**Содержание**

[Постановка задачи 2](#_Toc151049203)

[Таблица связей 3](#_Toc151049204)

[Пользовательский интерфейс программы 4](#_Toc151049205)

[Практическая часть 5](#_Toc151049206)

# Постановка задачи

**Цель:**

Разработать интернет-магазин еды, предоставляющий пользователям возможность регистрации, аутентификации, выбора и заказа продуктов. Администраторам предоставляется управление продуктами.

**Задачи:**

1. Разработка интерфейса для пользователей и администраторов;
2. Реализация системы аутентификации и авторизации;
3. Создание базы данных для хранения информации о пользователях, продуктах и заказах;
4. Разработка функциональности для просмотра каталога продуктов и размещения заказов;
5. Разработка панели администратора для управления продуктами, пользователями и заказами.

**Предметная область:**

1. *Пользователи и аутентификация:*

* Регистрация новых пользователей с указанием личных данных, таких как имя, фамилия, адрес электронной почты и пароль;
* Вход в систему для зарегистрированных пользователей;
* Разграничение доступа: обычные пользователи и администраторы.

1. *Продукты и заказы:*

* Каталог продуктов с информацией о наименовании, цене;
* Возможность пользователями выбирать продукты, добавлять их в корзину и размещать заказы.

1. *Управление:*

* Администраторы имеют доступ к панели управления для добавления и удаления продуктов и пользователей;
* Администраторы могут просматривать информацию о заказах и отменять их.

**Выходной документ:**

Общая стоимость всех заказов по выбранного клиента.

# Таблица связей

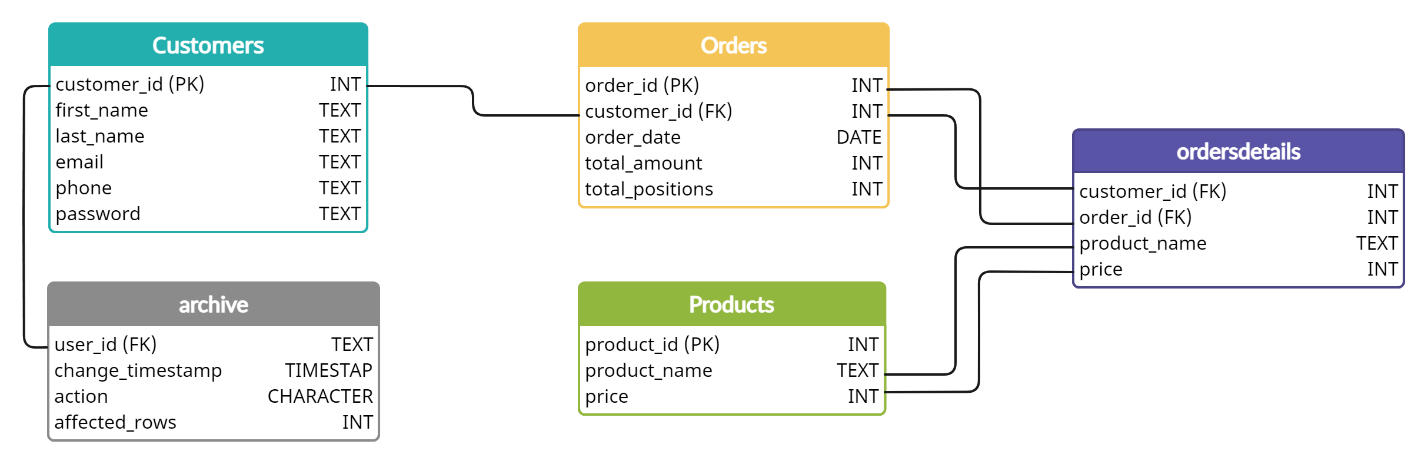


Рис. 1. Таблица связей

# Пользовательский интерфейс программы

После запуска программы открывается окно с таблицей продуктов (строки таблицы скрыты, пока пользователь не авторизован) и с кнопками «*Login*» и «*Sign up*» (см. рис. 2).

