

Министерство науки и высшего образования Российской Федерации

Муромский институт (филиал)

федерального государственного бюджетного образовательного учреждения высшего образования

«Владимирский государственный университет

Имени Александра Григорьевича и Николая Григорьевича  
Столетовых»

(МИ ВлГУ)

Факультет: ИТР

Кафедра: ПИН

# Курсовая работа

По: Системы управления базами данных

Тема: Информационная система магазина электротехники

Руководитель

к. т. н., доц. каф. ПИН

(уч.степень, звание)

Колпаков А.А.

(фамилия, инициалы)

Члены комиссии:

(подпись) (дата)

Студент ПИН – 122

(группа)

Миронов А.И.

(фамилия, инициалы)

(Оценка)

(Подпись Ф.И.О.)

(Подпись Ф.И.О.)

(подпись)

(дата)

Муром 2024

## Содержание

Введение.....	6
1. Анализ технического задания.....	8
1.1 Анализ предметной области.....	8
1.2 Анализ потребностей пользователей и требований заказчика.....	8
1.3 Исследование технической реализуемости.....	9
1.4 Выбор СУБД и обоснование выбора.....	9
1.5 Постановка задачи и предварительный анализ.....	10
2. Разработка моделей данных.....	11
2.1 Концептуальная модель данных.....	11
2.2 Логическая модель данных.....	12
2.3 Физическая модель данных.....	13
3. Разработка и реализация АИС.....	14
3.1 Создание базы данных.....	14
3.2 Описание программы.....	17
3.3 Руководство программиста.....	19
3.4 Руководство пользователя.....	20
4. Тестирование АИС.....	23
Заключение.....	26
Список используемой литературы.....	28
Приложение А. Модели данных.....	29
Приложение Б. Текст программы.....	32
Приложение В. Снимки окон программы (скриншоты программы).....	47

					МИВУ.09.03.04-13.000 ПЗ				
Изм.	Лист	№ докум.	Подп.	Дата	Информационная система магазина электротехники	Лит.		Лист	Листов
Студент		Миронов.А.И.					У	5	47
Руков.		Колпаков А.А.				МИ ВлГУ  ПИН-122			
Конс									
Н.контр.									
Утв.									

Автоматизированная информационная система или АИС – это совокупность различных программно – аппаратных средств, которые предназначены для автоматизации какой – либо деятельности, связанной с передачей, хранением и обработкой различной информации.

В автоматизированных информационных системах за хранение любой информации отвечают:

- 1) На физическом уровне:
  - а) Внешние накопители;
  - б) Встроенные устройства памяти (RAM);
  - в) Массивы дисков.
- 2) На программном уровне:
  - а) СУБД;
  - б) Файловая система ОС;
  - в) Системы хранения мультимедиа, документов и т. д.

На сегодняшний день достаточно широко применяются разнообразные программные средства при работе с компьютером. В их числе находятся и автоматизированные информационные системы. Информационная система или ИС – это система обработки, хранения и передачи какой-либо информации, которая представлена в определенной форме.

В современной вычислительной технике ИС представляет собой целый программный комплекс, который дает возможность надежно хранить данные в памяти, выполнять преобразования информации и производить вычисления с помощью удобного и легкого для пользователя интерфейса.

Исходя из вышесказанного, использование современных информационных систем позволяет нам:

- 1) Работать с огромными объемами данных;
- 2) Хранить какие-либо данные в течение довольно длительного временного периода;
- 3) Связать несколько компонентов, которые имеют свои определенные локальные цели, задачи и разнообразные приемы функционирования, в одну систему для работы с информацией;
- 4) Существенно снизить затраты на доступ и хранение к любым необходимым нам данным;
- 5) Довольно - таки быстро найти всю необходимую нам информацию и т. д.

В качестве классического примера современной информационной системы, стоит упомянуть банковские системы, АС управления предприятиями, системы резервирования железнодорожных или авиационных билетов и т. д.

Целью курсовой работы является создание информационной системы для добавления, удаления, изменения и поиска записей в базе данных.

Задачи, для реализации этой цели:

- 1) Проектирование логической и физической моделей базы данных.
- 2) Реализация физической модели в одной из современных СУБД.
- 3) Подключение БД к программе с помощью средств среды разработки.
- 4) Написание обработчиков событий и функций для добавления, удаления, изменения и поиска записей в БД.

## 1. Анализ технического задания

### 1.1 Анализ предметной области

Предметная область — магазин электротехники с различными функциональными зонами (склад, торговый зал) и определенным набором бизнес-процессов: закупка товаров у поставщиков, учет складских остатков, продажи товаров клиентам. Взаимодействие между клиентами, менеджерами и складом создает структуру данных, которую необходимо отразить в системе.

Цель информационной системы:

- 1) Обеспечить учет и управление данными о товаре, поставщиках, продажах и складских запасах.
- 2) Организовать хранение данных о продажах и поставках с возможностью быстрого доступа к ним для формирования отчетности.

### 1.2 Анализ потребностей пользователей и требований заказчика

Потребности пользователей включают:

- 1) Учет всех данных по продажам и закупкам;
- 2) Генерация сводных отчетов (количество проданной техники, выручка, прибыль) для анализа рентабельности и оценки работы магазина;
- 3) Учет остатков на складе и автоматическое обновление при поступлениях и продажах товаров.

Требования к системе включают:

- 1) Фиксация даты продажи, количества проданного товара, описания товара, стоимости, данных менеджера, информации о поставщике, а также текущих запасов на складе;
- 2) Хранение изображений товаров для удобства визуального восприятия;

					МИВУ.09.03.04-13.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		8

3) Реализация хранимых процедур и/или триггеров для автоматизации операций (например, обновление остатков при поступлениях и продажах).

### 1.3 Исследование технической реализуемости

Использование СУБД PostgreSQL соответствует поставленным задачам:

- 1) PostgreSQL поддерживает хранимые процедуры и триггеры, что позволит автоматизировать обновление данных в базе и обеспечить целостность информации.
- 2) СУБД позволяет хранить изображения (например, в виде полей типа BLOB), что соответствует требованиям.
- 3) PostgreSQL предоставляет высокую производительность и возможности масштабирования, необходимые для ведения учета продаж и складских остатков, особенно при расширении бизнеса.

### 1.4 Выбор СУБД и обоснование выбора

СУБД PostgreSQL выбрана с учетом следующих критериев:

- 1) Модель данных: реляционная модель данных PostgreSQL подходит для хранения табличных данных о товарах, поставках и продажах.
- 2) Производительность: PostgreSQL обеспечивает высокую производительность при обработке больших объемов данных и поддерживает сложные запросы, что особенно важно для отчетности.
- 3) Масштабируемость: PostgreSQL легко масштабируется, что позволяет расширять базу данных по мере роста объемов информации.
- 4) Удобство и надежность: PostgreSQL известна высокой надежностью, что делает её хорошим выбором для деловых приложений с критически важными данными.

					МИВУ.09.03.04-13.000 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

## 1.5 Постановка задачи и предварительный анализ

На данном этапе формулируются требования к системе, определяются ключевые функции и строится общий план. Основные функции системы:

- 1) Учет продаж и поставок: фиксировать продажи товаров, поступления от поставщиков и обновлять данные о текущих запасах.
- 2) Отчетность: создание отчетов для анализа продаж, прибыли и количества проданной техники.
- 3) Хранение изображений: обеспечение хранения изображений товаров.

Для того чтобы создать программу, необходимо учесть то, что она создается, прежде всего, для пользователя, и поэтому немаловажным требованием к программе должен стать удобный и интуитивно понятный интерфейс. Необходимо предусмотреть все возможности управления приложением, чтобы упростить работу пользователя и максимально обеспечить эффективность работы.

Программа должна правильно работать с данными, т.е. всегда должен выводиться нужный результат, требуемый пользователю. Приложение должно мгновенно реагировать на действия пользователя и в зависимости от запроса с его стороны формировать выходной результат.

## 2.Разработка моделей данных

Данный этап является самым важным при создании АИС. Здесь выделяются сущности, атрибуты сущностей и связи между сущностями. На основе полученной диаграммы “Сущность – связь” или логической модели строится физическая модель.

### 2.1 Концептуальная модель данных

Концептуальная модель отображает предметную область в виде взаимосвязанных объектов без указания способов их физического хранения. Концептуальная модель представляет интегрированные концептуальные требования всех пользователей к базе данных данной предметной области.

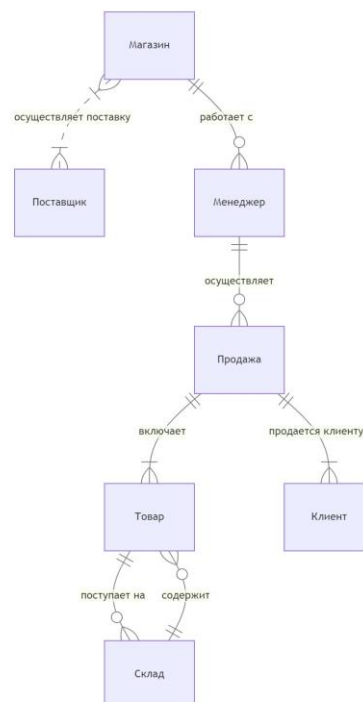


Рисунок 1 – Концептуальная модель данных



## 2.2 Логическая модель данных

Она отражает логические связи между атрибутами объектов вне зависимости от их содержания и среды хранения и может быть реляционной, иерархической или сетевой. Таким образом, логическая модель отображает логические связи между информационными данными в данной концептуальной модели.

Различным пользователям в информационной модели соответствуют различные подмножества ее логической модели. Таким образом, внешняя модель пользователя представляет собой отображение концептуальных требований этого пользователя в логической модели и соответствует тем представлениям, которые пользователь получает о предметной области на основе логической модели. Следовательно, насколько хорошо спроектирована внешняя модель, настолько полно и точно информационная модель отображает предметную область и настолько полно и точно работает автоматизированная система управления этой предметной областью.

