**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КОМИ**

**Государственное профессиональное образовательное учреждение**

**«Воркутинский арктический горно-политехнический колледж»**

**КУРСОВОЙ ПРОЕКТ**

По дисциплине МДК.05.02 Разработка кода информационных систем

**Разработка информационной системы «Продажа компьютерной периферии»**

Выполнил студент гр. ИСП-21 /\_\_\_\_\_\_\_\_\_\_\_\_/Олийнык Юрий Дмитриевич/

                                                                                    (подпись)                                          (Ф.И.О.)

**ОЦЕНКА:\_\_\_\_\_\_\_\_\_\_\_\_\_**

Дата:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ПРОВЕРИЛ**

Научный руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_/ Егоров Данил Павлович/

                                              (подпись)                                (Ф.И.О.)

Воркута

2024

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc166315090)

[ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ 4](#_Toc166315091)

[1.1 Платформа .NET 4](#_Toc166315092)

[1.3 Windows Presentation Foundation (WPF) 6](#_Toc166315093)

[1.4 СУБД SQL server 7](#_Toc166315094)

[1.5 Microsoft SQL Server Management Studio 7](#_Toc166315095)

[1.6 Entity Framework 8](#_Toc166315096)

[ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ 10](#_Toc166315097)

[2.1 Разработка диаграммы ERD 10](#_Toc166315098)

[2.2 Разработка базы данных 12](#_Toc166315099)

[ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ 18](#_Toc166315100)

[3.1 Разработка прототипа информационной системы 18](#_Toc166315101)

[3.2 Программирование информационной системы 21](#_Toc166315102)

[3.2.1 Разработка модуля «Авторизация» 21](#_Toc166315103)

[3.2.2 Разработка модуля «Добавить клиента» 22](#_Toc166315104)

[3.2.3 Разработка модуля «Удаление клиента» 23](#_Toc166315105)

[3.2.4 Разработка модуля «Редактирование» 24](#_Toc166315106)

[3.2.4 Разработка модуля «Отчеты» 25](#_Toc166315107)

[ЗАКЛЮЧЕНИЕ 28](#_Toc166315108)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 29](#_Toc166315109)

# 

# ВВЕДЕНИЕ

Актуальность выбранной темы заключается в растущем спросе на компьютерную технику и аксессуары. Компьютерная периферия – вспомогательные устройства, используемые для ввода информации в компьютер и получения информации из него – такие как мыши, клавиатуры, мониторы, принтеры и другие устройства, является неотъемлемой частью работы во многих сферах деятельность, так и в повседневной жизни людей.

С развитием технологий и увеличением числа пользователей компьютеров, спрос на качественную компьютерную периферию постоянно растет. Информационная система будет полезна для предпринимателей и компаний, занимающихся продажей компьютерной периферии, предоставляя им инструменты для управления запасами, заказами, продажами и другими аспектами бизнеса.

Также информационная система полезна для покупателей, предоставляя информацию о различных моделях компьютерной периферии, их характеристиках и ценах. Это поможет покупателям сделать осознанный выбор при покупке и получить максимально качественный товар.

**Объект:** информационная система «Продажа компьютерной периферии».

**Предмет:** анализ бизнес-процессов «Продажа компьютерной периферии».

**Цель работы:** разработать информационную систему «Продажа компьютерной периферией».

**Задачи:**

* выбор инструментария;
* проектирование базы данных;
* разработка информационной системы.

# ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ

# Платформа .NET

.NET — это бесплатная кроссплатформенная платформа разработчика с открытым кодом для создания множества приложений. Он может запускать программы, написанные на нескольких языках, при этом C# является наиболее популярным. Она использует высокопроизводительную среду выполнения, которая используется в рабочей среде многими высокомасштабируемыми приложениями.

Платформа .NET разработана для обеспечения производительности, производительности, безопасности и надежности. Она обеспечивает автоматическое управление памятью с помощью сборщика мусора (GC). Это типобезопасный и безопасный для памяти, из-за использования GC и строгих компиляторов языка. Он включает в себя большой набор библиотек, имеющих широкие функциональные возможности и оптимизированный для производительности в нескольких операционных системах и архитектурах микросхем.

.NET включает следующие компоненты:

* Среда выполнения — выполняет код приложения;
* Библиотеки — предоставляют служебные функции, такие как синтаксический анализ JSON;
* Компилятор — компилирует исходный код C# (и других языков) в исполняемый код (среда выполнения);
* Пакет SDK и другие средства — включение создания и мониторинга приложений с помощью современных рабочих процессов;
* Стеки приложений, такие как ASP.NET Core и Windows Forms, которые позволяют создавать приложения.

Среда выполнения, библиотеки и языки являются основными компонентами стека .NET. Компоненты более высокого уровня, такие как средства .NET и стеки приложений, такие как ASP.NET Core, создаются на основе этих основных компонентов. C# — это основной язык программирования для .NET, и большая часть .NET написана на C#.

* 1. **Язык программирования С#**

С# – это объектно-ориентированный язык программирования. Он был создан в период с 1998 по 2002 год командой инженеров Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота.

Язык входит в семью С-подобных языков. Синтаксис приближен к Java и C++. Его особенности:

* статистическая типизация;
* поддерживается полиморфизм;
* поддерживается перегрузка операторов;
* доступна делегация, атрибуты, события, обобщенные типы и анонимные функции.

Разработка Microsoft много особенностей унаследовала у Delphi, Smalltalk и Java. При этом создатели нового языка исключили из своего детища многие практики и спецификации, считающиеся «проблемными».

С# популярен за счет своей «простоты». Простоты для современных программистов и больших команд разработчиков, чтобы те могли в сжатые сроки создавать функциональные и производительные приложения. Этому способствуют нетипичные конструкции языка и специфичный синтаксис, помогающий максимально органично реализовать намеченные функции.

Популярность языка – еще одно значимое преимущество. Большое количество поклонников C# способствуют его развитию. Также это благоприятно влияет на рост числа вакансий, связанных с разработкой на языке Microsoft. Программисты, хорошо знакомые с С#, востребованы в индустрии, несмотря на их большое и постоянно увеличивающееся количество.

Понятный синтаксис C# заметно упрощает не только разработку как таковую, но и другие важные аспекты совместной работы, например, чтение чужого кода. Это упрощает процесс рефакторинга и исправления ошибок при работе над приложениями в больших командах.

Также нельзя не упомянуть низкий порог вхождения. С# – популярная и достаточно простая в освоении технология. Уже через полгода можно поднатореть в разработке и начать делать полноценные программы.

# Windows Presentation Foundation (WPF)

Технология WPF (Windows Presentation Foundation) является часть экосистемы платформы .NET и представляет собой подсистему для построения графических интерфейсов.

Если при создании традиционных приложений на основе WinForms за отрисовку элементов управления и графики отвечали такие части ОС Windows, как User32 и GDI+, то приложения WPF основаны на DirectX. В этом состоит ключевая особенность рендеринга графики в WPF: используя WPF, значительная часть работы по отрисовке графики, как простейших кнопочек, так и сложных 3D-моделей, ложиться на графический процессор на видеокарте, что также позволяет воспользоваться аппаратным ускорением графики.

Одной из важных особенностей является использование языка декларативной разметки интерфейса XAML, основанного на XML: вы можете создавать насыщенный графический интерфейс, используя или декларативное объявление интерфейса, или код на управляемых языках C#, VB.NET и F#, либо совмещать и то, и другое.

WPF существует в виде подмножества типов .NET, которые по большей части находятся в пространстве имен System.Windows. Если ранее вы создавали приложения с помощью .NET, используя управляемые технологии, такие как ASP.NET и Windows Forms, основные принципы программирования с помощью WPF должны быть вам знакомы: вы создаете экземпляры классов, задаете свойства, вызываете методы и обрабатываете события — все это с использованием своего любимого языка программирования .NET, например C# или Visual Basic.

