|  |  |  |
| --- | --- | --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики» | | |
| *Факультет социально-экономических и компьютерных наук* | | |
|  | | |
| Кривов Андрей Викторович | | |
|  | | |
| **РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ ПЕРСОНАЛА МАГАЗИНА КОМПЬЮТЕРНЫХ КОМПЛЕКТУЮЩИХ** | | |
| *Курсовая работа* | | |
| студента образовательной программы «Программная инженерия»  по направлению подготовки *09.03.04 Программная инженерия* | | |
|  |  | Руководитель  преподаватель кафедры информационных технологий в бизнесе  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Г. И. Рустамханова |

Пермь, 2023 год

**Аннотация**

Автор: студент НИУ ВШЭ Кривов Андрей Викторович, кафедра информационных технологий в бизнесе.

Тема работы: Разработка информационной системы для персонала магазина компьютерных комплектующих.

Актуальность работы обусловлена…

В первой главе рассматривается… Вторая глава посвящена… В третьей главе описана реализация… Четвертая глава представляет собой…

Работа состоит из N глав, включает N страниц основного текста, N иллюстраций, N таблиц и N приложений. Библиографический список состоит из N источников.

**ОГЛАВЛЕНИЕ**

[Введение 4](#_Toc130308346)

[Глава 1 Анализ задачи и разработка требований 5](#_Toc130308347)

[1.1 Постановка задачи 5](#_Toc130308348)

[1.2 Анализ существующих решений 5](#_Toc130308349)

[Глава 2 Проектирование информационной системы 8](#_Toc130308350)

[2.1 Проектирование базы данных 8](#_Toc130308351)

[2.1.1 Описание атрибутов 8](#_Toc130308352)

[2.1.2 Установление функциональных зависимостей 10](#_Toc130308353)

[2.1.3 Нормализация базы данных 11](#_Toc130308354)

[2.2 Проектирование интерфейса приложения 18](#_Toc130308355)

[2.3 Проектирование отчетов 19](#_Toc130308356)

[2.4 Результаты проектирования 19](#_Toc130308357)

[Глава 3 Реализация информационной системы 20](#_Toc130308358)

[3.1 Выбор инструментальных средств 20](#_Toc130308359)

[3.2 Создание моделей 20](#_Toc130308360)

[3.2.1 Создание модели пользователей 20](#_Toc130308361)

[3.2.2 Создание модели складов 21](#_Toc130308362)

[3.2.3 Создание модели поставщиков 22](#_Toc130308363)

[3.2.4 Создание модели категорий комплектующих 22](#_Toc130308364)

[3.2.5 Создание модели комплектующих 23](#_Toc130308365)

[3.2.6 Создание модели типов характеристик 24](#_Toc130308366)

[3.2.7 Создание модели характеристик 24](#_Toc130308367)

[3.2.8 Создание модели учета комплектующих на складах 25](#_Toc130308368)

[3.2.9 Создание модели поставок 26](#_Toc130308369)

[3.2.10 Создание модели заказов 27](#_Toc130308370)

[3.2.11 Создание модели составов заказов 28](#_Toc130308371)

[3.2.12 Создание модели конфигураций 29](#_Toc130308372)

[3.3 Подключение приложения к базе данных 30](#_Toc130308373)

[3.4 Реализация интерфейса 30](#_Toc130308374)

[3.4.1 Окно авторизации 30](#_Toc130308375)

[3.4.2 Главное меню системы 31](#_Toc130308376)

**Введение**

…

1. **Анализ задачи и разработка требований**

Данная глава обосновывает актуальность проекта, представляет обзор существующих решений информационной системы для магазина компьютерных комплектующих.

* 1. **Постановка задачи**

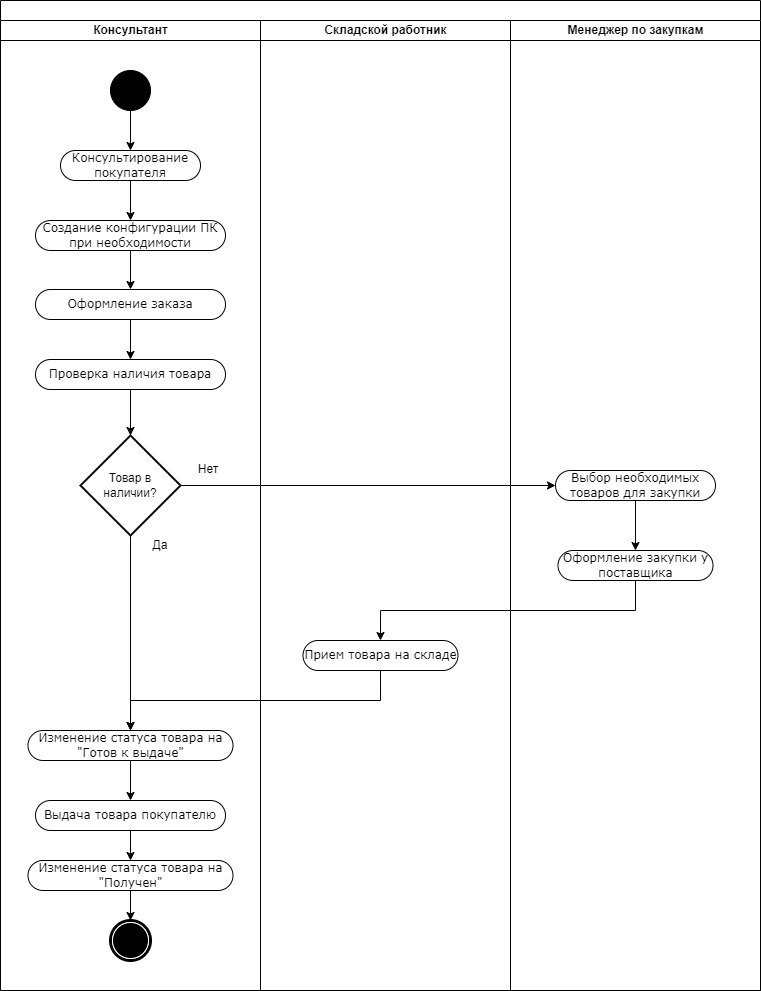
Целью курсовой работы является разработка информационной системы для персонала магазина компьютерных комплектующих с целью упрощения процессов управления продажами и инвентаризацией товаров, оптимизации времени и повышения эффективности работы магазина и персонала. Для достижения этой цели необходимо выполнить следующие задачи:

1. Проанализировать предметную область.
2. Создать модель данных.
3. Спроектировать приложение.
4. Разработать запросы к данным.
5. Спроектировать отчеты.
6. Реализовать информационную систему.
7. Протестировать и отладить приложение.
   1. **Анализ предметной области**

Магазин компьютерных комплектующих – место, где продаются комплектующие для сборки компьютера: процессоры, материнские платы, видеокарты, оперативная память, жесткие диски, корпуса, блоки питания. Возможная организация работы магазина: онлайн, оффлайн или комбинированный вариант.