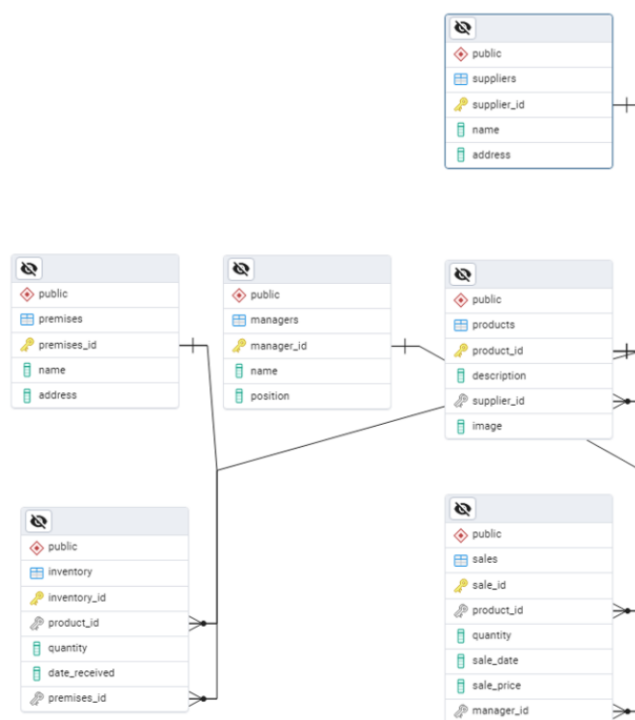


Рисунок 2 – Логическая модель данных

2.3 Физическая модель данных

На основе логической модели базы данных была спроектирована физическая модель, показанная на рисунке 3:

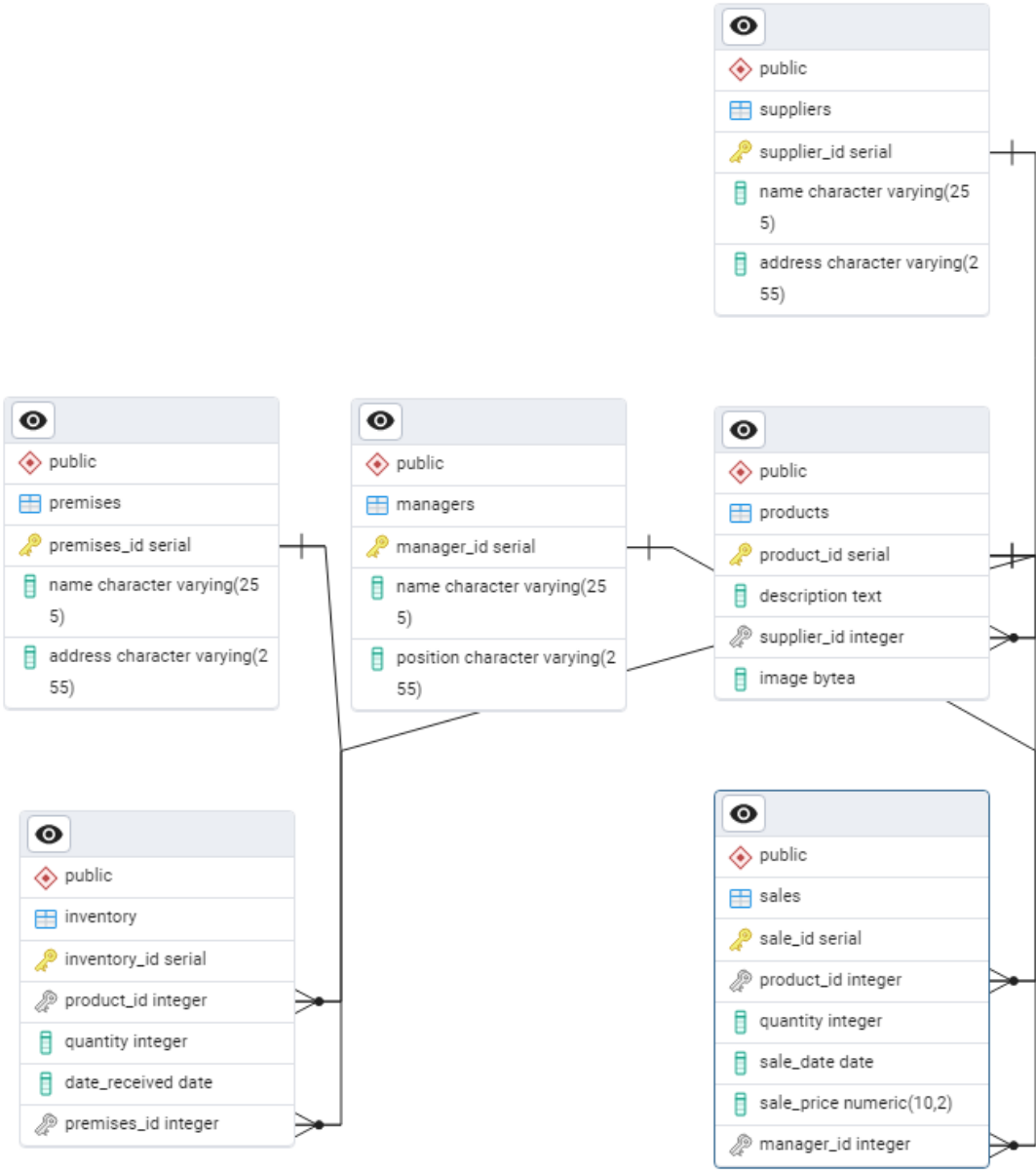


Рисунок 3 – Физическая модель данных

### 3.Разработка и реализация АИС

#### 3.1 Создание базы данных

База данных будет создана в программе PostgreSQL 17. Для её создания нужно использовать физическую модель.

Названия таблиц базы данных:

- 1) Inventory (содержит информацию о поступлении товаров на склад);
- 2) Managers (содержит информацию о менеджерах);
- 3) Premises (содержит информацию о помещениях);
- 4) Products (хранит информацию о товарах);
- 5) Sales (хранит информацию о продажах);
- 6) Suppliers (содержит информацию о поставщиках).

Типы полей для таблиц представлены ниже (Рис. 4 - 9).







	inventory
	inventory_id serial
	product_id integer
	quantity integer
	date_received date
	premises_id integer

Рисунок 4 – Таблица «Inventory»





	managers
	manager_id serial
	name character varying(255)
	position character varying (255)

Рисунок 5 – Таблица «Managers»





	premises
	premises_id serial
	name character varying(255)
	address character varying (255)

Рисунок 6 – Таблица «Premises»






	products
	product_id serial
	description text
	supplier_id integer
	image bytea

Рисунок 7 – Таблица «Products»








	sales
	sale_id serial
	product_id integer
	quantity integer
	sale_date date
	sale_price numeric(10,2)
	manager_id integer

Рисунок 8 – Таблица «Sales»





	suppliers
	supplier_id serial
	name character varying(255)
	address character varying(255)

Рисунок 9 – Таблица «Suppliers»

### 3.2 Описание программы

Приложение состоит из нескольких форм:

- 1) Главная форма;
- 2) Товары;
- 3) Поставщики;
- 4) Менеджеры;
- 5) Склад;
- 6) Продажи;
- 7) Генерация отчётов;
- 8) Помещения.

На главной форме, пользователь может выбирать между формами, нажатием специальных кнопок.

На форме «Товары», можно увидеть информацию из соответствующей таблицы, добавить новые данные, изменить данные в таблице, удалить данные из таблицы, выполнить поиск нужного товара по его описанию.

На форме «Поставщики», можно увидеть информацию из соответствующей таблицы, добавить новые данные, изменить данные в таблице, удалить данные из таблицы, выполнить поиск нужного поставщика по его названию.

На форме «Менеджеры», можно увидеть информацию из соответствующей таблицы, добавить новые данные, изменить данные в таблице, удалить данные из таблицы, выполнить поиск нужного менеджера по его имени.

					МИВУ.09.03.04-13.000 ПЗ	Лист
						17
Изм.	Лист	№ докум.	Подпись	Дата		

На форме «Склад», можно увидеть информацию из соответствующей таблицы, добавить новые данные, изменить данные в таблице, удалить данные из таблицы, выполнить поиск нужного товара по его ID.

На форме «Продажи», можно увидеть информацию из соответствующей таблицы, добавить новые данные, изменить данные в таблице, удалить данные из таблицы, выполнить поиск нужного товара по его ID.

На форме «Генерация отчётов», можно сгенерировать отчёты о количестве проданной техники и выручке от продаж.

На форме «Помещения» можно, можно увидеть информацию из соответствующей таблицы, добавить новые данные, изменить данные в таблице, удалить данные из таблицы, выполнить поиск нужного помещения по его названию.

Для предоставления информации в удобном для пользователя виде, а также для взаимодействия пользователя с программой были использованы следующие визуальные компоненты:

- button – компонент для выполнения различных действий над базой данных;
- dataGridView – отображает данные в табличном виде;
- textbox – компонент, необходимый для ввода информации и ее отображения;
- groupBox – компонент, необходимый для группировки других визуальных компонентов формы.

### 3.3 Руководство программиста

Приложение «Информационная система магазина электротехники» имеет открытый исходный код, распространяющийся с программой, поэтому любой желающий может его усовершенствовать или изменить.

