafeInt(reader, "зарплата")} руб.\n" +

$"Контактный телефон: {GetSafeString(reader, "телефон")}\n" +

$"Адрес работодателя: {GetSafeString(reader, "адрес")}\n" +

$"Статус: {GetSafeString(reader, "статус")}, Дата заполнения: {GetSafeDate(reader, "дата\_заполнения")}\n" +

$"Доп. информация: {GetSafeString(reader, "доп\_информация")}";

// Создание метки с информацией о соискателе

var label = new Label

{

AutoSize = false,

Width = panel.Width - 20,

Height = panel.Height - 10,

Location = new System.Drawing.Point(10, 10),

Text = infoText

};

// Добавление метки на панель

panel.Controls.Add(label);

// Добавление панели в FlowLayoutPanel

flowLayoutPanelApplicants.Controls.Add(panel);

}

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке данных: " + ex.Message);

}

}

// Вспомогательный класс для хранения данных в ComboBox

private class ComboBoxItem

{

public string Text { get; } // Отображаемый текст

public int Value { get; } // ID элемента

publicstringMetadata { get; } // Дополнительные данные (адрес)

public ComboBoxItem(string text, int value, string metadata)

{

Text = text;

Value = value;

Metadata = metadata;

}

// Переопределение метода для отображения текста

publicoverridestringToString()

{

return Text;

}

}

// Обработчик закрытия формы

private void FormClosingevent(object sender, FormClosingEventArgs e)

{

Application.Exit();

}

// Обработчик кнопки (назад)

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

{

this.Hide();

Form2 form2 = new Form2();

form2.Show();

}

}

}

Нарисунке60список претендентов:

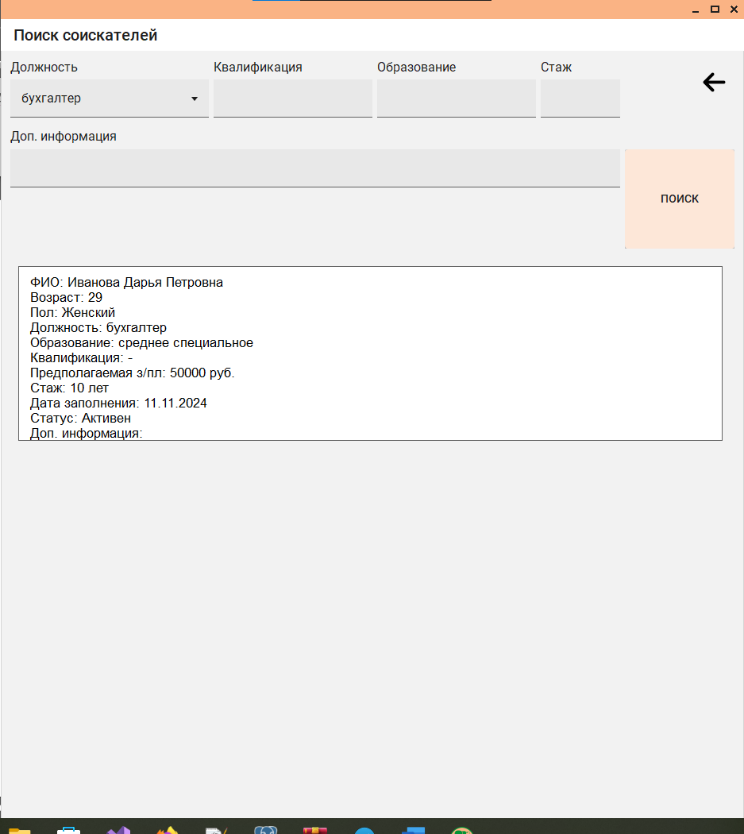


Рисунок 60 – Список претендентов

Работа № 9

Задание:Разработать подсистему для работы рыболовной базы «Золотая рыбка», обеспечивающую следующий функционал:

– оформление оплаты посетителями;

– добавление/редактирование данных клиентов;

– просмотр данных посетителей по условию (дата, клуб и др.).

Получить ER-диаграмму средствами СУБД: ER-диаграмма должна быть представлена в формате PDF и содержать таблицы, связи между ними, атрибуты и ключи (типами данных на данном этапе можно пренебречь).

Разработать программный модуль для учета соискателей. Необходимо реализовать вывод списка претендентов на должность вакансии, информация о которых хранятся в базе данных, согласно предоставленному макету:

|  |  |  |
| --- | --- | --- |
|  | Клуб Дата |  |
|  | ФИО посетителя  Возраст  Клуб  Дата последнего заезда  Количество посещений  Общая сумма % скидки |  |
|  |  |  |
|  | ФИО посетителя  Возраст  Клуб  Дата последнего заезда  Количество посещений  Общая сумма % скидки |  |
|  |  |  |
|  | ФИО посетителя  Возраст  Клуб  Дата последнего заезда  Количество посещений  Общая сумма % скидки |  |
|  |  |  |

Величина скидки для посетителя рассчитывается на основании суммы оплаты за весь период работы. Скидка зависит от общего количества визитов и суммы оплаты и составляет: до 5000 – 0%, от 5000 – до 7000 – 5%, от 7000 – до 10000 – 10%, более 10000 – 15%.

Разработать интерфейс программного модуля для работы с посетителями.

Реализовать функции добавления/редактирования данных посетителя в новом окне (странице) – форме для добавления/редактирования данных клиентов. Переходы на эту форму должны быть реализованы из формы списка клиентов: для редактирования клиентов из списка – при нажатии на конкретный элемент, для редактирования/добавления любого клиента рыболовной базы – при нажатии кнопки.

На форме для добавления/редактирования клиентов должны быть предусмотрены следующие поля: ФИО клиента, дата рождения, пол, рыболовный клуб.

**Ход работы**

Подготовим данные для импорта. Для этого приведём все данные в предоставленных для импорта таблицах к 3НФ (Рисунок 61).

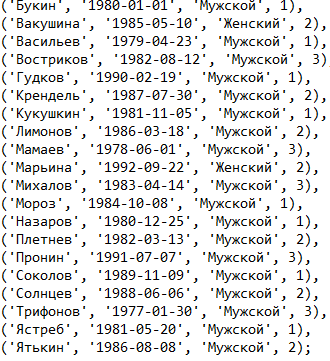


Рисунок 61 – Подготовка данных для импорта

Теперь необходимо создать базу данных (Рисунок 62)

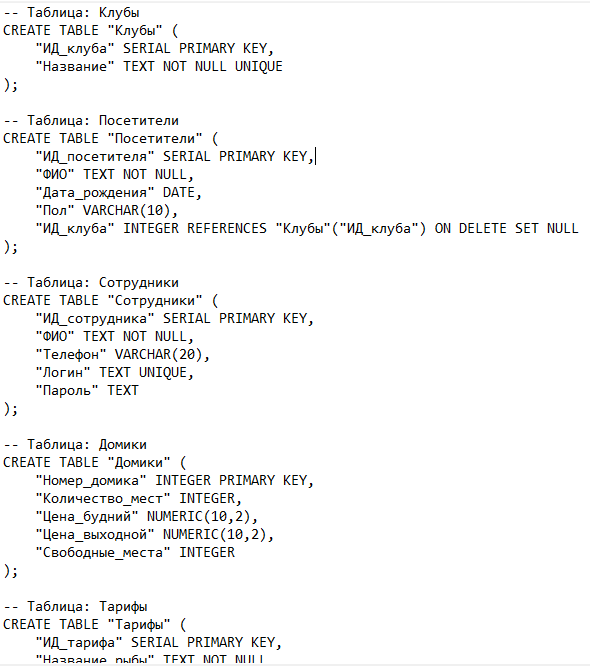


Рисунок 62 – Скрипт для создания БД

Теперь сделаем импорт данных (Рисунок 63)

