**Введение**

Автоматизированные информационные системы играют ключевую роль в современном управлении данными, особенно в сферах учета, анализа и обработки информации о товарных предложениях. В условиях стремительного роста строительной отрасли и увеличения спроса на разнообразные материалы для ремонта и строительства, такие системы становятся незаменимыми инструментами для эффективного управления данными. Они позволяют не только оптимизировать рабочие процессы, но и значительно сократить временные затраты, что, в свою очередь, способствует повышению качества обслуживания клиентов.

В последние годы наблюдается резкий рост потребности в строительных и отделочных материалах, включая напольные покрытия. Это подчеркивает значимость автоматизированных информационных систем для ведения баз данных, которые обеспечивают надежное и эффективное управление информацией о различных типах покрытий, их характеристиках, наличии на складе и ценах. Система, разработанная в рамках данной курсовой работы, призвана минимизировать время, затрачиваемое на поиск и обработку данных, а также повысить точность учета. Это особенно актуально в условиях высокой загруженности магазинов строительных материалов, где каждый момент на счету.

Целью курсовой работы является создание автоматизированной информационной системы «Магазин напольных покрытий». Разработка такой системы направлена не только на улучшение процессов учета и анализа информации, но и на предоставление пользователям удобных инструментов для работы с данными. В условиях повышенного спроса на строительные материалы и необходимости оперативного обслуживания клиентов, создание подобной системы становится актуальным и востребованным. Она позволяет автоматизировать процессы учета товаров, оформления заказов, анализа продаж и формирования отчетов. Программа может быть использована в небольших магазинах, где требуется оперативное управление данными и высокая точность учета.

В процессе проектирования необходимо решить следующие задачи:

* провести анализ существующих систем;
* сформулировать требования к программе;
* построить структуру базы данных;
* выбрать среду разработки;
* реализовать программный продукт;
* реализовать меры защиты информации;
* разработать техническую документацию.

**1. Проектирование**

**1.1. Обоснование выбора среды разработки**

Для разработки данного программного продукта можно использовать следующие инструментальные средства: ProjectRider, SharpDevelop и VisualStudio. Рассмотрим каждое и сравним их друг с другом.

* Project Rider — кроссплатформенная [интегрированная среда разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) для платформы .NET, разрабатываемая компанией [JetBrains](https://ru.wikipedia.org/wiki/JetBrains). Поддерживаются языки программирования [C#](https://ru.wikipedia.org/wiki/C_Sharp), [VB.NET](https://ru.wikipedia.org/wiki/Visual_Basic_.NET) и [F#](https://ru.wikipedia.org/wiki/F_Sharp). Быстрая и отзывчивая - Project Rider известна своей скоростью работы и отзывчивостью. Интеллектуальное автодополнение - предоставляет расширенное автодополнение кода, которое значительно ускоряет процесс разработки. Работает на Windows, macOS и Linux. Предоставляет мощные инструменты рефакторинга для улучшения качества кода. Хорошо интегрируется с другими продуктами JetBrains. Rider предоставляет мощные инструменты для отладки и тестирования, включая поддержку многопоточности и асинхронных операций. Project Rider является платным продуктом, что может быть дорого для небольших команд и индивидуальных разработчиков. Несмотря на оптимизации, Rider может потреблять значительное количество оперативной памяти и процессорного времени. Проекты Rider не предназначены для коммерческого использования в бесплатной версии приложения. В условиях санкций установка и получение обновлений может быть затруднительными.
* SharpDevelop— это бесплатная интегрированная среда разработки (IDE) с открытым исходным кодом, разработанная для создания приложений на платформах .NET и Mono. SharpDevelop полностью бесплатен и имеет открытый исходный код, что делает его доступным для всех разработчиков. У SharpDevelop относительно мало системных ресурсов, что позволяет использовать его даже на менее мощных машинах. Поддержка .NET: поддержка основных языков и технологий .NET. Но проект больше не так активно поддерживается и обновляется, что может вызвать проблемы с совместимостью и безопасностью.
* Visual Studio Community 2019 —это мощная интегрированная среда разработки (IDE) от компании Microsoft, которая используется для создания приложений под различные платформы, включая Windows, macOS, Android, iOS и веб. Visual Studio поддерживает разработку на C#, VB.NET, F#, C++, JavaScript, TypeScript, Python, и многих других языках. Также Visual Studio предоставляет одни из самых мощных и всесторонних средств отладки на рынке,включая поддержку многопоточности и отладки удаленных систем. Поддержка огромного количества плагинов и расширений через Visual Studio Marketplace, что позволяет адаптировать IDE под специфические нужды разработчиков. Microsoft предоставляет обширную документацию, учебные материалы и поддержку для Visual Studio. Существует огромное количество онлайн-ресурсов, форумов и сообществ, где можно найти ответы на вопросы и помощь от других пользователей.

**Вывод:** Исходя из вышеперечисленных характеристик инструментальных средств, выбираю VisualStudio 2019 версии Community. Данная версия является бесплатной, а благодаря своей универсальности и мощным возможностям эта интегрированная среда разработки является отличным выбором для создания приложения. Visual Studio Community 2019 поддерживает множество языков программирования, такие как C#, C++, JavaScriptи многие другие.

* С++ — это статически типизированный компилируемый язык программирования общего назначения, расширяющий язык программирования C. Он поддерживает как процедурное, так и объектно-ориентированное программирование, а также некоторые аспекты функционального программирования. C++ позволяет создавать приложения с высокой производительностью благодаря низкоуровневому управлению памятью и прямому доступу к аппаратному обеспечению. Но есть и минусы данного языка. Синтаксис и концепции могут быть сложными для начинающих разработчиков. Ручное управление памятью может приводить к ошибкам, таким как утечки памяти и ошибки сегментации. Компиляция больших проектов может занимать значительное время. Низкоуровневое управление памятью и указателями может приводить к уязвимостям безопасности. В отличие от языков с автоматическим управлением памятью, таких как Java или C#, в C++ разработчики должны вручную управлять памятью.
* C# — это объектно-ориентированный язык программирования, разработанный корпорацией Microsoft в рамках платформы .NET. Синтаксис C# интуитивно понятен и упорядочен, что облегчает изучение и использование языка для начинающих и опытных программистов. C# тесно интегрирован с платформой .NET, что предоставляет доступ к обширной библиотеке классов и инструментов для разработки различных типов приложений. С появлением .NET Core и .NET 5/6, приложения на C# можно разрабатывать и запускать на различных операционных системах, включая Windows, macOS и Linux; C# - строго типизированный язык, что помогает избежать множества ошибок на этапе компиляции и делает код более предсказуемым и надежным. C# идеально сочетается с Visual Studio, одной из самых мощных IDE на рынке, что обеспечивает разработчикам множество инструментов для написания, тестирования и отладки кода. Хотя C# и стал кроссплатформенным, его основное развитие и поддержка исходят от Microsoft, что может ограничивать некоторых разработчиков. А в некоторых случаях производительность приложений на C# может уступать программам на языках с более низким уровнем абстракции, таких как C++.

