**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 «Магазин Спортивного питания»**

Выполнил студент гр. ИСП-21 /\_\_\_\_\_\_\_\_\_/Волчков Александр Владиславович/

(подпись) (Ф.И.О.)

**ОЦЕНКА:\_\_\_\_\_\_\_\_\_\_\_\_\_**

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

**ПРОВЕРИЛ**

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

(подпись) (Ф.И.О.)

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc180498804)

[ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ 5](#_Toc180498805)

[1.1 Платформа .NET 5](#_Toc180498806)

[1.2 Язык программирования С# 6](#_Toc180498807)

[1.3 Windows Presentation Foundation (WPF) 8](#_Toc180498808)

[1.4 СУБД SQL server 9](#_Toc180498809)

[1.5 Microsoft SQL Server Management Studio 11](#_Toc180498810)

[1.6 Entity Framework 13](#_Toc180498811)

[ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ 15](#_Toc180498812)

[2.2 Разработка базы данных 18](#_Toc180498813)

[ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ 26](#_Toc180498814)

[3.1 Разработка прототипа информационной системы 26](#_Toc180498815)

[3.2 Программирование информационной системы 30](#_Toc180498816)

[3.2.1 Разработка модуля «Авторизация» 30](#_Toc180498817)

[3.2.2 Разработка модуля «Добавить клиента» 32](#_Toc180498818)

[3.2.3 Разработка модуля «Удаление клиента» 34](#_Toc180498819)

[3.2.4 Разработка модуля «Редактирование» 36](#_Toc180498820)

[3.2.4 Разработка модуля «Отчеты» 39](#_Toc180498821)

[ЗАКЛЮЧЕНИЕ 43](#_Toc180498822)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 44](#_Toc180498823)

# ВВЕДЕНИЕ

Из-за такой динамики рынка спортивного питания и роста активности пользователей в онлайн пространстве, компании, занимающиеся продажей спортивного питания, вынуждены адаптироваться к изменяющимся потребностям своих клиентов. Это означает, что необходимо постоянно анализировать поведение целевой аудитории, следить за новыми трендами и технологиями, чтобы дать возможность покупателям получить наилучший опыт покупок в интернет-магазине.

С развитием интернет-технологий появляются новые возможности для улучшения сервиса и удобства клиентов. Онлайн-платформы становятся все более инновационными и интерактивными, что помогает пользователям найти не только нужные им продукты, но и получить дополнительную информацию, советы по тренировкам и питанию, а также коммуницировать с другими участниками сообщества. Это создает уникальное сообщество, которое объединяет любителей спортивного питания и помогает им достигать своих целей более эффективно.

Для успешного развития интернет-магазина спортивного питания необходимо также проводить маркетинговые исследования, чтобы понять потребности и предпочтения клиентов, и на основе полученных данных создавать персонализированные предложения и акции. Это поможет привлечь новых покупателей, удержать существующих и увеличить лояльность к компании. В конечном итоге, быстрая и качественная обработка заказов, оперативная доставка товаров и высокий уровень обслуживания – важные компоненты успешного интернет-магазина спортивного питания, который стремится быть на шаг впереди конкурентов и удовлетворить потребности своих клиентов.

**Объект**: информационная система «Магазин Спортивного питания»

**Предмет**: анализ бизнес-процесса «Магазин Спортивного питания»

**Цель работы**: собрать материал и разработать автоматизированную информационную систему «Магазин Спортивного питания»

**Задачи**:

* Выбрать инструментарий;
* Спроектировать базу данных;
* Разработать прототип информационной системы
* Разработать информационную систему «Магазин спортивного питания».

# ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ

# 1.1 Платформа .NET

Платформа .NET Framework — это технология, которая поддерживает создание и выполнение веб-служб и приложений Windows. При разработке платформы .NET Framework учитывались следующие цели.

* Обеспечение согласованной объектно-ориентированной среды программирования для локального сохранения и выполнения объектного кода, для локального выполнения кода, распределенного в Интернете, либо для удаленного выполнения.
* Предоставление среды выполнения кода, в которой:
* сведена к минимуму вероятность конфликтов в процессе развертывания программного обеспечения и управления его версиями;
* гарантируется безопасное выполнение кода, включая код, созданный неизвестным или не полностью доверенным сторонним изготовителем;
* исключаются проблемы с производительностью сред выполнения скриптов или интерпретируемого кода;
* обеспечиваются единые принципы разработки для разных типов приложений, таких как приложения Windows и веб-приложения;
* обеспечивается взаимодействие на основе промышленных стандартов, которое гарантирует интеграцию кода платформы .NET Framework с любым другим кодом.

Платформа .NET Framework состоит из общеязыковой среды выполнения (среды CLR) и библиотеки классов .NET Framework. Основой платформы .NET Framework является среда CLR. Среду выполнения можно считать агентом, который управляет кодом во время выполнения и предоставляет основные службы, такие как управление памятью, управление потоками и удаленное взаимодействие. При этом средой накладываются условия строгой типизации и другие виды проверки точности кода, обеспечивающие безопасность и надежность. Фактически основной задачей среды выполнения является управление кодом. Код, который обращается к среде выполнения, называют управляемым кодом, а код, который не обращается к среде выполнения, называют неуправляемым кодом. Библиотека классов является комплексной объектно-ориентированной коллекцией повторно используемых типов, которые применяются для разработки приложений — начиная с обычных приложений, запускаемых из командной строки, и приложений с графическим интерфейсом (GUI) и заканчивая приложениями, использующими последние технологические возможности ASP.NET, такие как веб-формы и веб-службы XML.

Платформа .NET Framework может размещаться неуправляемыми компонентами, которые загружают среду CLR в собственные процессы и запускают выполнение управляемого кода, создавая таким образом программную среду, позволяющую использовать средства как управляемого, так и неуправляемого выполнения. Платформа .NET Framework не только предоставляет несколько базовых сред выполнения, но также поддерживает разработку базовых сред выполнения независимыми производителями.

# 1.2 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.

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.

C# — современный объектно-ориентированный и типобезопасный язык программирования. C# позволяет разработчикам создавать разные типы безопасных и надежных приложений, выполняющихся в .NET. C# относится к широко известному семейству языков C, и покажется хорошо знакомым любому, кто работал с C, C++, Java или JavaScript.

C# — объектно-ориентированный, ориентированный на компоненты язык программирования. C# предоставляет языковые конструкции для непосредственной поддержки такой концепции работы. Благодаря этому C# подходит для создания и применения программных компонентов. С момента создания язык C# обогатился функциями для поддержки новых рабочих нагрузок и современными рекомендациями по разработке ПО. В основном C# — объектно-ориентированный язык.

C# подчеркивает управление версиями, чтобы обеспечить совместимость программ и библиотек с течением времени. Вопросы управления версиями существенно повлияли на такие аспекты разработки C#, как раздельные модификаторы virtual и override, правила разрешения перегрузки методов и поддержка явного объявления членов интерфейса.

**Основные преимущества языка**

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 C#, востребованы в индустрии, несмотря на их большое и постоянно увеличивающееся количество.

# 1.3 Windows Presentation Foundation (WPF)

WPF существует в виде подмножества типов .NET, которые по большей части находятся в пространстве имен [System.Windows](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows). Если ранее вы создавали приложения в .NET с помощью таких платформ, как ASP.NET и Windows Forms, то должны быть знакомы с фундаментальными принципами программирования WPF. Вы выполняете следующие действия.

* Создание экземпляров классов
* Задание свойств
* Вызов методов
* Обработка событий

WPF включает в себя дополнительные конструкции программирования, которые расширяют возможности свойств и событий: [свойства зависимостей](https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/properties/dependency-properties-overview?view=netdesktop-8.0) и [перенаправленные события](https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/events/routed-events-overview?view=netdesktop-8.0).