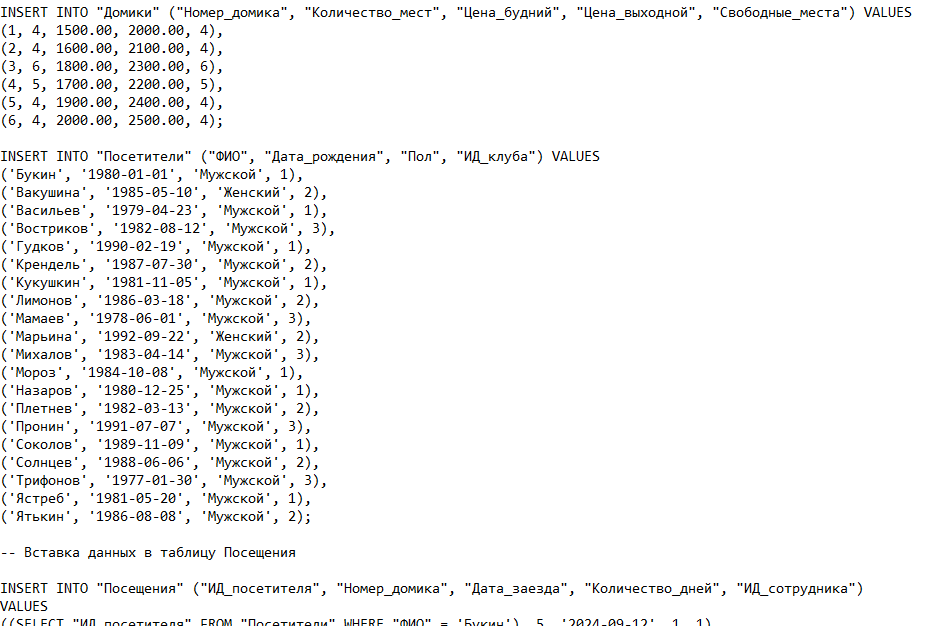


Рисунок 63 – Импорт данных в БД

Листинг кода авторизации:

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;

// Подключениебиблиотекидляработыс PostgreSQL

using Npgsql;

namespace YCHEBKA9

{

// Основнойклассформыавторизации

public partial class Form1: Form

{

// Объектсоединениясбазойданных

private NpgsqlConnection connection;

// Строкаподключениякбазеданных PostgreSQL

private string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=Zolotaya\_rb";

// Конструктор формы

public Form1()

{

InitializeComponent(); // Инициализация компонентов формы

// Создание и открытие подключения к базе данных при запуске формы

NpgsqlConnection connection = new NpgsqlConnection(connectionString);

connection.Open(); // ВАЖНО: Это локальная переменная, которая перекрывает поле. Лучше использовать this.connection

}

// Обработчик события нажатия на кнопку "Войти"

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

{

// Получение введённых пользователем логина и пароля

string username = textBox1.Text;

string password = textBox2.Text;

// Проверка данных пользователя

if (AuthenticateUser(username, password))

{

// Если пользователь найден — сообщение об успехе и переход к следующей форме

MessageBox.Show("Успешная аутентификация! Добро пожаловать, " + username + "!");

Form f2 = new Form2(); // Создание экземпляра следующей формы

this.Hide(); // Скрытие текущей формы

f2.Show(); // Отображение новой формы

}

else

{

// Ошибка аутентификации — вывод сообщения

MessageBox.Show("Ошибка аутентификации. Пользователь с такими данными не найден.");

}

}

// Метод для проверки существования пользователя в базе данных

private bool AuthenticateUser(string username, string password)

{

// Повторное объявление строки подключения (можно убрать и использовать поле класса)

string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=Zolotaya\_rb";

// Создание и открытие подключения с использованием using (гарантирует автоматическое закрытие)

using (NpgsqlConnection connection = new NpgsqlConnection(connectionString))

{

connection.Open();

// SQL-запрос с параметрами для безопасного поиска пользователя

string query = "SELECT COUNT(\*) FROM Сотрудники WHERE Логин = @Username AND Пароль = @Password";

using (NpgsqlCommand command = new NpgsqlCommand(query, connection))

{

// Передача параметров в запрос

command.Parameters.AddWithValue("@Username", username);

command.Parameters.AddWithValue("@Password", password);

// Выполнение запроса и получение количества совпадений

int count = Convert.ToInt32(command.ExecuteScalar());

// Если есть хотя бы одно совпадение — возвращаем true

return count > 0;

}

}

}

}

}

На рисунке 64 авторизация:

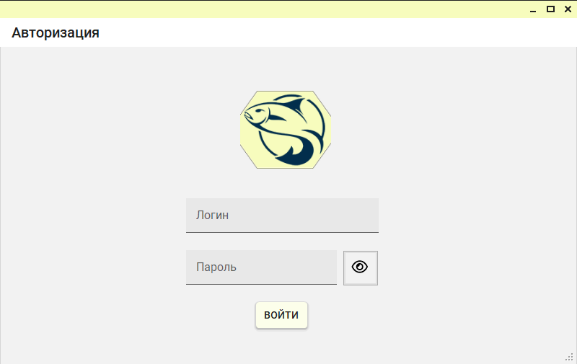


Рисунок 64 – Авторизация

Листинг кода основной формы:

using Npgsql; // Для работы с PostgreSQL

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Windows.Forms;

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

using YCHEBKA9;

namespace YCHEBKA9

{

public partial class Form2 : Form

{

NpgsqlConnection conn; // ПодключениекБД

NpgsqlDataAdapter adapter; // Адаптердляработысданными

DataTable dt; // Таблица для хранения данных

// Словарь для хранения справочных данных (для подстановок в ComboBox)

Dictionary<string, Dictionary<object, string>> lookupData = new Dictionary<string, Dictionary<object, string>>();

public Form2()

{

InitializeComponent();

this.FormClosing += new FormClosingEventHandler(FormClosingevent); // Обработчикзакрытияформы

// СтрокаподключениякБД

string connection = "Host=localhost;Username=postgres;Password=18273645;Database=Zolotaya\_rb";

conn = new NpgsqlConnection(connection);

LoadTablesToComboBox(); // Загружаемсписоктаблицв ComboBox

// Подписываемся на событие изменения выбранной таблицы

comboBox1.SelectedIndexChanged += comboBox1\_SelectedIndexChanged;

}

// Метод для загрузки всех таблиц из схемы public в ComboBox

private void LoadTablesToComboBox()

{

try

{

conn.Open();

// SQL запрос для получения списка таблиц

string sql = @"

SELECT table\_name

FROM information\_schema.tables

WHERE table\_schema='public' AND table\_type='BASE TABLE'

ORDER BY table\_name;

";

using (var cmd = new NpgsqlCommand(sql, conn))

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

comboBox1.Items.Add(reader.GetString(0)); // Добавляемтаблицыв ComboBox

}

}

}

finally

{

conn.Close(); // Всегда закрываем соединение

}

}

// Обработчик изменения выбранной таблицы в ComboBox

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

string table = comboBox1.Text;

if (string.IsNullOrEmpty(table)) return;

LoadLookupTables(); // Загружаем справочные данные для подстановок

// Запрос для получения данных из выбранной таблицы

string query = $"SELECT \* FROM \"{table}\"";

adapter = new NpgsqlDataAdapter(query, conn);

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

dt = new DataTable();

adapter.Fill(dt); // Заполняем DataTable данными

// Настраиваем DataGridView

dataGridView1.Columns.Clear();

dataGridView1.DataSource = dt;

// Скрываем первичные ключи и заменяем внешние ключи на ComboBox

HidePrimaryKeyColumns(table);

ReplaceForeignKeyColumns(table);

// Настраиваем обработчик для установки значений по умолчанию при добавлении новой строки

dataGridView1.DefaultValuesNeeded -= dataGridView1\_DefaultValuesNeeded;

dataGridView1.DefaultValuesNeeded += dataGridView1\_DefaultValuesNeeded;

}

// Метод для загрузки справочных данных (для внешних ключей)

private void LoadLookupTables()

{

lookupData.Clear();

// Загружаем данные из справочных таблиц

// Формат: Имя таблицы, поле ключа, поле значения для отображения

lookupData["Клубы"] = LoadLookup("Клубы", "ИД\_клуба", "Название");

lookupData["Посетители"] = LoadLookup("Посетители", "ИД\_посетителя", "ФИО");

lookupData["Сотрудники"] = LoadLookup("Сотрудники", "ИД\_сотрудника", "ФИО");

lookupData["Домики"] = LoadLookup("Домики", "Номер\_домика", "Номер\_домика");

lookupData["Тарифы"] = LoadLookup("Тарифы", "ИД\_тарифа", "Название\_рыбы");

}

// Вспомогательный метод для загрузки данных справочной таблицы

private Dictionary<object, string> LoadLookup(string table, string keyCol, string valCol)

{

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

string sql = $"SELECT \"{keyCol}\", \"{valCol}\" FROM \"{table}\"";

try

{

conn.Open();

using (var cmd = new NpgsqlCommand(sql, conn))