В магазине могут работать консультанты, аналитики, менеджеры по закупкам, складские работники (при наличии складов), администраторы. У каждого своя роль. Например, консультанты помогают покупателям выбрать нужные комплектующие, а также могут оказывать услуги по сборке и настройке компьютеров. Аналитики проводят анализ полученной информации и проверяют гипотезы, менеджеры по закупкам осуществляют закупки комплектующих у поставщиков, складские работники оформляют поставки и проводят учет товаров, администраторы управляют персоналом.

Описание процессов оформления заказа, закупки комплектующих, учет на складе представлено в виде диаграммы активностей в нотации UML на рисунке 1.1



***Рисунок 1.1 – Диаграмма активностей***

* 1. **Анализ существующих решений**

На сегодняшний день существует множество приложений для управления магазином и учета товаров. Но данные решения, как правило, являются универсальными. Информация система, которая должна быть разработана в рамках данной курсовой работы, является узконаправленной. Она охватывает магазины, продающие только компьютерные комплектующие.

1С:Предприятие – одна из платформ, наиболее распространенных на российском рынке. Для этой платформы существует решение «1С:Розница 8. Магазин бытовой техники и средств связи» для работы со специфическим ассортиментом (бытовая техника, сотовые телефоны, компьютеры и комплектующие) и бизнес-процессами (включая гарантийное обслуживание и обмен по схеме trade-in).

Основные функциональные возможности данного решения:

1. Заказ товаров поставщику.
2. Возврат товаров поставщику.
3. Возврат товаров от покупателей.
4. Сборка товаров.
5. Поступление товаров от контрагента на склад магазина.
6. Перемещение товаров между магазинами.
7. Перемещение товаров между складами магазина.
8. Пересчет товаров (инвентаризация).
9. Списание.
10. Реализация товаров и услуг.
11. Работа с эквайринговыми системами и с банковскими кредитами.
12. Оформление чеков продажи и сводного отчета по ККМ.
13. Доставка товаров.
14. Перемещение денежных средств между кассами магазина.
15. Заказ товаров от покупателя.
16. Поступление денежных средств от покупателей.
17. Прием товара в ремонт, выдача из ремонта.
18. Передача товара в сторонний сервисный центр, возврат товара из стороннего сервисного центра.

На рисунке 1.2 изображено окно оформления поставки, в котором необходимо указать поставщика, магазин, склад, организацию и другие данные. При добавлении товара заполняется артикул, номенклатура, количество товара в выбранной единице измерения, закупочная цена. НДС учитывается.



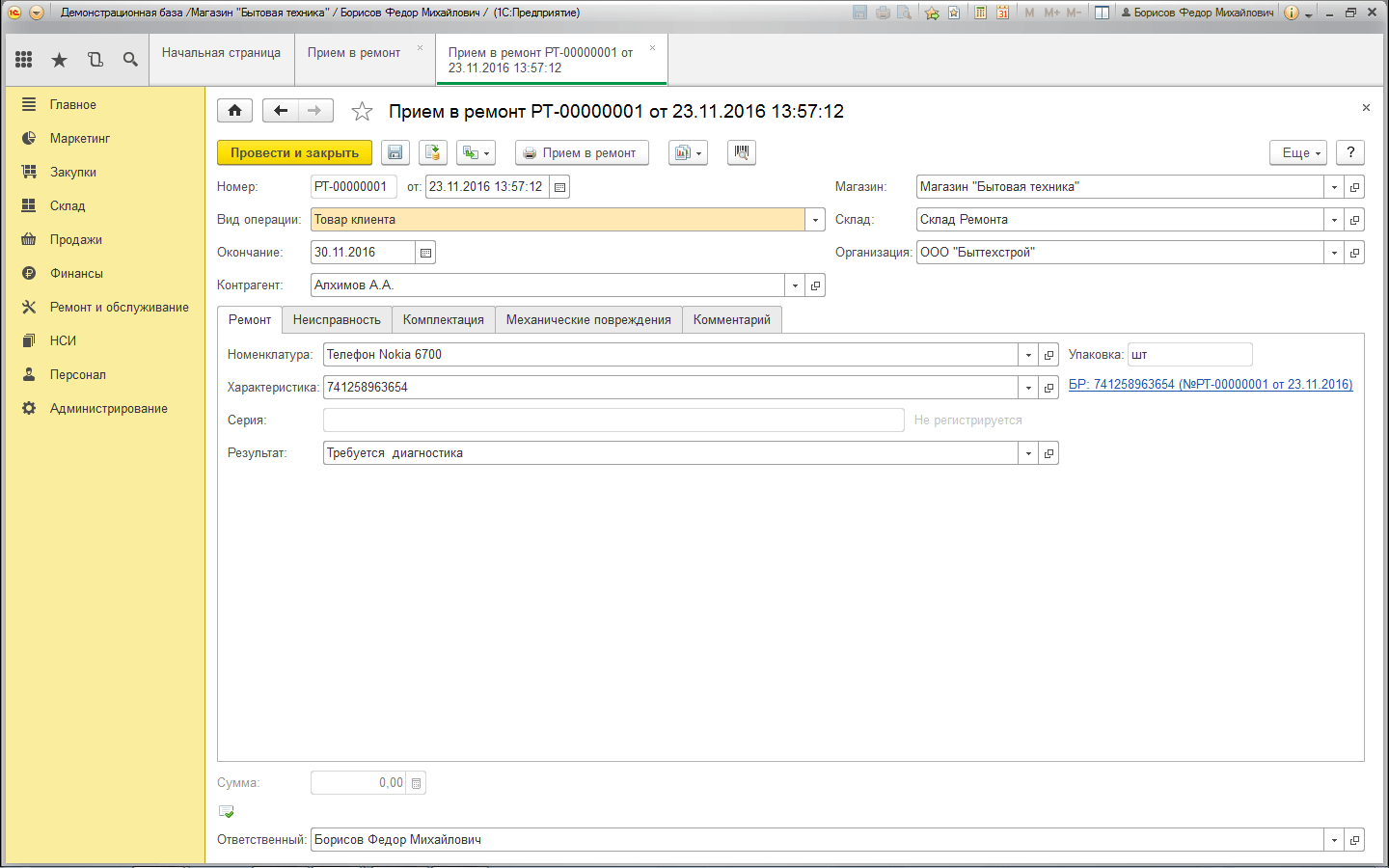
***Рисунок 1.2 – Оформление поступления товара в 1C:Предприятие***

Приложение позволяет генерировать отчеты по продажам, включая визуализацию в виде столбчатой диаграммы по годам (рисунок 1.3).