WPF позволяет разрабатывать приложения, используя как разметку, так и код программной части, что привычно для разработчиков на ASP.NET. Разметка XAML обычно используется для определения внешнего вида приложения, а управляемые языки программирования (код программной части) — для реализации его поведения.

Такое разделение внешнего вида и поведения имеет ряд преимуществ.

* Затраты на разработку и обслуживание снижаются, так как разметка, определяющая внешний вид, не связана тесно с кодом, обуславливающим поведение.
* Повышается эффективность разработки, так как дизайнеры, занимающиеся внешним видом приложения, могут работать параллельно с разработчиками, реализующими поведение приложения.
* [Глобализация и локализация](https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/advanced/wpf-globalization-and-localization-overview?view=netdesktop-8.0) приложений WPF упрощена.

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

# 1.4 СУБД SQL server

Microsoft SQL Server — это реляционная система управления базами данных (RDBMS). Приложения и средства подключаются к экземпляру ИЛИ базе данных SQL Server и взаимодействуют с помощью Transact-SQL (T-SQL).

Таблица 1

|  |  |
| --- | --- |
| Компонент | Description |
| Ядро СУБД | Компонент ядра СУБД представляет собой основную службу для хранения, обработки и обеспечения безопасности данных. Ядро СУБД обеспечивает контролируемый доступ и обработку транзакций в соответствии с требованиями наиболее требовательных приложений, использующих данные в вашей организации. Ядро СУБД также обеспечивает поддержку обеспечения непрерывности бизнес-процессов с помощью непрерывности бизнес-процессов и восстановления базы данных — SQL Server. |
| Службы Машинное обучение (MLS) | Службы SQL Server Машинное обучение поддерживают интеграцию машинного обучения с использованием популярных языков R и Python в корпоративные рабочие процессы. Службы машинного обучения (в базе данных) интегрируют R и Python с SQL Server, что позволяет легко создавать, повторно обучать и оценивать модели, вызывая хранимые процедуры. Машинное обучение Server предоставляет корпоративную поддержку R и Python, не требуя SQL Server. |
| Службы Integration Services (SSIS) | Службы SQL Server Integration Services — это платформа для создания решений для интеграции с данными высокой производительности, включая пакеты, обеспечивающие обработку извлечения, преобразования и загрузки (ETL) для хранения данных. |
| Службы Analysis Services (SSAS) | SQL Server Analysis Services — это платформа аналитических данных и набор инструментов для личной, командной и корпоративной бизнес-аналитики. Серверы и клиентские конструкторы поддерживают традиционные решения OLAP, новые решения табличного моделирования, а также самостоятельную аналитику и совместную работу с помощью Power Pivot, Excel и среды SharePoint Server. Службы Analysis Services также включают интеллектуальный анализ данных, чтобы выявить шаблоны и связи, скрытые внутри больших объемов данных. |
| Службы Reporting Services | Службы SQL Server Reporting Services предоставляют корпоративные функции отчетов с поддержкой Веб-приложений. Вы можете создавать отчеты, которые извлекает содержимое из различных источников данных, публиковать отчеты в различных форматах и централизованно управлять безопасностью и подписками. |
| Репликация | Репликация SQL Server — это набор технологий для копирования и распространения данных и объектов базы данных из одной базы данных в другую, а затем синхронизации между базами данных для обеспечения согласованности. Используя реплика tion, вы можете распространять данные в разные расположения, а также удаленным или мобильным пользователям с локальными и широкими сетями, подключениями по телефону, беспроводными подключениями и Интернетом. |
| Data Quality Services (DQS) | Службы качества данных предоставляют решение очистки данных на основе знаний. Службы DQS позволяют создать базу знаний, а затем выполнить в ней исправление данных и удаление дубликатов с помощью как автоматизированных, так и интерактивных средств. Можно использовать службы справочных данных на основе облачных вычислений, а также создавать решения по управлению данными, где службы DQS будут интегрированы со службами SQL Server Integration Services и Master Data Services. |
| Master Data Services (MDS) | Службы Master Data Services — это решение SQL Server для управления главными данными. Решение, созданное на основе служб Master Data Services, помогает гарантировать, что отчеты и анализ основаны на правильной информации. Используя службы Master Data Services, вы создаете центральный репозиторий для основных данных и сохраняете проверяемую, защищаемую запись этих данных по мере изменения с течением времени. |

# 1.5 Microsoft SQL Server Management Studio

SQL Server Management Studio (SSMS) — это интегрированная среда для управления любой инфраструктурой SQL, от SQL Server до баз данных SQL Azure. SSMS предоставляет средства для настройки, наблюдения и администрирования экземпляров SQL Server и баз данных. Используется SSMS для развертывания, мониторинга и обновления компонентов уровня данных, используемых приложениями, и создания запросов и скриптов.

SSMS используется для запроса, проектирования и управления базами данных и хранилищами данных, где бы они ни находились на локальном компьютере или в облаке.

Для клиентов, требующих кроссплатформенного компаньона sSMS для управления SQL и другими базами данных Azure, используйте Azure Data Studio.

Таблица 2

|  |  |
| --- | --- |
| Компонент | Description |
| Обозреватель объектов | **Обозреватель объектов** используется для просмотра всех объектов и управления ими в одном экземпляре SQL Server (или более). |
| Обозреватель шаблонов | **Template Explorer** используется для создания файлов со стандартным текстом, которые можно использовать для ускорения разработки запросов и скриптов, и управления ими. |
| Обозреватель решений | Устаревший **обозреватель решений** используется для создания проектов, применяемых для управления такими элементами администрирования, как скрипты и запросы. |
| Visual Database Tools | Используйте средства визуального проектирования, включенные в Management Studio, для создания запросов, таблиц и баз данных схем. |
| Редакторы запросов и текста | Используйте редакторы языка Management Studio для интерактивной сборки и отладки запросов и скриптов. |

# 1.6 Entity Framework

Entity Framework — это решение для работы с базами данных, которое используется в программировании на языках семейства .NET. Оно позволяет взаимодействовать с СУБД с помощью сущностей (entity), а не таблиц. Также код с использованием EF пишется гораздо быстрее.

Entity Framework является объектно-реляционным отображателем (ORM) для языка программирования C#. Он предоставляет разработчикам удобный способ работать с данными в базах данных, абстрагируясь от деталей работы с SQL.

Entity Framework позволяет работать с данными в виде объектов, а не таблиц, что делает программирование более удобным и интуитивно понятным. Он автоматически генерирует SQL-запросы на основе кода, что значительно упрощает процесс разработки и снижает количество кода, который нужно написать вручную.

Для работы с Entity Framework необходимо создать модель данных, которая представляет собой набор классов, соответствующих таблицам в базе данных. После этого можно использовать LINQ (Language Integrated Query) для выполнения запросов к данным.

Entity Framework позволяет значительно сократить код работы с базами данных. При этом он предоставляет большие возможности. Например, можно использовать:

* foreign keys;
* связи one-to-one, one-to-many и many-to-many;
* параметризацию запросов;
* хранимые процедуры.

Однако стоит учитывать, что EF выступает прослойкой между приложением и базой данных, поэтому может ухудшаться производительность. Для небольших проектов это допустимо, но если программа должна работать под большой нагрузкой, то лучше использовать чистый ADO.NET.

# ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ

**2.1 Разработка диаграммы ERD**

**ER-диаграмма** (Entity-Relationship diagram) — это графическое представление структуры базы данных, которое позволяет описать сущности и их отношения.