WPF включает в себя дополнительные конструкции программирования, которые расширяют возможности свойств и событий: свойства зависимостей и перенаправленные события.

XAML — это язык разметки на основе XML, который служит для определения внешнего вида приложения в декларативной форме. Обычно он используется для создания окон, страниц и пользовательских элементов управления, а также их заполнения элементами управления, фигурами и графическими элементами.

# СУБД SQL server

SQL Server - это система управления базами данных (СУБД) от Microsoft. Она позволяет создавать, модифицировать и управлять базами данных, а также предоставлять и контролировать доступ к ним. SQL Server используется для хранения и обработки данных в приложениях и веб-сайтах, он поддерживает различные языки программирования и интерфейсы. Центральным аспектом в MS SQL Server, как и в любой СУБД, является база данных. База данных представляет хранилище данных, организованных определенным способом. Нередко физически база данных представляет файл на жестком диске, хотя такое соответствие необязательно. Для хранения и администрирования баз данных применяются системы управления базами данных (database management system) или СУБД (DBMS).

# Microsoft SQL Server Management Studio

SQL Server Management Studio (SSMS) - это интегрированная среда для управления любой инфраструктурой SQL, от SQL Server до базы данных SQL Azure. SSMS предоставляет инструменты для настройки, мониторинга и администрирования экземпляров SQL Server и баз данных. Используйте SSMS для развертывания, мониторинга и обновления компонентов уровня данных, используемых вашими приложениями, а также для создания запросов и сценариев.

Главным инструментом SQL Server Management Studio является Object Explorer, который позволяет пользователю просматривать, извлекать объекты сервера, а также полностью ими управлять.

Также есть SQL Server Management Studio Express для Express версии сервера, которая является бесплатной. Однако в ней нет поддержки ряда компонентов (Analysis Services, Integration Services, Notification Services, Reporting Services) и SQL Server 2005 Mobile Edition.

# Entity Framework

Entity Framework представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работает с объектами.

Первая версия Entity Framework - 1.0 вышла еще в 2008 году и представляла очень ограниченную функциональность, базовую поддержку ORM (object-relational mapping - отображения данных на реальные объекты) и один единственный подход к взаимодействию с бд - Database First. С выходом версии 4.0 в 2010 году многое изменилось - с этого времени Entity Framework стал рекомендуемой технологией для доступа к данным, а в сам фреймворк были введены новые возможности взаимодействия с бд - подходы Model First и Code First.

Дополнительные улучшения функционала последовали с выходом версии 5.0 в 2012 году. И наконец, в 2013 году был выпущен Entity Framework 6.0, обладающий возможностью асинхронного доступа к данным.

Центральной концепцией Entity Framework является понятие сущности или entity. Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами.

Любая сущность, как и любой объект из реального мира, обладает рядом свойств. Например, если сущность описывает человека, то мы можем выделить такие свойства, как имя, фамилия, рост, возраст, вес. Свойства необязательно представляют простые данные типа int, но и могут представлять более комплексные структуры данных. И у каждой сущности может быть одно или несколько свойств, которые будут отличать эту сущность от других и будут уникально определять эту сущность. Подобные свойства называют ключами.

# ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ

# 2.1 Разработка диаграммы ERD

**Описание:**

Схема «сущность-связь» (также ERD или ER-диаграмма) — это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы, ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения и информационных систем для бизнеса. ERдиаграммы (или ER-модели) полагаются на стандартный набор символов, включая прямоугольники, ромбы, овалы и соединительные лини для отображения сущностей, их атрибутов и связей. Эти диаграммы устроены по тому же принципу, что и грамматические структуры: сущности выполняют роль существительных, а связи - глаголов.

B ER-моделях и моделях данных обычно выделяют до трех уровней детализации:

* **Концептуальная модель данных** - схема наивысшего уровня с минимальным количеством подробностей. Достоинство этого подхода заключается в возможности отобразить общую структуру модели и всю архитектуру системы. Менее масштабные системы могут обойтись и без этой модели. В этом случае можно сразу переходить к логической модели;
* **Логическая модель** **данных** содержит более подробную информацию, нежели концептуальная модель. На этом уровне определяются более подробные операционные и транзакционные сущности. Логическая модель не зависит от технологии, в которой она будет применяться;
* **Физическая модель данных**: на основе каждой логической модели данных можно составить одну или две физических модели. В последних должно присутствовать достаточно технических подробностей для составления и внедрения самой базы данных.

**Элементы:**

Для того чтобы построить ER-диаграмму, можно использовать разные нотации:

* Нотация IDEF1X. Ее относят к фундаментальным, но на практике давно не используют, потому что есть более удобные варианты;
* Нотация Чена. Классическая нотация, которая состоит из простых символов – прямоугольников, овалов и линий. Из-за этого нотацию часто используют для концептуальных моделей, которые презентуют заказчику;
* Нотация Мартина. Ее еще называют «воронья лапка» (от англ. Crow's Foot). Она компактнее нотации Чена, поэтому ее используют для построения ER-моделей логического уровня, когда нужно описать в модели все атрибуты сущностей.

В нотациях Чена и Мартина есть одинаковые элементы: сущности, атрибуты и связи. Но эти элементы диаграмм обозначают разными символами.

Элементы ER-диаграммы в нотации Чена соединяют линиями. Если линия соединяет две сущности, сверху обозначают тип связи:

● 1:1 — «один-к-одному»;

● 1:N — «один-ко-многим»;

● M:N — «многие-ко-многим».

В нотации Мартина сущность также вписывают в прямоугольник, а атрибуты и связи обозначают по-другому:

● атрибуты перечисляют прямо под сущностью;

● связи рисуют разными соединительными линиями (рис. 2.1).