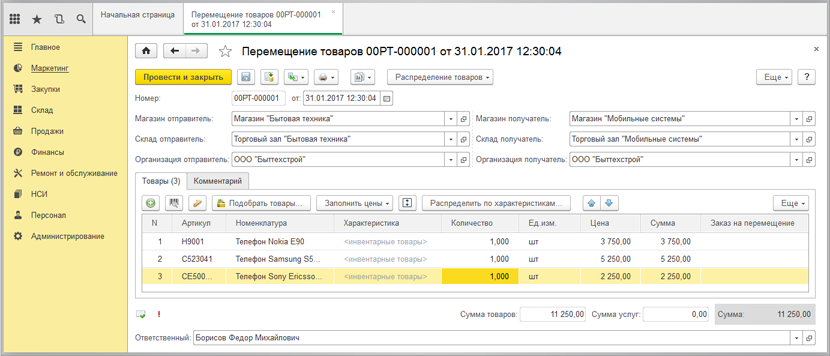
***Рисунок 1.3 – Окно продаж в 1C:Предприятие***

Система учета товаров позволяет осуществлять прием, передачу в сторонний сервисный центр и выдачу подмены товаров, находящихся в ремонте. Используется как для товаров, принадлежащих клиентам, так и для товаров, принадлежащих компании. Таким образом, система обеспечивает полный и точный учет товаров в процессе ремонта (рисунок 1.4).



***Рисунок 1.4 – Окно приема товара в ремонт***

В приложении также доступны две схемы перемещения, реализации и поступления товаров на склады магазина: ордерная и безордерная. Ордерная схема включает перемещение товаров в буферный список, а фактическая операция с товарным остатком производится при помощи расходного или приходного ордера. Для перемещения товаров между магазинами используется безордерная схема, при которой списание товаров с остатком в одном магазине и оприходование их в другом происходит в момент проведения документа «Перемещение товаров». В данном случае дополнительное оформление складских ордеров не требуется. На рисунке 1.5 изображено окно перемещения товаров.



***Рисунок 1.5 – Окно перемещения товаров***

Описанная система представляет собой универсальное решение, предназначенное для управления широким спектром операций, связанных с продажей компьютерных комплектующих. Она обладает множеством функций, необходимых для полноценной и успешной деятельности магазина, таких как управление складами, продажами, финансами и отчетностью. Благодаря многофункциональности данной системы, магазин может эффективно управлять всеми аспектами своей деятельности.

* 1. **Анализ требований к системе**

В системе можно выделить два типа требований: функциональные и нефункциональные. Функциональные требования определяют задачи, которые должна выполнять система. Нефункциональные требования, в свою очередь, определяют свойства и ограничения системы, не связанные с её поведением и не являющиеся задачами, которые система должна выполнять.

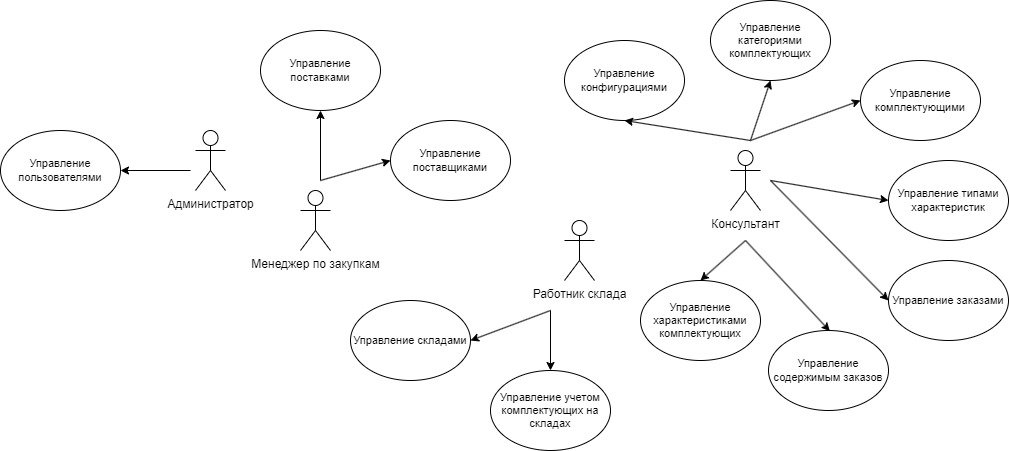
Следует выделить следующие функциональные требования:

1. Добавление, просмотр, изменение и удаление данных пользователей, складов, поставщиков, комплектующих, категорий комплектующих, типов характеристик, характеристик, учета комплектующих, поставок, конфигураций, заказов и их содержимое.
2. Генерирование отчетов для любой информации в формате xlsx.
3. Визуализация долей категорий комплектующих в виде круговой диаграммы, числа продаж и числа поставок за последние 30 дней в виде столбчатой диаграммы.
4. Система должна быть многопользовательской.
5. Каждый пользователь имеет группу прав, которая определяет доступ к определенным ресурсам и операциям.

Нефункциональные требования:

1. Система должна быть устойчивой при вводе некорректных данных.
2. Пароли должны храниться в захэшированном виде.
3. База данных должна быть нормализована до 3НФ.

С учетом рассмотренных требований к информационной системе составлена диаграмма вариантов использования (рисунок 1.6).



***Рисунок 1.6 – Диаграмма вариантов***

|  |  |
| --- | --- |
| ID |  |
| Название |  |
| Акторы |  |
| Краткое описание |  |
| Триггер |  |
| Основной поток |  |
| Альтернативный поток |  |

1. **Проектирование информационной системы**

В процессе проектирования информационной системы для персонала магазина компьютерных комплектующих необходимо решить следующие задачи:

1. Спроектировать базу данных.
2. Выполнить проектирование интерфейса приложения.
3. Спроектировать отчеты.
   1. **Проектирование базы данных**

В данном разделе приведено детальное описание процесса проектирования базы данных для магазина компьютерных комплектующих. Процесс включает следующие этапы:

1. Описание атрибутов, которые будут храниться в базе данных.
2. Установление функциональных зависимостей.
3. Нормализация базы данных до третьей нормальной формы.
   * 1. **Описание атрибутов**

Для выполнения описанных на этапе анализа операций по автоматизации учета компьютерных комплектующих необходимо, чтобы в базе данных хранилась информация, описанная в таблице 2.1.