**ER-диаграмма нужна** для моделирования и визуализации различных аспектов базы данных. Она помогает разработчикам и аналитикам лучше понять структуру данных, включая сущности, их атрибуты и связи между ними. Диаграмма ER также может использоваться для создания базы данных или для документирования уже существующей базы данных.

**Основными элементами в ER-диаграмме являются:**

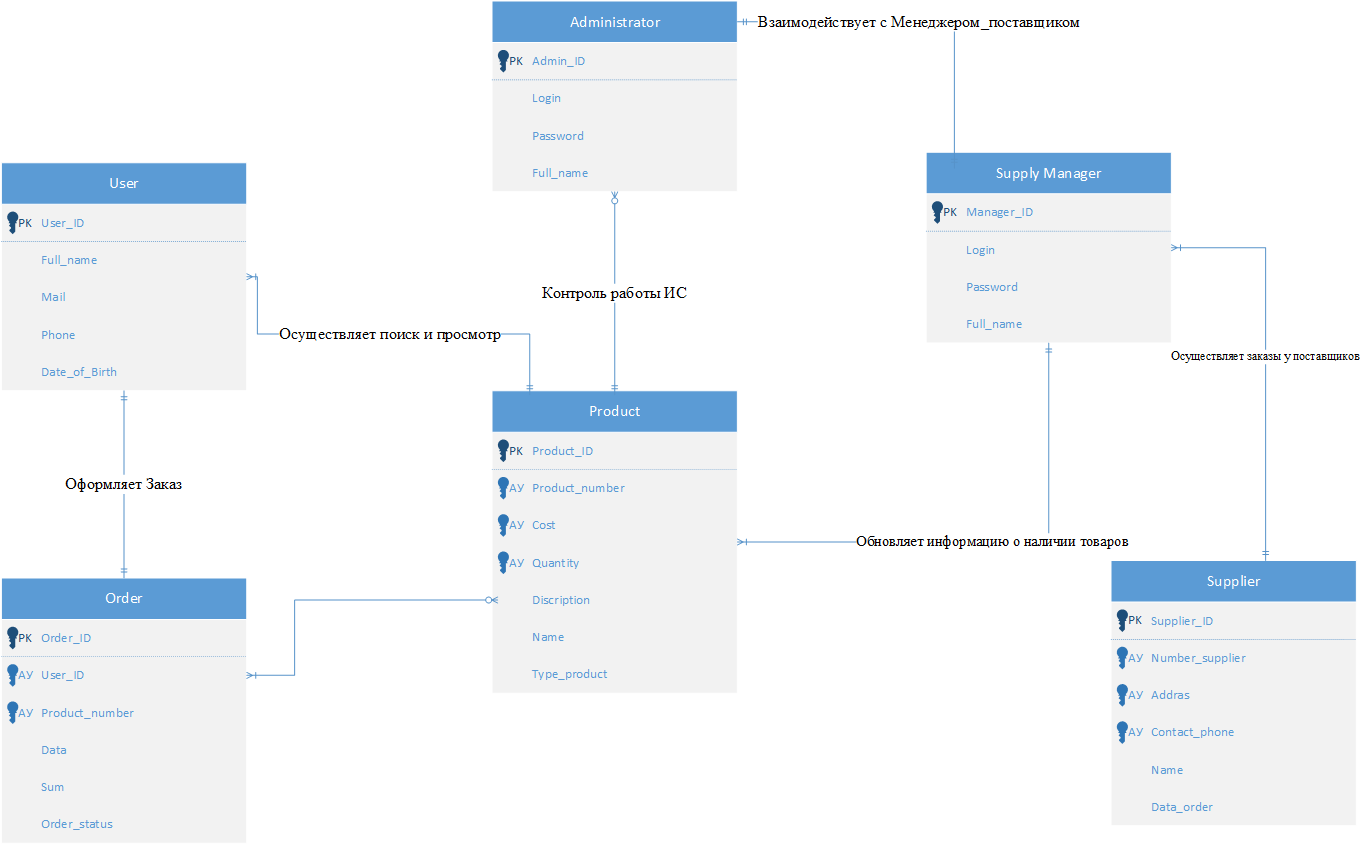
1. Сущности (entities): представляют объекты в базе данных, имеют уникальный идентификатор (первичный ключ) и атрибуты.

2. Атрибуты (attributes): описывают свойства сущностей.

3. Отношения (relationships): связывают сущности друг с другом.

4. Ключи (keys): определяют уникальность сущностей. Имеются первичные ключи, которые уникально идентифицируют каждую сущность, и внешние ключи, которые связывают отношения между сущностями.

В ER-диаграмме эти элементы помогают визуализировать структуру базы данных, позволяют легко понять связи между объектами и логику взаимодействия между ними.

****

*Рисунок 2.1. Диаграмма ER (ER Diagram)*

1. **Сущности (Entities):**

**Пользователь:** Сущность, представляющая клиентов системы. Включает атрибуты: ID, Логин, Пароль, Адрес эл. почты, ФИО.

**Товар:** Сущность, описывающая товары в системе. Содержит атрибуты: ID, Номер товара, Цена, Кол-во товара на складе, Описание, Название, Категории товаров.

**Заказ:** Сущность, представляющая заказы пользователей. Включает атрибуты: ID, Дата, Стоимость, ID\_Пользователя, Статус заказа.

**Администратор ИС:** Сущность, представляющая администратора. Содержит атрибуты: ID Логин, Пароль, Дата регистрации, ФИО.

**Менеджер поставок:** Сущность, описывающая менеджера поставок. Включает атрибуты: ID Логин, Пароль, ФИО, Дата регистрации.

**Поставщик:** Сущность, представляющая поставщиков товаров. Содержит атрибуты: ID, Номер поставщика, Адрес, Контактный телефон, Название, Дата заказа.

1. **Отношения (Relationships):**

Осуществляет поиск и просмотр: Связь между Пользователем и Товаром, позволяющая пользователю искать и просматривать товары.

**Оформляет заказ:** Связь между Пользователем и Заказом, показывающая, что пользователь может оформлять заказы.

**Поиск товаров:** Связь между Пользователем и Товаром, указывающая на процесс поиска и просмотра заказов.

**Следит и управляет акциями:** Связь между Администратором и Товаром, демонстрирующая, что администратор следит и управляет акциями товаров.

**Взаимодействует с менеджером поставок:** Связь между Администратором и Менеджером поставок, указывающая на взаимодействие между ними.

**Отслеживает уровень запасов и обновляет информацию**: Связь между Менеджером поставок и Поставщиком, показывающая, что менеджер отслеживает уровень запасов и обновляет информацию о товарах.

**Осуществляет заказы у поставщиков:** еще одна связь между Менеджером поставок и Поставщиком, указывающая, что менеджер может осуществлять заказы у поставщиков.

**Обновляет информацию о наличии товаров:** Связь между Менеджером поставок и Товаром, демонстрирующая, что менеджер обновляет информацию о наличии товаров.

2.2 Разработка базы данных

SQL Server Management Studio — это набор административных средств для управления компонентами, относящимися к SQL Server. Эта интегрированная среда позволяет пользователям выполнять разнообразные задачи, например резервное копирование данных, редактирование запросов и автоматизацию общих функций в одном интерфейсе.

Среда SQL Server Management Studio включает в себя следующие средства:

Редактор кода — богатый возможностями редактор скриптов для написания и редактирования скриптов. Среда SQL Server Management Studio предоставляет четыре версии редактора кода: редактор запросов ядра СУБД для скриптов Transact-SQL, редактор запросов многомерных выражений, редактор запросов расширения интеллектуального анализа данных и редактор запросов XML/A.