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

dict[reader[0]] = reader[1].ToString(); // Заполняемсловарь: ключ - значение

}

}

}

finally

{

conn.Close();

}

return dict;

}

// Метод для скрытия столбцов с первичными ключами

private void HidePrimaryKeyColumns(string table)

{

// Словарь: таблица - массив имен полей первичных ключей

Dictionary<string, string[]> pkFields = new Dictionary<string, string[]>()

{

{"Клубы", new[] {"ИД\_клуба"}},

{"Посетители", new[] {"ИД\_посетителя"}},

{"Сотрудники", new[] {"ИД\_сотрудника"}},

{"Домики", new[] {"Номер\_домика"}},

{"Тарифы", new[] {"ИД\_тарифа"}},

{"Посещения", new[] {"ИД\_посещения"}},

{"Улов", new[] {"ИД\_улова"}}

};

if (!pkFields.ContainsKey(table)) return;

// Скрываемуказанныестолбцы

foreach (var pk in pkFields[table])

{

if (dataGridView1.Columns.Contains(pk))

dataGridView1.Columns[pk].Visible = false;

}

}

// Метод для замены столбцов с внешними ключами на ComboBox

private void ReplaceForeignKeyColumns(string table)

{

// Словарь: таблица - словарь (поле внешнего ключа - справочная таблица)

Dictionary<string, Dictionary<string, string>> fkMap = new Dictionary<string, Dictionary<string, string>>()

{

{ "Посетители", new Dictionary<string, string> { { "ИД\_клуба", "Клубы" } } },

{ "Посещения", new Dictionary<string, string> {

{ "ИД\_посетителя", "Посетители" },

{ "Номер\_домика", "Домики" },

{ "ИД\_сотрудника", "Сотрудники" }

} },

{ "Улов", new Dictionary<string, string> {

{ "ИД\_посещения", "Посещения" }

} }

};

if (!fkMap.ContainsKey(table)) return;

// Для каждого внешнего ключа создаем ComboBox

foreach (var kvp in fkMap[table])

{

string fkColumn = kvp.Key;

string lookupTable = kvp.Value;

if (!dt.Columns.Contains(fkColumn) || !lookupData.ContainsKey(lookupTable))

continue;

// Создаемколонкутипа ComboBox

var combo = new DataGridViewComboBoxColumn

{

Name = fkColumn,

DataPropertyName = fkColumn,

HeaderText = fkColumn,

FlatStyle = FlatStyle.Flat,

DataSource = new BindingSource(lookupData[lookupTable], null), // Данныеизсправочника

DisplayMember = "Value", // Чтопоказываем

ValueMember = "Key" // Чтохраним

};

// Заменяемтекстовуюколонкуна ComboBox

int idx = dataGridView1.Columns[fkColumn].Index;

dataGridView1.Columns.Remove(fkColumn);

dataGridView1.Columns.Insert(idx, combo);

}

}

// Обработчик для установки значений по умолчанию при добавлении новой строки

private void dataGridView1\_DefaultValuesNeeded(object sender, DataGridViewRowEventArgs e)

{

string table = comboBox1.Text;

// Для таблицы "Посещения" устанавливаем значения по умолчанию

if (table == "Посещения")

{

// Устанавливаем первого посетителя, домик и сотрудника по умолчанию

if (lookupData.ContainsKey("Посетители") && lookupData["Посетители"].Count > 0)

e.Row.Cells["ИД\_посетителя"].Value = lookupData["Посетители"].First().Key;

if (lookupData.ContainsKey("Домики") && lookupData["Домики"].Count > 0)

e.Row.Cells["Номер\_домика"].Value = lookupData["Домики"].First().Key;

if (lookupData.ContainsKey("Сотрудники") && lookupData["Сотрудники"].Count > 0)

e.Row.Cells["ИД\_сотрудника"].Value = lookupData["Сотрудники"].First().Key;

// Устанавливаем текущую дату и 1 день по умолчанию

e.Row.Cells["Дата\_заезда"].Value = DateTime.Today;

e.Row.Cells["Количество\_дней"].Value = 1;

}

}

// Обработчиккнопки "Сохранить"

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

{

try

{

// Создаем команды для обновления данных

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

adapter.UpdateCommand = builder.GetUpdateCommand();

adapter.InsertCommand = builder.GetInsertCommand();

adapter.DeleteCommand = builder.GetDeleteCommand();

adapter.Update(dt); // СохраняемизменениявБД

MessageBox.Show("Изменениясохранены.");

}

catch (Exception ex)

{

MessageBox.Show("Ошибкаприсохранении: " + ex.Message);

}

}

// Обработчиккнопки "Удалить"

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

{

if (dataGridView1.CurrentRow != null)

{

try

{

// Удаляемвыбраннуюстроку

dataGridView1.Rows.RemoveAt(dataGridView1.CurrentRow.Index);

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

adapter.Update(dt); // Сохраняем изменения в БД

MessageBox.Show("Записьудалена.");

}

catch (Exception ex)

{

MessageBox.Show("Ошибкаприудалении: " + ex.Message);

}

}

else

{

MessageBox.Show("Выберите строку для удаления.");

}

}

// Обработчик кнопки для перехода на Form3

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

{

this.Hide(); // Скрываем текущую форму

Form3 form3 = new Form3(); // Создаем новую форму

form3.Show(); // Показываем новую форму

}

// Обработчикзакрытияформы

private void FormClosingevent(object sender, FormClosingEventArgs e)

{

Application.Exit(); // Завершаем приложение

}

}

}

На рисунках 65, 66 работа основной формы:

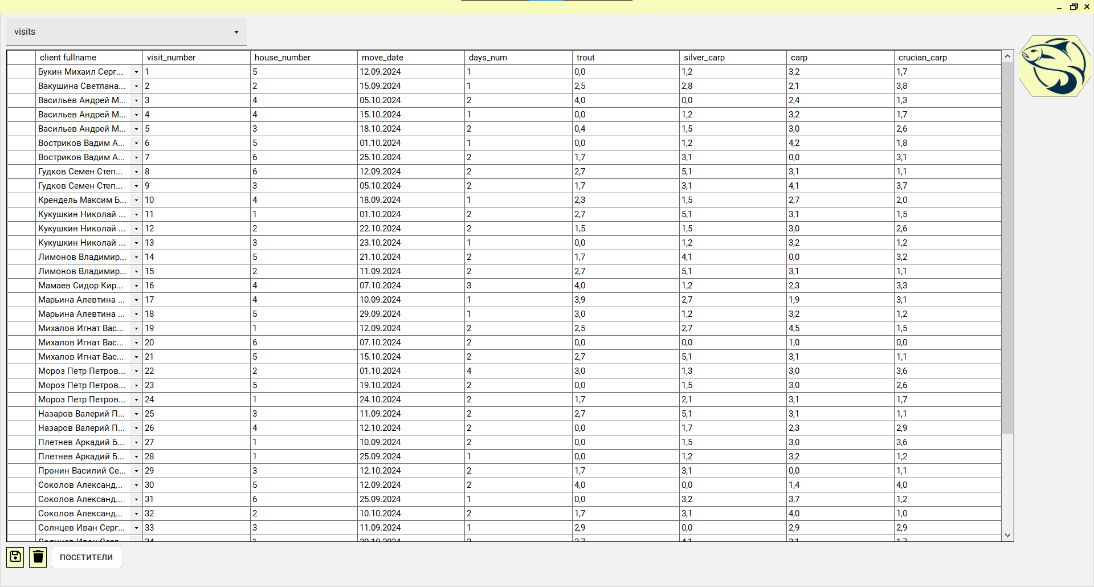


Рисунок 65 – Результат работы формы

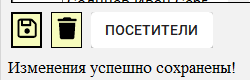


Рисунок 66 – Результат работы формы

Листинг кода формы списка клиентов:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Windows.Forms;

using Npgsql;

namespace YCHEBKA9

{

public partial class Form3 : Form

{

private NpgsqlConnection connection; // Подключениекбазеданных PostgreSQL

public Form3()

{

InitializeComponent();

InitializeDatabaseConnection(); // ИнициализацияподключениякБД

LoadClubs(); // Загрузка списка клубов

LoadDates(); // Загрузка списка дат

LoadVisitors(); // Загрузка информации о посетителях

}

// Инициализация подключения к базе данных

private void InitializeDatabaseConnection()