Для подключения к базе данных используется класс `NpgsqlConnection`, который предоставляет открытое подключение к базе данных PostgreSQL:

```
private string connectionString=  
"Host=localhost;Username=postgres;Password=admin1234;Database=electrost";  
  
var conn = new NpgsqlConnection(connectionString)
```

Запрос для вывода информации в PostgreSQL, выглядит таким образом:

```
SELECT premises_id, name, address FROM premises;
```

Для вывода всех данных используя язык высокого уровня C#, используется класс `NpgsqlCommand`, который представляет набор выполняемых над данными команд.

```
using (var cmd = new NpgsqlCommand("SELECT premises_id, name, address FROM  
premises", conn))  
{  
    var reader = cmd.ExecuteReader();  
    var dt = new DataTable();  
    dt.Load(reader);  
    dataGridViewPremises.DataSource = dt;  
}
```

Для удаления записей из таблицы в SQL Server используется такой запрос:

```
DELETE FROM premises WHERE premises_id = @premises_id;
```

Для удаления записи всё также используется класс `NpgsqlCommand`:

```
using (var cmd = new NpgsqlCommand("DELETE FROM premises WHERE premises_id =  
@premises_id", conn))  
{  
    cmd.Parameters.AddWithValue("@premises_id", int.Parse(txtPremisesId.Text));  
    cmd.ExecuteNonQuery();  
}
```

Для добавления записей в PostgreSQL используется запрос:

```
INSERT INTO premises (name, address) VALUES (@name, @address);
```

Для добавления записи также используется `NpgsqlCommand`:



Для изменения данных в PostgreSQL используется запрос:

```
UPDATE premises SET name = @name, address = @address WHERE premises_id = @premises_id;
```

Для изменения данных все также используется NpgsqlCommand:

```
using (var cmd = new NpgsqlCommand("UPDATE premises SET name = @name, address = @address WHERE premises_id = @premises_id", conn))
{
    cmd.Parameters.AddWithValue("@premises_id", int.Parse(txtPremisesId.Text));
    cmd.Parameters.AddWithValue("@name", txtName.Text);
    cmd.Parameters.AddWithValue("@address", txtAddress.Text);
    cmd.ExecuteNonQuery();
}
```

### 3.4 Руководство пользователя

Для запуска приложения нужно открыть файл SYBDKR.exe. После этого появится главная форма программы, на которой находятся кнопки вызова форм.

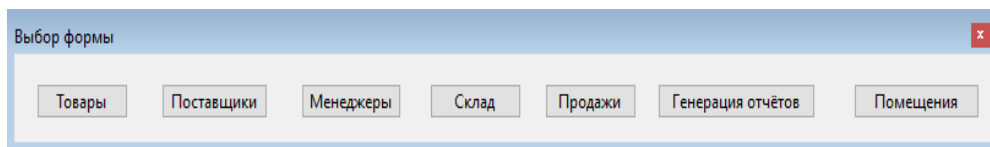


Рисунок 10 – Главная форма

Если нажать на кнопку «Товары», то появится форма «Товары».

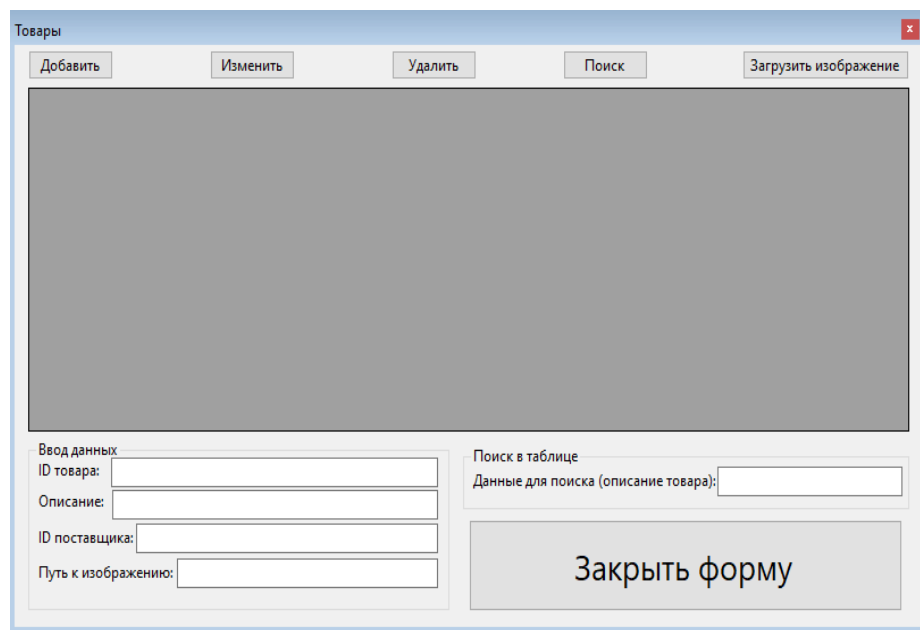


Рисунок 11 – Форма «Товары»

На этой форме можно увидеть всю доступную информацию о товарах.

Чтобы добавить новый товар, сначала нужно загрузить изображение товара. Для этого нужно нажать на кнопку «Загрузить изображение» и выбрать нужное изображение. Потом нужно ввести описание товара и ID поставщика в соответствующие текстовые поля и нажать на кнопку «Добавить».

Чтобы изменить данные о выбранном товаре, нужно нажать на кнопку «Изменить» и ввести новое описание и ID поставщика. Если нужно изменить изображение, то надо нажать на кнопку «Загрузить изображение» и выбрать нужное изображение.

Чтобы удалить данные о выбранном товаре из таблицы, нужно выбрать этот товар в таблице и нажать на кнопку «Удалить».

Чтобы найти нужный товар, нужно ввести его описание в специальное текстовое поле и нажать кнопку «Поиск».

Если нажать на кнопку «Поставщики», то появится форма «Поставщики».

Поставщики

Добавить Изменить Удалить Поиск поставщика

Ввод данных

ID:

Название:

Адрес:

Поиск данных (по названию):

Название:

Закреть форму

Рисунок 12 – Форма «Поставщики»

На этой форме можно увидеть всю доступную информацию о поставщиках.

Чтобы добавить нового поставщика, сначала нужно ввести название и адрес поставщика в соответствующие текстовые поля и потом нажать на кнопку «Добавить».

Чтобы изменить данные о выбранном поставщике, нужно нажать на кнопку «Изменить» и ввести новое название и адрес поставщика.

Чтобы удалить данные о выбранном поставщике из таблицы, нужно выбрать этого поставщика в таблице и нажать на кнопку «Удалить».

Чтобы найти нужного поставщика, нужно ввести его название в специальное текстовое поле и нажать кнопку «Поиск».

Аналогичные действия нужно делать в формах: «Менеджеры», «Склад», «Продажи», «Помещения».

Если нажать на кнопку «Генерация отчётов», то появится форма «Генерация отчётов».

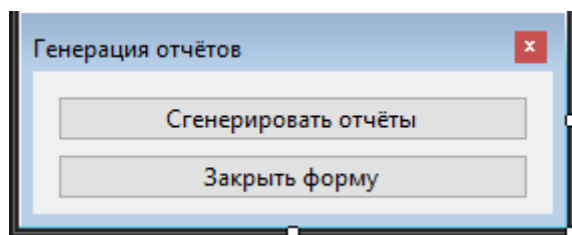


Рисунок 13 – Форма «Генерация отчётов»

Если нажать на кнопку «Сгенерировать отчёт», то сгенерируются два отчёта: отчёт о количестве проданной техники, и отчёт о выручке от продаж. Они появятся в папке с программой.

## 4.Тестирование АИС

Одним из важнейших этапов создания приложения является его тестирование и отладка. Тестирование позволяет выявить скрытые и явные недостатки программы, либо убедиться в ее пригодности для применения. Обнаруженные недостатки устраняются в ходе отладки.

Целью тестирования является проверка работоспособности программы, правильности выполнения всех функций, описанных выше, а также правильности обработки всех исключений, возникающих в ходе работы программы.

Процесс тестирования:

### 1) Добавление нового объекта в БД

Добавим новый смартфон в базу данных:

Товары

	product_id	description	supplier_id	image
▶	3	POCO C61	1	
*				

Ввод данных

ID товара:

Описание:

ID поставщика:

Путь к изображению:

Поиск в таблице

Данные для поиска (описание товара):

Заккрыть форму

Рисунок 14 – Результат добавления нового объекта

Теперь снова попробуем добавить тот же смартфон, но укажем ID поставщика, которого нет в таблице «Поставщики».

Возникает исключение, которое говорит нам о том, что поставщика с таким ID в таблице «Поставщики» нет:

```
Npgsql.PostgresException: "23503: INSERT или UPDATE в таблице "products" нарушает ограничение внешнего ключа "products_supplier_id_fkey"
```

Рисунок 15 – Ошибка добавления товара

## 2) Изменение объекта в БД

Изменим данные о смартфоне:

Товары

Добавить Изменить Удалить Поиск Загрузить изображение

	product_id	description	supplier_id	image
▶	3	POCO C62	1	
*				

Ввод данных

ID товара:

Описание:

ID поставщика:

Путь к изображению:

Поиск в таблице

Данные для поиска (описание товара):

Заккрыть форму

Рисунок 16 – Результат изменения объекта

## 3) Удаление объекта из БД

Удалим смартфон из базы данных:

Товары

Добавить

Изменить

Удалить

Поиск

Загрузить изображение

	product_id	description	supplier_id	image
*				

Ввод данных

ID товара:

Описание:

ID поставщика:

Путь к изображению:

Поиск в таблице

Данные для поиска (описание товара):

Заккрыть форму

Рисунок 17 – Результат удаления объекта

В результате тестирования программы, недостатков, влияющих на корректную работу приложения, не выявлено.

## Заключение

В ходе выполнения курсовой работы была разработана автоматизированная информационная система для управления магазином электротехники. Система включает в себя модули для управления товарами, поставщиками, менеджерами, складом и продажами, а также для генерации отчетов.

Основные достижения проекта включают:

- 1) Разработка базы данных: Создана реляционная база данных с использованием СУБД PostgreSQL, которая включает таблицы для хранения информации о товарах, поставщиках, менеджерах, складе и продажах. База данных обеспечивает надежное хранение и быстрый доступ к данным.
- 2) Разработка пользовательского интерфейса: Создано приложение на платформе Windows Forms с использованием языка программирования C#. Интерфейс включает формы для управления каждой из таблиц базы данных, а также для генерации отчетов. Пользовательский интерфейс интуитивно понятен и удобен в использовании, что позволяет пользователям эффективно управлять данными.
- 3) Реализация функциональности: Внедрены функции для добавления, обновления, удаления и поиска данных в каждой из таблиц базы данных. Реализована возможность загрузки изображений товаров через компонент OpenFileDialog. Также реализована хранящая процедура для добавления новых товаров на склад и обновления их количества.