Обозреватель объектов для размещения, изменения, создания скрипта или выполнения объектов, принадлежащих экземплярам SQL Server.

Обозреватель шаблонов для размещения и написания сценариев шаблонов.

Обозреватель решений для организации и хранения связанных скриптов как части проекта.

Окно свойств для отображения текущих свойств выбранных объектов.

Среда SQL Server Management Studio обеспечивает эффективность рабочих процессов, предоставляя:

Отключенный доступ. Можно писать и изменять скрипты, не соединяясь с экземпляром SQL Server.

Создание сценариев из любого диалогового окна. Можно создать скрипт из любого диалогового окна, а также читать, изменять, сохранять и многократно использовать скрипты после создания.

Немодальные диалоговые окна. При обращении к диалоговому окну интерфейса можно просмотреть другие ресурсы в среде SQL Server Management Studio, не закрывая диалоговое окно.

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

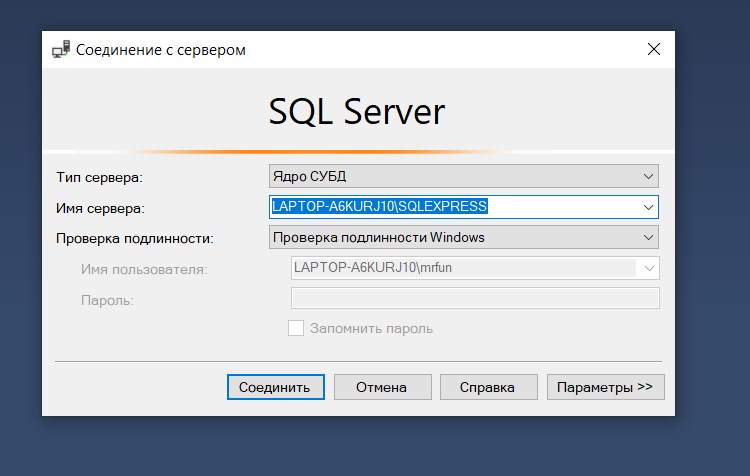


Рисунок 2.2.1 Соединение с сервером

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».

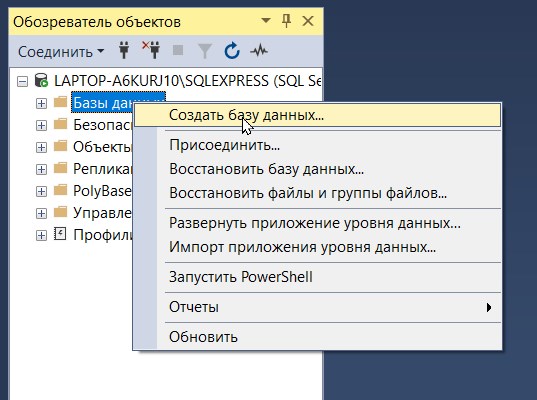


Рисунок 2.2.2 Создание базы данных

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

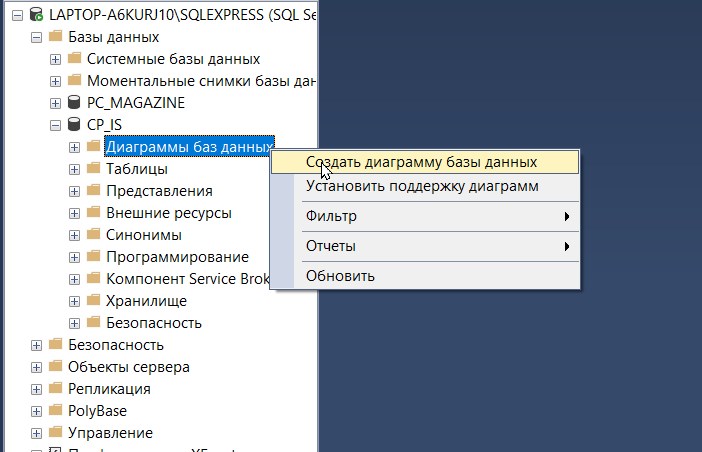


Рисунок 2.2.3 Создание диаграммы

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

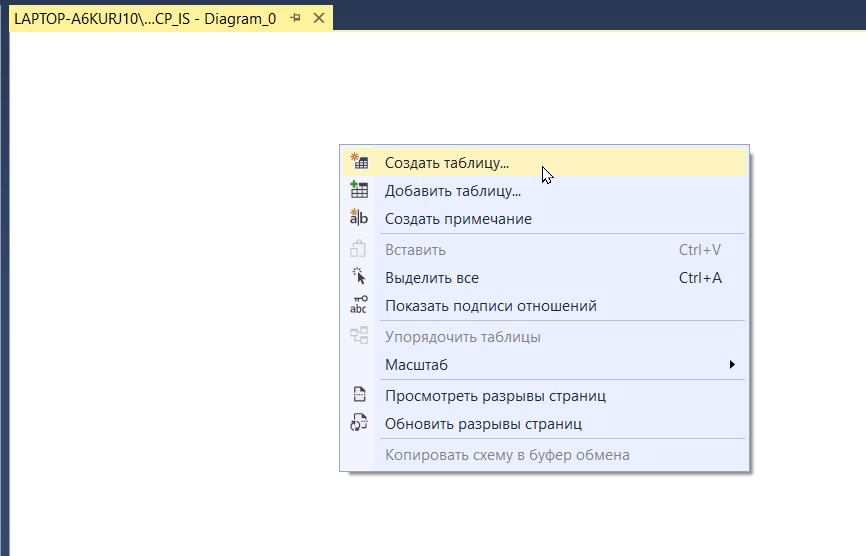


Рисунок 2.2.4 Создание таблицы

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

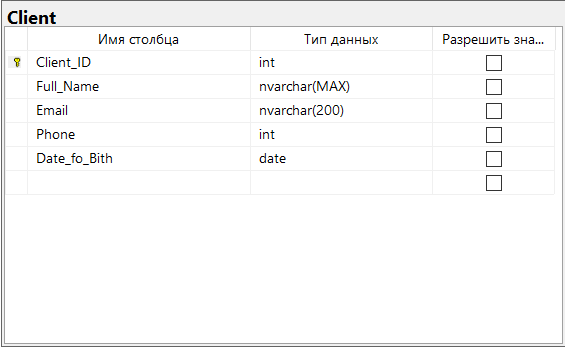


Рисунок 2.2.5 Задаем имена столбцов и указываем типы данных в таблице

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.

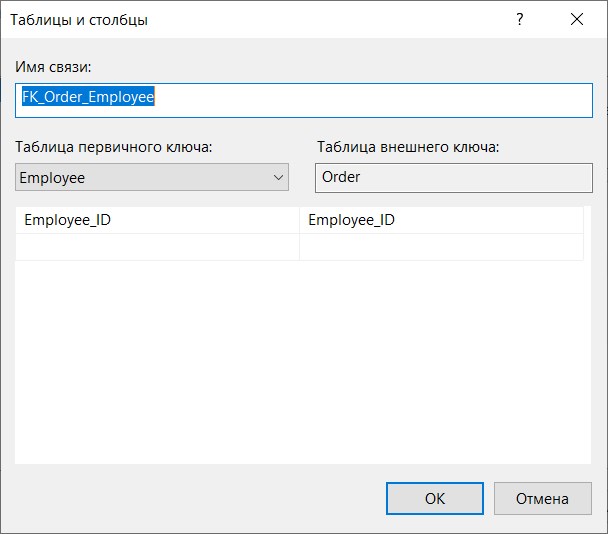


Рисунок 2.2.6 Создаем связи между атрибутами

После создания связей наша таблица полностью готова.