{

string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=Zolotaya\_rb";

connection = new NpgsqlConnection(connectionString);

}

// Загрузкаспискаклубовизбазыданных

private void LoadClubs()

{

try

{

connection.Open();

string query = "SELECT \"Название\" FROM \"Клубы\" ORDER BY \"Название\"";

using (var command = new NpgsqlCommand(query, connection))

using (var reader = command.ExecuteReader())

{

comboBoxClub.Items.Clear();

comboBoxClub.Items.Add("Всеклубы"); // Добавляемвариант "Всеклубы"

// Заполняем комбобокс названиями клубов

while (reader.Read())

{

comboBoxClub.Items.Add(reader["Название"].ToString());

}

comboBoxClub.SelectedIndex = 0; // Выбираем первый элемент по умолчанию

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке клубов: " + ex.Message);

}

finally

{

if (connection.State == ConnectionState.Open)

connection.Close(); // Закрываем подключение в любом случае

}

}

// Загрузка списка дат заездов из базы данных

private void LoadDates()

{

try

{

connection.Open();

string query = "SELECT DISTINCT \"Дата\_заезда\" FROM \"Посещения\" ORDER BY \"Дата\_заезда\" DESC";

using (var command = new NpgsqlCommand(query, connection))

using (var reader = command.ExecuteReader())

{

comboBoxDate.Items.Clear();

comboBoxDate.Items.Add("Вседаты"); // Добавляемвариант "Вседаты"

// Заполняемкомбобоксдатами

while (reader.Read())

{

comboBoxDate.Items.Add(Convert.ToDateTime(reader["Дата\_заезда"]).ToShortDateString());

}

comboBoxDate.SelectedIndex = 0; // Выбираем первый элемент по умолчанию

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке дат: " + ex.Message);

}

finally

{

if (connection.State == ConnectionState.Open)

connection.Close(); // Закрываем подключение в любом случае

}

}

// Загрузка и отображение информации о посетителях

private void LoadVisitors()

{

flowLayoutPanel1.Controls.Clear(); // Очищаем панель перед загрузкой новых данных

if (connection == null) return;

try

{

connection.Open();

// Основной SQL-запрос для получения информации о посетителях

string query = @"

SELECT

п.""ФИО"",

EXTRACT(YEAR FROM AGE(CURRENT\_DATE, п.""Дата\_рождения"")) AS ""Возраст"",

COALESCE(к.""Название"", 'Без клуба') AS ""Клуб"",

MAX(пос.""Дата\_заезда"") AS ""Дата\_заезда"",

COUNT(DISTINCT пос.""ИД\_посещения"") AS ""Количество\_посещений"",

COALESCE(SUM(д.""Цена\_будний"" \* пос.""Количество\_дней""), 0) AS ""Общая\_сумма""

FROM ""Посетители"" п

LEFT JOIN ""Клубы"" к ON п.""ИД\_клуба"" = к.""ИД\_клуба""

LEFT JOIN ""Посещения"" пос ON п.""ИД\_посетителя"" = пос.""ИД\_посетителя""

LEFT JOIN ""Домики"" д ON пос.""Номер\_домика"" = д.""Номер\_домика""

WHERE 1=1";

// Добавляем условия фильтрации, если выбраны конкретные клуб или дата

if (comboBoxClub.SelectedIndex > 0)

{

query += " AND к.\"Название\" = @ClubName";

}

if (comboBoxDate.SelectedIndex > 0)

{

query += " AND пос.\"Дата\_заезда\"::date = @VisitDate::date";

}

query += @"

GROUP BY п.""ИД\_посетителя"", п.""ФИО"", п.""Дата\_рождения"", к.""Название""

ORDER BY п.""ФИО""";

using (var command = new NpgsqlCommand(query, connection))

{

// Добавляемпараметры, еслионинужны

if (comboBoxClub.SelectedIndex > 0)

{

command.Parameters.AddWithValue("@ClubName", comboBoxClub.SelectedItem.ToString());

}

if (comboBoxDate.SelectedIndex > 0)

{

command.Parameters.AddWithValue("@VisitDate", DateTime.Parse(comboBoxDate.SelectedItem.ToString()));

}

using (var reader = command.ExecuteReader())

{

// Для каждого посетителя создаем панель с информацией

while (reader.Read())

{

Panel visitorPanel = new Panel

{

Width = flowLayoutPanel1.Width - 25,

Height = 100,

BorderStyle = BorderStyle.FixedSingle,

Margin = new Padding(5)

};

// Получаем данные из результата запроса

decimal totalAmount = reader["Общая\_сумма"] != DBNull.Value ? Convert.ToDecimal(reader["Общая\_сумма"]) : 0;

int visitCount = Convert.ToInt32(reader["Количество\_посещений"]);

decimal discount = CalculateDiscount(totalAmount); // Рассчитываемскидку

string lastVisitDate = reader["Дата\_заезда"] != DBNull.Value ?

Convert.ToDateTime(reader["Дата\_заезда"]).ToShortDateString() : "Нетпосещений";

// Создаем элементы управления для отображения информации

Label nameLabel = new Label

{

Text = reader["ФИО"].ToString(),

Location = new Point(10, 10),

AutoSize = true,

Font = new Font(Font, FontStyle.Bold)

};

Label ageLabel = new Label

{

Text = $"Возраст: {reader["Возраст"]}",

Location = new Point(10, 30),

AutoSize = true

};

Label clubLabel = new Label

{

Text = $"Клуб: {reader["Клуб"]}",

Location = new Point(10, 50),

AutoSize = true

};

Label dateLabel = new Label

{

Text = $"Датапоследнегозаезда: {lastVisitDate}",

Location = new Point(200, 10),

AutoSize = true

};

Label visitsLabel = new Label

{

Text = $"Количествопосещений: {visitCount}",

Location = new Point(200, 30),

AutoSize = true

};

Label amountLabel = new Label

{

Text = $"Общаясумма: {totalAmount:F2} руб.",

Location = new Point(200, 50),

AutoSize = true

};

Label discountLabel = new Label

{

Text = $"Скидка: {discount}%",

Location = new Point(400, 10),

AutoSize = true,

ForeColor = Color.Green,

Font = new Font(Font, FontStyle.Bold)

};

// Добавляемэлементыуправлениянапанель

visitorPanel.Controls.Add(nameLabel);

visitorPanel.Controls.Add(ageLabel);

visitorPanel.Controls.Add(clubLabel);

visitorPanel.Controls.Add(dateLabel);

visitorPanel.Controls.Add(visitsLabel);

visitorPanel.Controls.Add(amountLabel);

visitorPanel.Controls.Add(discountLabel);

// Добавляемпанельв flowLayoutPanel

flowLayoutPanel1.Controls.Add(visitorPanel);

}

}

}

}

catch (Exception ex)

{

// Обработка ошибок (пустой блок, что не рекомендуется)

}

finally

{

if (connection.State == ConnectionState.Open)

connection.Close(); // Закрываем подключение в любом случае

}

}

// Расчет скидки на основе общей суммы расходов

private decimal CalculateDiscount(decimal totalAmount)

{

if (totalAmount > 10000) return 15;

if (totalAmount > 7000) return 10;

if (totalAmount > 5000) return 5;

return 0;

}

// Обработчик изменения выбранного клуба

private void comboBoxClub\_SelectedIndexChanged(object sender, EventArgs e)

{

LoadVisitors(); // Перезагружаем список посетителей при изменении фильтра

}

// Обработчикизменениявыбраннойдаты

private void comboBoxDate\_SelectedIndexChanged(object sender, EventArgs e)

{

LoadVisitors(); // Перезагружаем список посетителей при изменении фильтра

}

// Обработчиккнопки "Назад"

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

{

this.Hide(); // Скрываемтекущуюформу

Form2 form2 = new Form2();

form2.Show(); // Показываем форму 2

}

// Обработчик закрытия формы

protected override void OnFormClosing(FormClosingEventArgs e)

{

base.OnFormClosing(e);

if (e.CloseReason == CloseReason.UserClosing)

{

Application.Exit(); // Завершаем приложение при закрытии формы

}

}

// Обработчикизмененияразмераформы

private void Form3\_Resize(object sender, EventArgs e)

{

// Изменяем ширину всех панелей с посетителями при изменении размера формы

foreach (Control control in flowLayoutPanel1.Controls)

{

if (control is Panel panel)

{

panel.Width = flowLayoutPanel1.Width - 25;