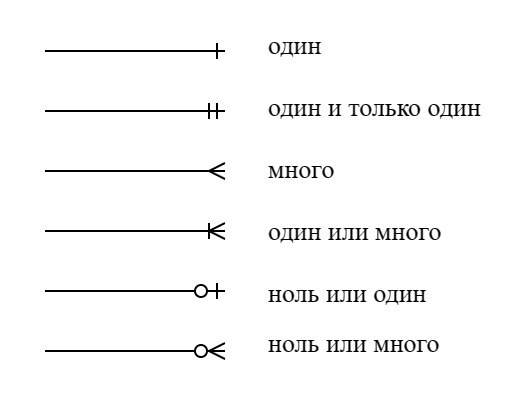


Рисунок 2.1 Виды соединительных линий нотации Мартина

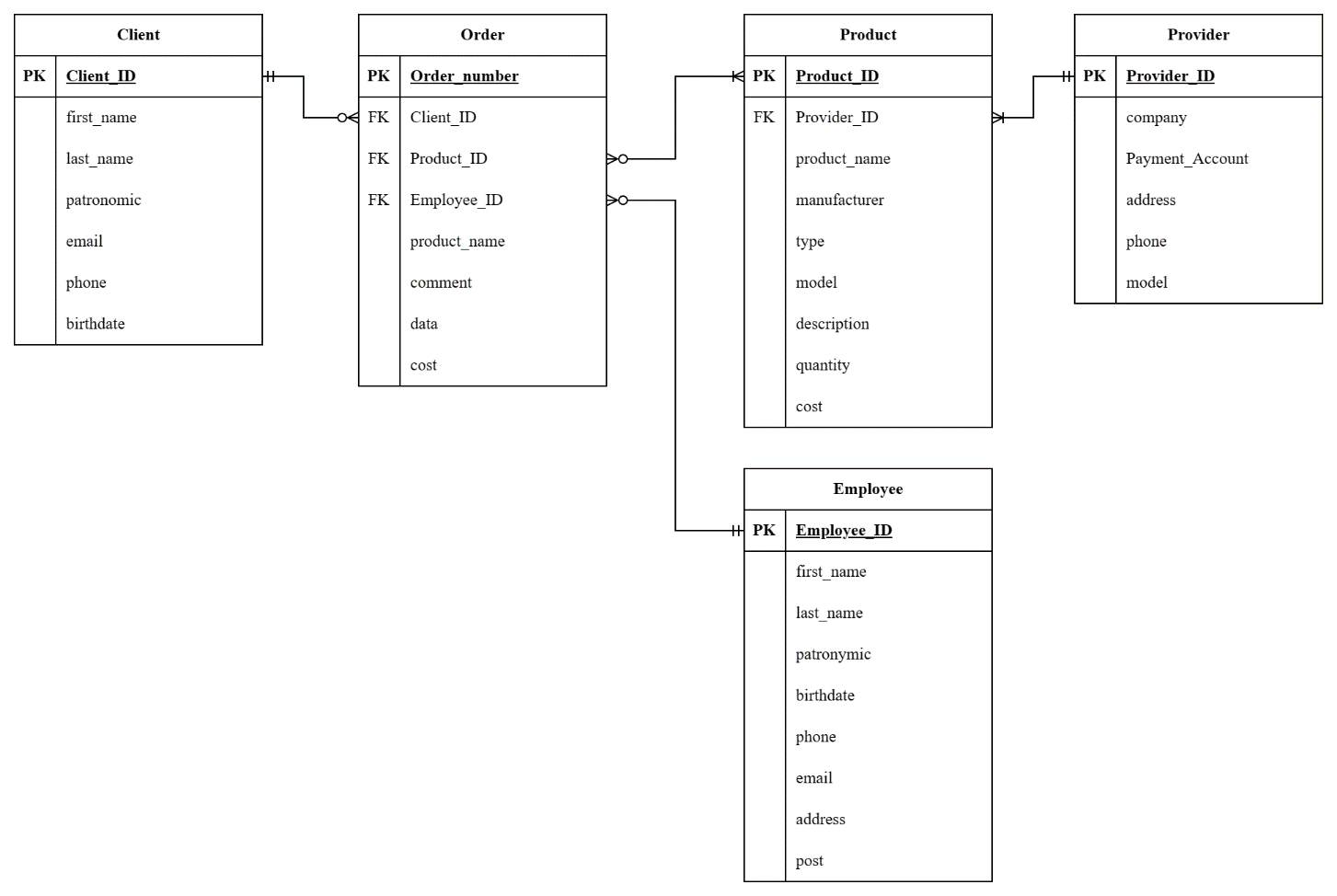


Рисунок 2.2 Диаграмма «Сущность-связь» (Entity-Relationship diagram)

На рис. 2.2 показан ER-диаграмма информационной системы «Продажа компьютерной периферией». Она показывает сущности и как они связаны между собой внутри. Сущностями являются: Сотрудники (Employee), Товар (Product), Заказ (Order), Поставщик (Provider), Клиент (Client).

# 2.2 Разработка базы данных

SQL Server Management Studio (SSMS) - это интегрированная среда для управления любой инфраструктурой SQL, от SQL Server до базы данных SQL Azure. SSMS предоставляет инструменты для настройки, мониторинга и администрирования экземпляров SQL Server и баз данных. Используйте SSMS для развертывания, мониторинга и обновления компонентов уровня данных, используемых вашими приложениями, а также для создания запросов и сценариев.

Разработка базы данных начинается с ранее разработанной ERD диаграммой. Откроем Microsoft SQL Server Management Studio и подключаемся к нашему локальному серверу.

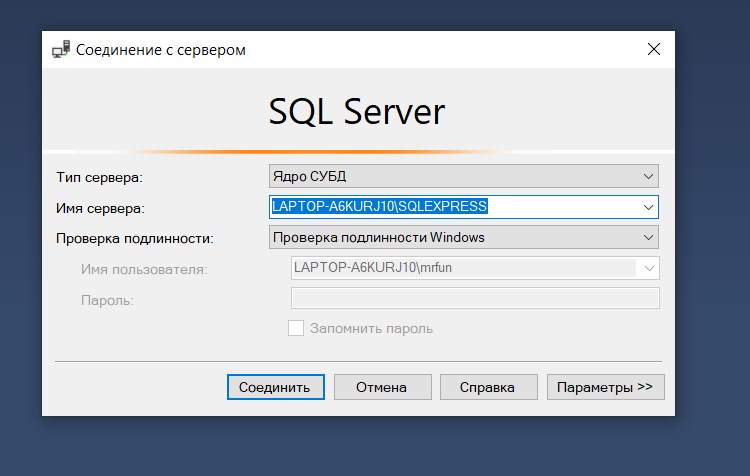


Рисунок 2.2.1 Соединение с сервером

После чего у нас станет доступна таблица Object Explorer в которой будут показаны файлы и базы данных на локальном сервере. Создадим базу данных, нажимаем ПКМ по папке «Базы данных» и выбираем «Создать базу данных».

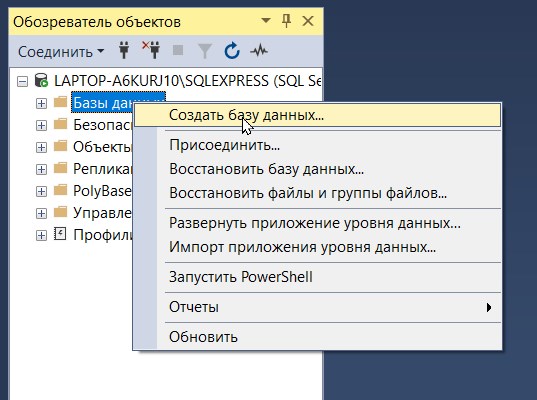


Рисунок 2.2.2 Создание базы данных

После чего нам выведется поле для создания диаграммы базы данных, на котором мы можем создавать таблицы с название столбцов и их типом данных.

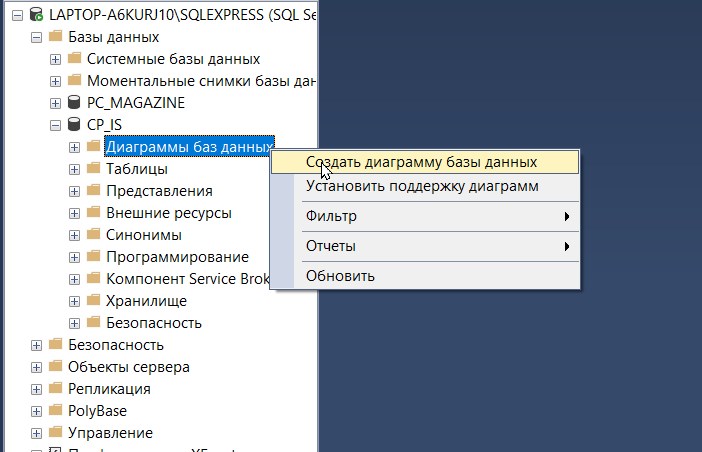


Рисунок 2.2.3 Создание диаграммы

Для создания таблицы данных необходимо нажать ПКМ в любом месте диаграммы, выбираем «Создать таблицу» и даём название для нашей таблицы