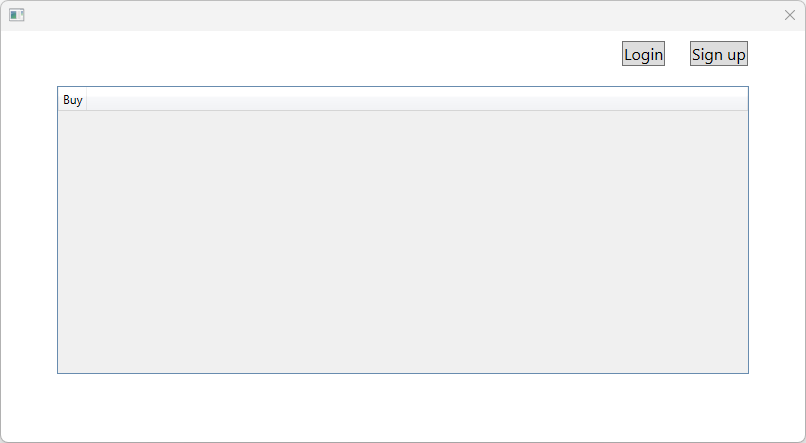


Рис. 2. Основное окно программы

При нажатии на кнопку «*Sign up*», открывается окно регистрации нового пользователя (см. рис. 3).

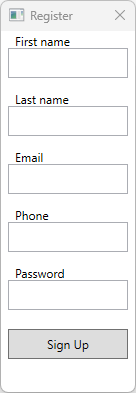


Рис. 3. Окно регистрации

При нажатии на кнопку «*Login*», открывается окно входа для существующего пользователя (см. рис. 4).

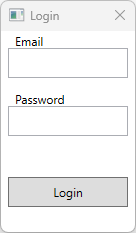


Рисунок 4. Окно авторизации

Когда пользователь авторизован, у него появляется возможность добавлять необходимые продукты в корзину (кнопка «*Order*») и оформлять заказы (кнопка «*MAKE ORDER*») (см. рис. 5).

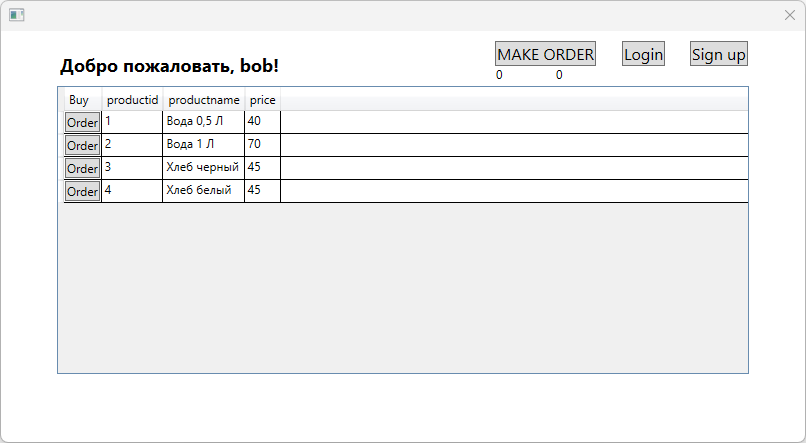


Рис. 5. Основное окно программы для авторизованного пользователя

После нажатия на кнопку «*MAKE ORDER*» пользователю показывается оповещение о том, что заказ оформлен (см. рис. 6).

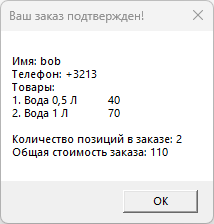


Рис. 6. Оповещение об успешном оформлении заказа

Далее рассмотрим интерфейс админ-панели. Главное окно для администратора включает в себя: таблицу продуктов, таблицу пользователей, таблицу заказов, таблицу заказов со списком продуктов. Также имеется функция добавления новых продуктов и пользователей в базу данных (кнопка «*Add*»). Администратору разрешено удаление продуктов, пользователей и заказов (кнопка «*Delete*»). Существует возможность создания выходного документа в формате «*txt*» для экспорта общей стоимости всех заказов отдельного клиента, путем ввода в нужное поле *id* клиента и нажатия кнопки «*Output file*» (см. рис. 7–10).