}

}

}

}

}

На рисунке 67список клиентов:

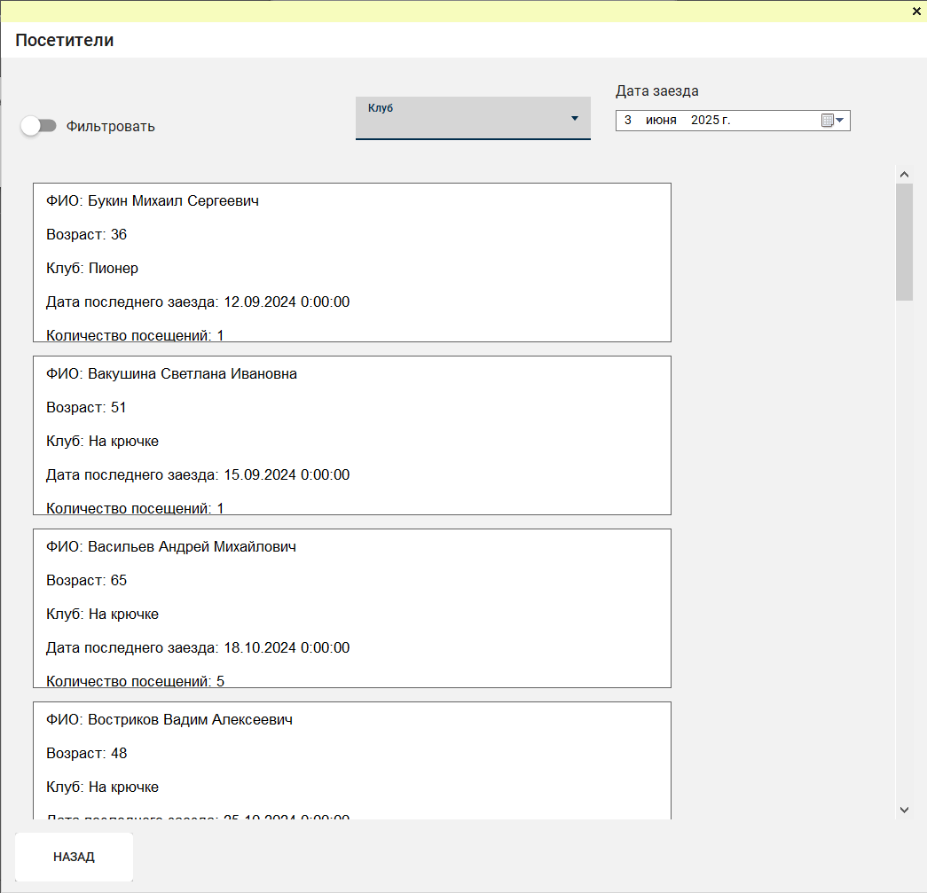


Рисунок 67 – Список клиентов

Работа № 10

Задание: Разработать подсистему для работы с клиентами компании по грузоперевозкам, обеспечивающую следующий функционал:

– просмотр списка клиентов;

– добавление/редактирование данных клиентов;

– просмотр истории взаимодействия с клиентами.

Получить ER-диаграмму средствами СУБД: ER-диаграмма должна быть представлена в формате PDF и содержать таблицы, связи между ними, атрибуты и ключи (типами данных на данном этапе можно пренебречь).

Разработать программный модуль для учета клиентов компании. Необходимо реализовать вывод списка клиентов, информация о которых хранятся в базе данных, согласно предоставленному макету:

|  |  |  |
| --- | --- | --- |
|  | Клуб Дата |  |
|  | Тип и наименование компании клиента  ФИО представителя  Телефон  Количество перевозок  Общая сумма % скидки |  |
|  |  |  |
|  | Тип и наименование компании клиента  ФИО представителя  Телефон  Количество перевозок  Общая сумма % скидки |  |
|  |  |  |
|  | Тип и наименование компании клиента  ФИО представителя  Телефон  Количество перевозок  Общая сумма % скидки |  |
|  |  |  |

Величина скидки для заказчика рассчитывается на основании суммы оплаты за весь период работы. Скидка зависит от общего количества заказов и суммы оплаты и составляет: до 10000 – 0%, от 10000 – до 50000 – 5%, от 50000 – до 100000 – 10%, более 100000 – 15%.

Реализовать функции добавления/редактирования данных клиента в новом окне (странице) – форме для добавления/редактирования данных клиентов. Переходы на эту форму должны быть реализованы из формы списка клиентов: для редактирования клиентов из списка – при нажатии на конкретный элемент, для редактирования/добавления любого клиента – при нажатии кнопки.

На форме для добавления/редактирования клиентов должны быть предусмотрены поля согласно предметной области.

**Ход работы**

Подготовим данные для импорта. Для этого приведём все данные в предоставленных для импорта таблицах к 3НФ (Рисунок 68).

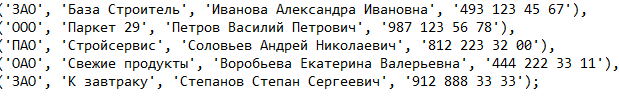


Рисунок 68 – Подготовка данных для импорта

Теперь необходимо создать базу данных (Рисунок 69)

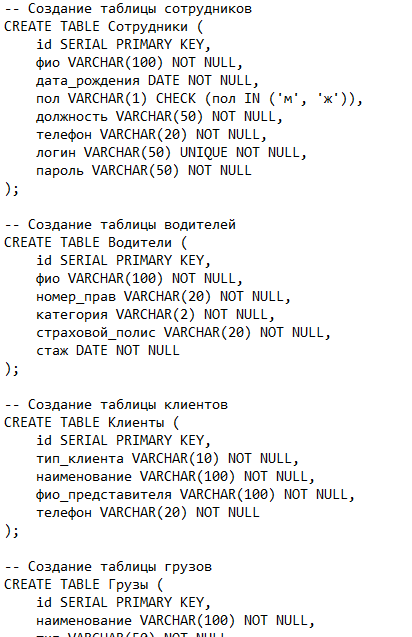


Рисунок 69 – Скрипт для создания БД

Теперь сделаем импорт данных (Рисунок 70)

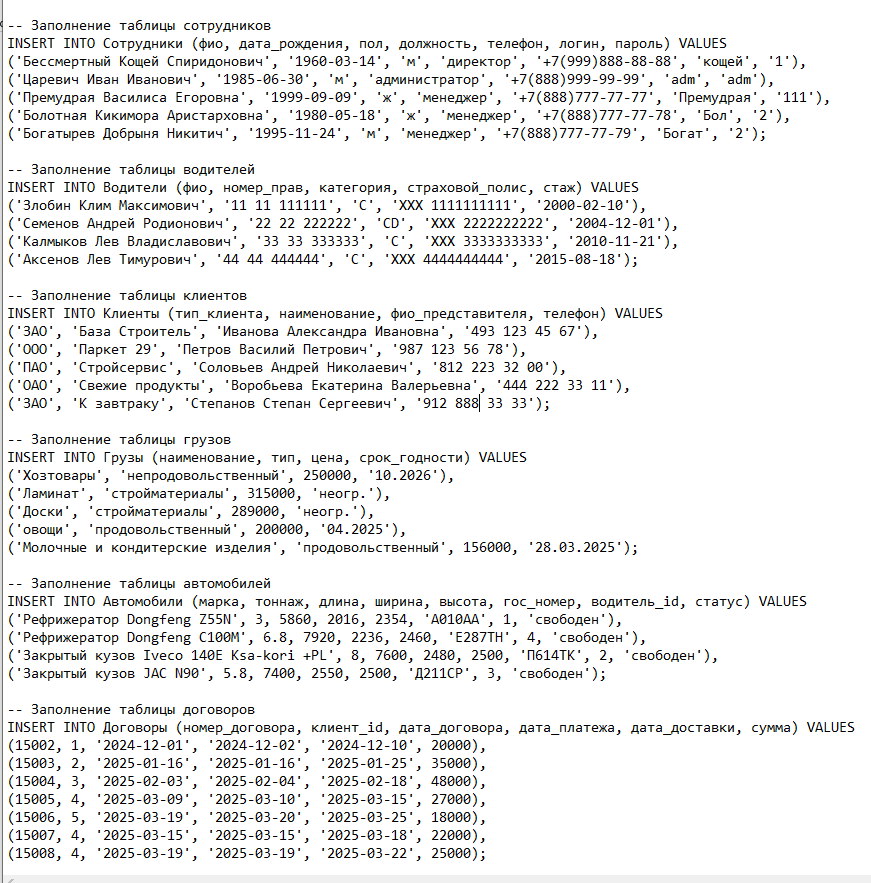


Рисунок 70 – Импорт данных в БД

Листинг кода авторизации:

// Подключение необходимых пространств имён

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;

// Подключение библиотеки для работы с PostgreSQL

using Npgsql;

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

namespace YCHEBKA10

{

// Основной класс формы авторизации

public partial class Form1 : Form

{

// Объект соединения с базой данных

private NpgsqlConnection connection;

// Строка подключения к базе данных PostgreSQL

private string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=Gryz";

// Конструктор формы

public Form1()

{

InitializeComponent(); // Инициализация компонентов формы

// Создание и открытие подключения к базе данных при запуске формы

NpgsqlConnection connection = new NpgsqlConnection(connectionString);

connection.Open(); // ВАЖНО: Это локальная переменная, которая перекрывает поле. Лучше использовать this.connection

}

// Обработчик события нажатия на кнопку "Войти"

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

{

// Получение введённых пользователем логина и пароля

string username = textBox1.Text;

string password = textBox2.Text;

// Проверка данных пользователя

if (AuthenticateUser(username, password))

{

// Если пользователь найден — сообщение об успехе и переход к следующей форме

MessageBox.Show("Успешная аутентификация! Добро пожаловать, " + username + "!");

Form f2 = new Form2(); // Создание экземпляра следующей формы

this.Hide(); // Скрытие текущей формы

f2.Show(); // Отображение новой формы

}

else

{

// Ошибка аутентификации — вывод сообщения

MessageBox.Show("Ошибка аутентификации. Пользователь с такими данными не найден.");

}

}

// Метод для проверки существования пользователя в базе данных

private bool AuthenticateUser(string username, string password)

{

// Повторное объявление строки подключения (можно убрать и использовать поле класса)

string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=Gryz";

// Создание и открытие подключения с использованием using (гарантирует автоматическое закрытие)

using (NpgsqlConnection connection = new NpgsqlConnection(connectionString))

{

connection.Open();

// SQL-запрос с параметрами для безопасного поиска пользователя

string query = "SELECT COUNT(\*) FROM Сотрудники WHERE логин = @Username AND пароль = @Password";

using (NpgsqlCommand command = new NpgsqlCommand(query, connection))

{

// Передача параметров в запрос

command.Parameters.AddWithValue("@Username", username);

command.Parameters.AddWithValue("@Password", password);

// Выполнение запроса и получение количества совпадений

int count = Convert.ToInt32(command.ExecuteScalar());

// Если есть хотя бы одно совпадение — возвращаем true

return count > 0;

}

}

}

}

}

На рисунке 71показана работа формы авторизации:

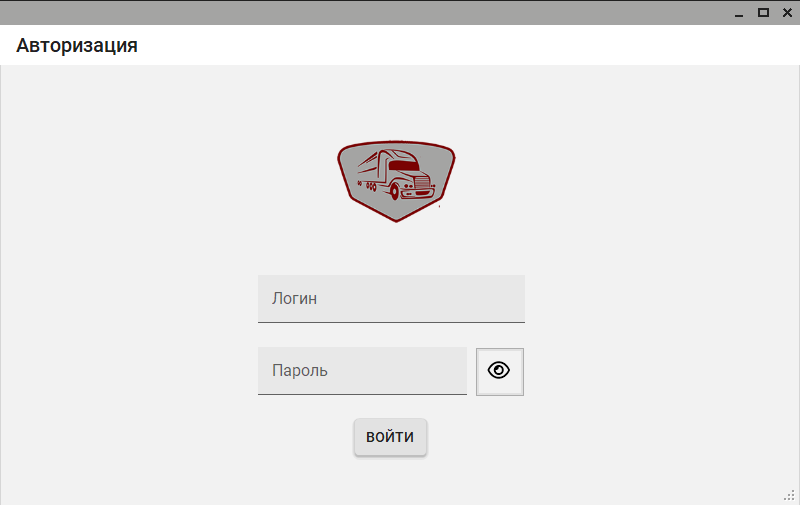


Рисунок 71 – Авторизация

Листинг основной формы:

using Npgsql;

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Windows.Forms;

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

namespace YCHEBKA10

{

public partial class Form2 : Form

{

// Объявление переменных для работы с базой данных

NpgsqlConnection conn; // Подключение к PostgreSQL

NpgsqlDataAdapter adapter; // Адаптер для работы с данными

DataTable dt; // Таблица для хранения данных

// Словарь для хранения справочных данных (например, для выпадающих списков)

Dictionary<string, Dictionary<object, string>> lookupData = new Dictionary<string, Dictionary<object, string>>();

public Form2()

{

InitializeComponent();

// Подписка на событие закрытия формы

this.FormClosing += new FormClosingEventHandler(FormClosingevent);

// Строкаподключениякбазеданных

string connection = "Host=localhost;Username=postgres;Password=18273645;Database=Gryz";

conn = new NpgsqlConnection(connection);

// Добавлениеэлементовввыпадающийсписок

comboBox1.Items.AddRange(new object[] {

"Сотрудники", "Водители", "Клиенты", "Грузы", "Автомобили",

"Договоры", "Перевозки", "История\_взаимодействия"

});

// Подписка на событие изменения выбранного элемента в comboBox1

comboBox1.SelectedIndexChanged += comboBox1\_SelectedIndexChanged;

}

// Обработчик изменения выбранной таблицы в comboBox1

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

try

{

string table = comboBox1.Text;

LoadLookupTables(); // Загрузка справочных данных

// SQL-запрос для выбора всех данных из выбранной таблицы

string query = $"SELECT \* FROM \"{table}\"";

adapter = new NpgsqlDataAdapter(query, conn);

NpgsqlCommandBuilder builder = new NpgsqlCommandBuilder(adapter);

dt = new DataTable();

adapter.Fill(dt); // Заполнение DataTable даннымиизбазы

// Настройка DataGridView

dataGridView1.Columns.Clear();

dataGridView1.DataSource = dt;

HidePrimaryKey(); // Скрытиепервичныхключей

ReplaceForeignKeyColumns(table); // Замена внешних ключей на выпадающие списки

// Подписка на событие необходимости значений по умолчанию для новых строк

dataGridView1.DefaultValuesNeeded -= dataGridView1\_DefaultValuesNeeded;

dataGridView1.DefaultValuesNeeded += dataGridView1\_DefaultValuesNeeded;

}

catch (Exception ex)

{

MessageBox.Show("Ошибкапризагрузкеданных: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

// Обработчикнажатиякнопки "Сохранить"

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

{

try

{

// Проверканаличияизменений

if (dt == null || dt.GetChanges() == null)

{

MessageBox.Show("Нетизмененийдлясохранения.", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

// Сохранение изменений в базе данных

adapter.Update(dt);

dt.AcceptChanges();

MessageBox.Show("Изменениясохраненыуспешно.", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show("Ошибкаприсохранении: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

// Обработчикнажатиякнопки "Удалить"

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

{

// Проверкавыбраннойстроки

if (dataGridView1.CurrentRow == null || dataGridView1.CurrentRow.IsNewRow)

{

MessageBox.Show("Выберитестрокудляудаления.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

try

{

// Подтверждение удаления

if (MessageBox.Show("Вы уверены, что хотите удалить эту запись?", "Подтверждение",

MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

// Удалениестроки

dataGridView1.Rows.Remove(dataGridView1.CurrentRow);

adapter.Update(dt);

MessageBox.Show("Записьудаленауспешно.", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибкаприудалении: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

// Метод для скрытия первичных ключей в DataGridView

private void HidePrimaryKey()

{

foreach (DataColumn col in dt.Columns)

{

if ((col.ColumnName.ToLower() == "id" || col.ColumnName.ToLower().EndsWith("\_id")) &&

dataGridView1.Columns.Contains(col.ColumnName))

{

dataGridView1.Columns[col.ColumnName].Visible = false;

}

}

}

// Методдлязагрузкисправочныхданных

private void LoadLookupTables()

{

lookupData["Сотрудники"] = LoadLookup("Сотрудники", "id", "фио");

lookupData["Водители"] = LoadLookup("Водители", "id", "фио");

lookupData["Клиенты"] = LoadLookup("Клиенты", "id", "наименование");

lookupData["Грузы"] = LoadLookup("Грузы", "id", "наименование");

lookupData["Автомобили"] = LoadLookup("Автомобили", "id", "гос\_номер");

lookupData["Договоры"] = LoadLookup("Договоры", "номер\_договора", "номер\_договора");