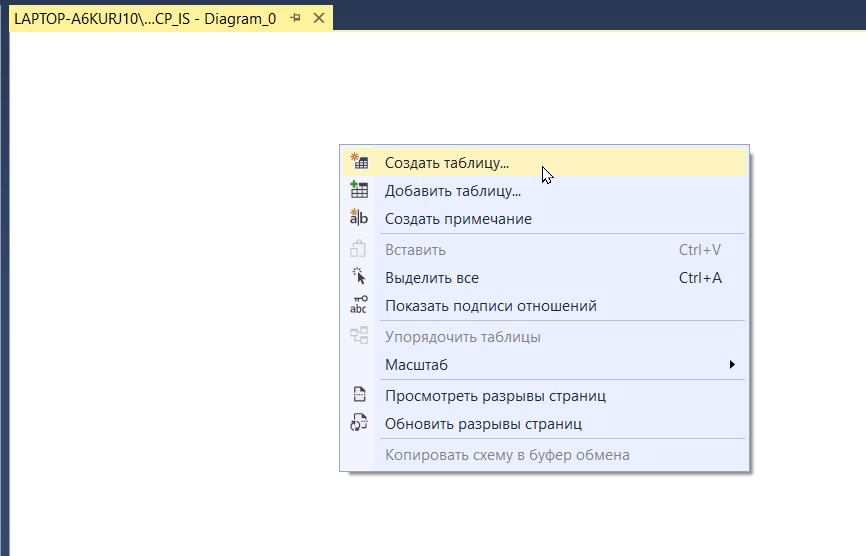


Рисунок 2.2.4 Создание таблицы

После создания таблицы, записываем атрибуты и типы данных, которые будут храниться в таблице.

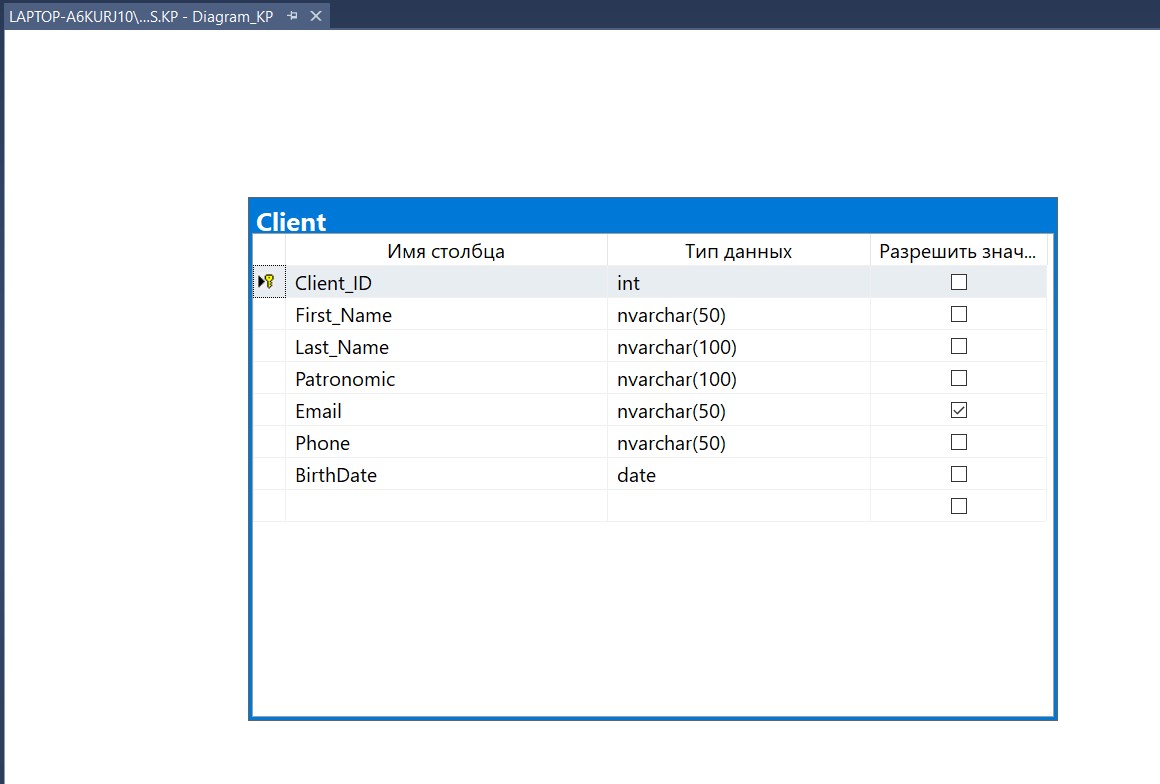


Рисунок 2.2.5 Задаем имена столбцов и указываем типы данных в таблице

Для информационной системы необходимо создать пять таблиц: Товары (Product), Сотрудники (Employee), Клиенты (Client), Заказы (Order), Поставщик (Provider). После чего заполним таблицы необходимыми полями и типами данных из ERD таблицы.

Далее создаём связи как в ERD сущностях и после сохраняем таблицы нажав сочетание клавиш «Ctrl + S» и в высветившимся окне пишем название нашей диаграммы.

Созданные нами таблицы можно увидеть в Object Explorer кликнув на название базы данных и перейдя в папку «Таблицы». Зажав уникальный ключ, тянем к полу в нужной таблице и создаётся связь «один ко многим». Для создания связи «Многие ко многим», необходимо создать дополнительную добавляем в нее названия нужных таблиц, делаем их ключевыми, и соединяем ключи с таблицами.

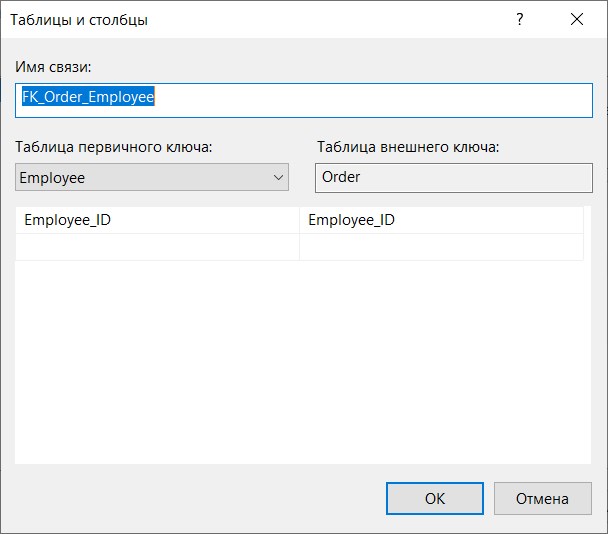


Рисунок 2.2.6 Создаем связи между атрибутами

После создания связей наша таблица полностью готова.