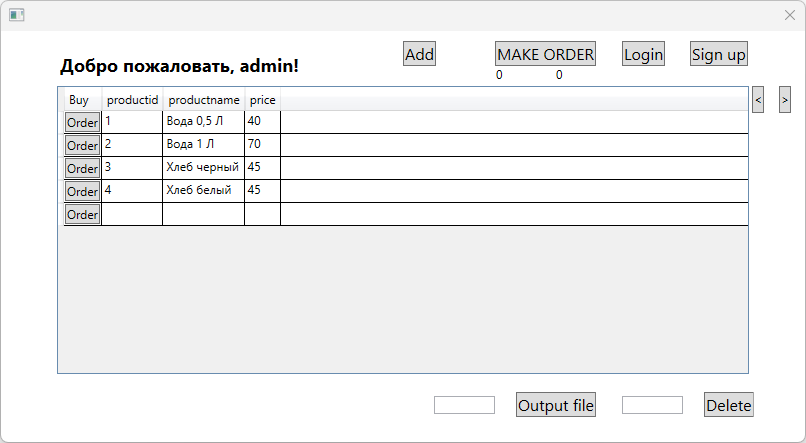


Рис. 7. Таблица «Products»

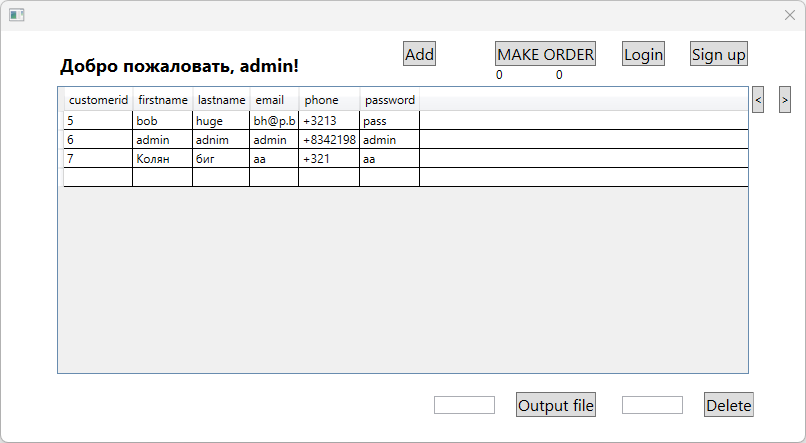


Рис. 8. Таблица «Customers»

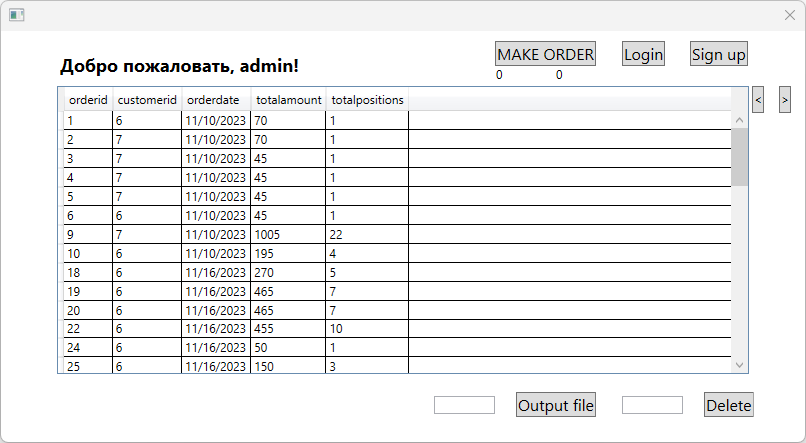


Рис. 9. Таблица «Orders»