**Вывод:** Исходя из вышеперечисленного сравнения языков программирования, наиболее подходящим для разработки программного продукта «Магазин напольных покрытий» является C#. C# представляет собой оптимальный выбор языка программирования, обеспечивая необходимую производительность, удобство разработки и доступность широкого спектра инструментов и библиотек.

**1.2 Выбор способа организации входных и выходных данных**

Для организации входных и выходных данных необходима база данных. Visual Studio поддерживает работу с различными базами данных, напримерMySQL, SQLite и PostgreSQL.

* MySQL—популярная реляционная СУБД, разработанная и поддерживаемая компанией Oracle, является одним из самых широко используемых СУБД в мире благодаря своей надежности, быстродействию и простоте использования.

Преимущества:

* удобство и эффективность разработки: плагины и интеграции позволяют эффективно управлять базами данных прямо из Visual Studio, что сокращает время на переключение между различными инструментами и упрощает разработку;
* надежность и стабильность: MySQL является одной из самых надежных и проверенных систем управления базами данных, которая используется многими крупными компаниями по всему миру;
* обширные возможности масштабирования: если ваш проект требует обработки большого объема данных или предполагает рост нагрузки, MySQL может предложить необходимые возможности для масштабирования.

Недостатки:

* ограниченная интеграция: MySQL не имеет такой глубокой интеграции с Visual Studio, как SQL Server, что может привести к дополнительным настройкам и установке плагинов для полноценной работы;
* поддержка инструментов: хотя существуют расширения для работы с MySQL в Visual Studio, они могут не предоставлять такой же широкий функционал и стабильность, как родные инструменты для SQL Server;
* производительность: в некоторых сценариях производительность MySQL может уступать другим СУБД, особенно при работе с очень большими объемами данных или сложными запросами.
* SQLite —это легковесная, встраиваемая реляционная система управления базами данных (СУБД), которая не требует отдельного серверного процесса. В основном используется для хранения данных небольших проектов и мобильных приложений.

Преимущества:

* легковесность: SQLite — это компактная и легковесная СУБД, не требующая отдельного сервера, что делает ее идеальной для небольших проектов, прототипирования и встроенных систем;
* простота в использовании: SQLite легко интегрируется и настраивается в Visual Studio, что упрощает начальную настройку и ускоряет процесс разработки;
* поддержка стандарта SQL: SQLite поддерживает большинство стандартных команд SQL, что облегчает работу с данными и переносимость знаний между различными СУБД.

Недостатки:

* ограниченная масштабируемость: SQLite не предназначена для высоконагруженных приложений и масштабирования на большие объемы данных или высокие нагрузки;
* ограниченные возможности по безопасности: встроенные механизмы безопасности SQLite менее развиты по сравнению с другими СУБД, такими как MySQL или SQL Server;
* проблемы с большими данными: при работе с очень большими объемами данных производительность SQLite может значительно снижаться, что делает ее менее подходящей для аналитических и высоконагруженных приложений.
* PostgreSQL — это мощная и популярная система управления реляционными базами данных с открытым исходным кодом. Она известна своей надежностью, масштабируемостью и гибкостью и используется в различных областях, от небольших приложений до крупных корпоративных систем и высоконагруженных веб-сайтов.

Преимущества:

* бесплатность и открытый исходный код: PostgreSQL является бесплатной СУБД с открытым исходным кодом, что делает его доступным для широкого круга разработчиков и организаций;
* мощные возможности SQL: поддержка сложных SQL-запросов, транзакций, подзапросов, триггеров и хранимых процедур;
* расширяемость: PostgreSQL позволяет добавлять пользовательские функции, типы данных, операторы и индексы, что делает его высоко настраиваемым и расширяемым.

Недостатки:

* ограниченная интеграция с Visual Studio: PostgreSQL не имеет такой глубокой интеграции с Visual Studio, что может потребовать дополнительных настроек и использования сторонних расширений;
* сложность настройки: настройка и администрирование PostgreSQL могут быть более сложными по сравнению с другими СУБД, что требует определенного уровня технических знаний;
* производительность в Windows: хотя PostgreSQL работает на Windows, производительность и стабильность могут быть несколько ниже по сравнению с другими операционными системами, такими как Linux;
* сложность изучения: из-за обширных возможностей и настроек PostgreSQL может потребовать больше времени на изучение и освоение, особенно для начинающих разработчиков.

Сравнив вышеперечисленные СУБД, можем сделать вывод, что PostgreSQL требует какое-то количество времени на изучение и больше подходит для больших проектов. SQLite отлично подходит для небольших приложений, не требующихотдельного сервера, но при выполнении сложных запросов и транзакций может снижаться производительность. Самым оптимальным вариантом для разрабатываемого приложения является MySQL, совмещающая в себе высокую производительность и отличную интеграцию с Visual Studio.

База данных состоит из 7 таблиц в соответствии с рисунком 1.2.1.