}

// Метод для загрузки данных справочника из таблицы

private Dictionary<object, string> LoadLookup(string table, string keyCol, string valCol)

{

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

string query = $"SELECT \"{keyCol}\", \"{valCol}\" FROM \"{table}\"";

try

{

conn.Open();

using (var cmd = new NpgsqlCommand(query, conn))

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

dict[reader[0]] = reader[1].ToString();

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибкапризагрузкесправочника {table}: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

if (conn.State == ConnectionState.Open)

conn.Close();

}

return dict;

}

// Метод для замены столбцов с внешними ключами на выпадающие списки

private void ReplaceForeignKeyColumns(string table)

{

var fkMap = new Dictionary<string, string>();

// Определение соответствий между внешними ключами и справочными таблицами

switch (table)

{

case "Автомобили":

fkMap.Add("водитель\_id", "Водители");

break;

case "Договоры":

fkMap.Add("клиент\_id", "Клиенты");

break;

case "Перевозки":

fkMap.Add("договор\_id", "Договоры");

fkMap.Add("автомобиль\_id", "Автомобили");

fkMap.Add("сотрудник\_id", "Сотрудники");

fkMap.Add("груз\_id", "Грузы");

break;

case "История\_взаимодействия":

fkMap.Add("клиент\_id", "Клиенты");

fkMap.Add("сотрудник\_id", "Сотрудники");

break;

}

// Замена столбцов на выпадающие списки

foreach (var kvp in fkMap)

{

string column = kvp.Key;

string lookupTable = kvp.Value;

if (!dt.Columns.Contains(column)) continue;

var combo = new DataGridViewComboBoxColumn

{

Name = column,

DataPropertyName = column,

HeaderText = column.Replace("\_", " "),

DataSource = new BindingSource(lookupData[lookupTable], null),

DisplayMember = "Value",

ValueMember = "Key",

FlatStyle = FlatStyle.Flat

};

int index = dataGridView1.Columns[column].Index;

dataGridView1.Columns.Remove(column);

dataGridView1.Columns.Insert(index, combo);

}

}

// Обработчик события необходимости значений по умолчанию для новых строк

private void dataGridView1\_DefaultValuesNeeded(object sender, DataGridViewRowEventArgs e)

{

string table = comboBox1.Text;

// Установка значений по умолчанию для разных таблиц

switch (table)

{

case "Перевозки":

if (lookupData["Договоры"].Count > 0)

e.Row.Cells["договор\_id"].Value = lookupData["Договоры"].First().Key;

if (lookupData["Автомобили"].Count > 0)

e.Row.Cells["автомобиль\_id"].Value = lookupData["Автомобили"].First().Key;

if (lookupData["Сотрудники"].Count > 0)

e.Row.Cells["сотрудник\_id"].Value = lookupData["Сотрудники"].First().Key;

if (lookupData["Грузы"].Count > 0)

e.Row.Cells["груз\_id"].Value = lookupData["Грузы"].First().Key;

e.Row.Cells["дата\_отправления"].Value = DateTime.Now;

e.Row.Cells["дата\_прибытия"].Value = DateTime.Now.AddDays(1);

e.Row.Cells["вес\_груза"].Value = 0;

break;

case "История\_взаимодействия":

if (lookupData["Клиенты"].Count > 0)

e.Row.Cells["клиент\_id"].Value = lookupData["Клиенты"].First().Key;

if (lookupData["Сотрудники"].Count > 0)

e.Row.Cells["сотрудник\_id"].Value = lookupData["Сотрудники"].First().Key;

e.Row.Cells["дата\_взаимодействия"].Value = DateTime.Now;

e.Row.Cells["тип\_взаимодействия"].Value = "Звонок";

break;

}

}

// Обработчик нажатия кнопки для перехода к Form3

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

{

this.Hide();

Form3 form3 = new Form3();

form3.Show();

}

// Обработчик события закрытия формы

private void FormClosingevent(object sender, FormClosingEventArgs e)

{

Application.Exit();

}

}

}

На рисунках 72, 73 работа этой формы:

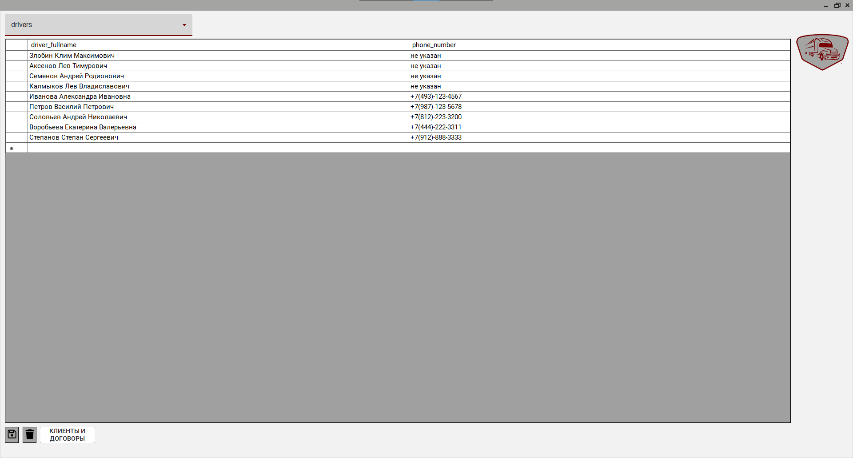


Рисунок 72 – Работа главной формы

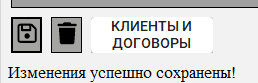


Рисунок 73 – Работа главной формы

Листинг кода формы списка клиентов и истории взаимодействий:

using System;

using System.Windows.Forms;

using Npgsql;

namespace YCHEBKA10

{

public partial class Form3 : Form

{

// Флаг, указывающий, отображается ли в данный момент история взаимодействий

bool showingHistory = false;

// Строка подключения к базе данных PostgreSQL

string connectionString = "Host=localhost;Username=postgres;Password=18273645;Database=Gryz";

// Конструкторформы

public Form3()

{

InitializeComponent();

// Подписываемсянасобытиезакрытияформы

this.FormClosing += new FormClosingEventHandler(FormClosingevent);

// Загружаем список клиентов при инициализации формы

LoadClients();

}

// Обработчиксобытиязакрытияформы

private void FormClosingevent(object sender, FormClosingEventArgs e)

{

Application.Exit();

}

// Обработчик кнопки "Назад" - возврат к предыдущей форме

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

{

this.Hide();

Form2 form2 = new Form2();

form2.Show();

}

// Обработчик кнопки переключения между списком клиентов и историей взаимодействий

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

{

listBox1.Items.Clear();

comboBox1.Visible = !showingHistory;

if (showingHistory)

{

// Если показывали историю, переключаемся обратно на список клиентов

LoadClients();

button1.Text = "Историявзаимодейсвтия";

label1.Text = "Списокклиентов";

}

else

{

// Если показывали список клиентов, переключаемся на историю

LoadClientNamesToComboBox();

button1.Text = "Списокклиентов";

label1.Text = "Историявзаимодеействия";

}

showingHistory = !showingHistory;

}

// Метод загрузки списка клиентов из базы данных

private void LoadClients()

{

listBox1.Items.Clear();

using (var conn = new NpgsqlConnection(connectionString))

{

conn.Open();

// SQL-запрос для получения информации о клиентах и их перевозках

string query = @"

SELECT

k.id,

k.тип\_клиента || ' ' || k.наименование AS company,

k.фио\_представителя,

k.телефон,

COUNT(d.номер\_договора) AS перевозки\_количество,

COALESCE(SUM(d.сумма), 0) AS перевозки\_сумма

FROM

Клиенты k

LEFT JOIN

Договоры d ON k.id = d.клиент\_id

GROUP BY

k.id

ORDER BY

k.наименование";

using (var cmd = new NpgsqlCommand(query, conn))

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

// Чтение данных о клиенте из результата запроса

string company = reader.GetString(1);

string contact = reader.GetString(2);

string phone = reader.GetString(3);

int shipmentsCount = reader.GetInt32(4);

decimal totalAmount = reader.GetDecimal(5);

// Расчетскидкинаосновеобщейсуммыперевозок

int discount = CalculateDiscount(totalAmount);