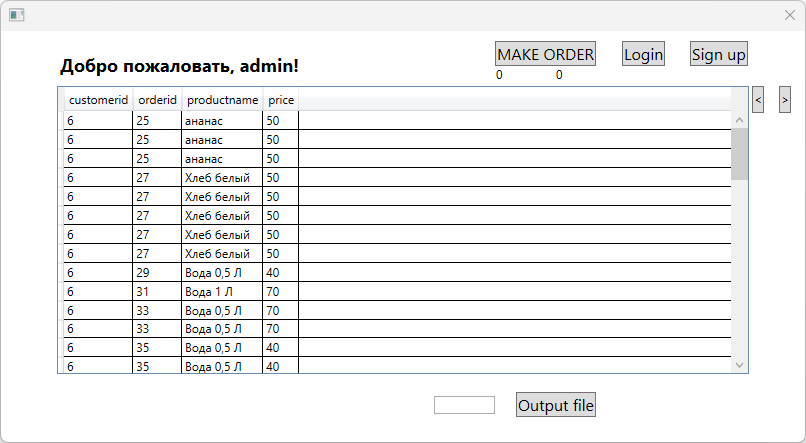


Рис. 10. Таблица «ordersdetails»

# Практическая часть

**Программный код для создания таблиц**

CREATE TABLE public."Orders" (

order\_id SERIAL PRIMARY KEY,

customer\_id INT,

order\_date DATE,

total\_amount INT,

total\_positions INT

);

CREATE TABLE public."Customers" (

customer\_id SERIAL PRIMARY KEY,

first\_name TEXT,

last\_name TEXT,

email TEXT,

phone TEXT,

password TEXT

);

CREATE TABLE public."Products" (

product\_id SERIAL PRIMARY KEY,

product\_name TEXT,

price INT

);

CREATE TABLE public."OrdersDetails" (

customer\_id INT,

order\_id INT,

product\_name TEXT,

price INT,

FOREIGN KEY (customer\_id) REFERENCES "Customers"(customer\_id),

FOREIGN KEY (order\_id) REFERENCES "Orders"(order\_id)

);

INSERT INTO public."Products"(product\_name, price) VALUES ('Вода 0,5 Л', 40);

INSERT INTO public."Products"(product\_name, price) VALUES ('Вода 1 Л', 70);

INSERT INTO public."Products"(product\_name, price) VALUES ('Хлеб черный', 45);

INSERT INTO public."Products"(product\_name, price) VALUES ('Хлеб белый', 45);

INSERT INTO public."Customers"(first\_name, last\_name, email, phone, password) VALUES ('admin', 'adnim', 'admin', +78932132143, 'admin');

CREATE TABLE ARCHIVE (

user\_id TEXT,

change\_timestamp TIMESTAMP,

action VARCHAR(10),

affected\_rows INT

);

CREATE OR REPLACE FUNCTION archive\_products\_changes()

RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'INSERT' THEN

INSERT INTO ARCHIVE (user\_id, change\_timestamp, action, affected\_rows)

VALUES (current\_user, NOW(), 'Insert', 1);

ELSIF TG\_OP = 'UPDATE' THEN

INSERT INTO ARCHIVE (user\_id, change\_timestamp, action, affected\_rows)

VALUES (current\_user, NOW(), 'Update', 1);

ELSIF TG\_OP = 'DELETE' THEN

INSERT INTO ARCHIVE (user\_id, change\_timestamp, action, affected\_rows)

VALUES (current\_user, NOW(), 'Delete', 1);

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER products\_changes\_trigger

AFTER INSERT OR UPDATE OR DELETE ON "Products"

FOR EACH ROW EXECUTE FUNCTION archive\_products\_changes();

CREATE OR REPLACE FUNCTION get\_total\_amount\_by\_customer\_id(input\_customer\_id INT)

RETURNS INT AS $$

DECLARE

total\_amount\_sum INT := 0;

BEGIN

SELECT COALESCE(SUM(total\_amount), 0)

INTO total\_amount\_sum

FROM "Orders"

WHERE customer\_id = input\_customer\_id;

RETURN total\_amount\_sum;

END;

$$ LANGUAGE plpgsql;

**Программный код реализации основной части программы**

В программе имеется отдельный класс, в котором 14 методов, для управлений данными в таблицах базы данных.