Таблица 2.1 – Атрибуты базы данных

| **Имя атрибута** | **Тип данных** | **Ограничение на значение атрибута** | **Комментарий** |
| --- | --- | --- | --- |
| Код пользователя | Целое число | Уникально, обязательно для заполнения |  |
| Пароль | Строка символов | Обязательно для заполнения | Хранится в захэшированном (SHA256) виде |
| Дата регистрации | Целое число | Обязательно для заполнения | В формате Unix Timestamp, значение по умолчанию – текущая дата |
| Дата последнего входа | Целое число |  | В формате Unix Timestamp |
| Код группы прав | Целое число | Обязательно для заполнения | 5 групп прав (от 0 до 4) |
| Код склада | Целое число | Уникально, обязательно для заполнения |  |
| Город | Строка символов | Обязательно для заполнения, максимальная длина – 64 символа |  |
| Улица | Строка символов | Обязательно для заполнения, максимальная длина – 64 символа |  |
| Дом | Строка символов | Обязательно для заполнения, максимальная длина – 8 символов |  |
| Индекс | Целое число | Обязательно для заполнения | Маска - 000000 |
| Код поставщика | Целое число | Уникально, обязательно для заполнения |  |
| Название организации поставщика | Строка символов | Уникально, обязательно для заполнения, максимальная длина – 128 символов |  |
| Код категории комплектующих | Целое число | Уникально, обязательно для заполнения |  |
| Название категории комплектующих | Строка символов | Уникально, обязательно для заполнения, максимальная длина – 64 символа |  |
| Код комплектующего | Целое число | Уникально, обязательно для заполнения |  |
| Название модели комплектующего | Строка символов | Уникально, обязательно для заполнения, максимальная длина – 256 символов |  |
| Гарантия комплектующего | Целое число | Обязательно для заполнения, не может быть меньше 1 |  |
| Код типа характеристики | Целое число | Уникально, обязательно для заполнения |  |
| Название типа характеристики | Строка символов | Уникально, обязательно для заполнения, максимальная длина – 64 символа |  |
| Код характеристики | Целое число | Уникально, обязательно для заполнения |  |
| Значение характеристики | Строка символов | Обязательно для заполнения, максимальная длина – 128 символов |  |
| Код хранения комплектующего на складе | Целое число | Уникально, обязательно для заполнения |  |
| Цена комплектующего | Вещественное число | Обязательно для заполнения, не может быть равно или меньше 0 |  |
| Количество комплектующего | Целое число | Обязательно для заполнения, не может быть меньше 1 |  |
| Код поставки | Целое число | Уникально, обязательно для заполнения |  |
| Закупочная цена комплектующего | Вещественное число | Обязательно для заполнения, не может быть равно или меньше 0 |  |
| Дата поставки | Целое число | Обязательно для заполнения | В формате Unix Timestamp |
| Код заказа | Целое число | Уникально, обязательно для заполнения |  |
| Дата заказа | Целое число | Обязательно для заполнения | В формате Unix Timestamp |
| Статус заказа | Целое число | Обязательно для заполнения | 5 статусов (от 0 до 4) |
| Код состава заказа | Целое число | Уникально, обязательно для заполнения |  |
| Часть конфигурации | Логический тип | Обязательно для заполнения |  |
| Код конфигурации | Целое число | Уникально, обязательно для заполнения |  |
| Код части конфигурации | Целое число | Уникально, обязательно для заполнения |  |

* + 1. **Установление функциональных зависимостей**

Рассмотрим следующие функциональные зависимости между атрибутами в каждом отношении:

1. Код пользователя однозначно определяет пароль, дату регистрации, дату последнего входа, группу прав.
2. Код склада однозначно определяет город, улицу, дом, индекс.
3. Код поставщика однозначно определяет название организации поставщика.
4. Код категории комплектующих однозначно определяет название категории комплектующих.
5. Код комплектующего однозначно определяет название модели комплектующего, гарантию комплектующего, код категории комплектующего.
6. Код типа характеристики однозначно определяет название типа характеристики.
7. Код характеристики однозначно определяет код типа характеристики, код комплектующего, значение характеристики.
8. Код хранения комплектующего на складе однозначно определяет цену комплектующего, количество комплектующего, код склада, код комплектующего.
9. Коду поставки однозначно определяет закупочную цену комплектующего, количество комплектующего, дату поставки, код поставщика, код склада, код комплектующего.
10. Код заказа однозначно определяет дату заказа, статус заказа.
11. Код состава заказа однозначно определяет цену комплектующего, количество комплектующего, является ли комплектующее частью конфигурации, код заказа, код склада, код комплектующего.
12. Код части конфигурации однозначно определяет код конфигурации, количество комплектующего, код комплектующего.
    * 1. **Нормализация базы данных**

В данном разделе описывается процесс нормализации базы данных с целью приведения её до третьей нормальной формы. Процесс включает в себя выделение отдельных таблиц и установление связей между ними, что позволяет устранить аномалии ввода, редактирования и удаления.

* + - 1. **Приведение к 1НФ**

Отношение находится в первой нормальной форме, когда все его атрибуты являются атомарными, то есть имеют единственное значение, и все кортежи уникальны, так как содержат первичный ключ.

В исходном универсальном отношении содержатся указанные ниже атрибуты:

1. Код пользователя.
2. Пароль.
3. Дата регистрации.
4. Дата последнего входа.
5. Код группы прав.
6. Код склада.
7. Город.
8. Улица.
9. Дом.
10. Индекс.
11. Код поставщика.
12. Название организации поставщика.
13. Код категории комплектующих.
14. Название категории комплектующих.
15. Код комплектующего.
16. Название модели комплектующего.
17. Гарантия комплектующего.
18. Код типа характеристики.
19. Название типа характеристики.
20. Код характеристики.
21. Значение характеристики.
22. Код хранения комплектующего на складе.
23. Цена комплектующего.
24. Количество комплектующего.
25. Код поставки.
26. Закупочная цена комплектующего.
27. Дата поставки.
28. Код заказа.
29. Дата заказа.
30. Статус заказа.
31. Код состава заказа.
32. Часть конфигурации.
33. Код конфигурации.
34. Код части конфигурации.