// Добавлениеинформацииоклиентев listBox

listBox1.Items.Add($"{company}");

listBox1.Items.Add($"ФИОпредставителя: {contact}");

listBox1.Items.Add($"Телефон: {phone}");

listBox1.Items.Add($"Количествоперевозок: {shipmentsCount}");

listBox1.Items.Add($"Общаясумма: {totalAmount:N2} руб.");

listBox1.Items.Add($"Скидка: {discount} %");

listBox1.Items.Add("");

}

}

}

}

// Метод расчета скидки в зависимости от общей суммы перевозок

private int CalculateDiscount(decimal totalAmount)

{

if (totalAmount >= 100000) return 15;

if (totalAmount >= 50000) return 10;

if (totalAmount >= 10000) return 5;

return 0;

}

// Методзагрузкиименклиентовввыпадающийсписок (ComboBox)

private void LoadClientNamesToComboBox()

{

comboBox1.Items.Clear();

comboBox1.Visible = true;

using (var conn = new NpgsqlConnection(connectionString))

{

conn.Open();

// SQL-запрос для получения ID и наименований клиентов

string query = "SELECT id, тип\_клиента || ' ' || наименование AS company FROM Клиенты ORDER BY наименование";

using (var cmd = new NpgsqlCommand(query, conn))

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

// Добавлениеклиентовв ComboBox какобъектов ComboBoxItem

comboBox1.Items.Add(new ComboBoxItem

{

Id = reader.GetInt32(0),

Name = reader.GetString(1)

});

}

}

}

// Переподписка на событие изменения выбранного элемента в ComboBox

comboBox1.SelectedIndexChanged -= comboBox1\_SelectedIndexChanged;

comboBox1.SelectedIndexChanged += comboBox1\_SelectedIndexChanged;

}

// Обработчик изменения выбранного клиента в ComboBox

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox1.SelectedItem is ComboBoxItem selectedClient)

{

// Загрузка истории взаимодействий для выбранного клиента

LoadClientHistory(selectedClient.Id);

}

}

// Метод загрузки истории взаимодействий с клиентом

private void LoadClientHistory(int clientId)

{

listBox1.Items.Clear();

using (var conn = new NpgsqlConnection(connectionString))

{

conn.Open();

// SQL-запрос для получения истории договоров и перевозок клиента

string query = @"

SELECT

d.номер\_договора,

d.дата\_договора,

d.дата\_доставки,

d.сумма,

p.дата\_отправления,

p.дата\_прибытия,

p.пункт\_отправления,

p.пункт\_назначения,

a.марка || ' ' || a.гос\_номер AS автомобиль,

g.наименование AS груз

FROM

Договоры d

JOIN

Перевозки p ON d.номер\_договора = p.договор\_id

JOIN

Автомобили a ON p.автомобиль\_id = a.id

JOIN

Грузы g ON p.груз\_id = g.id

WHERE

d.клиент\_id = @clientId

ORDER BY

d.дата\_договора DESC";

using (var cmd = new NpgsqlCommand(query, conn))

{

cmd.Parameters.AddWithValue("clientId", clientId);

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

// Форматирование информации о договоре и перевозке

string contract = $"Договор №{reader.GetInt32(0)} от {reader.GetDateTime(1):dd.MM.yyyy}";

string delivery = $"Датадоставки: {reader.GetDateTime(2):dd.MM.yyyy}";

string amount = $"Сумма: {reader.GetDecimal(3):N2} руб.";

string dates = $"Периодперевозки: {reader.GetDateTime(4):dd.MM.yyyy} - {reader.GetDateTime(5):dd.MM.yyyy}";

string route = $"Маршрут: {reader.GetString(6)} → {reader.GetString(7)}";

string vehicle = $"Автомобиль: {reader.GetString(8)}";

string cargo = $"Груз: {reader.GetString(9)}";

// Добавлениеинформациив listBox

listBox1.Items.Add(contract);

listBox1.Items.Add(delivery);

listBox1.Items.Add(amount);

listBox1.Items.Add(dates);

listBox1.Items.Add(route);

listBox1.Items.Add(vehicle);

listBox1.Items.Add(cargo);

listBox1.Items.Add("");

}

}

}

}

}

// Вспомогательный класс для хранения ID и наименования клиента в ComboBox

private class ComboBoxItem

{

public int Id { get; set; }

public string Name { get; set; }

public override string ToString() => Name;

}

}

}

На рисунке 74список клиентов:

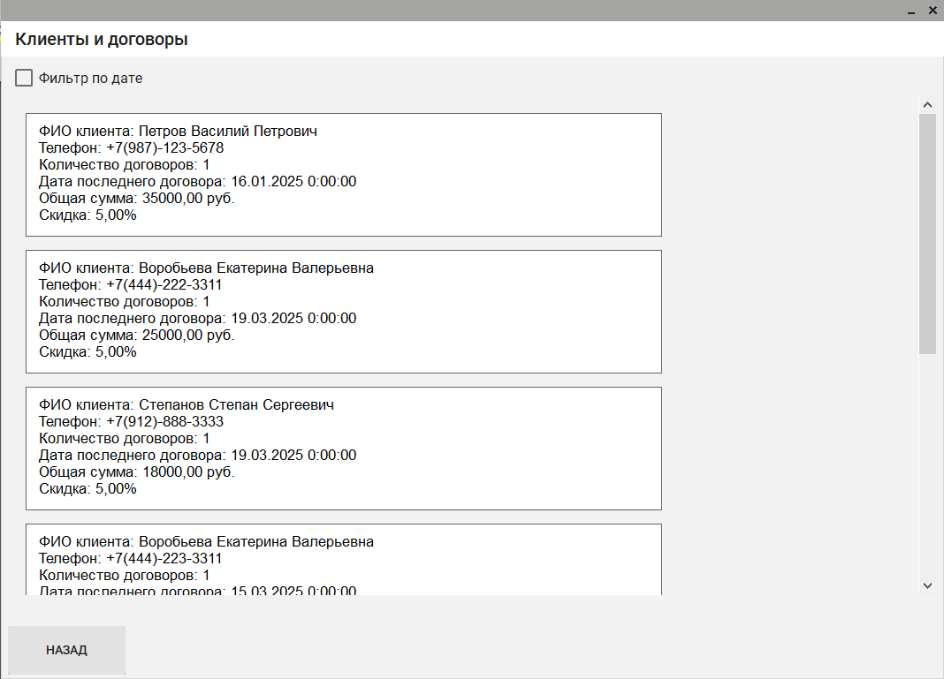


Рисунок 74 – Список клиентов

На рисунке 75 история взаимодействий:

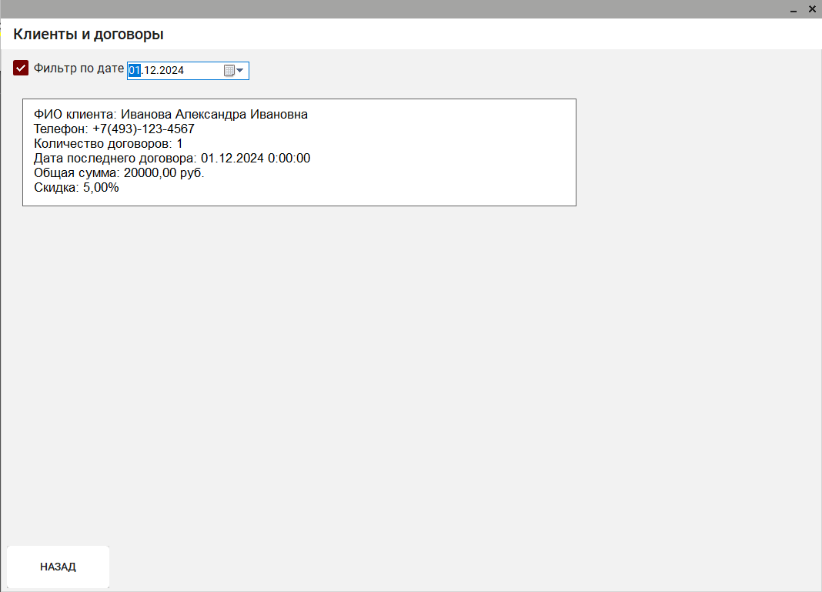


Рисунок 75 – Отфильтрованный список

Ссылка на github, со всеми практическими: **https://github.com/MaxyUMus/PR\_up2025JUNE**