Рисунок 2.2.7 Готовые таблицы диаграммы БД

На данной схеме используются 5 таблиц базы данных. Вот подробное описание каждой из них:

1. Client:

- Client\_ID (Primary Key) — уникальный идентификатор клиента.

- Full\_name — полное имя клиента.

- Email — электронная почта клиента.

- Phone — номер телефона клиента.

- Date\_of\_Birth — дата рождения клиента.

Таблица предназначена для хранения информации о клиентах, включая их имя, контактные данные и дату рождения.

2. Order:

- Order\_ID (Primary Key) — уникальный идентификатор заказа.

- Client\_ID (Foreign Key) — ссылка на клиента, который сделал заказ.

- Product\_ID (Foreign Key) — ссылка на продукт, который был заказан.

- Employee\_ID (Foreign Key) — ссылка на сотрудника, который обработал заказ.

- Product\_Name — наименование продукта.

- Data — дата оформления заказа.

- Sum — общая сумма заказа.

Таблица Order хранит информацию о заказах клиентов, включая информацию о продукте, сотруднике, дате оформления заказа и общей сумме.

3. Employee:

- Employee\_ID (Primary Key) — уникальный идентификатор сотрудника.

- Full\_name — полное имя сотрудника.

- Date\_of\_Birth — дата рождения сотрудника.

- Phone — номер телефона сотрудника.

- Role — должность сотрудника (роль в системе).

- Login — логин для авторизации в системе.

- Password — зашифрованный пароль для авторизации в системе.

Таблица содержит данные о сотрудниках, которые могут управлять заказами и клиентами в системе. Включает их персональные данные, должность и учётные данные для входа в систему.

4. Product:

- Product\_ID (Primary Key) — уникальный идентификатор продукта.

- Provider\_ID (Foreign Key) — ссылка на поставщика.

- Product\_Name — наименование продукта.

- Quantity — количество продукта на складе.

- Description — описание продукта.

- Cost — цена продукта.

Таблица Product хранит информацию о продуктах, включая их наименование, количество, описание и стоимость. Также есть ссылка на поставщика продукта.

5. Provider:

- Provider\_ID (Primary Key) — уникальный идентификатор поставщика.

- Company — название компании поставщика.

- Payment\_Acc — платёжные реквизиты поставщика.

- Address — адрес компании.

- Phone — контактный номер телефона поставщика.

- Email — электронная почта поставщика.

Таблица Provider содержит информацию о поставщиках, таких как название компании, контактные данные и платёжные реквизиты.

Каждая таблица в этой схеме связана через первичные и внешние ключи для обеспечения целостности данных.

# ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

# 3.1 Разработка прототипа информационной системы

Проволочная диаграмма (или wireframe) — это визуальный макет пользовательского интерфейса, который детализирует структуру и функциональность будущего веб-сайта или мобильного приложения. Это своего рода "скелет" интерфейса, представляющий расположение элементов на странице без акцента на их точный дизайн или стиль. Проволочные диаграммы играют ключевую роль на ранних этапах проектирования, помогая визуализировать идеи, уточнить требования и обеспечить единство видения среди членов команды.

Преимущества использования проволочных диаграмм:

- Четкая структура: Они демонстрируют расположение элементов интерфейса, таких как кнопки, текстовые поля, изображения, формы и навигационные элементы, что упрощает обсуждение структуры сайта или приложения.

- Простота корректировки: Проволочные диаграммы не перегружены графическими элементами, поэтому они позволяют легко и быстро вносить изменения.

- Фокус на функциональности: Основное внимание уделяется функциональности интерфейса, а не его внешнему виду, что помогает избежать отвлечения на визуальные детали на этапе концептуального проектирования.

Шаблоны проволочных диаграмм предлагают начальные макеты, которые можно адаптировать под конкретные нужды проекта. Обычно такие шаблоны включают:

- Контейнеры: Эти элементы группируют связанные компоненты, представляя собой структуры страниц, диалоговые окна или карточки.

- Элементы управления: Кнопки, переключатели, чекбоксы и другие интерактивные элементы.

- Мультимедиа: Заглушки для изображений, видео или иконок, позволяющие представить их место в макете.

- Текстовые блоки: Места для заголовков, параграфов и других текстовых данных.

- Навигационные элементы: Меню, панели навигации, хлебные крошки (breadcrumbs) и ссылки.

Существует множество интеллектуальных фигур для создания проволочных диаграмм, таких как ползунки диапазона или индикаторы прогресса, которые можно легко настраивать. Эти элементы позволяют задавать состояние, изменять значения или стили через контекстное меню, что ускоряет процесс проектирования. Проволочные шаблоны универсальны и адаптируются под различные платформы, включая настольные браузеры и мобильные устройства, что делает их полезным инструментом как для веб-разработки, так и для мобильных приложений.

Таким образом, проволочные диаграммы являются отличным инструментом для быстрого создания эскизов интерфейсов, которые позволяют получить общее представление о проекте, согласовать детали с командой и подготовить основу для разработки полноценных каркасных моделей.

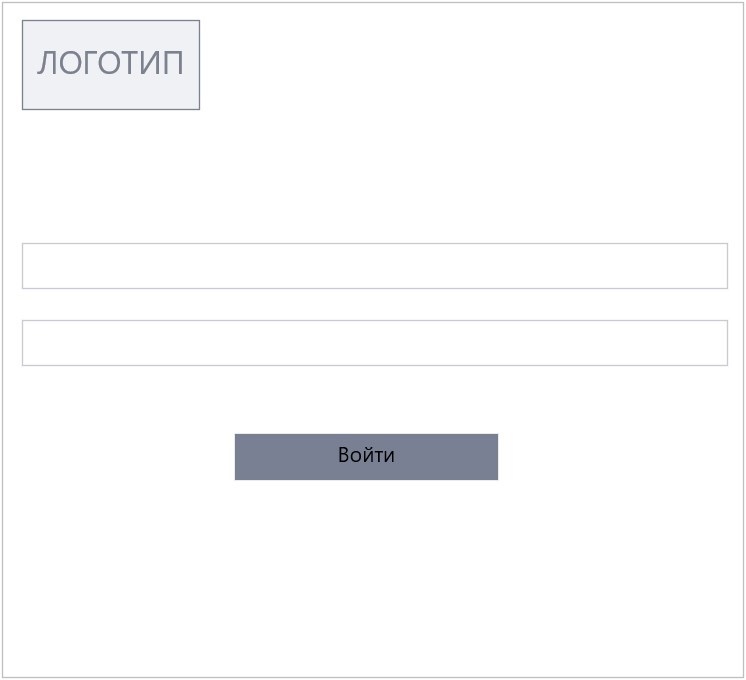


Рисунок 3.1.1 Разработка прототипа формы окна «Авторизация»

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.