Согласно функциональным зависимостям, описанным в разделе 2.1.2, формируется первичный ключ для отношения, включающий следующие атрибуты:

1. Код пользователя.
2. Код склада.
3. Код поставщика.
4. Код категории комплектующих.
5. Код комплектующего.
6. Код типа характеристики.
7. Код характеристики.
8. Код хранения комплектующего на складе.
9. Код поставки.
10. Код заказа.
11. Код состава заказа.
12. Код части конфигурации.

Комбинация этих атрибутов уникальна для каждой строки, что позволяет однозначно идентифицировать запись.

Для обеспечения атомарности требуется, чтобы в каждой строке таблицы содержались все значения указанных атрибутов без группировки данных. Однако это может привести к избыточности данных и нежелательным аномалиям при выполнении операций:

1. Аномалии ввода:

* невозможно ввести данные о заказах, если нет информации о комплектующих;
* невозможно ввести данные о поставках, если нет информации о складах;
* невозможно ввести данные о комплектующих, если нет информации о категориях комплектующих.

1. Аномалии редактирования:

* при изменении информации о комплектующем, потребуется продублировать изменения во всех строках, в которых хранится информация о данном комплектующем;
* при изменении информации о заказе, потребуется продублировать изменения во всех строках, в которых хранится информация о данном заказе.

1. Аномалии удаления:

* при удалении информации о поставках одного поставщика все данные о поставщике могут быть утеряны.

Для их устранения необходимо выполнить декомпозицию отношения в соответствии с алгоритмом нормализации.

* + - 1. **Приведение ко 2НФ**

Отношение находится во второй нормальной форме, если оно удовлетворяет критериям первой нормальной формы и не имеет частичной функциональной зависимости не ключевых атрибутов от ключа, то есть не должно быть не ключевых полей, зависящих от части первичного ключа.

В соответствии с описанными выше зависимостями можно сделать вывод, что отношение уже находится во второй нормальной форме.

* + - 1. **Приведение к 3НФ**

Отношение находится в третьей нормальной форме, если оно удовлетворяет критериям второй нормальной формы, отсутствует функциональная зависимость между не ключевыми атрибутами, то есть не ключевые атрибуты взаимно независимы, и каждый не ключевой атрибут не транзитивно зависит от ключа.

Текущее отношение не находится в третьей нормальной форме, так как по атрибуту «Код пользователя» можно однозначно определить следующие атрибуты:

1. Пароль.
2. Дата регистрации.
3. Дата последнего входа.
4. Группа прав.

Атрибут «Код склада» позволяет однозначно определить следующие атрибуты:

1. Город.
2. Улица.
3. Дом.
4. Индекс.

Атрибут «Код поставщика» позволяет однозначно определить атрибут «Название организации поставщика».

Атрибут «Код категории» позволяет однозначно определить атрибут «Название категории комплектующих».

Атрибут «Код комплектующего» позволяет однозначно определить следующие атрибуты:

1. Название модели комплектующего.
2. Гарантия комплектующего.
3. Код категории комплектующего.

Атрибут «Код типа характеристики» позволяет однозначно определить атрибут «Название типа характеристики».

Атрибут «Код характеристики» позволяет однозначно определить следующие атрибуты:

1. Код типа характеристики.
2. Код комплектующего.
3. Значение характеристики.

Атрибут «Код хранения комплектующего на складе» позволяет однозначно определить следующие атрибуты:

1. Цена комплектующего.
2. Количество комплектующего.
3. Код склада.
4. Код комплектующего.

Атрибут «Код поставки» позволяет однозначно определить следующие атрибуты:

1. Цена комплектующего.
2. Количество комплектующего.
3. Дата поставки.
4. Код поставщика.
5. Код склада.
6. Код комплектующего.

Атрибут «Код заказа» позволяет однозначно определить следующие атрибуты:

1. Дата заказа.
2. Статус заказа.

Атрибут «Код состава заказа» позволяет однозначно определить следующие атрибуты:

1. Цена комплектующего.
2. Количество комплектующего.
3. Часть конфигурации.
4. Код заказа.
5. Код склада.
6. Код комплектующего.

Атрибут «Код части конфигурации» позволяет однозначно определить следующие атрибуты:

1. Код конфигурации.
2. Количество комплектующего.
3. Код комплектующего.

Таким образом, в результате приведения универсального отношения к третьей нормальной форме, должны быть выделены следующие таблицы (подчеркиванием выделены первичные ключи, штриховым подчеркиванием – внешние ключи):

1. Таблица «Пользователи»:

* код пользователя;
* пароль;
* дата регистрации;
* дата последнего входа;
* группа прав.

1. Таблица «Склады»:

* код склада;
* город;
* улица;
* дом;
* индекс.

1. Таблица «Поставщики»:

* код поставщика;
* название организации поставщика.

1. Таблица «Категории комплектующих»:

* код категории комплектующих;
* название категории комплектующих.

1. Таблица «Комплектующие»:

* код комплектующего;
* название модели комплектующего;
* гарантия комплектующего;
* код категории комплектующего.

1. Таблица «Типы характеристик»:

* код типа характеристики;
* название типа характеристики.

1. Таблица «Характеристики»:

* код характеристики;
* код типа характеристики;
* код комплектующего;
* значение характеристики.

1. Таблица «Учет комплектующих на складах»:

* код хранения комплектующего на складе;
* цена комплектующего;
* количество комплектующего;
* код склада;
* код комплектующего.

1. Таблица «Поставки»:

* код поставки;
* закупочная цена комплектующего;
* количество комплектующего;
* дата поставки;
* код поставщика;
* код склада;
* код комплектующего.

1. Таблица «Заказы»:

* код заказа;
* дата заказа;
* статус заказа.