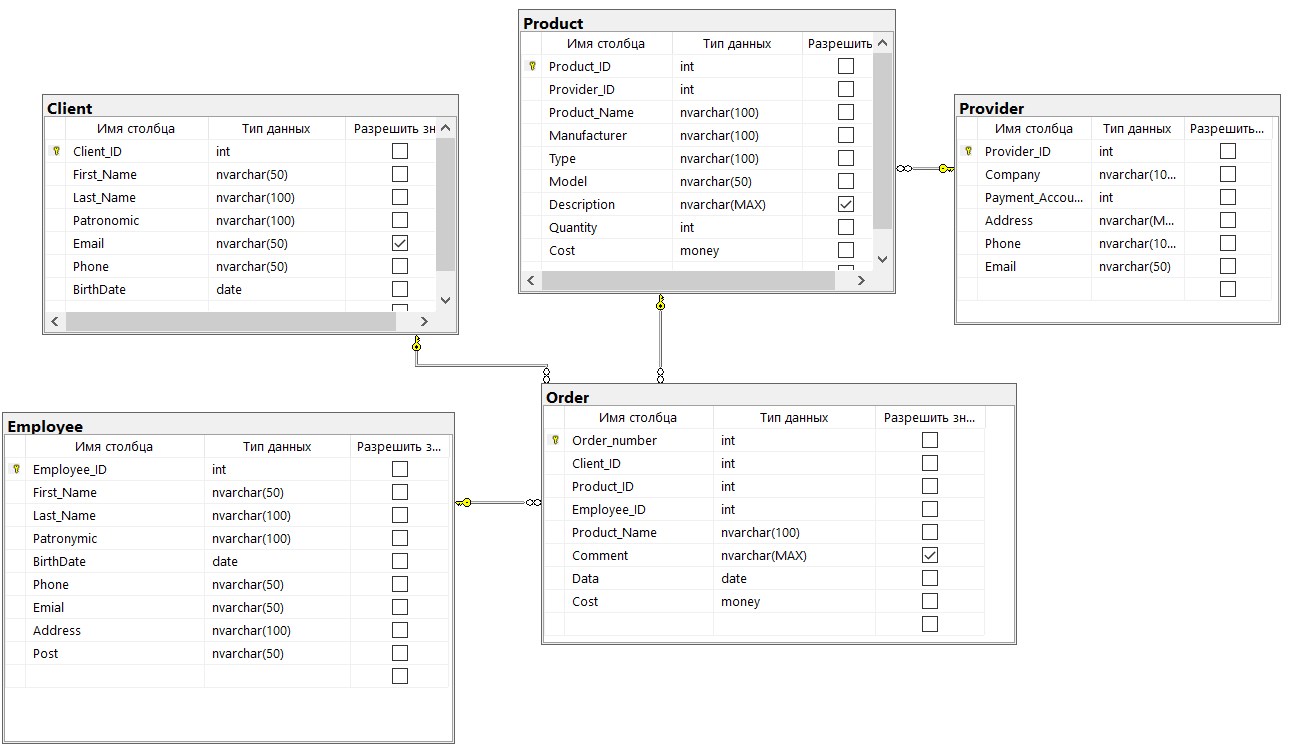


Рисунок 2.2.7 Готовые таблицы диаграммы БД

В данной работе используется 5 таблиц базы данных. Таблица Employee содержит Уникальный ключ, Имя, Фамилия, Отчество, Дата рождения, Номер телефона, Почта, Адрес и Должность, которую сотрудник занимает. Order содержит Уникальный ключ, Название продукта, Комментарий заказа, Дата оформления заказа, Сумма заказа. Client содержит Уникальный ключ, Фамилию, Имя, Отчество, Номер телефона, Дату рождения и Почту. Product содержит Уникальный ключ, Наименование, Цену, Количество, Тип, Модель, Производителя, Описание. Provider содержит в себе следующие атрибуты: Уникальный ключ, Название компании, Платежные данные, Адрес, Телефон и Почту.

# ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

# 3.1 Разработка прототипа информационной системы

Проволочная диаграмма представляет собой наглядный детализированный макет пользовательского интерфейса – своеобразный рабочий чертеж, описывающий функциональные возможности и содержимое приложения. Такие проволочные шаблоны для веб-сайтов и мобильных приложений идеально подходят для создания нестрогих проектных эскизов, демонстрирующих идеи и позволяющих достичь единого мнения в команде и сформировать основу для разработки высококачественной каркасной модели.

Для каждого шаблона можно выбрать пустую схему для быстрого начала или пустой схемы. Каждый шаблон имеет пять элементов (контейнеры, элементы управления, мультимедиа, текст и навигация) с большим количеством фигур. Контейнеры предоставляют предварительно созданные группы фигур, представляющих различные диалоговые окно и карточки, что помогает быстро создать проволочное представление диаграммы. Проволочная фигура также включает интеллектуальные фигуры, например ползунок диапазона или ползунок хода выполнения, которые можно настроить с помощью пунктов управления и действий в меню, щелкнув правой кнопкой мыши, чтобы изменить состояние, значения или визуальные стили. Эти шаблоны являются универсальными и могут использоваться для разных браузеров и мобильных устройств.

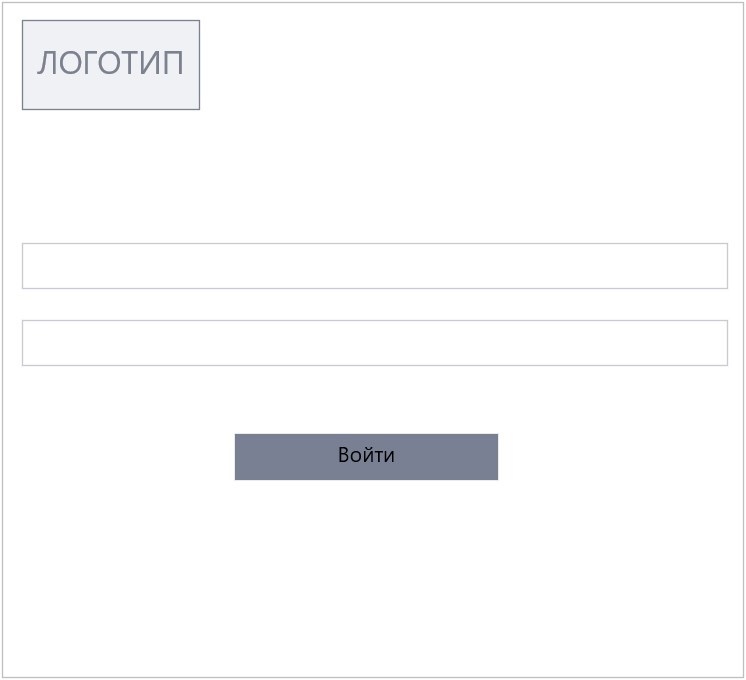


Рисунок 3.1.1 Разработка прототипа формы окна «Авторизация»

На рисунке 3.1.1 показано окно «Авторизация», включающее в себя поле для логина и пароля, кнопку «Войти». После правильного ввода логина и пароля, мы попадаем на главное окно.



Рисунок 3.1.2 Разработка прототипа формы окна «Главное окно»

На рисунке 3.1.2 показана главная страница информационной системы, которая является связующим звеном всех последующих страниц. Она включает в себя кнопки переходов на разные страницы.

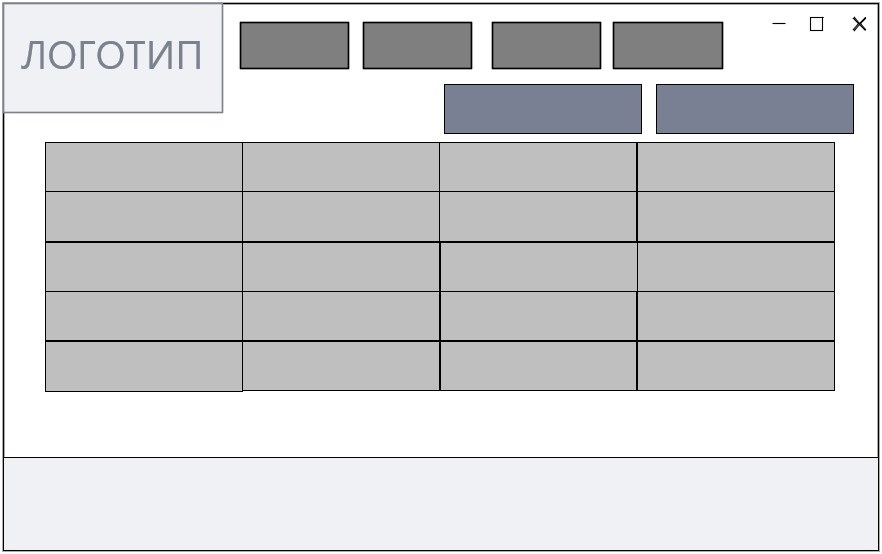


Рисунок 3.1.3 Разработка прототипа формы окон «Клиенты, продажа, товары»

На рисунке 3.1.3 показано окно форм Клиенты, Продажи, Товары, включающее в себя ряд полей, содержащих данные из базы данных; кнопки переходов на разные страницы, после добавляются кнопки добавления, редактирования и удаления.