* **Методы «*showProducts*», «*showUsers*», «*showOrders*», «*showOrdersDetails*»**

Данные методы предназначены для отображения списка продуктов, пользователей, заказов и списка товаров в заказах в *DataGrid* *WPF*-приложения. Извлекаются данные о продуктах из таблиц базы данных и отображаются в *DataGrid*.

public void showProducts(Label custname, DataGrid DG)

        {

            custname.Content = "Добро пожаловать, " + Data.firstName + "!";

            custname.Visibility = Visibility.Visible;

            DG.ItemsSource = null;

            using (NpgsqlConnection conn = new NpgsqlConnection(connString))

            {

                conn.Open();

                string selectQuery = "SELECT \* FROM public.\"Products\" ORDER BY product\_id ASC";

                using (NpgsqlCommand cmd = new NpgsqlCommand(selectQuery, conn))

                {

                    using (NpgsqlDataAdapter adapter = new NpgsqlDataAdapter(cmd))

                    {

                        DataTable dataTable = new DataTable();

                        adapter.Fill(dataTable);

                        DG.ItemsSource = dataTable.DefaultView;

                    }

                }

                conn.Close();

            }

        }

        public void showUsers(DataGrid DG)

        {

            DG.ItemsSource = null;

            using (NpgsqlConnection conn = new NpgsqlConnection(connString))

            {

                conn.Open();

                string selectQuery = "SELECT \* FROM public.\"Customers\" ORDER BY customer\_id ASC";

                using (NpgsqlCommand cmd = new NpgsqlCommand(selectQuery, conn))

                {

                    using (NpgsqlDataAdapter adapter = new NpgsqlDataAdapter(cmd))

                    {

                        DataTable dataTable = new DataTable();

                        adapter.Fill(dataTable);

                        DG.ItemsSource = dataTable.DefaultView;

                    }

                }

                conn.Close();

            }

        }

        public void showOrders(DataGrid DG)

        {

            DG.ItemsSource = null;

            using (NpgsqlConnection conn = new NpgsqlConnection(connString))

            {

                conn.Open();

                string selectQuery = "SELECT \* FROM public.\"Orders\" ORDER BY order\_id ASC";

                using (NpgsqlCommand cmd = new NpgsqlCommand(selectQuery, conn))

                {

                    using (NpgsqlDataAdapter adapter = new NpgsqlDataAdapter(cmd))

                    {

                        DataTable dataTable = new DataTable();

                        adapter.Fill(dataTable);

                        DG.ItemsSource = dataTable.DefaultView;

                    }

                }

                conn.Close();

            }

        }

        public void showOrdersDetails(DataGrid DG)

        {

            DG.ItemsSource = null;

            using (NpgsqlConnection conn = new NpgsqlConnection(connString))

            {

                conn.Open();

                string selectQuery = "SELECT \* FROM public.\"ordersdetails\" ORDER BY order\_id ASC";

                using (NpgsqlCommand cmd = new NpgsqlCommand(selectQuery, conn))

                {

                    using (NpgsqlDataAdapter adapter = new NpgsqlDataAdapter(cmd))

                    {

                        DataTable dataTable = new DataTable();

                        adapter.Fill(dataTable);

                        DG.ItemsSource = dataTable.DefaultView;

                    }

                }

                conn.Close();

            }

        }

* **Метод «*loginCheck*»**

Этот метод выполняет проверку аутентификации для введенных электронной почты и пароля в базе данных пользователей. Метод выполняет аутентификацию пользователя, управляет флагом администратора и открывает новое главное окно, скрывая текущее окно входа в систему.

public void loginCheck(TextBox tb1, TextBox tb2, Window wn)

        {

            using (NpgsqlConnection conn = new NpgsqlConnection(connString))

            {

                conn.Open();

                string selectQuery = "SELECT customer\_id, first\_name, last\_name, email, phone, password FROM public.\"Customers\" WHERE email = @email AND password = @password";

                using (NpgsqlCommand cmd = new NpgsqlCommand(selectQuery, conn))

                {

                    cmd.Parameters.AddWithValue("@email", tb1.Text);

                    cmd.Parameters.AddWithValue("@password", tb2.Text);

                    if (tb1.Text == "admin" && tb2.Text == "admin")

                        Data.admin = true;

                    else

                        Data.admin = false;

                    using (NpgsqlDataReader reader = cmd.ExecuteReader())

                    {

                        if (reader.Read())

                        {

                            Data.customerId = reader.GetInt32(0);

                            Data.firstName = reader.GetString(1);