4) Генерация отчетов: Разработана функциональность для генерации сводных отчетов в формате PDF, включая отчеты о количестве проданной техники, выручке от продаж и прибыли. Это позволяет руководству магазина получать актуальную информацию для принятия управленческих решений.

5) Интеграция и тестирование: Все компоненты системы были интегрированы и протестированы на предмет корректности работы и производительности. Система показала высокую надежность и удобство в использовании.

Разработанная автоматизированная информационная система значительно упрощает управление магазином электротехники, повышает эффективность работы сотрудников и улучшает качество обслуживания клиентов. Внедрение такой системы позволяет магазину оптимизировать бизнес-процессы, снизить затраты на управление и повысить конкурентоспособность на рынке.

В будущем планируется расширение функциональности системы, включая добавление модулей для управления клиентами, учета финансов и интеграции с внешними системами. Также планируется улучшение пользовательского интерфейса и повышение производительности системы.

Таким образом, разработанная автоматизированная информационная система для магазина электротехники является важным шагом на пути к цифровизации и автоматизации бизнес-процессов, что способствует устойчивому развитию и росту компании.



## Список используемой литературы

- 1) Тарасов, С. В. СУБД для программиста. Базы данных изнутри / С. В. Тарасов. — Москва : СОЛОН-Пресс, 2018. — 320 с. — ISBN 978-2-7466-7383-0. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/90409.html> (дата обращения: 05.12.2024). — Режим доступа: для авторизир. Пользователей
- 2) Прокушев, Я. Е. Базы данных : учебник с практикумом / Я. Е. Прокушев. — 2-е изд. — Санкт-Петербург : Интермедия, 2022. — 264 с. — ISBN 978-5-4383-0250-6. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/120171.html> (дата обращения: 30.11.2024). — Режим доступа: для авторизир. Пользователей
- 3) Маркин, А. В. СУБД «Ред База Данных». Основы SQL : учебное пособие / А. В. Маркин. — Москва : Ай Пи Ар Медиа, 2022. — 460 с. — ISBN 978-5-4497-1605-7. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/119617.html> (дата обращения: 04.12.2024). — Режим доступа: для авторизир. пользователей