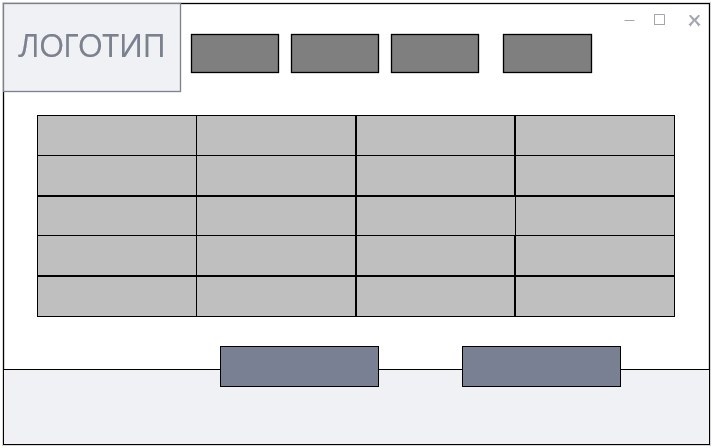


Рисунок 3.1.4 Разработка прототипа формы окна «Отчеты»

На рисунке показано окно «Отчеты», включающее в себя ряд полей, содержащих данные о товарах (номер продукта, название, цену, категорию, описание и количество), либо о продажа. Кнопки переходов на другие страницы и кнопки сохранения в Word и Excel.

# 3.2 Программирование информационной системы

Для программирования информационной системы «Продажа компьютерной периферии» использовался язык программирования С# и программа VisualStudio. Интерфейс и функционал разрабатывался технологией WPF и Entity Framework.

# 3.2.1 Разработка модуля «Авторизация»

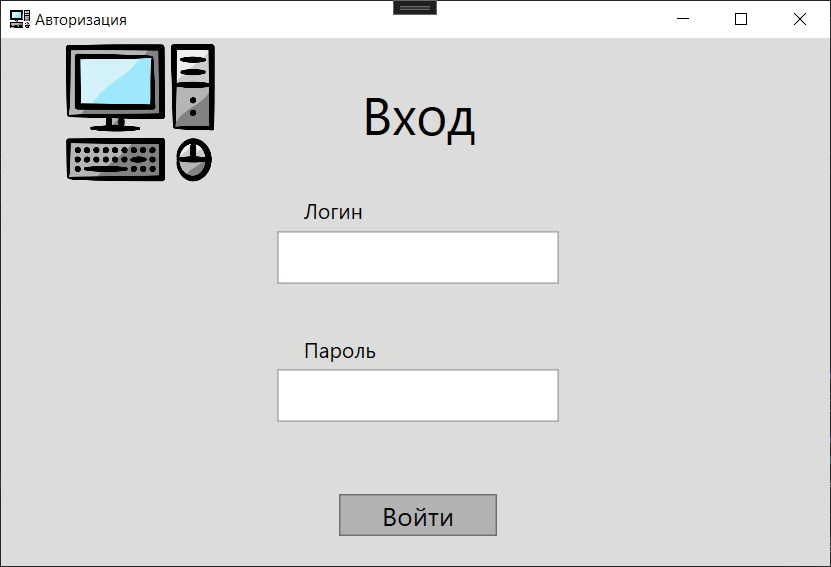


Рисунок 3.2.1. Модуль авторизации

Код «Authorixation.xaml.cs»

namespace KP

{

public partial class Authorization : Window

{

public Authorization()

{

InitializeComponent();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

Seller SellerWindow = new Seller();

Manager ManagerWindow = new Manager();

Administrator AdministratorWindow = new Administrator();

string login = Login.Text.ToLower();

string password = Password.Password;

if (login == "Seller" && password == "1")

{

this.Close();

SellerWindow.Show();

}

else if (login == "Manager" && password == "2")

{

this.Close();

ManagerWindow.Show();

}

else if (login == "Admin" && password == "3")

{

this.Close();

AdministratorWindow.Show();

}

else

{

MessageBox.Show("Неверный логин или пароль");

}

}

}

При разработке данного модуля, была создана страница «Авторизация». На ней есть поля для ввода логина и пароля, которые переходят в дальнейшем на нужные страницы в зависимости от роли пользователя(Администратор, Продавец-консультант, Работник склада).

# 3.2.2 Разработка модуля «Добавить клиента»

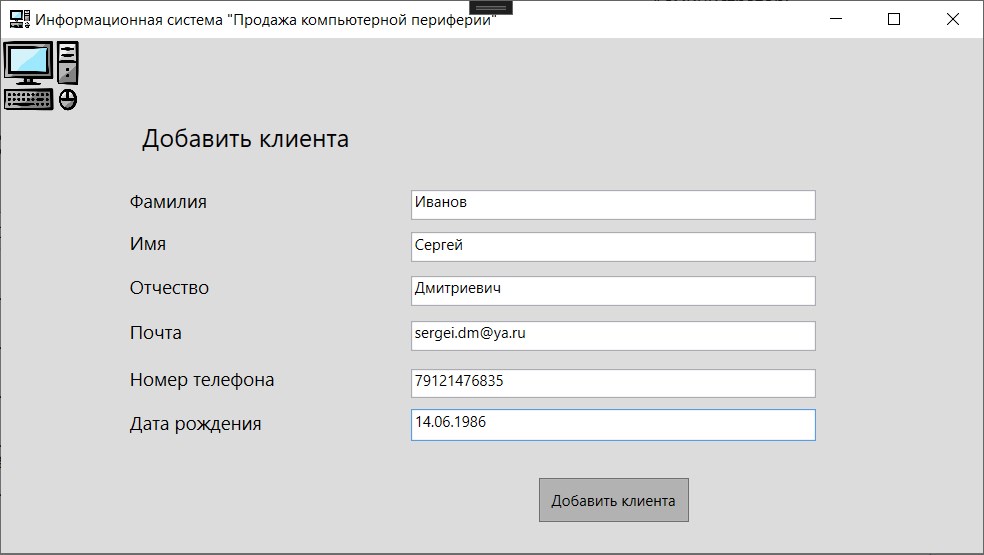


Рисунок 3.2.2 Модуль добавить клиента

Код «Admin\_Add\_Client.xaml.cs»

namespace KP.Admin\_Folder.Add\_Folder

{

public partial class Admin\_Add\_Client : Window

{

private Client \_currentClient = new Client();

public Admin\_Add\_Client()

{

InitializeComponent();

DataContext = \_currentClient;

}

private void Button\_Click(object sender, object e)

{

StringBuilder errors = new StringBuilder();

if (string.IsNullOrWhiteSpace(\_currentClient.First\_Name))

errors.AppendLine("Укажите имя клиента");

if (string.IsNullOrWhiteSpace(\_currentClient.Last\_Name))

errors.AppendLine("Укажите фамилию клиента");

if (string.IsNullOrWhiteSpace(\_currentClient.Patronomic))

errors.AppendLine("Укажите отчество клиента");

if (string.IsNullOrWhiteSpace(\_currentClient.Email))

errors.AppendLine("Укажите почту клиента");

if (string.IsNullOrWhiteSpace(\_currentClient.Phone))

errors.AppendLine("Укажите номер телефона клиента");

if (errors.Length > 0)

{

MessageBox.Show(errors.ToString());

return;

}

if (\_currentClient.Client\_ID == 0)

KPEntities.GetContext().Client.Add(\_currentClient);

try

{

KPEntities.GetContext().SaveChanges();

MessageBox.Show("Клиент добавлен");

this.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

}

В данном модуле была разработана функция добавления клиентов, товаров и заказов в базу данных. При нажатии на кнопку «Добавить клиента» откроется окно с полями, которые нужно будет заполнить. Доступ к данной функции имеют Администратор, Продавец-консультант и Работник склада.