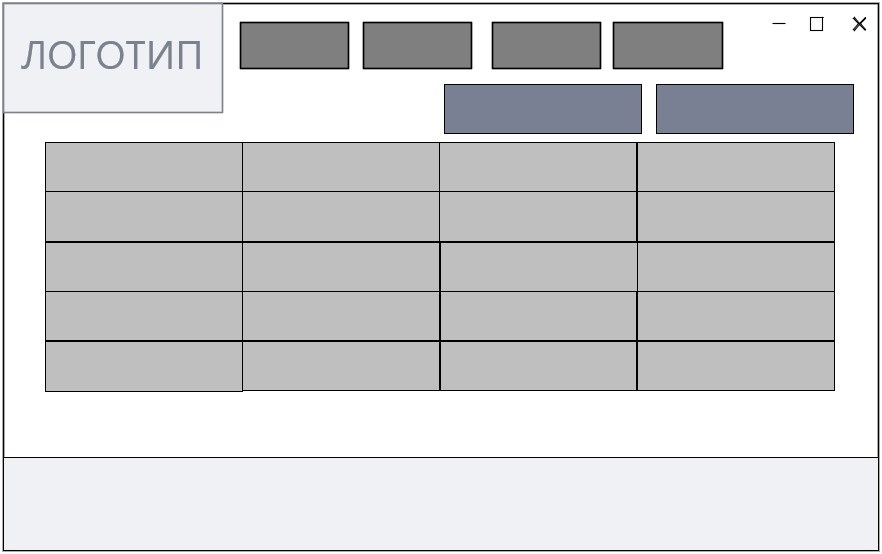


Рисунок 3.1.3 Разработка прототипа формы окон «Клиенты, продажа, товары»

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.

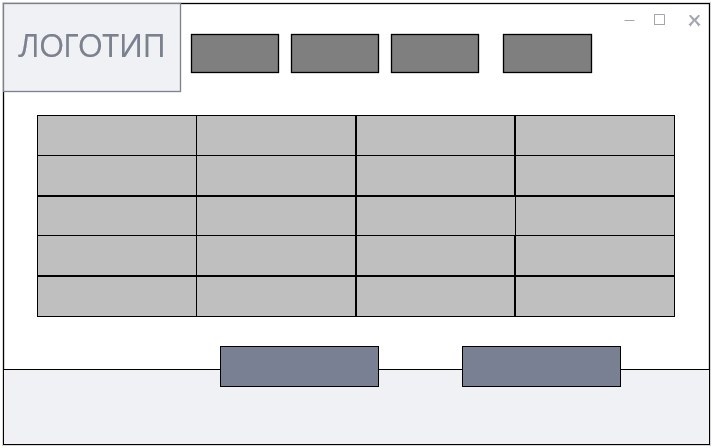


Рисунок 3.1.4 Разработка прототипа формы окна «Отчеты»

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»

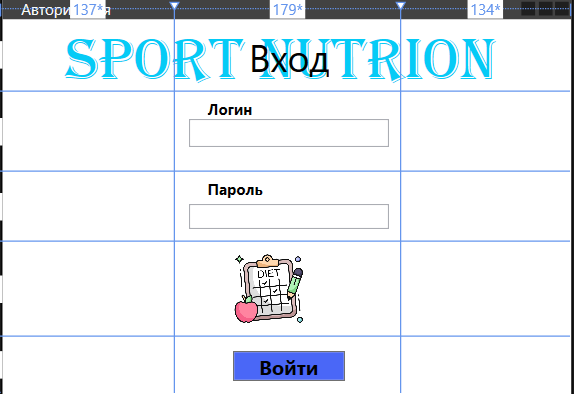


Рисунок 3.2.1. Модуль авторизации

Код «Authorixation.xaml.cs»

namespace KP

{

/// <summary>

/// Класс, отвечающий за логику взаимодействия окна авторизации (Authorization.xaml).

/// </summary>

public partial class Authorization : Window

{

// Конструктор класса Authorization, который инициализирует компоненты окна.

public Authorization()

{

InitializeComponent();

}

/// <summary>

/// Метод, срабатывающий при нажатии на кнопку (Button\_Click). Выполняет проверку логина и пароля.

/// </summary>

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

{

// Создание объектов окон для Менеджера и Администратора

Manager ManagerWindow = new Manager();

Administrator AdministratorWindow = new Administrator();

// Получение логина и пароля, введенных пользователем

string login = Login.Text.ToLower(); // Приведение логина к нижнему регистру для унификации

string password = Password.Password; // Получение пароля из поля ввода

// Проверка логина и пароля для роли менеджера

if (login == "manager" && password == "managermanager")

{

this.Close(); // Закрытие текущего окна авторизации

ManagerWindow.Show(); // Открытие окна Менеджера

}

// Проверка логина и пароля для роли администратора

else if (login == "admin" && password == "adminadmin")

{

this.Close(); // Закрытие текущего окна авторизации

AdministratorWindow.Show(); // Открытие окна Администратора

}

// Если логин или пароль не совпадают, вывести сообщение об ошибке

else

{

MessageBox.Show("Повторите попытку позже"); // Сообщение пользователю о неправильных данных

}

}

}

}

При разработке данного модуля, была создана страница” Авторизация”.

В ней расположены поля для ввода логина и пароля, которые переводят в дальнейшем вас на нужные страницы в зависимости от роли пользователя(Администратор, Менеджер).

# 3.2.2 Разработка модуля «Добавить клиента»

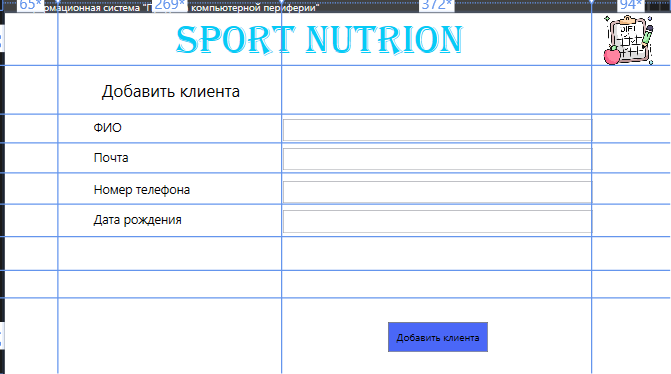


Рисунок 3.2.2 Модуль добавить клиента

Код ”Admin\_Add\_Client.xaml.cs”

namespace KP.Admin\_Folder.Add\_Folder

{

/// <summary>

/// Логика взаимодействия для Admin\_Add\_Client.xaml

/// </summary>

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.Full\_Name))

errors.AppendLine("Укажите имя клиента");

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

errors.AppendLine("Укажите почту клиента");

if (errors.Length > 0)

{

MessageBox.Show(errors.ToString());

return;

}

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

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

try

{

PITEntities.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, Менеджер.

# 3.2.3 Разработка модуля «Удаление клиента»

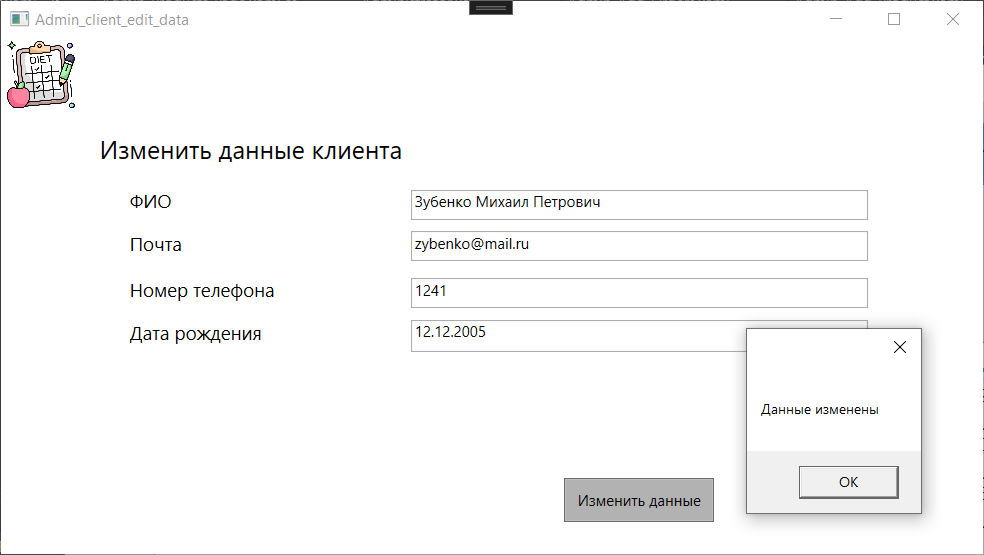


Рисунок 3.2.3 Модуль «Удаление клиента»