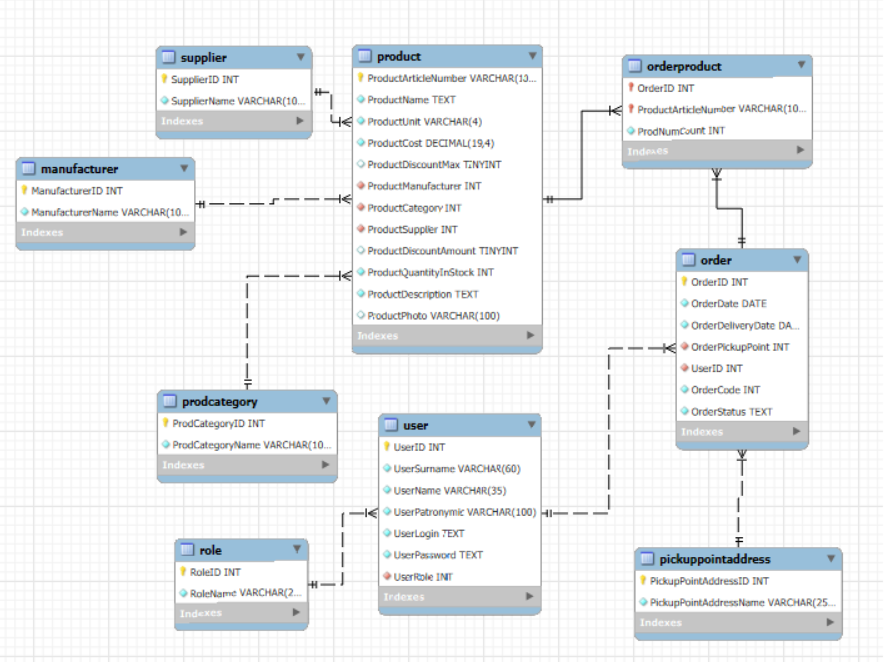


Рисунок 1.2.1—Структура сущностей базы данных

Основной таблицей в базе данных является «Товар» (Product). Таблица «Товар» (Product) состоит из 7 полей, в соответствии с таблицей 1.2.1.

Таблица 1.2.1 — Структура таблицы «Заказы»

|  |  |  |  |
| --- | --- | --- | --- |
| **Название столбца** | **Название столбца (в БД)** | **Тип данных** | **Ограничения** |
| Артикул товара | ProductArticleNumber | varchar(100) | PRIMARY KEY |
| Название товара | ProductName | text | NOT NULL |
| Единица измерения | ProductUnit | varchar(4) | NOT NULL |
| Цена товара | ProductCost | decimal(19,4) | NOT NULL |
| Максимальная скидка | ProductDiscountMax | tinyint | DEFAULT NULL |
| Идентификатор производителя | ProductManufacturer | int | NOT NULL |
| Идентификатор категории | ProductCategory | int | NOT NULL |

Таблица «Категория товаров» (ProdCategory) состоит из 2 полей, в соответствии с таблицей 1.2.2.

Таблица 1.2.2 — Структура таблицы «Категория товаров»

|  |  |  |  |
| --- | --- | --- | --- |
| **Название столбца** | **Название столбца (в БД)** | **Тип данных** | **Ограничения** |
| Уникальный идентификатор | ProdCategoryID | int | AUTO\_INCREMENT, PRIMARY KEY |
| Название категории | ProdCategoryName | varchar(100) | NOT NULL |

Таблица «Пользователи» (User) состоит из 7 полей, в соответствии с таблицей 1.2.3.

Таблица 1.2.3 — Структура таблицы «Клиенты»

|  |  |  |  |
| --- | --- | --- | --- |
| **Название столбца** | **Название столбца (в БД)** | **Тип данных** | **Ограничения** |
| Уникальный идентификатор | UserID | int | AUTO\_INCREMENT, PRIMARY KEY |
| Фамилия пользователя | UserSurname | varchar(60) | NOT NULL |
| Имя пользователя | UserName | varchar(35) | NOT NULL |
| Отчество пользователя | UserPatronymic | varchar(100) | NOT NULL |
| Логин пользователя | UserLogin | text | NOT NULL |
| Пароль пользователя | UserPassword | text | NOT NULL |
| Идентификатор роли | UserRole | int | NOT NULL |

Таблица «Заказы» (Order) состоит из 5 полей, в соответствии с таблицей 1.2.4.

Таблица 1.2.4 — Структура таблицы «Заказы»

|  |  |  |  |
| --- | --- | --- | --- |
| **Название столбца** | **Название столбца (в БД)** | **Тип данных** | **Ограничения** |
| Уникальный идентификатор | OrderID | int | AUTO\_INCREMENT, PRIMARY KEY |
| Дата заказа | OrderDate | date | NOT NULL |
| Дата доставки | OrderDeliveryDate | date | NOT NULL |
| Идентификатор пользователя | UserID | int | NOT NULL |
| Код заказа | OrderCode | int | NOT NULL |

Таблица «Состав заказов» (OrderProduct) состоит из 5 полей, в соответствии с таблицей 1.2.5.

Таблица 1.2.5 — Структура таблицы «Состав заказов»

|  |  |  |  |
| --- | --- | --- | --- |
| **Название столбца** | **Название столбца (в БД)** | **Тип данных** | **Ограничения** |
| Идентификатор заказа | OrderID | int | NOT NULL |
| Артикул товара | ProductArticleNumber | varchar(100) | NOT NULL |
| Количество товара в заказе | ProdNumCount | int | NOT NULL |

Таблица «Роль пользователей» (Role) состоит из 2 полей, в соответствии с таблицей 1.2.6.

Таблица 1.2.6 — Структура таблицы «Роль пользователей»

|  |  |  |  |
| --- | --- | --- | --- |
| **Название столбца** | **Название столбца (в БД)** | **Тип данных** | **Ограничения** |
| Уникальный идентификатор | RoleID | int | AUTO\_INCREMENT, PRIMARY KEY |
| Название роли | RoleName | varchar(20) | NOT NULL |

Таблица «Производители» (Manufacturer) состоит из 2 полей, в соответствии с таблицей 1.2.7.

Таблица 1.2.7 — Структура таблицы «Состав заказов»

|  |  |  |  |
| --- | --- | --- | --- |
| **Название столбца** | **Название столбца (в БД)** | **Тип данных** | **Ограничения** |
| Уникальный идентификатор | ManufacturerID | int | AUTO\_INCREMENT, PRIMARY KEY |
| Название производителя | ManufacturerName | varchar(100) | NOT NULL |

**1.3. Проектирование моделей данных**

Для проектирования модели данных были разработаны диаграммы вариантов использования, деятельности и последовательности действий.

Диаграмма вариантов использования — диаграмма, отражающая варианты использования и ожидаемое поведение программы. Диаграммы вариантов использования помогают описать функциональные требования к системе и показывают, какие функции системы будут использовать пользователи.

Диаграмма вариантов использования представлена в соответствии с рисунком 1.3.1.