# 3.2.3 Разработка модуля «Удаление клиента»

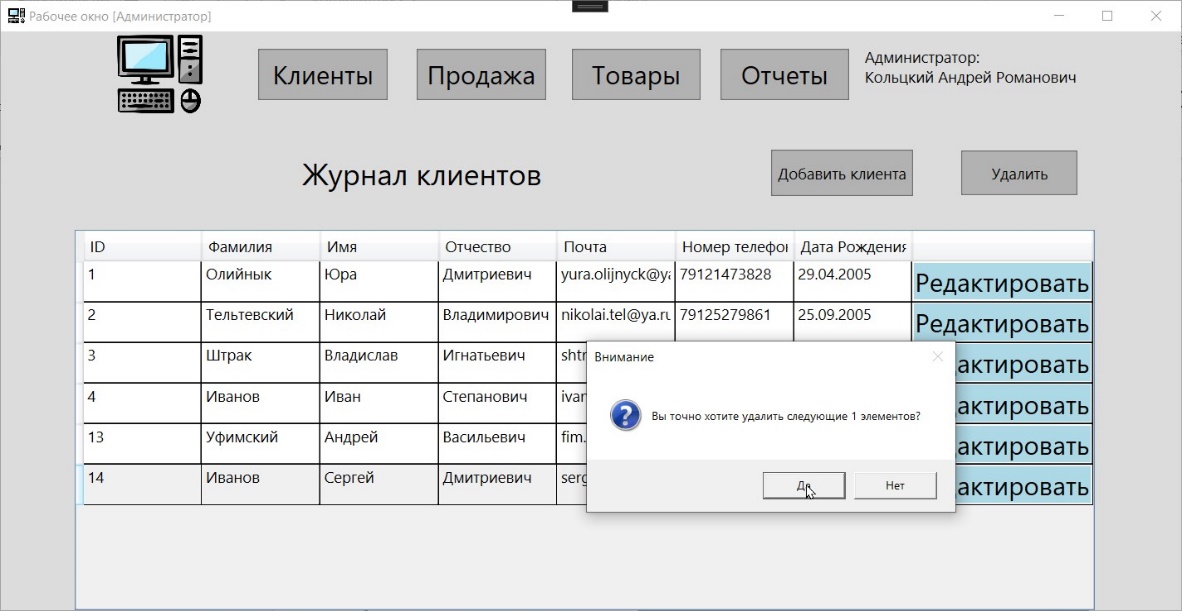


Рисунок 3.2.3 Модуль «Удаление клиента»

Код «Admin\_Clients\_Page.xaml.cs»

private void Button\_Remove(object sender, RoutedEventArgs e)

{

var clientRemoving = DGrid\_Clients.SelectedItems.Cast<Client>().ToList();

if (MessageBox.Show($"Вы точно хотите удалить следующие {clientRemoving.Count()} элементов?", "Внимание",

MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

try

{

KPEntities.GetContext().Client.RemoveRange((IEnumerable<Client>)clientRemoving);

KPEntities.GetContext().SaveChanges();

MessageBox.Show("Данные удалены");

DGrid\_Clients.ItemsSource = KPEntities.GetContext().Client.ToList();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

}

В этом модуле была разработана функция удаления клиентов из базы данных. Кроме того, данная функция также работает с товарами и заказами. Для этого нужно в таблице выделить строки, которые необходимо удалить. Доступ к данной функции имеет только Администратор.

# 3.2.4 Разработка модуля «Редактирование»

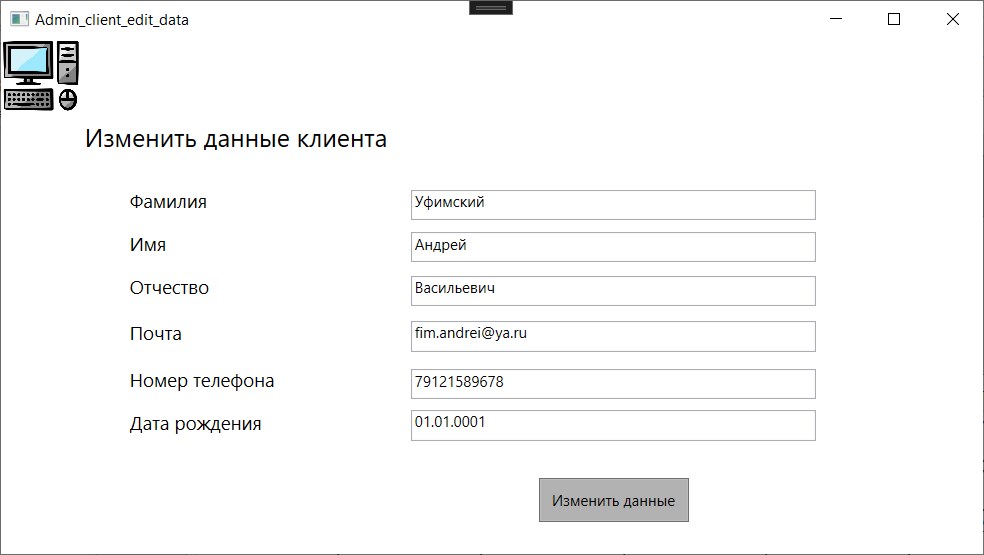


Рисунок 3.2.4 Модуль «Редактирование»

Код «Admin\_client\_edit\_data.xaml.cs»:

namespace KP.Admin\_Folder.Edit\_data\_Folder

{

public partial class Admin\_client\_edit\_data : Window

{

private Client \_currentClient = new Client();

public Admin\_client\_edit\_data(Client selectedClient)

{

InitializeComponent();

DataContext = \_currentClient;

if (selectedClient != null)

\_currentClient = selectedClient;

DataContext = \_currentClient;

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

StringBuilder errors = new StringBuilder();

if (string.IsNullOrWhiteSpace(\_currentClient.First\_Name))

errors.AppendLine("Укажите имя клиента");

if (string.IsNullOrWhiteSpace(\_currentClient.Last\_Name))

errors.AppendLine("Укажите фамилию клиента");

if (string.IsNullOrWhiteSpace(\_currentClient.Patronomic))

errors.AppendLine("Укажите отчество клиента");

if (string.IsNullOrWhiteSpace(\_currentClient.Email))

errors.AppendLine("Укажите почту клиента");

if (string.IsNullOrWhiteSpace(\_currentClient.Phone))

errors.AppendLine("Укажите номер телефона клиента");

if (errors.Length > 0)

{

MessageBox.Show(errors.ToString());

return;

}

if (\_currentClient.Client\_ID == 0)

KPEntities.GetContext().Client.Add(\_currentClient);

try

{

KPEntities.GetContext().SaveChanges();

MessageBox.Show("Данные изменены");

this.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

}

В этом модуле была разработана функция редактирования данных. При нажатии на кнопку редактировать откроется окно, где пользователь сможет изменить какую-либо информацию о клиенте, товаре или заказе. Доступ к данной функции имеет Администратор и Работник склада.