Код «Admin\_Clients\_Page.xaml.cs»

namespace KP.Admin\_Folder

{

public partial class Admin\_Clients\_Page : Page

{

// Создание объекта для страницы добавления клиента

Admin\_Add\_Client Admin\_Add\_Client = new Admin\_Add\_Client();

// Конструктор страницы Admin\_Clients\_Page

public Admin\_Clients\_Page()

{

InitializeComponent();

}

// Метод, который срабатывает при изменении видимости страницы

private void Page\_IsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e)

{

// Проверяем, что страница видима

if (Visibility == Visibility.Visible)

{

// Перезагружаем данные клиентов для обновления после возможных изменений

PITEntities.GetContext().ChangeTracker.Entries().ToList().ForEach(p => p.Reload());

// Обновляем источник данных для таблицы клиентов

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

}

}

// Метод, срабатывающий при нажатии на кнопку добавления клиента

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

{

// Открытие окна для добавления клиента

Admin\_Add\_Client.Show();

}

// Метод для редактирования данных клиента, срабатывает при нажатии кнопки "Редактировать"

private void Button\_edit\_data(object sender, RoutedEventArgs e)

{

try

{

// Открытие окна редактирования данных клиента с передачей информации о выбранном клиенте

Edit\_data\_Folder.Admin\_client\_edit\_data admin\_Client\_Edit\_Data

= new Edit\_data\_Folder.Admin\_client\_edit\_data((sender as Button).DataContext as Client);

// Показать окно редактирования

admin\_Client\_Edit\_Data.Show();

}

catch (Exception ex)

{

// В случае ошибки выводим сообщение с текстом исключения

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

}

}

// Метод для удаления клиентов, срабатывает при нажатии кнопки "Удалить"

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

{

// Получаем список клиентов, выбранных для удаления

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

// Подтверждение удаления

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

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

{

try

{

// Удаление выбранных клиентов из базы данных

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

// Сохранение изменений в базе данных

PITEntities.GetContext().SaveChanges();

// Уведомление о завершении удаления

MessageBox.Show("Данные удалены");

// Обновление данных на странице после удаления

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

}

catch (Exception ex)

{

// В случае ошибки выводим сообщение с текстом исключения

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»

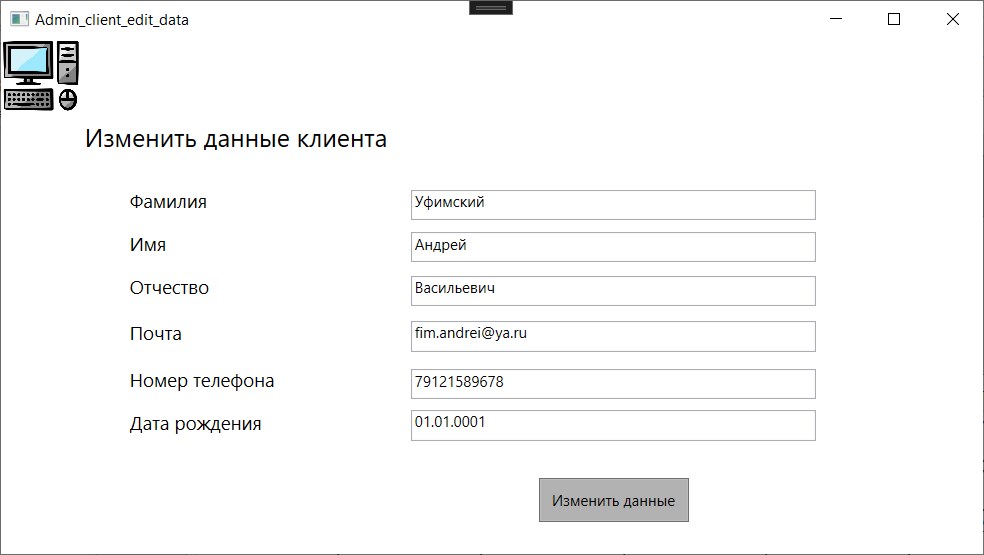


Рисунок 3.2.4 Модуль «Редактирование»

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

namespace KP.Admin\_Folder

{

// Класс для работы с страницей управления клиентами

public partial class Admin\_Clients\_Page : Page

{

// Создание экземпляра окна для добавления клиента

Admin\_Add\_Client Admin\_Add\_Client = new Admin\_Add\_Client();

// Конструктор страницы, вызывается при создании объекта Admin\_Clients\_Page

public Admin\_Clients\_Page()

{

InitializeComponent(); // Инициализация компонентов страницы

}

// Обработчик события, когда видимость страницы изменяется

private void Page\_IsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e)

{

// Если страница становится видимой

if (Visibility == Visibility.Visible)

{

// Обновление данных клиентов с перезагрузкой записей из базы данных

PITEntities.GetContext().ChangeTracker.Entries().ToList().ForEach(p => p.Reload());

// Обновление данных в таблице с клиентами

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

}

}

// Обработчик события нажатия кнопки добавления клиента

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

{

// Открытие окна для добавления клиента

Admin\_Add\_Client.Show();

}

// Обработчик нажатия кнопки редактирования данных клиента

private void Button\_edit\_data(object sender, RoutedEventArgs e)

{

try

{

// Создание экземпляра окна редактирования клиента, передавая выбранного клиента

Edit\_data\_Folder.Admin\_client\_edit\_data admin\_Client\_Edit\_Data

= new Edit\_data\_Folder.Admin\_client\_edit\_data((sender as Button).DataContext as Client);

// Открытие окна редактирования клиента

admin\_Client\_Edit\_Data.Show();

}

catch (Exception ex)

{

// В случае ошибки выводим сообщение об ошибке

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

}

}

// Обработчик нажатия кнопки удаления клиента

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

{

// Получаем список выбранных клиентов для удаления

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

// Подтверждение действия удаления у пользователя

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

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

{

try

{

// Удаление выбранных клиентов из базы данных

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

// Сохранение изменений в базе данных

PITEntities.GetContext().SaveChanges()

// Уведомление пользователя об успешном удалении данных

MessageBox.Show("Данные удалены");

// Обновление таблицы клиентов после удаления

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

}

catch (Exception ex)

{

// В случае ошибки выводим сообщение с текстом исключения

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.