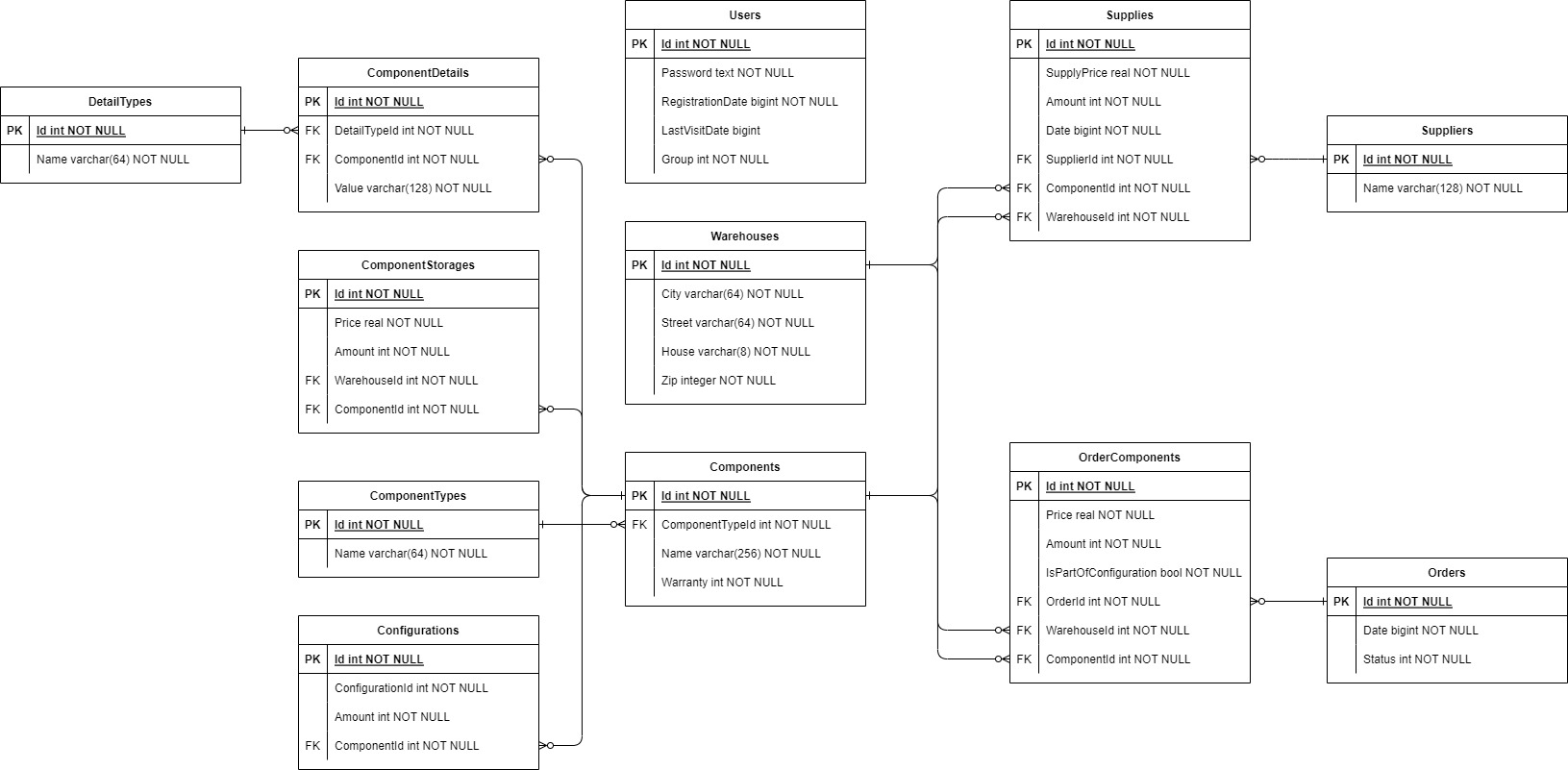
1. Таблица «Состав заказов»:

* код состава заказа;
* цена комплектующего;
* количество комплектующего;
* часть конфигурации;
* код заказа;
* код склада;
* код комплектующего.

1. Таблица «Конфигурации»:

* код части конфигурации;
* код конфигурации;
* количество комплектующего;
* код комплектующего.

На рисунке 2.1 представлена схема базы данных, полученная в результате её проектирования и приведения к третьей нормальной форме.



***Рисунок 2.1 – Схема базы данных***

* 1. **Проектирование интерфейса приложения**

Информация система разрабатывается для персонала, следовательно, в приложении должно быть окно авторизации, включающее в себя следующие элементы пользовательского интерфейса:

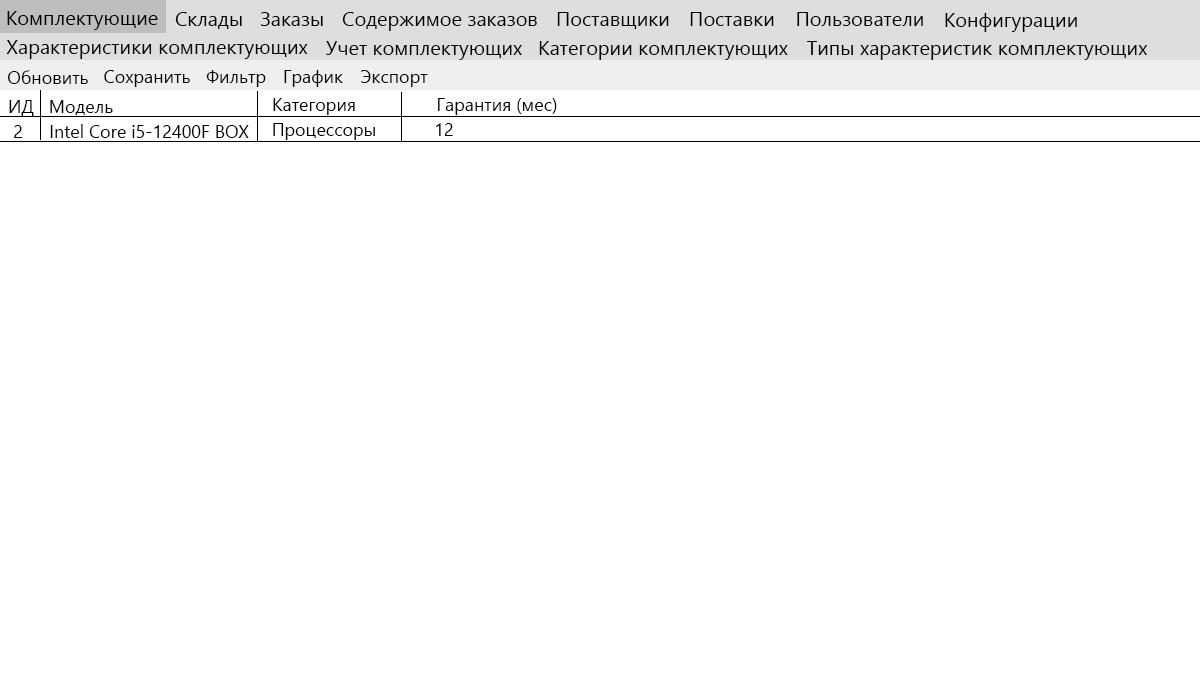
1. Поле для ввода идентификатора сотрудника.
2. Поле для ввода пароля.
3. Кнопка авторизации.