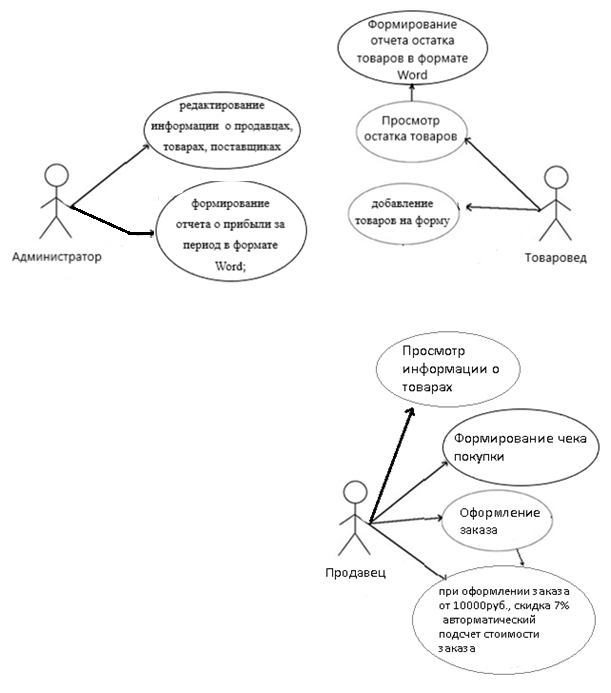


Рисунок 1.3.1 — Диаграмма вариантов использования

**Сценарий диаграммы вариантов использования** (use case, юзкейс) —это сценарная техника описания взаимодействия пользователей с продуктом, которая приводит к достижению конкретной цели.

Сценарий диаграммы вариантов использования представлена в соответствии с таблицей 1.3.1.

Таблица 1.3.1 – Сценарий диаграммы вариантов использования

|  |  |
| --- | --- |
| Вариант использования | Оформление заказов |
| Актеры | Администратор, Товаровед, Продавец |
| Краткое описание | Администратор добавляет новых пользователей, редактирует или удаляет их данные. Администратор формирует отчет о прибыли за период в формате Word;  Продавец может осуществлять поиск информации о товарах и просматривать их, создавать заказы и формировать чеки покупок;  Товаровед занимается ведением базы товаров. Товаровед формирует отчет по продажам за период в формате Word. |
| Цель | Получение прибыли |
| Тип | Базовый |
| Ссылки на другие варианты использования | Добавление, редактирование и удаление данных пользователей; добавление и редактирование данных сотрудников; удаление данных сотрудников; запись заказов; просмотр информации о товарах; формирование отчетов в формате Word |

**Таблица последовательности действий —** это структурированный способ представления шагов, которые необходимо выполнить для достижения определенной цели или выполнения задачи в системе. Она помогает визуализировать, как пользователи (или системы) взаимодействуют с системой, и описывает последовательность действий и реакций системы на эти действия.

Таблица последовательности действий представлена в соответствии с таблицей 1.3..

Таблица 1.3.2 – Последовательность действий

|  |  |
| --- | --- |
| Действия актеров | Отклик системы |
| 1. Продавец просматривает товары  3. Продавец создает заказ  5. Продавец рассчитывает заказ с учетом скидки  7. Продавец формирует чек покупки  9. Товаровед просматривает товары  11. Товаровед просматривает заказы  13. Товаровед оформляет заказ у товароведа. Исключения: 1. Пустые поля для ввода, 2. Удаление всех товаров из заказа | 2. Отображение формы «Товары»  4. Создается заказ в базе данных  6. В итоговую сумму заказа добавляется скидка 7% от 10000 рублей  8. Открывается документ Word, в котором создается чек покупки и подставляются данные из программы.  10. Отображение формы «Товары»  12. Отображение формы «Заказы»  14. Отображение формы «Заказы»  15. Отображение ошибки, что поля не заполнены  16. Отображение сообщения, что заказ будет удалён, так как товаров в нём не осталось. |

Продолжение таблицы 1.3.2

|  |  |
| --- | --- |
| Действия актеров | Отклик системы |
| 17. Товаровед удаляет заказ  19. Товаровед формирует и экспортирует отчет по продажам  21. Администратор просматривает товары  23. Администратор просматривает поставщики  25. Администратор просматривает заказы  27. Администратор просматривает категории товаров  29. Администратор просматривает пользователей  31. Администратор редактирует товары. Исключения: 1. Пустые поля для ввода  34. Администратор редактирует поставщиков. Исключения: 1. Пустые поля для ввода  37. Администратор редактирует категории товаров. Исключения: 1. Пустые поля для ввода  40. Администратор редактирует товары. Исключения: 1. Пустые поля для ввода  43. Администратор редактирует пользователей. Исключения: 1. Пустые поля для ввода | 18. Отображение формы «Заказы»  20. Открывается документ Excel, в котором формируется и экспортируется отчет по заказам магазина одежды с подсчетом дохода за указанный период.  22. Отображение формы «Товары»  24. Отображение формы «Поставщики»  26. Отображение формы «Заказы»  28. Отображение формы «Товары»  30. Отображение формы «Пользователи»  32. Отображение формы «Товары»  33. Отображение ошибки, что поля не заполнены  35. Отображение формы «Пользователи»  36. Отображение ошибки, что поля не заполнены  38. Отображение формы «Товары»  39. Отображение ошибки, что поля не заполнены  41. Отображение формы «Товары»  42. Отображение ошибки, что поля не заполнены  44. Отображение формы «Пользователи»  45. Отображение ошибки, что поля не заполнены |

**Диаграмма деятельности** описывает последовательность действий в системе. Диаграмма деятельности предоставлена в соответствии с рисунком 1.3.2