                            Data.lastName = reader.GetString(2);

                            Data.email = reader.GetString(3);

                            Data.phone = reader.GetString(4);

                            MainWindow main = new MainWindow();

                            wn.Hide();

                            main.Show();

                        }

                    }

                }

                conn.Close();

            }

        }

* **Метод «*registration*»**

Следующий метод выполняет регистрацию нового пользователя в базе данных, а затем отображает главное окно приложения, скрывая текущее окно регистрации.

public void registration(TextBox tb1, TextBox tb2, TextBox tb3, TextBox tb4, TextBox tb5, Window wn)

        {

            using (NpgsqlConnection conn = new NpgsqlConnection(connString))

            {

                conn.Open();

                string insertQuery = "INSERT INTO public.\"Customers\" (first\_name, last\_name, email, phone, password) VALUES (@first\_name, @last\_name, @email, @phone, @password)";

                using (NpgsqlCommand cmd = new NpgsqlCommand(insertQuery, conn))

                {

                    cmd.Parameters.AddWithValue("@first\_name", tb1.Text);

                    cmd.Parameters.AddWithValue("@last\_name", tb2.Text);

                    cmd.Parameters.AddWithValue("@email", tb3.Text);

                    cmd.Parameters.AddWithValue("@phone", tb4.Text);

                    cmd.Parameters.AddWithValue("@password", tb5.Text);

                    cmd.ExecuteNonQuery();

                }

                conn.Close();

                MainWindow main = new MainWindow();

                wn.Hide();

                main.Show();

            }

        }

* **Метод «*Order*»**

Этот метод выполняет вставку нового заказа в базу данных.

public void Order(int price, int kolvo)

        {

            using (NpgsqlConnection conn = new NpgsqlConnection(connString))

            {

                conn.Open();

                string insertQuery = "INSERT INTO public.\"Orders\" (customer\_id, order\_date, total\_amount, total\_positions) VALUES (@customer\_id, @order\_date, @total\_amount, @total\_positions) RETURNING order\_id;";

                using (NpgsqlCommand cmd = new NpgsqlCommand(insertQuery, conn))

                {

                    cmd.Parameters.AddWithValue("@customer\_id", Data.customerId);

                    cmd.Parameters.AddWithValue("@order\_date", DateTime.Now);

                    cmd.Parameters.AddWithValue("@total\_amount", price);

                    cmd.Parameters.AddWithValue("@total\_positions", kolvo);

                    Data.orderId = (int)cmd.ExecuteScalar();

                }

            }

        }

* **Метод «*AddOrderDetails*»**

Этот метод предназначен для добавления информации о деталях заказа в таблицу «*ordersdetails*». В метод передаются списки с именами продуктов и их ценами. Метод выполняет множество вставок в базу данных, каждая из которых представляет отдельный продукт в заказе.

public void AddOrderDetails(List<string> namesOfProducts, List<int> pricesOfProducts)

        {

            using (NpgsqlConnection conn = new NpgsqlConnection(connString))

            {

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

                {

                    conn.Open();

                    string insertQuery = "INSERT INTO public.\"ordersdetails\" (customer\_id, order\_id, product\_name, price) VALUES (@customer\_id, @order\_id, @product\_name, @price);";

                    using (NpgsqlCommand cmd = new NpgsqlCommand(insertQuery, conn))

                    {

                        cmd.Parameters.AddWithValue("@customer\_id", Data.customerId);

                        cmd.Parameters.AddWithValue("@order\_id", Data.orderId);

                        cmd.Parameters.AddWithValue("@product\_name", namesOfProducts[i]);

                        cmd.Parameters.AddWithValue("@price", pricesOfProducts[i]);

                        cmd.ExecuteNonQuery();

                    }

                    conn.Close();

                }

            }

        }

* **Методы «*AddProductToDatabase*», «*AddCustomerToDatabase*»**

Методы выполняет вставку нового продукта или пользователя в таблицы «*Products*» и «*Customers*» с указанными параметрами.

public void AddProductToDatabase(string productName, int price)

        {

            using (NpgsqlConnection conn = new NpgsqlConnection(connString))