В случае ввода некорректных данных интерфейс должен сообщить об этом пользователю и предоставить возможность ввести их заново. Если данные соответствуют действительности, то должно открыться главное окно, а окно авторизации закрыться. Макет окна авторизации представлен на рисунке 2.2.



***Рисунок 2.2 – Макет окна авторизации***

В главном окне должны быть расположены вкладки, соответствующие таблицам из базы данных (всего 12 таблиц). Взаимодействие (в том числе выполнение CRUD операций) будет происходить через таблицы. В случае с таблицей «Пользователи» добавление и редактирование записей должно осуществляться через диалоговое окно, так как это необходимо для ввода пароля. На каждой вкладке требуется наличие кнопки создания отчета и кнопки фильтрации, если в таблице присутствует больше 1 столбца (помимо столбца с уникальным идентификатором). Макет главного окна представлен на рисунке 2.3.



***Рисунок 2.3 – Макет главного окна***

* 1. **Проектирование отчетов**

Информационная система предлагает возможность генерирования 12 отчетов, столько же, сколько таблиц в базе данных. Отчеты повторяют структуру соответствующих таблиц. Формат отчетов – Excel (расширение .xlsx). Ниже приведен пример - структура отчета «Комплектующие»:

1. ИД.
2. Модель.
3. ИД категории.
4. Название категории.
5. Гарантия (мес).

В структуру некоторых отчетах также входят дополнительные столбцы для предоставления более детальной информации. Например, в отчет комплектующих добавлен столбец «Название категории», в самой таблице комплектующих в базе данных его нет.

* 1. **Выводы по второй главе**

По итогам выполнения этапа проектирования были получены следующие результаты:

1. Выполнено проектирование и нормализация базы данных, в итоге получено 12 таблиц.
2. Спроектирован интерфейс приложения, который состоит из окна авторизации и главного окна, которое включает в себя 12 вкладок для взаимодействия с таблицами из базы данных.
3. Выполнено проектирование отчетов, их общее число равно 12, что соответствует количеству таблиц из базы данных.
4. **Реализация информационной системы**

В данной главе приведено подробное описание процесса реализации информационной системы для персонала магазина компьютерных комплектующих. Глава включает следующие разделы:

1. Выбор инструментальных средств.
2. Создание моделей.
3. Подключение приложения к базе данных.
4. Разработка запросов.
5. Реализация интерфейса.
   1. **Выбор инструментальных средств**

В качестве языка программирования выбран C#, один из самых популярных и наиболее часто используемых в разработке прикладных приложений. Интерфейс построен при помощи UI фреймворка WPF (Windows Presentation Foundation).

База данных хранится в СУБД PostgreSQL, что является отличным решением для малых и средних проектов с низкими минимальными требованиями к производительности и возможностям масштабирования. Помимо этого, PostgreSQL предоставляет большой набор функций и возможностей для работы с данными, что позволяет легко управлять базой данных. Для доступа к таблицам и атрибутам используется ORM фреймворк – Entity Framework Core 7.

* 1. **Создание моделей**

В данном разделе описывается процесс создания классов на языке программирования C#, которые будут соответствовать таблицам из базы данных. Создание таких классов позволит обращаться к базе данных без написания SQL-кода, что упростит разработку и облегчит поддержку системы.

Каждый класс создается с использованием фреймворка Entity Framework Core 7 и подхода «Data Annotations», что позволяет задавать метаданные для модели напрямую в коде класса. Такой подход облегчает процесс создания модели данных и позволяет легко изменять её в соответствии с требованиями системы.

* + 1. **Создание модели «Пользователь»**

Атрибуты таблицы «Пользователи»:

* код пользователя;
* пароль;
* дата регистрации;
* дата последнего входа;
* группа прав.

Для создания модели необходимо написать класс, повторяющий структуру таблицы. Атрибут «Not mapped» означает, что свойство не требуется сохранять в базе данных. На рисунке 3.1 приведен код модели на C#.



***Рисунок 3.1 – Класс «User»***

В класс также добавлены свойства RegistrationDateText и LastVisitDateText, которые используются для вывода даты регистрации и даты последнего входа в текстовом формате.

* + 1. **Создание модели «Склад»**

Атрибуты таблицы «Склады»:

* код склада;
* город;
* улица;
* дом;
* индекс.

Аналогично создаем класс «Warehouse» (см. рисунок 3.2).



***Рисунок 3.2 – Класс «Warehouse»***

Для модели дополнительно указывается индекс, состоящий из трех столбцов: City, Street, House. При этом индекс должен быть уникальным, что исключает возможность дублирования складов с одинаковым адресом. Устанавливается максимальная длина для City (64 символа), Street (64 символа), House (8 символов).

* + 1. **Создание модели «Поставщик»**

Атрибуты таблицы «Поставщики»:

* код поставщика;
* название организации поставщика.

Представим таблицу поставщиков в виде класса «Supplier» на C# (см. рисунок 3.3).



***Рисунок 3.3 – Класс «Supplier»***

На свойство «Name» наложено ограничение: максимальная длина равна 128. Кроме того, наименование организации должно быть уникальным.

* + 1. **Создание модели «Категория комплектующих»**

Атрибуты таблицы «Категории комплектующих»:

* код категории комплектующих;
* название категории комплектующих.

Представим таблицу категорий комплектующих в виде класса «ComponentType» на C# (см. рисунок 3.4).



***Рисунок 3.4 – Класс «ComponentType»***

На свойство «Name» наложено ограничение: максимальная длина равна 64. Кроме того, название категории должно быть уникальным.

* + 1. **Создание модели «Комплектующее»**

Атрибуты таблицы «Комплектующие»:

* код комплектующего;
* название модели комплектующего;
* гарантия комплектующего;
* код категории комплектующего.

Представим таблицу комплектующих в виде класса «Component» на C# (см. рисунок 3.5).



***Рисунок 3.5 – Класс «Component»***

На свойство «Name» наложено ограничение на максимальное число символов – 256. Кроме того, из данного класса можно обратиться к характеристикам, заказам, конфигурациям, поставкам и информации о хранении заданного комплектующего.

* + 1. **Создание модели «Тип характеристик»**

Атрибуты таблицы «Типы характеристик»:

* код типа характеристики;
* название типа характеристики.

Представим таблицу типов характеристик в виде класса «DetailType» на C# (см. рисунок 3.6).



***Рисунок 3.6 – Класс «DetailType»***

На «Name» наложено ограничение – значения данного свойства должны быть уникальными.