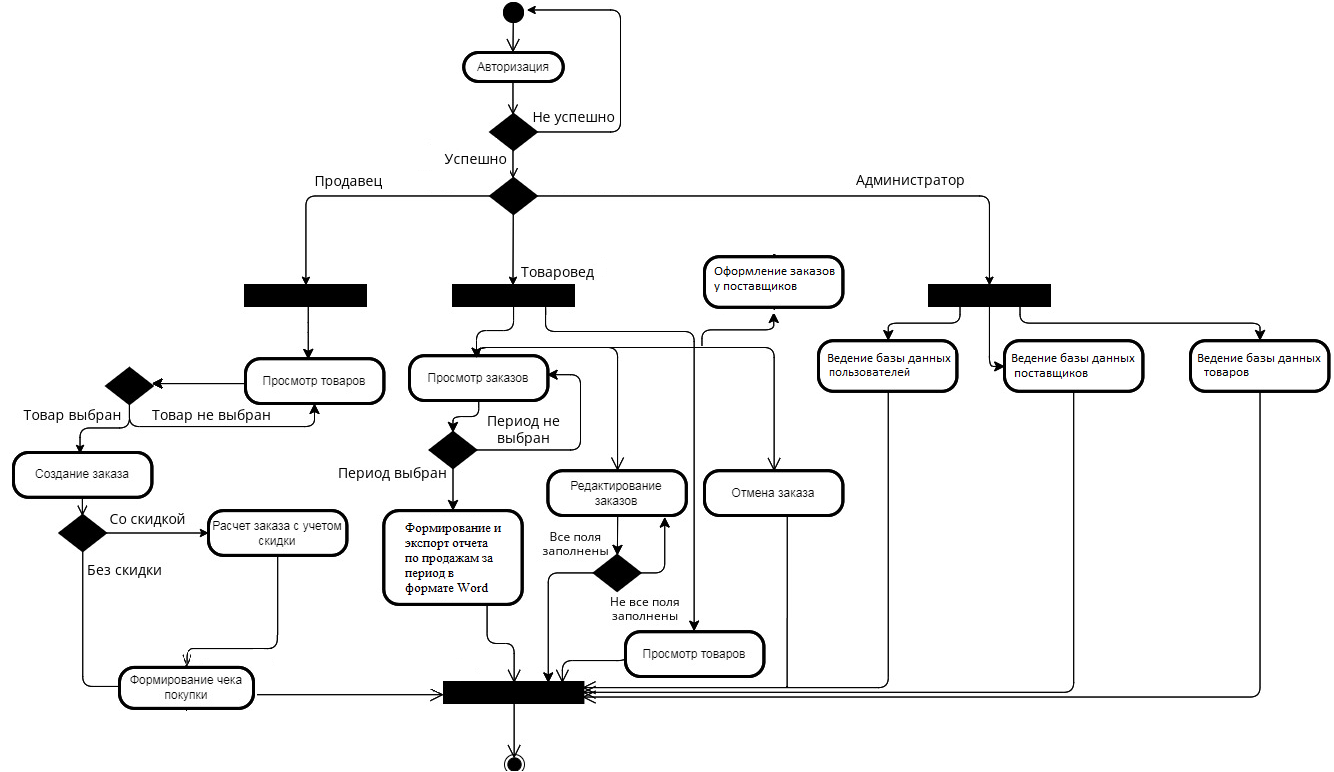


Рисунок 1.3.2 – диаграмма деятельности

**2 Алгоритм представления задачи**

**2.1 Детальное проектирование пользовательского интерфейса**

**Прототип** — это предварительно созданный макет дизайна, дающий пользователям представление о продукте и возможность взаимодействовать с ним.

**Виды прототипов:**

**Горизонтальный прототип** — это прототип демонстрирует широкий спектр функций продукта, но с ограниченной глубиной проработки каждой из них. Представьте, что вы смотрите на дом снаружи — вы видите все комнаты, но не можете заглянуть внутрь. Горизонтальный прототип показывает основные возможности продукта, но не углубляется в детали их реализации.

**Вертикальный прототип** — подробно прорабатывает ограниченное количество функций продукта. Это как заглянуть в одну комнату дома — вы видите все детали, но не видите другие комнаты. Вертикальный прототип позволяет глубоко изучить конкретные функции, но не дает полного представления о продукте.

**Одноразовый прототип** — это прототип, созданный для одной конкретной цели, обычно для демонстрации концепции, получения обратной связи или исследования конкретной идеи. Он не предполагает дальнейшего развития и обычно не используется после достижения своей цели.

**Эволюционный прототип** — это прототип, который развивается и совершенствуется по мере продвижения проекта. Он используется как рабочая модель, которая подвергается модификациям и дополнениям на протяжении всей разработки продукта, отражая эволюцию идей и требований. Эволюционный прототип является «живым» документом, который активно развивается и меняется в ответ на новые знания и обратную связь.

**Бумажный прототип** — создается на бумаге с помощью ручки и карандаша. Простой и быстрый способ визуализации идеи продукта. Позволяет быстро проверить концепцию и получить первые отзывы от пользователей.

Исходя из вышеперечисленных прототипов, было решено выбрать эволюционный прототип для своего программного продукта. В отличие от одноразовых или бумажных прототипов, он позволяет построить рабочую модель, которая постоянно совершенствуется, отражая изменения требований и получаемую обратную связь. В сравнении с горизонтальными и вертикальными прототипами, эволюционный подход даёт возможность не только увидеть широкую картину или углубиться в детали, но и постепенно наращивать функциональность и улучшать качество продукта в целом. Такой итеративный подход минимизирует риски создания невостребованного продукта и обеспечивает соответствие конечного решения потребностям пользователей.