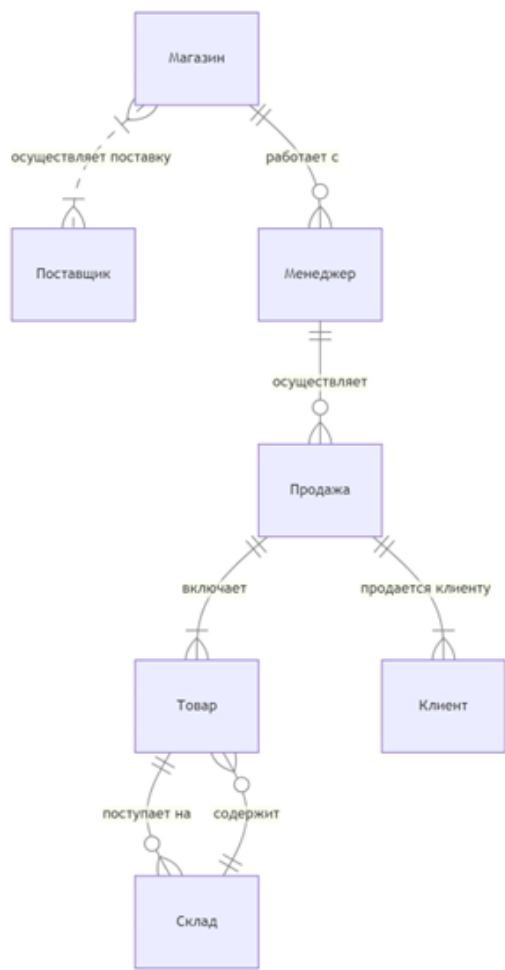


Рисунок А.1 – Концептуальная модель данных

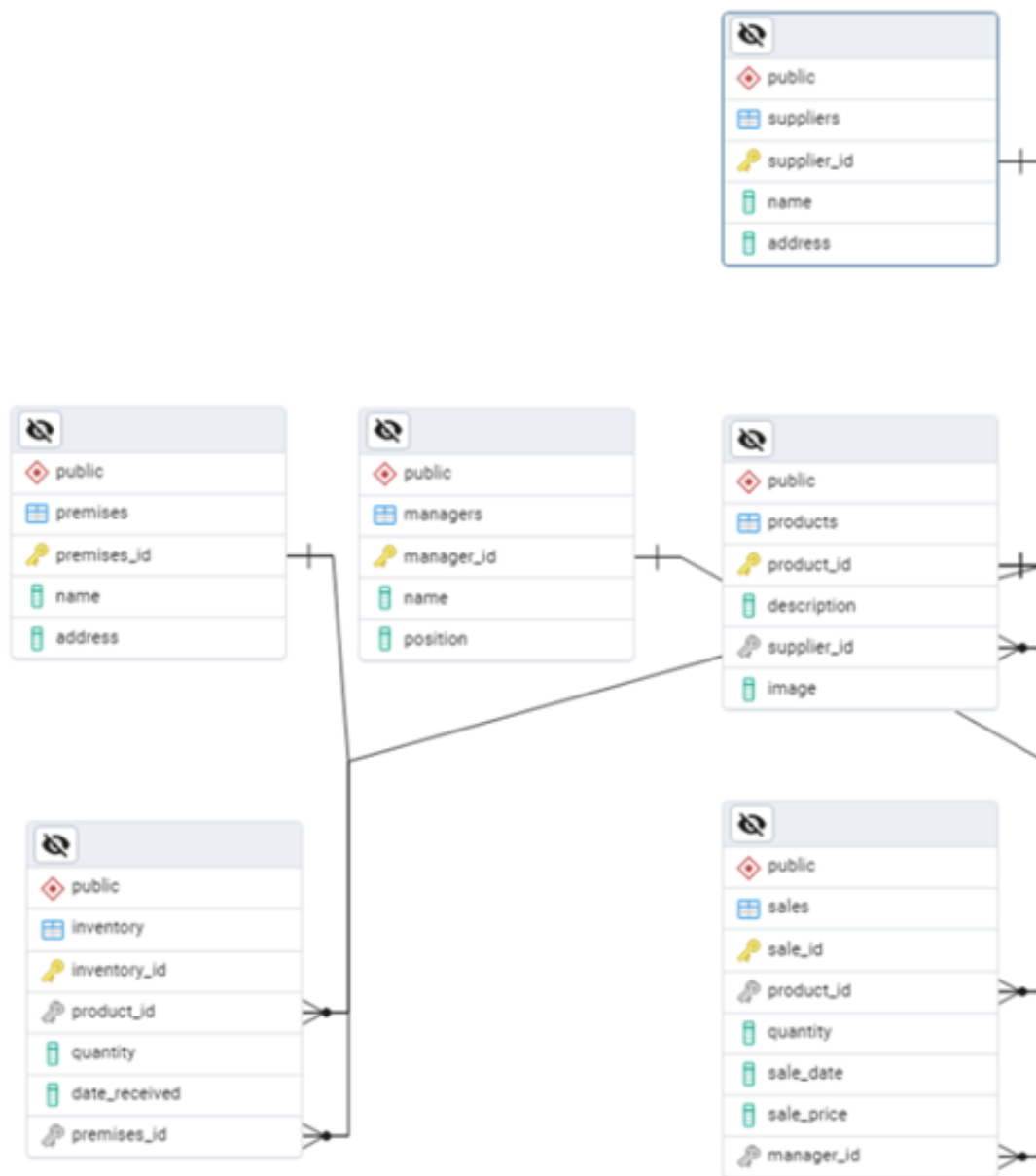


Рисунок А.2 – Логическая модель данных

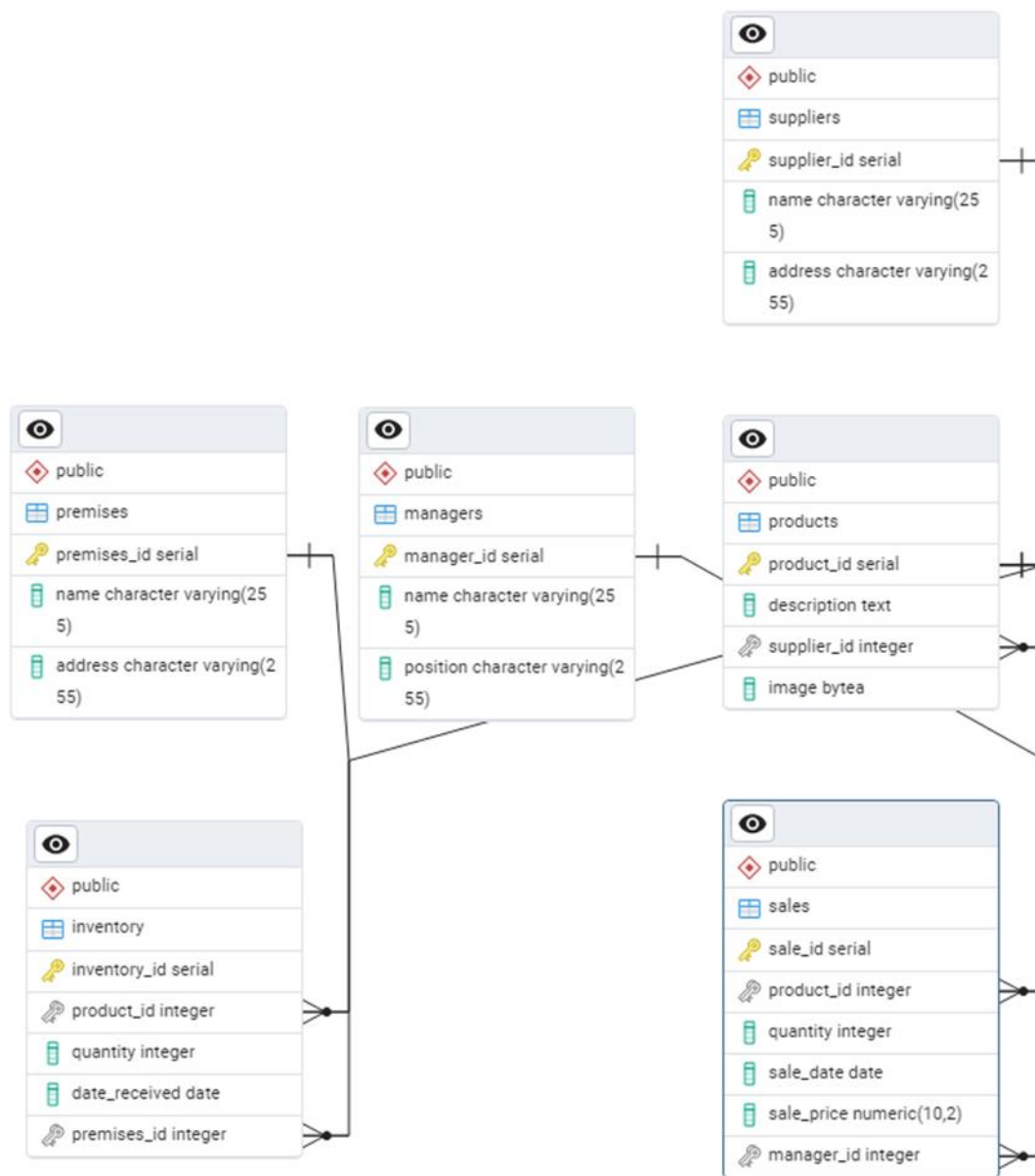


Рисунок А.3 – Физическая модель данных

## Приложение Б. Текст программы

### Form1.cs

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void btnProducts_Click(object sender, EventArgs e)
    {
        ProductsForm productsForm = new ProductsForm();
        productsForm.Show();
    }

    private void btnManagers_Click(object sender, EventArgs e)
    {
        ManagersForm managersForm = new ManagersForm();
        managersForm.Show();
    }

    private void btnSuppliers_Click(object sender, EventArgs e)
    {
        SuppliersForm suppliersForm = new SuppliersForm();
        suppliersForm.Show();
    }

    private void btnInventory_Click(object sender, EventArgs e)
    {
        InventoryForm inventoryForm = new InventoryForm();
        inventoryForm.Show();
    }

    private void btnSales_Click(object sender, EventArgs e)
    {
        SalesForm salesForm = new SalesForm();
        salesForm.Show();
    }

    private void btnReports_Click(object sender, EventArgs e)
    {
        ReportsForm reportsForm = new ReportsForm();
        reportsForm.Show();
    }

    private void btnPremises_Click(object sender, EventArgs e)
    {
        PremisesForm premisesForm = new PremisesForm();
        premisesForm.Show();
    }
}
```

### ProductsForm.cs

```
public partial class ProductsForm : Form
{
    private string connectionString =
        "Host=localhost;Username=postgres;Password=admin1234;Database=electrost";
}
```

					МИВУ.09.03.04-13.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		32

```

public ProductsForm()
{
    InitializeComponent();
    LoadProducts();
}

private void LoadProducts()
{
    using (var conn = new NpgsqlConnection(connectionString))
    {
        conn.Open();
        using (var cmd = new NpgsqlCommand("SELECT product_id, description,
supplier_id, image FROM products", conn))
        {
            var reader = cmd.ExecuteReader();
            var dt = new DataTable();
            dt.Load(reader);
            dataGridViewProducts.DataSource = dt;
        }
    }
}

private void ClearFields()
{
    txtProductId.Clear();
    txtDescription.Clear();
    txtSupplierId.Clear();
    txtImagePath.Clear();
}

private void btnAddProduct_Click(object sender, EventArgs e)
{
    using (var conn = new NpgsqlConnection(connectionString))
    {
        conn.Open();
        using (var cmd = new NpgsqlCommand("INSERT INTO products (description,
supplier_id, image) VALUES (@description, @supplier_id, @image)", conn))
        {
            cmd.Parameters.AddWithValue("@description", txtDescription.Text);
            cmd.Parameters.AddWithValue("@supplier_id",
int.Parse(txtSupplierId.Text));
            cmd.Parameters.AddWithValue("@image",
File.ReadAllBytes(txtImagePath.Text));
            cmd.ExecuteNonQuery();
        }
    }
    LoadProducts();
    ClearFields();
}

private void btnUpdateProduct_Click(object sender, EventArgs e)
{
    using (var conn = new NpgsqlConnection(connectionString))
    {
        conn.Open();
        using (var cmd = new NpgsqlCommand("UPDATE products SET description =
@description, supplier_id = @supplier_id, image = @image WHERE product_id =
@product_id", conn))
        {
            cmd.Parameters.AddWithValue("@product_id",
int.Parse(txtProductId.Text));
            cmd.Parameters.AddWithValue("@description", txtDescription.Text);
            cmd.Parameters.AddWithValue("@supplier_id",
int.Parse(txtSupplierId.Text));

```

					МИВУ.09.03.04-13.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		33

```

        cmd.Parameters.AddWithValue("@image",
File.ReadAllBytes(txtImagePath.Text));
        cmd.ExecuteNonQuery();
    }
}
LoadProducts();
ClearFields();
}

private void btnDeleteProduct_Click(object sender, EventArgs e)
{
    using (var conn = new NpgsqlConnection(connectionString))
    {
        conn.Open();
        using (var cmd = new NpgsqlCommand("DELETE FROM products WHERE
product_id = @product_id", conn))
        {
            cmd.Parameters.AddWithValue("@product_id",
int.Parse(txtProductId.Text));
            cmd.ExecuteNonQuery();
        }
    }
    LoadProducts();
    ClearFields();
}

private void btnSearchProduct_Click(object sender, EventArgs e)
{
    using (var conn = new NpgsqlConnection(connectionString))
    {
        conn.Open();
        using (var cmd = new NpgsqlCommand("SELECT product_id, description,
supplier_id FROM products WHERE description LIKE @description", conn))
        {
            cmd.Parameters.AddWithValue("@description", "%" + txtSearch.Text +
"%");
            var reader = cmd.ExecuteReader();
            var dt = new DataTable();
            dt.Load(reader);
            dataGridViewProducts.DataSource = dt;
        }
    }
}

private void btnUploadImage_Click(object sender, EventArgs e)
{
    using (OpenFileDialog openFileDialog = new OpenFileDialog())
    {
        openFileDialog.Filter = "Image Files|*.jpg;*.jpeg;*.png;*.gif;*.bmp";
        if (openFileDialog.ShowDialog() == DialogResult.OK)
        {
            txtImagePath.Text = openFileDialog.FileName;
        }
    }
}

private void dataGridViewProducts_CellClick(object sender,
DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0)
    {
        DataGridViewRow row = dataGridViewProducts.Rows[e.RowIndex];
        txtProductId.Text = row.Cells["product_id"].Value.ToString();
        txtDescription.Text = row.Cells["description"].Value.ToString();
        txtSupplierId.Text = row.Cells["supplier_id"].Value.ToString();
    }
}

```

```

    }

    private void exitBtn_Click(object sender, EventArgs e)
    {
        this.Close();
    }
}

```

## SuppliersForm.cs

```

public partial class SuppliersForm : Form
{
    private string connectionString =
"Host=localhost;Username=postgres;Password=admin1234;Database=electrost";

    public SuppliersForm()
    {
        InitializeComponent();
        LoadSuppliers();
    }

    private void LoadSuppliers()
    {
        using (var conn = new NpgsqlConnection(connectionString))
        {
            conn.Open();
            using (var cmd = new NpgsqlCommand("SELECT supplier_id, name, address
FROM suppliers", conn))
            {
                var reader = cmd.ExecuteReader();
                var dt = new DataTable();
                dt.Load(reader);
                dataGridViewSuppliers.DataSource = dt;
            }
        }
    }

    private void ClearFields()
    {
        txtSupplierId.Clear();
        txtName.Clear();
        txtAddress.Clear();
    }

    private void btnAddSupplier_Click(object sender, EventArgs e)
    {
        using (var conn = new NpgsqlConnection(connectionString))
        {
            conn.Open();
            using (var cmd = new NpgsqlCommand("INSERT INTO suppliers (name,
address) VALUES (@name, @address)", conn))
            {
                cmd.Parameters.AddWithValue("@name", txtName.Text);
                cmd.Parameters.AddWithValue("@address", txtAddress.Text);
                cmd.ExecuteNonQuery();
            }
        }
        LoadSuppliers();
        ClearFields();
    }

    private void btnUpdateSupplier_Click(object sender, EventArgs e)
    {
        using (var conn = new NpgsqlConnection(connectionString))
        {

```



```

        conn.Open();
        using (var cmd = new NpgsqlCommand("UPDATE suppliers SET name = @name,
address = @address WHERE supplier_id = @supplier_id", conn))
        {
            cmd.Parameters.AddWithValue("@supplier_id",
int.Parse(txtSupplierId.Text));
            cmd.Parameters.AddWithValue("@name", txtName.Text);
            cmd.Parameters.AddWithValue("@address", txtAddress.Text);
            cmd.ExecuteNonQuery();
        }
    }
    LoadSuppliers();
    ClearFields();
}

private void btnSearchSupplier_Click(object sender, EventArgs e)
{
    using (var conn = new NpgsqlConnection(connectionString))
    {
        conn.Open();
        using (var cmd = new NpgsqlCommand("SELECT supplier_id, name, address
FROM suppliers WHERE name LIKE @name", conn))
        {
            cmd.Parameters.AddWithValue("@name", "%" + txtSearch.Text + "%");
            var reader = cmd.ExecuteReader();
            var dt = new DataTable();
            dt.Load(reader);
            dataGridViewSuppliers.DataSource = dt;
        }
    }
}

private void btnDeleteSupplier_Click(object sender, EventArgs e)
{
    using (var conn = new NpgsqlConnection(connectionString))
    {
        conn.Open();
        using (var cmd = new NpgsqlCommand("DELETE FROM suppliers WHERE
supplier_id = @supplier_id", conn))
        {
            cmd.Parameters.AddWithValue("@supplier_id",
int.Parse(txtSupplierId.Text));
            cmd.ExecuteNonQuery();
        }
    }
    LoadSuppliers();
    ClearFields();
}

private void dataGridViewSuppliers_CellClick(object sender,
DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0)
    {
        DataGridViewRow row = dataGridViewSuppliers.Rows[e.RowIndex];
        txtSupplierId.Text = row.Cells["supplier_id"].Value.ToString();
        txtName.Text = row.Cells["name"].Value.ToString();
        txtAddress.Text = row.Cells["address"].Value.ToString();
    }
}

private void btnExit_Click(object sender, EventArgs e)
{
    this.Close();
}
}

```

## ManagersForm.cs