            {

                conn.Open();

                string insertQuery = "INSERT INTO public.\"Products\" (product\_name, price) VALUES (@product\_name, @price)";

                using (NpgsqlCommand cmd = new NpgsqlCommand(insertQuery, conn))

                {

                    cmd.Parameters.AddWithValue("@product\_name", productName);

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

                    cmd.ExecuteNonQuery();

                }

                conn.Close();

            }

        }

        public void AddCustomerToDatabase(string firstName, string lastName, string email, string phone, string pass)

        {

            using (NpgsqlConnection conn = new NpgsqlConnection(connString))

            {

                conn.Open();

                string insertQuery = "INSERT INTO public.\"Customers\" (first\_name, last\_name, email, phone, password) VALUES (@first\_name, @last\_name, @email, @phone, @password)";

                using (NpgsqlCommand cmd = new NpgsqlCommand(insertQuery, conn))

                {

                    cmd.Parameters.AddWithValue("@first\_name", firstName);

                    cmd.Parameters.AddWithValue("@last\_name", lastName);

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

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

                    cmd.Parameters.AddWithValue("@password", pass);

                    cmd.ExecuteNonQuery();

                }

                conn.Close();

            }

        }

* **Методы «*DeleteCustomer*», «*DeleteOrder*», «*DeleteProduct*»**

Метод удаляет строку из таблицы «*Products*», «*Orders*», «*Customers*» с указанным уникальным идентификатором.

public void DeleteCustomer(int userId)

        {

            using (NpgsqlConnection conn = new NpgsqlConnection(connString))

            {

                conn.Open();

                string insertQuery = "DELETE FROM public.\"Customers\" WHERE customer\_id = @customer\_id";

                using (NpgsqlCommand cmd = new NpgsqlCommand(insertQuery, conn))

                {

                    cmd.Parameters.AddWithValue("@customer\_id", userId);

                    cmd.ExecuteNonQuery();

                }

                conn.Close();

            }

        }

        public void DeleteOrder(int orderId)

        {

            using (NpgsqlConnection conn = new NpgsqlConnection(connString))

            {

                conn.Open();

                string insertQuery = "DELETE FROM public.\"ordersdetails\" WHERE order\_id = @order\_id";

                using (NpgsqlCommand cmd = new NpgsqlCommand(insertQuery, conn))

                {

                    cmd.Parameters.AddWithValue("@order\_id", orderId);

                    cmd.ExecuteNonQuery();

                }

                conn.Close();

                conn.Open();

                insertQuery = "DELETE FROM public.\"Orders\" WHERE order\_id = @order\_id";

                using (NpgsqlCommand cmd = new NpgsqlCommand(insertQuery, conn))

                {

                    cmd.Parameters.AddWithValue("@order\_id", orderId);

                    cmd.ExecuteNonQuery();

                }

                conn.Close();

            }

        }

        public void DeleteProduct(int productId)

        {

            using (NpgsqlConnection conn = new NpgsqlConnection(connString))

            {

                conn.Open();

                string insertQuery = "DELETE FROM public.\"Products\" WHERE product\_id = @product\_id";

                using (NpgsqlCommand cmd = new NpgsqlCommand(insertQuery, conn))

                {

                    cmd.Parameters.AddWithValue("@product\_id", productId);

                    cmd.ExecuteNonQuery();

                }

                conn.Close();

            }

        }

* **Метод «*OutputFile*»**

Этот метод используется для выполнения запроса на получение суммы всех заказов для указанного *customer\_id* и записи этой суммы в файл «*total\_amount*.*txt*».

public void OutputFile(int cId)

        {

            using (NpgsqlConnection conn = new NpgsqlConnection(connString))

            {

                conn.Open();

                using (NpgsqlCommand cmd = new NpgsqlCommand("SELECT get\_total\_amount\_by\_customer\_id(@customer\_id)", conn))

                {

                    cmd.Parameters.AddWithValue("@customer\_id", cId);

                    int totalAmountSum = (int)cmd.ExecuteScalar();

                    string filePath = "total\_amount.txt";

                    File.WriteAllText(filePath, totalAmountSum.ToString());

                }

                conn.Close();

            }

        }