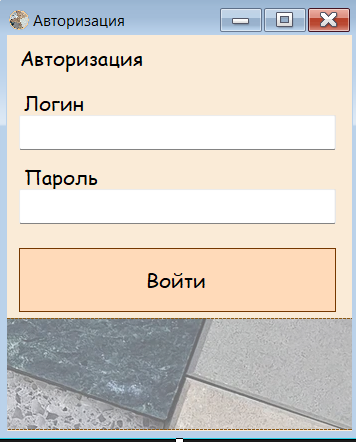


Рисунок 2.1.1 – Форма авторизации

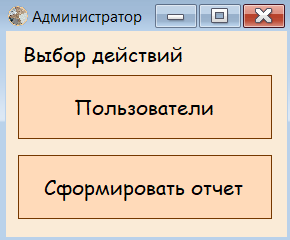


Рисунок 2.1.2 – Форма «Администратор» под ролью администратора

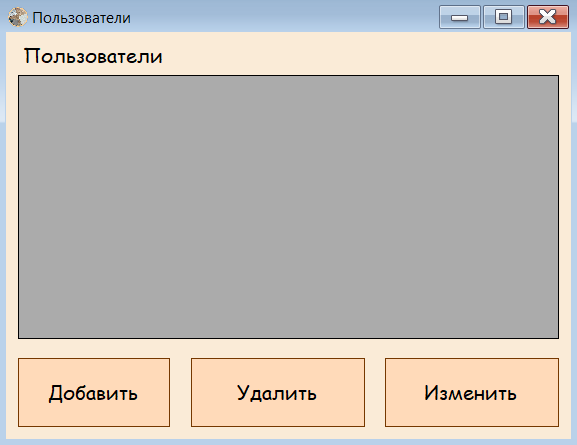


Рисунок 2.1.3 – Форма «Пользователи» под ролью администратора

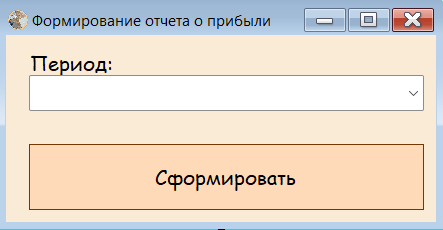


Рисунок 2.1.4 – Форма «Формирование отчета» под ролью администратора

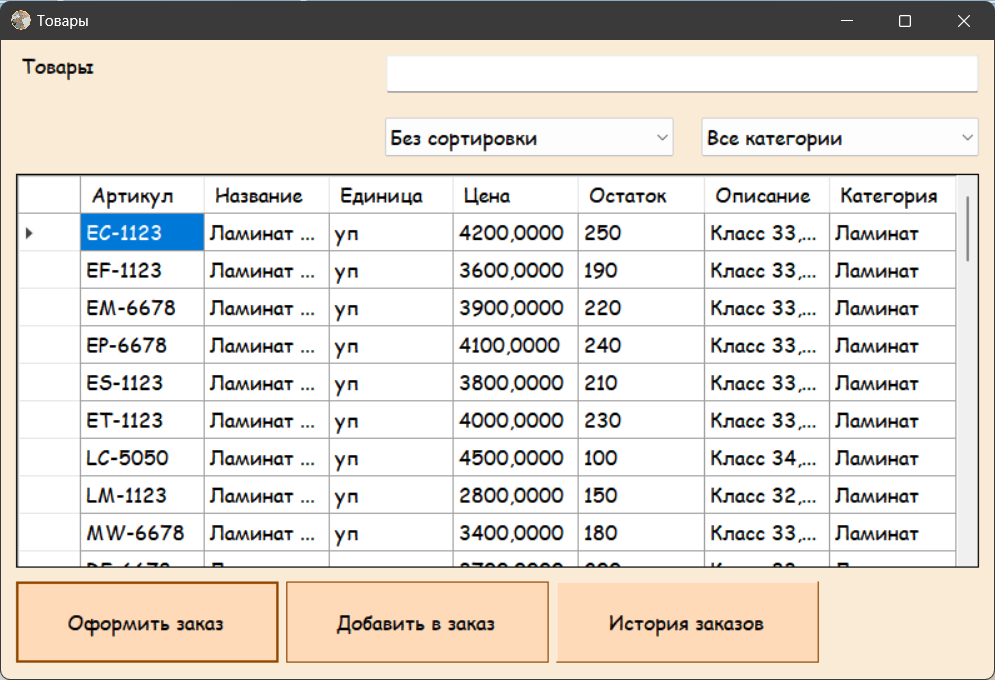


Рисунок 2.1.6 – Форма «Товары» под ролью продавец

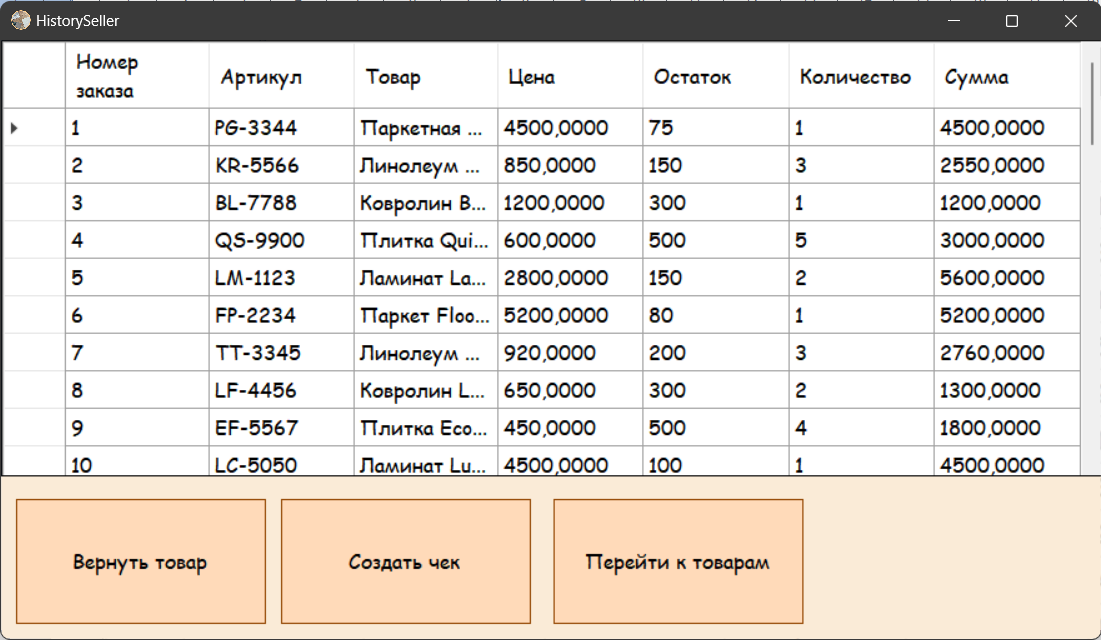


Рисунок 2.1.7 – Форма «История товаров» под ролью продавца

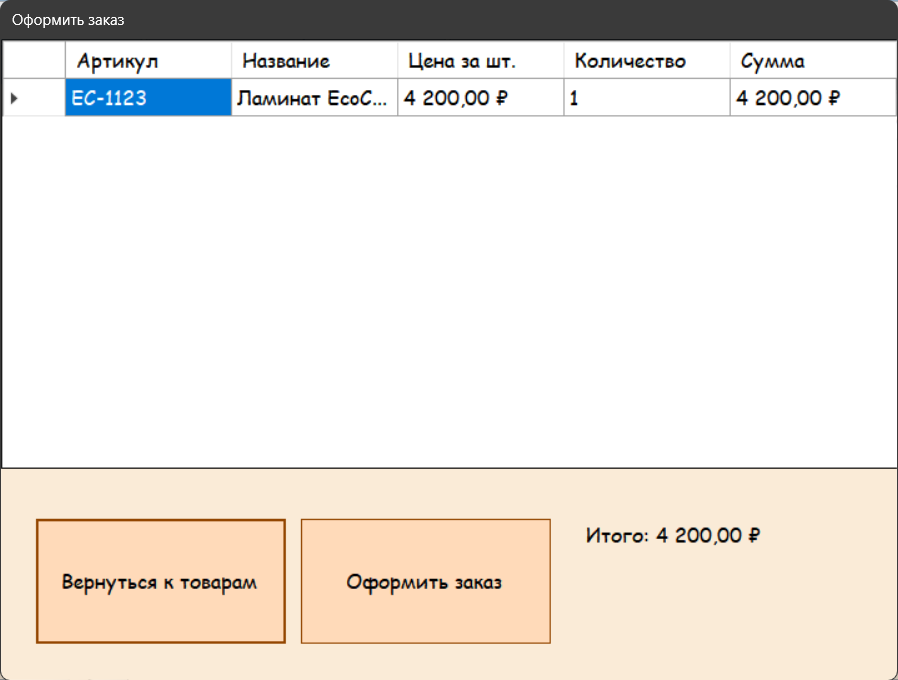


Рисунок 2.1.11 – Форма «Создание заказа» под ролью продавца

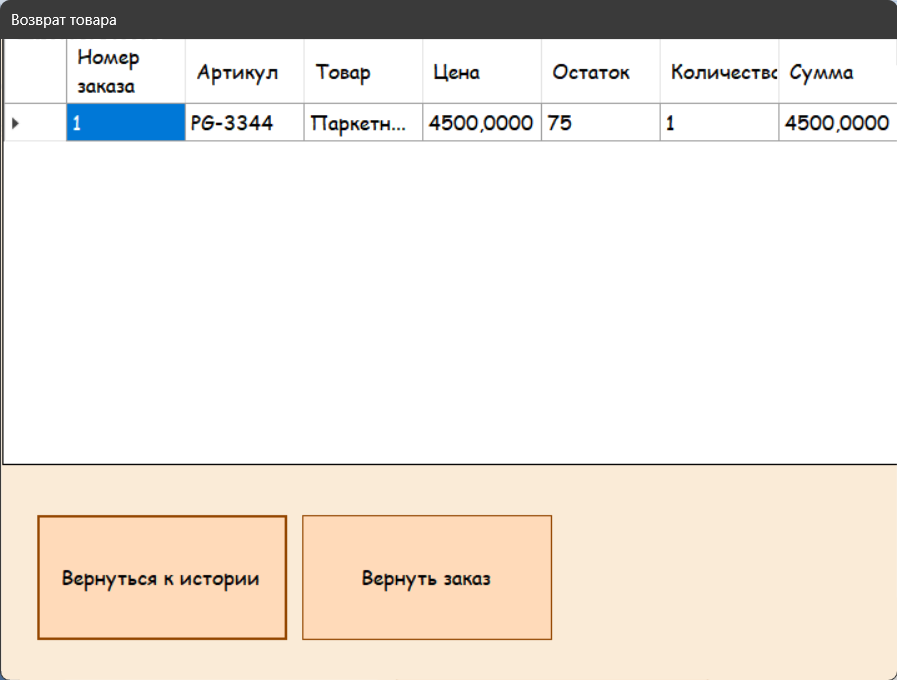


Рисунок 2.1.12 – Форма «Отмена заказа» под ролью продавца

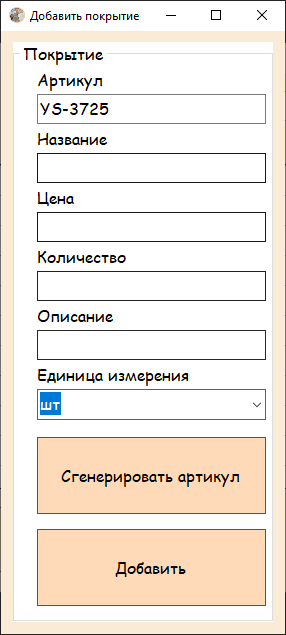


Рисунок Б.13 – Форма «Добавление товара» под ролью товаровед

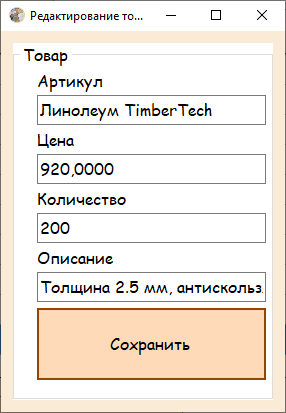


Рисунок Б.14 – Форма «Редактирование товара» под ролью товаровед

**2.2 Описание основных программных модулей**

AddOrder\_Click(object sender, EventArgs e): этот модуль обрабатывает нажатие кнопки "Добавить в заказ". Передает данные товаров на форму «Заказ», для оформления заказа.

OrderBtn\_Click(object sender, EventArgs e): этот модуль обрабатывает нажатие кнопки «Оформить заказ». Открывает форму «Заказ», вставляет в DataGridView переданные данные товаров для оформления заказа.

SellerOrder\_FormClosing(object sender, FormClosingEventArgs e): этот модуль обрабатывает закрытие формы «SellerOrder»(Товары). При закрытии формы, открывается форма (с авторизацией) Authorization.cs.