```
public partial class ManagersForm : Form
{
    private string connectionString =
        "Host=localhost;Username=postgres;Password=admin1234;Database=electrost";

    public ManagersForm()
    {
        InitializeComponent();
        LoadManagers();
    }

    private void LoadManagers()
    {
        using (var conn = new NpgsqlConnection(connectionString))
        {
            conn.Open();
            using (var cmd = new NpgsqlCommand("SELECT manager_id, name, position
FROM managers", conn))
            {
                var reader = cmd.ExecuteReader();
                var dt = new DataTable();
                dt.Load(reader);
                dataGridViewManagers.DataSource = dt;
            }
        }
    }

    private void btnAddManager_Click(object sender, EventArgs e)
    {
        using (var conn = new NpgsqlConnection(connectionString))
        {
            conn.Open();
            using (var cmd = new NpgsqlCommand("INSERT INTO managers (name,
position) VALUES (@name, @position)", conn))
            {
                cmd.Parameters.AddWithValue("@name", txtName.Text);
                cmd.Parameters.AddWithValue("@position", txtPosition.Text);
                cmd.ExecuteNonQuery();
            }
        }
        LoadManagers();
        ClearFields();
    }

    private void ClearFields()
    {
        txtManagerId.Clear();
        txtName.Clear();
        txtPosition.Clear();
    }

    private void btnUpdateManager_Click(object sender, EventArgs e)
    {
        using (var conn = new NpgsqlConnection(connectionString))
        {
            conn.Open();
            using (var cmd = new NpgsqlCommand("UPDATE managers SET name = @name,
position = @position WHERE manager_id = @manager_id", conn))
            {
                cmd.Parameters.AddWithValue("@manager_id",
int.Parse(txtManagerId.Text));
                cmd.Parameters.AddWithValue("@name", txtName.Text);
                cmd.Parameters.AddWithValue("@position", txtPosition.Text);
            }
        }
    }
}
```

					МИВУ.09.03.04-13.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		37

```

        cmd.ExecuteNonQuery();
    }
}
LoadManagers();
ClearFields();
}

private void btnDeleteManager_Click(object sender, EventArgs e)
{
    using (var conn = new NpgsqlConnection(connectionString))
    {
        conn.Open();
        using (var cmd = new NpgsqlCommand("DELETE FROM managers WHERE
manager_id = @manager_id", conn))
        {
            cmd.Parameters.AddWithValue("@manager_id",
int.Parse(txtManagerId.Text));
            cmd.ExecuteNonQuery();
        }
    }
    LoadManagers();
    ClearFields();
}

private void btnSearchManager_Click(object sender, EventArgs e)
{
    using (var conn = new NpgsqlConnection(connectionString))
    {
        conn.Open();
        using (var cmd = new NpgsqlCommand("SELECT manager_id, name, position
FROM managers WHERE name LIKE @name", conn))
        {
            cmd.Parameters.AddWithValue("@name", "%" + txtSearch.Text + "%");
            var reader = cmd.ExecuteReader();
            var dt = new DataTable();
            dt.Load(reader);
            dataGridViewManagers.DataSource = dt;
        }
    }
}

private void btnExit_Click(object sender, EventArgs e)
{
    this.Close();
}

private void dataGridViewManagers_CellClick(object sender,
DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0)
    {
        DataGridViewRow row = dataGridViewManagers.Rows[e.RowIndex];
        txtManagerId.Text = row.Cells["manager_id"].Value.ToString();
        txtName.Text = row.Cells["name"].Value.ToString();
        txtPosition.Text = row.Cells["position"].Value.ToString();
    }
}
}

```

## InventoryForm.cs

```

public partial class InventoryForm : Form
{
    private string connectionString =
"Host=localhost;Username=postgres;Password=admin1234;Database=electrost";

```

					МИВУ.09.03.04-13.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		38

```

public InventoryForm()
{
    InitializeComponent();
    LoadInventory();
}

private void LoadInventory()
{
    using (var conn = new NpgsqlConnection(connectionString))
    {
        conn.Open();
        using (var cmd = new NpgsqlCommand("SELECT inventory_id, product_id,
quantity, date_received, premises_id FROM inventory", conn))
        {
            var reader = cmd.ExecuteReader();
            var dt = new DataTable();
            dt.Load(reader);
            dataGridViewInventory.DataSource = dt;
        }
    }
}

private void ClearFields()
{
    txtInventoryId.Clear();
    txtProductId.Clear();
    txtQuantity.Clear();
    txtDateReceived.Clear();
    txtPremisesId.Clear();
}

private void btnAddInventory_Click(object sender, EventArgs e)
{
    using (var conn = new NpgsqlConnection(connectionString))
    {
        conn.Open();
        using (var cmd = new NpgsqlCommand("INSERT INTO inventory (product_id,
quantity, date_received, premises_id) VALUES (@product_id, @quantity,
@date_received, @premises_id)", conn))
        {
            cmd.Parameters.AddWithValue("@product_id",
int.Parse(txtProductId.Text));
            cmd.Parameters.AddWithValue("@quantity",
int.Parse(txtQuantity.Text));
            cmd.Parameters.AddWithValue("@date_received",
DateTime.Parse(txtDateReceived.Text));
            cmd.Parameters.AddWithValue("@premises_id",
int.Parse(txtPremisesId.Text));
            cmd.ExecuteNonQuery();
        }
        LoadInventory();
        ClearFields();
    }
}

private void btnUpdateInventory_Click(object sender, EventArgs e)
{
    using (var conn = new NpgsqlConnection(connectionString))
    {
        conn.Open();
        using (var cmd = new NpgsqlCommand("UPDATE inventory SET product_id =
@product_id, quantity = @quantity, date_received = @date_received, premises_id =
@premises_id WHERE inventory_id = @inventory_id", conn))
        {

```

```

        cmd.Parameters.AddWithValue("@inventory_id",
int.Parse(txtInventoryId.Text));
        cmd.Parameters.AddWithValue("@product_id",
int.Parse(txtProductId.Text));
        cmd.Parameters.AddWithValue("@quantity",
int.Parse(txtQuantity.Text));
        cmd.Parameters.AddWithValue("@date_received",
DateTime.Parse(txtDateReceived.Text));
        cmd.Parameters.AddWithValue("@premises_id",
int.Parse(txtPremisesId.Text));
        cmd.ExecuteNonQuery();
    }
}
LoadInventory();
ClearFields();
}

private void btnDeleteInventory_Click(object sender, EventArgs e)
{
    using (var conn = new NpgsqlConnection(connectionString))
    {
        conn.Open();
        using (var cmd = new NpgsqlCommand("DELETE FROM inventory WHERE
inventory_id = @inventory_id", conn))
        {
            cmd.Parameters.AddWithValue("@inventory_id",
int.Parse(txtInventoryId.Text));
            cmd.ExecuteNonQuery();
        }
    }
    LoadInventory();
    ClearFields();
}

private void btnSearchInventory_Click(object sender, EventArgs e)
{
    using (var conn = new NpgsqlConnection(connectionString))
    {
        conn.Open();
        using (var cmd = new NpgsqlCommand("SELECT inventory_id, product_id,
quantity, date_received, premises_id FROM inventory WHERE product_id = @product_id",
conn))
        {
            cmd.Parameters.AddWithValue("@product_id",
int.Parse(txtSearch.Text));
            var reader = cmd.ExecuteReader();
            var dt = new DataTable();
            dt.Load(reader);
            dataGridViewInventory.DataSource = dt;
        }
    }
}

private void btnExit_Click(object sender, EventArgs e)
{
    this.Close();
}

private void dataGridViewInventory_CellClick(object sender,
DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0)
    {
        DataGridViewRow row = dataGridViewInventory.Rows[e.RowIndex];
        txtInventoryId.Text = row.Cells["inventory_id"].Value.ToString();
        txtProductId.Text = row.Cells["product_id"].Value.ToString();
    }
}

```

					МИВУ.09.03.04-13.000 ПЗ	Лист
						40
Изм.	Лист	№ докум.	Подпись	Дата		

```

        txtQuantity.Text = row.Cells["quantity"].Value.ToString();
        txtDateReceived.Text = row.Cells["date_received"].Value.ToString();
        txtPremisesId.Text = row.Cells["premises_id"].Value.ToString();
    }
}

```

## SalesForm.cs