# 3.2.4 Разработка модуля «Отчеты»

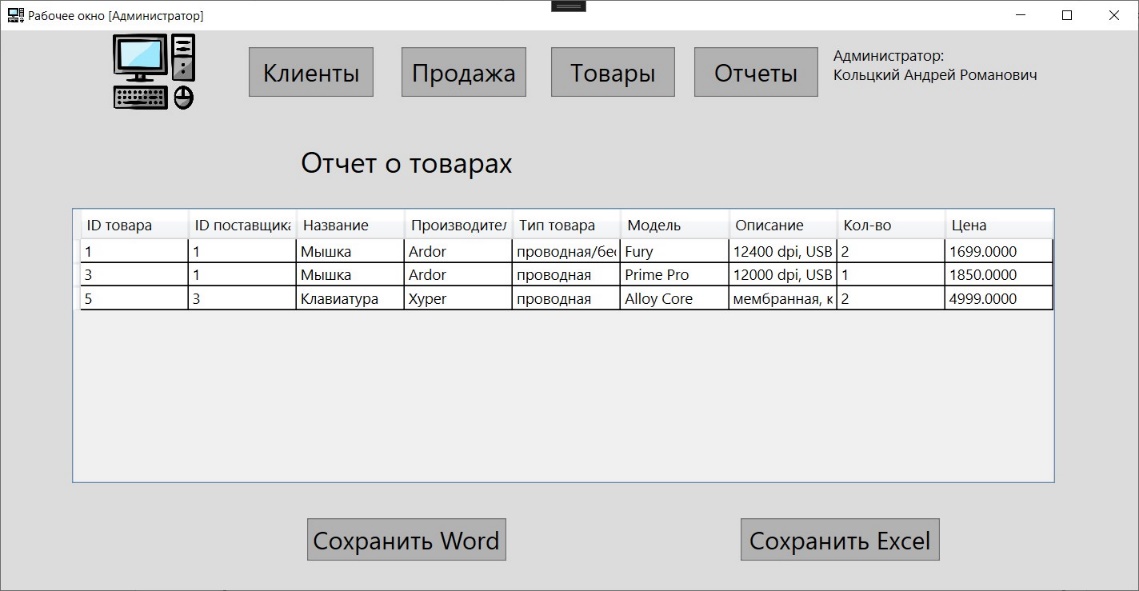


Рисунок 3.2.4 Модуль «Отчеты»

Код «Admin\_report.xaml.cs»

namespace KP.Admin\_Folder

{

public partial class Admin\_report : System.Windows.Controls.Page

{

public Admin\_report()

{

InitializeComponent();

DGrid\_Clients.ItemsSource = KPEntities.GetContext().Product.ToList();

}

private void Button\_Click\_Word(object sender, RoutedEventArgs e)

{

Word.Application word = new Word.Application();

word.Visible = false;

Word.Document document = word.Documents.Add();

Word.Paragraph para = document.Content.Paragraphs.Add();

para.Range.Text = "Отчет о товарах";

Word.Table table = document.Tables.Add(para.Range, DGrid\_Clients.Items.Count + 1, DGrid\_Clients.Columns.Count);

table.Borders.Enable = 1;

for (int j = 0; j < DGrid\_Clients.Columns.Count; j++)

{

table.Cell(1, j + 1).Range.Text = DGrid\_Clients.Columns[j].Header.ToString();

}

for (int i = 0; i < DGrid\_Clients.Items.Count; i++)

{

for (int j = 0; j < DGrid\_Clients.Columns.Count; j++)

{

TextBlock b = DGrid\_Clients.Columns[j].GetCellContent(DGrid\_Clients .Items[i]) as TextBlock;

table.Cell(i + 2, j + 1).Range.Text = b.Text;

}

}

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Word files (\*.docx)|\*.docx|All files (\*.\*)|\*.\*";

if (saveFileDialog.ShowDialog() == true)

{

string filePath = saveFileDialog.FileName;

document.SaveAs2(filePath);

document.Close();

word.Quit();

}

}

private void Button\_Click\_Excel(object sender, RoutedEventArgs e)

{

Excel.Application excel = new Excel.Application();

excel.Visible = false;

Workbook workbook = excel.Workbooks.Add();

Worksheet worksheet = workbook.ActiveSheet;

if (DGrid\_Clients != null)

{

for (int j = 0; j < DGrid\_Clients.Columns.Count; j++)

{

worksheet.Cells[2, j + 1] = DGrid\_Clients.Columns[j].Header.ToString();

}

for (int i = 0; i < DGrid\_Clients.Items.Count; i++)

{

for (int j = 0; j < DGrid\_Clients.Columns.Count; j++)

{

TextBlock b = DGrid\_Clients.Columns[j].GetCellContent(DGrid\_Clients.Items[i]) as TextBlock;

worksheet.Cells[i + 2, j + 1] = b.Text;

}

}

}

else

{

MessageBox.Show("datagridkolvo не инициализирован или равен null.");

worksheet.Cells[1, 1] = "Отчет о товарах";

}

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Excel files (\*.xlsx)|\*.xlsx|All files (\*.\*)|\*.\*";

if (saveFileDialog.ShowDialog() == true)

{

string filePath = saveFileDialog.FileName;

workbook.SaveAs(filePath);

workbook.Close();

excel.Quit();

}

}

}

}

В данном модуле была разработана функция создания отчетов. При нажатии на кнопку откроется окно, где нужно будет выбрать путь куда будет сохраняться файл и его имя. Доступ к данной функции имеют Администратор, Работник склада и Продавец-консультант.

# ЗАКЛЮЧЕНИЕ

При разработке курсового проекта, а именно создание автоматизированной информационной системы «Продажа компьютерной периферией», были спроектированы Проволочные и Entity-Relationship диаграммы. Автоматизированная информационная система была разработана для сотрудников (продавца-консультанта, работника склада, администратора информационной системы). С ее помощью можно быстро создавать, скачивать и печатать необходимые отчеты.

В конечном итоге данный программный продукт упрощает работу персонала, улучшает обслуживание клиентов и помогает автоматизировать продажи, повысить точность учета и контроля за товарами. От внедрения автоматизированной информационной системы «Продажа компьютерной периферией», экономический эффект ожидается за счет сокращения затрат времени на создание отчетов, исключения ошибок при их формировании (так как при ручном создании могут они могут быть допущены) и увеличения времени на анализ данных.

Полученная система соблюдает все требования технического задания, поставленные задачи были выполнены, система разработана и цель достигнута.

Ссылка на репозиторий: [Yuras67/Course\_project (github.com)](https://github.com/Yuras67/Course_project)

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. SQL Server [Электронный ресурс] // Metanit – Режим доступа: [MS SQL Server 2022 и T-SQL (metanit.com)](https://metanit.com/sql/sqlserver/) (дата обращения: 18.03.2024)
2. Диаграмма ER-модель [Электронный ресурс] // WorldSkills – Режим доступа: <https://nationalteam.worldskills.ru/skills/proektirovanie-er-diagrammy/> (дата обращения: 20.12.2023)
3. Язык программирования C# [Электронный ресурс] // Metanit – Режим доступа: [WPF и C# | Полное руководство (metanit.com)](https://metanit.com/sharp/wpf/?ysclid=lw1bmb1xhb498396661) (дата обращения: 15.02.2024)
4. Введение в .NET [Электронный ресурс] // Microsoft – Режим доступа: [Введение в .NET - .NET | Microsoft Learn](https://learn.microsoft.com/ru-ru/dotnet/core/introduction) (дата обращения: 17.01.2024)
5. Работа с Базой Данных [Видео ресурс] // WorldSkills – Режим доступа: [Работа с базой данных в приложении: чтение, добавление, редактирование, удаление данных (часть 1) — Национальная сборная Worldskills Россия](https://nationalteam.worldskills.ru/skills/rabota-s-bazoy-dannykh-v-prilozhenii-chtenie-dobavlenie-redaktirovanie-udalenie-dannykh-chast-1/?ysclid=lw4p42n3g9933379553) (дата обращения: 26.03.2024)
6. Разработка каркаса приложения [Электронный ресурс] // WorldSkills – Режим доступа: [Создание каркаса приложения. Создание и использование стилей — Национальная сборная Worldskills Россия](https://nationalteam.worldskills.ru/skills/sozdanie-karkasa-prilozheniya-sozdanie-i-ispolzovanie-stiley/?ysclid=lw4p2dm5u49767104)