GIT\_HUB-https://github.com/SashaDibiloid/KP.git

# 3.2.4 Разработка модуля «Отчеты»

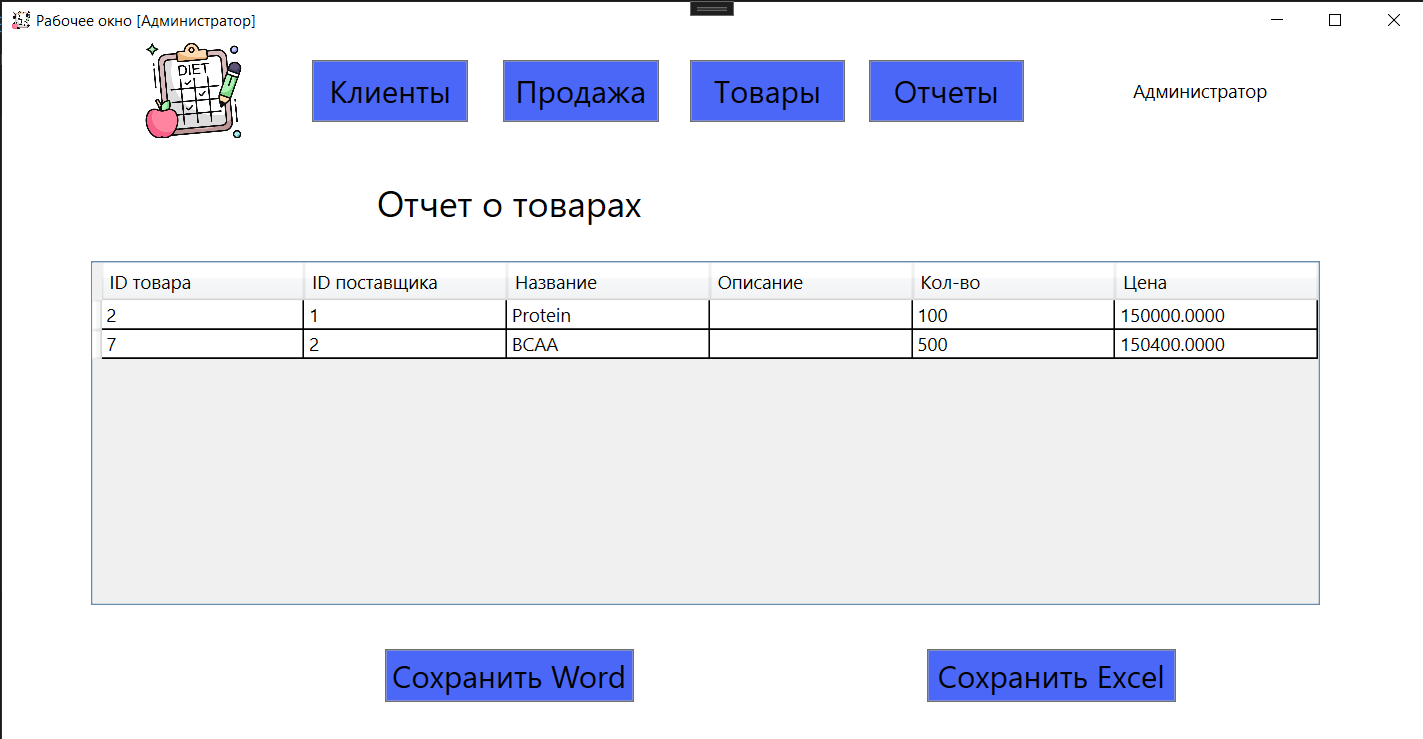


Рисунок 3.2.4 Модуль «Отчеты»

Код «Admin\_report.xaml.cs»

namespace KP.Admin\_Folder

{

/// <summary>

/// Логика взаимодействия для страницы Admin\_report.xaml

/// </summary>

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

{

// Конструктор страницы отчета

public Admin\_report()

{

InitializeComponent();

// Установка источника данных для DataGrid с информацией о продуктах

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

}

// Обработчик нажатия кнопки для экспорта отчета в Word

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

{

// Создание нового экземпляра приложения Word

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

word.Visible = false; // Приложение будет работать в фоновом режиме

// Создание нового документа

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

// Добавление заголовка "Отчет о товарах"

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

para.Range.Text = "Отчет о товарах";

// Создание таблицы в документе Word на основе данных из DataGrid

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

table.Borders.Enable = 1; // Включение границ таблицы

// Заполнение первой строки (заголовков столбцов)

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

{

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

}

// Заполнение строк таблицы данными из DataGrid

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

{

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

{

// Получение содержимого ячейки DataGrid и вставка в таблицу Word

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

if (b != null)

{

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

}

}

}

// Диалог сохранения файла Word

SaveFileDialog saveFileDialog = new SaveFileDialog();

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

if (saveFileDialog.ShowDialog() == true)

{

string filePath = saveFileDialog.FileName;

// Сохранение документа

document.SaveAs2(filePath);

document.Close(); // Закрытие документа

word.Quit(); // Закрытие приложения Word

}

}

// Обработчик нажатия кнопки для экспорта отчета в Excel

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

{

// Создание нового экземпляра приложения Excel

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

excel.Visible = false; // Приложение будет работать в фоновом режиме

// Создание новой книги и листа

Workbook workbook = excel.Workbooks.Add();

Worksheet worksheet = workbook.ActiveSheet;

if (DGrid\_Clients != null) // Проверка, что DataGrid не пуст

{

// Заполнение заголовков столбцов на втором ряду

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

{

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

}

// Заполнение строк таблицы данными из DataGrid

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

{

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

{

// Получение содержимого ячейки DataGrid и вставка в ячейку Excel

TextBlock b = DGrid\_Clients.Columns[j].GetCellContent(DGrid\_Clients.

Items[i]) as TextBlock;

if (b != null)

{

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

}

}

}

}

else

{

// Если DataGrid не инициализирован, вывод сообщения об ошибке

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

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

}

// Диалог сохранения файла Excel

SaveFileDialog saveFileDialog = new SaveFileDialog();

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

if (saveFileDialog.ShowDialog() == true)

{

string filePath = saveFileDialog.FileName;

// Сохранение книги Excel

workbook.SaveAs(filePath);

workbook.Close(); // Закрытие книги

excel.Quit(); // Закрытие приложения Excel

}

}

}

}

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

Таким образом, динамика рынка спортивного питания и растущая активность пользователей в онлайн-пространстве создают уникальные вызовы и возможности для компаний, занимающихся продажей спортивного питания. Адаптация к изменениям потребностей клиентов и использование современных технологий становятся неотъемлемыми условиями успешной работы в данной сфере. Создание интерактивных и инновационных онлайн-платформ позволяет не только эффективно предлагать продукты, но и формировать сообщество, поддерживающее пользователей в достижении их целей.

В ходе выполнения данного курсового проекта была разработана автоматизированная информационная система «Магазин Спортивного питания», которая направлена на улучшение качества обслуживания и персонализацию предложений. Проведенный анализ бизнес-процесса и проектирование базы данных обеспечивают надежную основу для функционирования системы, что позволит эффективно обрабатывать заказы, улучшать взаимодействие с клиентами и предлагать актуальные акции.

С учетом постоянно меняющихся трендов и потребностей пользователей, дальнейшее развитие системы должно быть направлено на постоянное совершенствование сервиса и адаптацию к новым вызовам рынка. Только так можно добиться высоких результатов и сохранить конкурентоспособность на рынке спортивного питания. Настоящая работа, безусловно, является важным шагом на пути к созданию успешного интернет-магазина, который будет отвечать всем современным требованиям и ожиданиям потребителей.

Ссылка на репозиторий: <https://github.com/SashaDibiloid/PIT.git>

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Общие сведения о платформе .NET

[Общие сведения о платформе .NET - .NET Framework | Microsoft Learn](https://learn.microsoft.com/ru-ru/dotnet/framework/get-started/overview) (дата обращения: 18.03.2024)

1. Руководство по классическим приложениям (WPF .NET)

[Что такое Windows Presentation Foundation - WPF .NET | Microsoft Learn](https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/overview/?view=netdesktop-8.0)

(дата обращения: 26.03.2024)

1. Что такое SQL Server Management Studio (SSMS) //[SQL Server Management Studio (SSMS) - SQL Server Management Studio (SSMS) | Microsoft Learn](https://learn.microsoft.com/ru-ru/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16&source=recommendations)// (дата обращения: 15.02.2024)
2. 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)
3. 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)

Дата обращения(25.03.2024)