```

public partial class SalesForm : Form
{
    private string connectionString =
        "Host=localhost;Username=postgres;Password=admin1234;Database=electrost";

    public SalesForm()
    {
        InitializeComponent();
        LoadSales();
    }

    private void LoadSales()
    {
        using (var conn = new NpgsqlConnection(connectionString))
        {
            conn.Open();
            using (var cmd = new NpgsqlCommand("SELECT sale_id, product_id,
quantity, sale_date, sale_price, manager_id FROM sales", conn))
            {
                var reader = cmd.ExecuteReader();
                var dt = new DataTable();
                dt.Load(reader);
                dataGridViewSales.DataSource = dt;
            }
        }
    }

    private void btnAddSale_Click(object sender, EventArgs e)
    {
        using (var conn = new NpgsqlConnection(connectionString))
        {
            conn.Open();
            using (var cmd = new NpgsqlCommand("INSERT INTO sales (product_id,
quantity, sale_date, sale_price, manager_id) VALUES (@product_id, @quantity,
@sale_date, @sale_price, @manager_id)", conn))
            {
                cmd.Parameters.AddWithValue("@product_id",
int.Parse(txtProductId.Text));
                cmd.Parameters.AddWithValue("@quantity",
int.Parse(txtQuantity.Text));
                cmd.Parameters.AddWithValue("@sale_date",
DateTime.Parse(txtSaleDate.Text));
                cmd.Parameters.AddWithValue("@sale_price",
decimal.Parse(txtSalePrice.Text));
                cmd.Parameters.AddWithValue("@manager_id",
int.Parse(txtManagerId.Text));
                cmd.ExecuteNonQuery();
            }
        }
        LoadSales();
        ClearFields();
    }

    private void ClearFields()
    {
        txtSaleId.Clear();
    }
}

```

					МИВУ.09.03.04-13.000 ПЗ	Лист
						41
Изм.	Лист	№ докум.	Подпись	Дата		

```

        txtProductId.Clear();
        txtQuantity.Clear();
        txtSaleDate.Clear();
        txtSalePrice.Clear();
        txtManagerId.Clear();
    }

    private void btnUpdateSale_Click(object sender, EventArgs e)
    {
        using (var conn = new NpgsqlConnection(connectionString))
        {
            conn.Open();
            using (var cmd = new NpgsqlCommand("UPDATE sales SET product_id =
@product_id, quantity = @quantity, sale_date = @sale_date, sale_price = @sale_price,
manager_id = @manager_id WHERE sale_id = @sale_id", conn))
            {
                cmd.Parameters.AddWithValue("@sale_id", int.Parse(txtSaleId.Text));
                cmd.Parameters.AddWithValue("@product_id",
int.Parse(txtProductId.Text));
                cmd.Parameters.AddWithValue("@quantity",
int.Parse(txtQuantity.Text));
                cmd.Parameters.AddWithValue("@sale_date",
DateTime.Parse(txtSaleDate.Text));
                cmd.Parameters.AddWithValue("@sale_price",
decimal.Parse(txtSalePrice.Text));
                cmd.Parameters.AddWithValue("@manager_id",
int.Parse(txtManagerId.Text));
                cmd.ExecuteNonQuery();
            }
        }
        LoadSales();
        ClearFields();
    }

    private void btnDeleteSale_Click(object sender, EventArgs e)
    {
        using (var conn = new NpgsqlConnection(connectionString))
        {
            conn.Open();
            using (var cmd = new NpgsqlCommand("DELETE FROM sales WHERE sale_id =
@sale_id", conn))
            {
                cmd.Parameters.AddWithValue("@sale_id", int.Parse(txtSaleId.Text));
                cmd.ExecuteNonQuery();
            }
        }
        LoadSales();
        ClearFields();
    }

    private void btnSearchSale_Click(object sender, EventArgs e)
    {
        using (var conn = new NpgsqlConnection(connectionString))
        {
            conn.Open();
            using (var cmd = new NpgsqlCommand("SELECT sale_id, product_id,
quantity, sale_date, sale_price, manager_id FROM sales WHERE product_id =
@product_id", conn))
            {
                cmd.Parameters.AddWithValue("@product_id",
int.Parse(txtSearch.Text));
                var reader = cmd.ExecuteReader();
                var dt = new DataTable();
                dt.Load(reader);
                dataGridViewSales.DataSource = dt;
            }
        }
    }

```

```

    }
}

private void dataGridViewSales_CellClick(object sender,
DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0)
    {
        DataGridViewRow row = dataGridViewSales.Rows[e.RowIndex];
        txtSaleId.Text = row.Cells["sale_id"].Value.ToString();
        txtProductId.Text = row.Cells["product_id"].Value.ToString();
        txtQuantity.Text = row.Cells["quantity"].Value.ToString();
        txtSaleDate.Text = row.Cells["sale_date"].Value.ToString();
        txtSalePrice.Text = row.Cells["sale_price"].Value.ToString();
        txtManagerId.Text = row.Cells["manager_id"].Value.ToString();
    }
}

private void btnExit_Click(object sender, EventArgs e)
{
    this.Close();
}
}

```

## ReportForm.cs

```

public partial class ReportsForm : Form
{
    private string connectionString =
        "Host=localhost;Username=postgres;Password=admin1234;Database=electrost";

    public ReportsForm()
    {
        InitializeComponent();
    }

    private void btnGenerateReport_Click(object sender, EventArgs e)
    {
        // Генерация отчета о количестве проданной техники
        GenerateSalesQuantityReport();

        // Генерация отчета о выручке от продаж
        GenerateSalesRevenueReport();
    }

    private void GenerateSalesQuantityReport()
    {
        using (var conn = new NpgsqlConnection(connectionString))
        {
            conn.Open();
            using (var cmd = new NpgsqlCommand("SELECT product_id, SUM(quantity) AS
total_quantity FROM sales GROUP BY product_id", conn))
            {
                var reader = cmd.ExecuteReader();
                var dt = new DataTable();
                dt.Load(reader);

                // Создание PDF документа
                PdfDocument document = new PdfDocument();
                PdfPage page = document.AddPage();
                XGraphics gfx = XGraphics.FromPdfPage(page);
                XFont font = new XFont("Verdana", 20);

                gfx.DrawString("Отчёт о количестве продаж", font, XBrushes.Black,
new XRect(0, 0, page.Width, page.Height), XStringFormats.Center);
            }
        }
    }
}

```

					МИВУ.09.03.04-13.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		43



```

        // Добавление данных в PDF
        XFont dataFont = new XFont("Verdana", 12);
        double yOffset = 50;

        foreach (DataRow row in dt.Rows)
        {
            string productId = row["product_id"].ToString();
            string totalQuantity = row["total_quantity"].ToString();
            gfx.DrawString($"ID товара: {productId}, Общее количество: {totalQuantity}", dataFont, XBrushes.Black, new XRect(50, yOffset, page.Width - 100, page.Height), XStringFormats.TopLeft);
            yOffset += 20;
        }

        string filename = "SalesQuantityReport.pdf";
        document.Save(filename);
        MessageBox.Show("Отчёт о количестве продаж сгенерирован успешно!");
    }
}

private void GenerateSalesRevenueReport()
{
    using (var conn = new NpgsqlConnection(connectionString))
    {
        conn.Open();
        using (var cmd = new NpgsqlCommand("SELECT product_id, SUM(sale_price * quantity) AS total_revenue FROM sales GROUP BY product_id", conn))
        {
            var reader = cmd.ExecuteReader();
            var dt = new DataTable();
            dt.Load(reader);

            // Создание PDF документа
            PdfDocument document = new PdfDocument();
            PdfPage page = document.AddPage();
            XGraphics gfx = XGraphics.FromPdfPage(page);
            XFont font = new XFont("Verdana", 20);

            gfx.DrawString("Отчёт о выручке от продаж", font, XBrushes.Black, new XRect(0, 0, page.Width, page.Height), XStringFormats.Center);

            // Добавление данных в PDF
            XFont dataFont = new XFont("Verdana", 12);
            double yOffset = 50;

            foreach (DataRow row in dt.Rows)
            {
                string productId = row["product_id"].ToString();
                string totalRevenue = row["total_revenue"].ToString();
                gfx.DrawString($"ID товара: {productId}, Общая выручка: {totalRevenue}", dataFont, XBrushes.Black, new XRect(50, yOffset, page.Width - 100, page.Height), XStringFormats.TopLeft);
                yOffset += 20;
            }

            string filename = "SalesRevenueReport.pdf";
            document.Save(filename);
            MessageBox.Show("Отчёт о выручке от продаж сгенерирован успешно!");
        }
    }
}

private void btnExit_Click(object sender, EventArgs e)
{

```

```

        this.Close();
    }
}

```

## PremisesForm.cs

```

public partial class PremisesForm : Form
{
    private string connectionString =
        "Host=localhost;Username=postgres;Password=admin1234;Database=electrost";

    public PremisesForm()
    {
        InitializeComponent();
        LoadPremises();
    }

    private void LoadPremises()
    {
        using (var conn = new NpgsqlConnection(connectionString))
        {
            conn.Open();
            using (var cmd = new NpgsqlCommand("SELECT premises_id, name, address
FROM premises", conn))
            {
                var reader = cmd.ExecuteReader();
                var dt = new DataTable();
                dt.Load(reader);
                dataGridViewPremises.DataSource = dt;
            }
        }
    }

    private void btnUpdatePremises_Click(object sender, EventArgs e)
    {
        using (var conn = new NpgsqlConnection(connectionString))
        {
            conn.Open();
            using (var cmd = new NpgsqlCommand("UPDATE premises SET name = @name,
address = @address WHERE premises_id = @premises_id", conn))
            {
                cmd.Parameters.AddWithValue("@premises_id",
int.Parse(txtPremisesId.Text));
                cmd.Parameters.AddWithValue("@name", txtName.Text);
                cmd.Parameters.AddWithValue("@address", txtAddress.Text);
                cmd.ExecuteNonQuery();
            }
        }
        LoadPremises();
        ClearFields();
    }

    private void ClearFields()
    {
        txtPremisesId.Clear();
        txtName.Clear();
        txtAddress.Clear();
    }

    private void btnAddPremises_Click(object sender, EventArgs e)
    {
        using (var conn = new NpgsqlConnection(connectionString))
        {
            conn.Open();

```

					МИВУ.09.03.04-13.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		45