CloseOrder\_Click(object sender, EventArgs e): этот модуль обрабатывает событие нажатия кнопки «Вернуть товар». Открывает форму «Оформление отмены заказа» с переданными данными о заказе.

LoginIn\_Click(objectsender, RoutedEventArgs e): этот модуль обрабатывает событие нажатия кнопки "Войти". Он получает логин и пароль из полей ввода, хеширует пароль с помощью SHA256, проверяет наличие пустых полей и выполняет запрос к базе данных MySQL для проверки подлинности пользователя. В зависимости от роли пользователя (полученной из базы данных), открывает форму Administrator.cs или SellerForm.cs, или выводит сообщение об ошибке.

Authorization\_Closing(objectsender, RoutedEventArgs e): модуль обрабатывает действие выхода с формы Authorization. Выводит диалоговое окно с подтверждением выхода и закрывает приложение, если пользователь подтверждает действие.

SVPasswordBtn\_Click(objectsender, RoutedEventArgs e): модуль управляет отображением пароля. При нажатии кнопки переключает видимость отображения пароля (passwordTxt), а также изменяет содержимое кнопки (символ глаза). Обеспечивает возможность пользователю увидеть вводимый пароль.

**2.3 Применяемые меры защиты информации**

Защита данных — это комплекс мер, направленных на обеспечение конфиденциальности, целостности и доступности данных. Это включает в себя предотвращение несанкционированного доступа, использования, раскрытия, изменения или уничтожения данных, а также обеспечение их доступности для авторизованных пользователей в нужное время.

Функции:

* обеспечение безопасности данных;
* обеспечение секретности данных.