```

        using (var cmd = new NpgsqlCommand("INSERT INTO premises (name, address)
VALUES (@name, @address)", conn))
        {
            cmd.Parameters.AddWithValue("@name", txtName.Text);
            cmd.Parameters.AddWithValue("@address", txtAddress.Text);
            cmd.ExecuteNonQuery();
        }
    }
    LoadPremises();
    ClearFields();
}

private void btnDeletePremises_Click(object sender, EventArgs e)
{
    using (var conn = new NpgsqlConnection(connectionString))
    {
        conn.Open();
        using (var cmd = new NpgsqlCommand("DELETE FROM premises WHERE
premises_id = @premises_id", conn))
        {
            cmd.Parameters.AddWithValue("@premises_id",
int.Parse(txtPremisesId.Text));
            cmd.ExecuteNonQuery();
        }
    }
    LoadPremises();
    ClearFields();
}

private void btnSearchPremises_Click(object sender, EventArgs e)
{
    using (var conn = new NpgsqlConnection(connectionString))
    {
        conn.Open();
        using (var cmd = new NpgsqlCommand("SELECT premises_id, name, address
FROM premises WHERE name LIKE @name", conn))
        {
            cmd.Parameters.AddWithValue("@name", "%" + txtSearch.Text + "%");
            var reader = cmd.ExecuteReader();
            var dt = new DataTable();
            dt.Load(reader);
            dataGridViewPremises.DataSource = dt;
        }
    }
}

private void dataGridViewPremises_CellClick(object sender,
DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0)
    {
        DataGridViewRow row = dataGridViewPremises.Rows[e.RowIndex];
        txtPremisesId.Text = row.Cells["premises_id"].Value.ToString();
        txtName.Text = row.Cells["name"].Value.ToString();
        txtAddress.Text = row.Cells["address"].Value.ToString();
    }
}

private void btnExit_Click(object sender, EventArgs e)
{
    this.Close();
}
}

```

## Приложение В. Снимки окон программы (скриншоты программы)

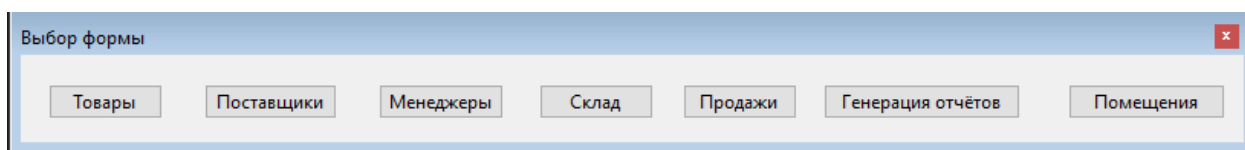


Рисунок В.1 – Главная форма программы

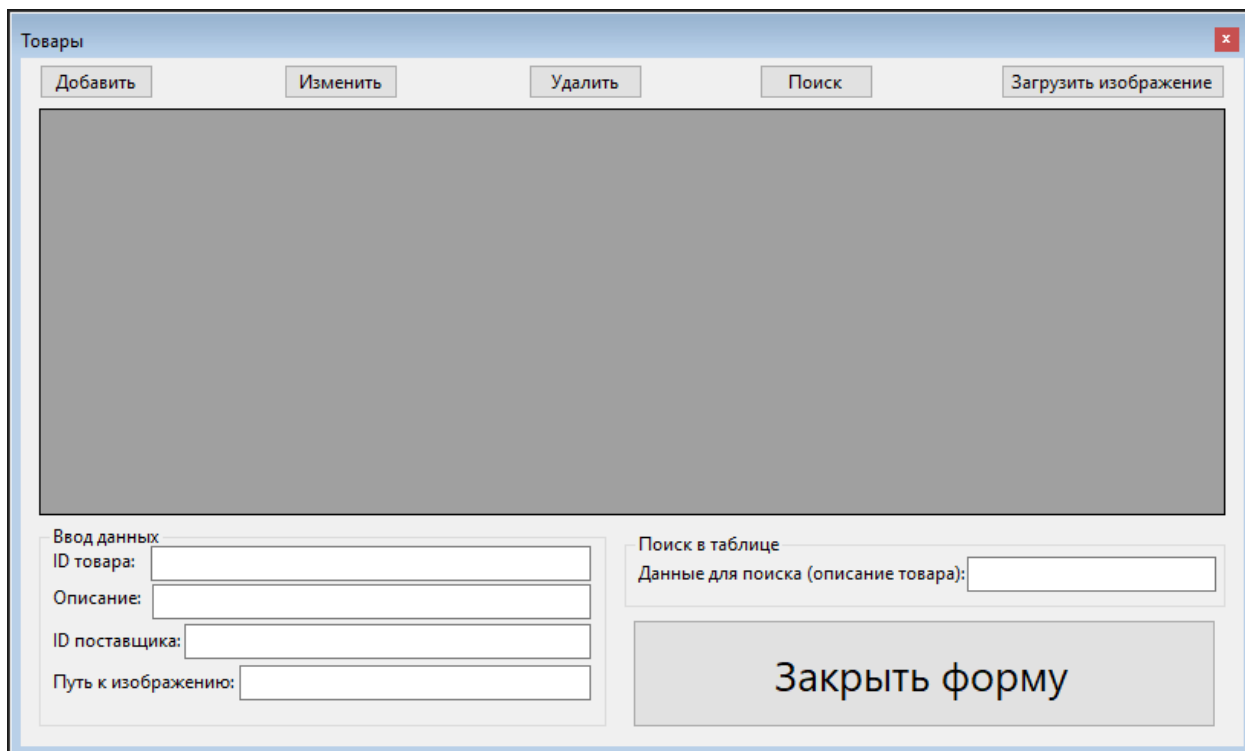


Рисунок В.2 – Форма «Товары»

Поставщики

Добавить      Изменить      Удалить      Поиск поставщика

Ввод данных

ID:

Название:

Адрес:

Поиск данных (по названию):  
Название:

Заккрыть форму

Рисунок В.3 – Форма «Поставщики»

Менеджеры

Добавить      Изменить      Удалить      Поиск

Ввод данных

ID:

Имя:

Должность:

Поиск данных (по имени):  
Имя:

Заккрыть форму

Рисунок В.4 – Форма «Менеджеры»

Склад

Добавить

Изменить

Удалить

Поиск

Ввод данных

ID склада:

ID помещения:

ID товара:

Кол-во товара:

Дата поступления:

Поиск склада (по ID товара)

ID товара:

Заккрыть форму

Рисунок В.5 – Форма «Склад»

Продажи

Добавить

Изменить

Удалить

Поиск

Ввод данных

ID продажи:

Цена продажи:

ID товара:

ID менеджера:

Кол-во товара:

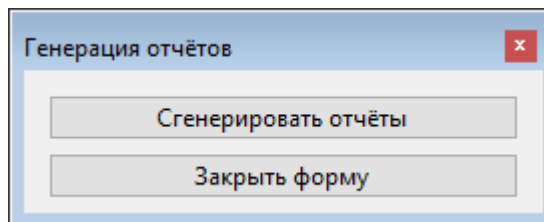
Дата продажи:

Поиск данных (по ID товара)

ID товара:

Заккрыть форму

Рисунок В.6 – Форма «Продажи»

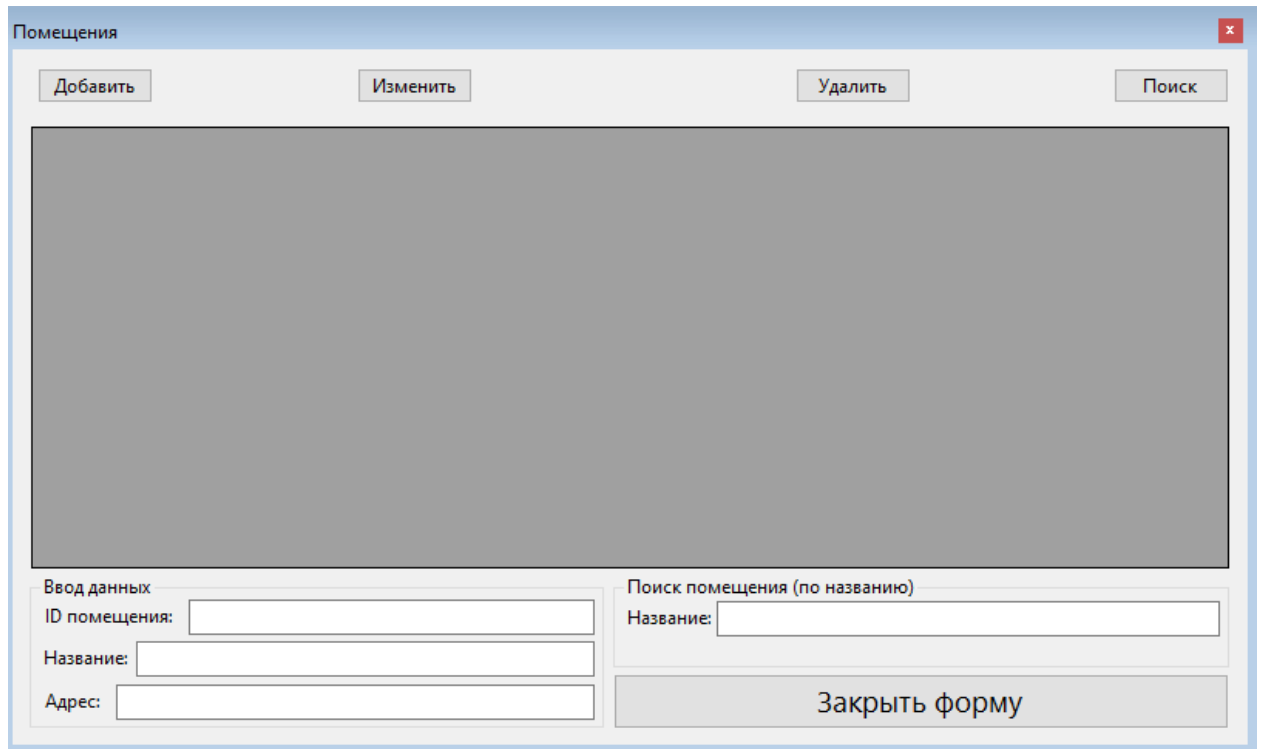


Генерация отчётов

Сгенерировать отчёты

Заккрыть форму

Рисунок В.7 – Форма «Генерация отчётов»



Помещения

Добавить Изменить Удалить Поиск

Ввод данных

ID помещения:

Название:

Адрес:

Поиск помещения (по названию)

Название:

Заккрыть форму

Рисунок В.8 – Форма «Помещения»