В процессе разработки информационной системы были реализованы следующие меры защиты информации:

1. *Разграничение прав доступа*

Разграничение прав доступа — это процесс определения и управления правами пользователей или групп пользователей на доступ к ресурсам, данным или функциональности системы. Это позволяет контролировать, кто может выполнять определенные действия (например, чтение, запись, изменение или удаление данных) и обеспечивает безопасность информации, предотвращая несанкционированный доступ и использование ресурсов. Разграничение прав доступа часто реализуется через системы аутентификации и авторизации, а также с помощью ролей и политик безопасности. Разграничение прав доступа в соответствии с рисунком 2.3.1.

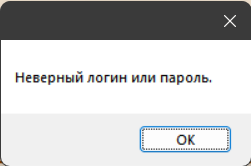


Рисунок 2.3.1 — Парольная защита базы данных

1. *Скрытие пароля при его вводе* представлено в соответствии с рисунком 2.3.2

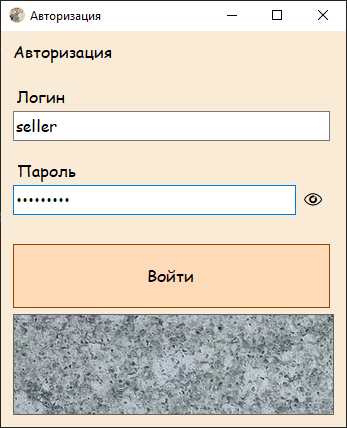


Рисунок 2.3.2 — Скрытие пароля

1. *Проверка на заполнение обязательных полей* в соответствии с рисунком 2.3.3.

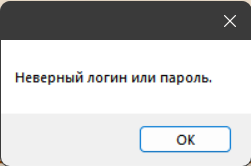


Рисунок 2.3.3 — Проверка на заполнения поля «Логин и пароль»

1. *При неудачной попытке авторизации система выдаёт сообщения* о неуспешной авторизации в соответствии с рисунком 2.3.4.

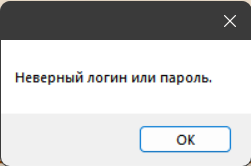


Рисунок 2.3.4 — Проверка при неправильном логине или пароле

1. *Блокировка некорректного ввода данных* или некорректных действий пользователя.

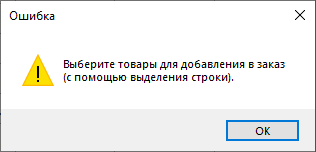


Рисунок 2.3.5 — Главная форма продавца

1. *При повторной авторизации* требуется заполнение полей: логин и пароль.

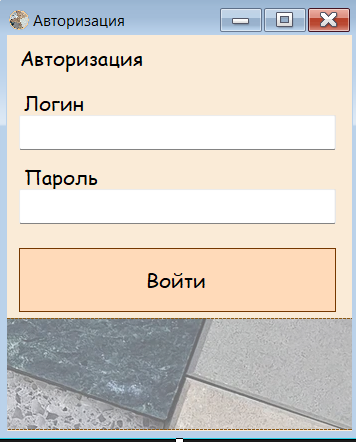


Рисунок 2.3.6 — Главная форма

1. *Выход из программы* только через главную форму.

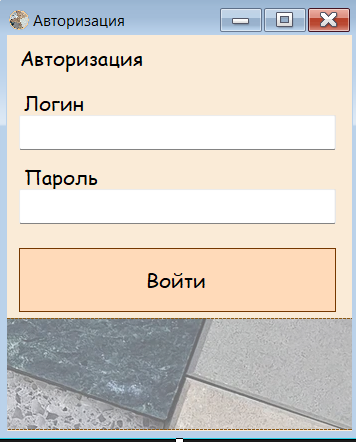


Рисунок 2.3.7 — Главная форма

**Заключение**

В ходе выполнения курсового проекта было создано программное приложение «Магазин напольных покрытий».

В процессе описания программного кода, использовался язык программирования высокого уровня C#. Были улучшены навыки платформы .NET.

Программное приложение обеспечивает выполнение следующих функций:

* разграничение прав для 3-х ролей: администратор, продавец и товаровед;
* «живой» поиск при работе с информацией о товарах по наименованию товара;
* сортировка информации о товарах по возрастанию, убыванию цены и по алфавиту названия товаров;
* фильтрация на формах с информацией о товарах по категориям и по статусу заказа;
* совместное выполнение функций «Фильтрация» и «Поиск»;
* в режиме «Администратор» пользователю предоставляются возможности:
* добавление/редактирование и удаление информации о продавцах, товарах, поставщиках;
* формирование отчета о прибыли с продаж за период в формате Word;
* в режиме «Продавец» пользователю предоставляются возможности:
* просмотр информации о товарах;
* оформление заказа;
* автоматический подсчет стоимости заказа с учетом скидки при оформлении заказа от 10000 руб, скидка 7%;
* просмотр истории оформленных заказов на форме «История заказов»;
* возможность отмены заказа на форме «История заказов»;
* формирование чека покупки;
* в режиме «Товаровед» пользователю предоставляются возможности:
* просмотр остатка товаров на форме «Информация о товарах»;
* возможность добавления/изменения изображения товаров на форме «Информация о товарах»;
* ведение базы товаров
* формирование отчета по продажам за период в формате Word;
* «живой» поиск информации на форме «Товары» будет происходить по наименованию товара и артикулу;
* реализация функции сортировки информации на форме «Товары» по количеству (по убыванию или по возрастанию)на складе;
* реализация функции фильтрации данных формах «Товары» по категориям;
* поиск, сортировка и фильтрация происходят в реальном времени без необходимости нажатия кнопки «Найти/Фильтровать»;
* сортировка, выбранная ранее сохраняется во время фильтрации и поиска;
* формирование отчета в Word о выручке за определённый период времени;
* формирование в Word чека на форме «Товары»;
* хэширование пароля на формах «Авторизация», «Добавление пользователя», «Редактирование пользователя»;

Входными данными являются: товары, заказы;

Выходными данными являются: чек, отчёт о выручке.

*В дальнейшем требуется доработка следующих функций:*

* для каждой формы реализовать пропорциональное масштабирование всех элементов;
* при неудачной попытке авторизации вылезает captcha;
* на основной форме предусмотреть организацию пагинации;
* реализовать функции резервное копирование, «Импорт» и «Экспорт» данных в других форматах, например .csv;
* при работе с персональными данными предусмотреть функции защиты информации в соответствии с законом «О персональных данных»;
* использовать условное форматирование данных (выделение цветом) для визуального мониторинга данных или контроля ввода данных.