* + 1. **Создание модели «Характеристика»**

Атрибуты таблицы «Характеристики»:

* код характеристики;
* код типа характеристики;
* код комплектующего;
* значение характеристики.

Представим таблицу характеристик в виде класса «ComponentDetail» на C# (см. рисунок 3.7).



***Рисунок 3.7 – Класс «ComponentDetail»***

На значения данного класса наложено два ограничения. Первое ограничение – пара значений «DetailTypeId» и «ComponentId» должна быть уникальной, второе – максимальная длина значения характеристики не должна превышать 128 символов.

* + 1. **Создание модели «Учет комплектующих на складах»**

Атрибуты таблицы «Учет комплектующих на складах»:

* код хранения комплектующего на складе;
* цена комплектующего;
* количество комплектующего;
* код склада;
* код комплектующего.

Представим таблицу учета комплектующих на складах в виде класса «ComponentStorage» на C# (см. рисунок 3.8).



***Рис 3.8 – Класс «ComponentStorage»***

Из данного класса можно обратиться к комплектующему и складу, на котором он хранится.

* + 1. **Создание модели «Поставка»**

Атрибуты таблицы «Поставки»:

* код поставки;
* закупочная цена комплектующего;
* количество комплектующего;
* дата поставки;
* код поставщика;
* код склада;
* код комплектующего.

Представим таблицу поставок в виде класса «Supply» на C# (см. рисунок 3.9).



***Рисунок 3.9 – Класс «Supply»***

Таблица «Поставки» связана с таблицами «Поставщики», «Комплектующие», «Склады» отношением многие-к-одному. Поэтому к ним можно обратиться напрямую из класса «Supply» через свойства «Supplier», «Component» и «Warehouse».

* + 1. **Создание модели «Заказ»**

Атрибуты таблицы «Заказы»:

* код заказа;
* дата заказа;
* статус заказа.

Представим таблицу заказов в виде класса «Order» на C# (см. рисунок 3.10).



***Рисунок 3.10 – Класс «Order»***

К составу заказа можно обратиться через свойство «OrderComponents», так как таблицы «Order» и «OrderComponents» связаны отношением один-ко-многим.

* + 1. **Создание модели составов заказов**

Атрибуты таблицы «Состав заказов»:

* код состава заказа;
* цена комплектующего;
* количество комплектующего;
* часть конфигурации;
* код заказа;
* код склада;
* код комплектующего.

Представим таблицу состава заказов в виде класса «OrderComponent» на C# (см. рисунок 3.11).



***Рисунок 3.11 – Класс «OrderComponent»***

Через данный класс можно получить информацию о комплектующем, выбранном складе, заказе через соответствующие свойства «Component», «Warehouse» и «Order».

* + 1. **Создание модели конфигураций**

Атрибуты таблицы «Конфигурации»:

* код части конфигурации;
* код конфигурации;
* количество комплектующего;
* код комплектующего.

Представим таблицу конфигураций в виде класса «Configuration» на C# (см. рисунок 3.12).



***Рисунок 3.12 – Класс «Configuration»***

В классе «Configuration» пара значений «ConfigurationId» и «ComponentId» должна быть уникальной.

* 1. **Подключение приложения к базе данных**

Взаимодействие программы с базой данных реализуется через библиотеку Entity Framework. В разделе 3.2 были рассмотрены модели, которые представлены в виде таблиц в СУБД, но для возможности работы с ними необходимо определить класс «StoreContext», который наследуется от «DbContext».

Каждая коллекция имеет тип DbSet<T>, где T – название класса модели (например, User или Component). В конструкторе вызывается метод «EnsureCreated», который используется для создания базы данных, связанных таблиц и других объектов хранения. В методе «OnConfiguring» вызывается функция «UseNpgsql» с указанием данных для подключения к СУБД. Код класса «StoreContext» приведен в приложении.

* 1. **Разработка запросов**

В разработанной программе используется технология ORM Entity Framework, поэтому в коде программы отсутствует явное применение SQL запросов. Также реализация CRUD-операций (за исключением реализации CRUD для таблицы пользователей), фильтра, группировки данных не требует вызова специальных методов Entity Framework (Where, Add, Remove и других), они реализуются за счет встроенных механизмов WPF и Entity Framework.

Кроме того, Entity Framework применяется при визуализации данных, выполнении CRUD-операций для таблицы пользователей и при экспортировании данных в Excel.

Для получения количества комплектующих каждой категории используется метод «GroupBy», вызов выглядит следующим образом: \_context.Components.GroupBy(c => c.ComponentTypeId).Select(c => new { categoryTypeId = c.Key, count = c.Count() }).ToList(). Этот код можно записать на SQL: “SELECT id, COUNT(\*) FROM Components GROUP BY ComponentTypeId”.

Чтобы получить список заказов за последние 30 дней по возрастанию даты для построения графика, можно сделать следующий вызов: \_context.Orders.Where(o => timeStampLastMoth <= o.Date && o.Date <= Utilities.GetCurrentUnixTimeStamp()).OrderBy(o => o.Date).ToList(). Аналогично для числа продаж, если заменить «Orders» на «Supplies». Функция «GetCurrentUnixTimeStamp» возвращает текущее время в формате unix timestamp.

Выполнение добавления пользователя выглядит следующим образом:

1. Создается объект класса «User», заполняется данными.
2. Вызывается метод Add(user), где user – созданный объект.

В случае изменения данных пользователя – производится поиск с помощью метода Where, после старые данные заменяются новыми. Для удаления необходимо вызвать метод Remove с указанием нужного объекта.

Результаты запросов проиллюстрированы в следующем разделе.

* 1. **Реализация интерфейса**

В данном разделе рассматривается процесс реализации интерфейса, для которого был разработан макет в ходе проектирования, для настольного приложения на языке программирования C# с использованием библиотеки WPF.

* + 1. **Окно авторизации**

При каждом запуске программы пользователю системы необходимо авторизоваться, для чего открывается соответствующее окно (рисунок 3.13).

Окно авторизации включает поля для ввода логина и пароля и кнопку «Войти». Логином является идентификатор сотрудника магазина. В случае ввода некорректных данных пользователь получает сообщение об ошибке авторизации. Если логин и пароль верны, то окно авторизации закрывается и открывается главное меню.



***Рисунок 3.13 – Окно авторизации***

* + 1. **Главное меню системы**

После авторизации пользователю открывается главное меню (рисунок 3.14), в котором происходит взаимодействие с таблицами из базы данных.

Главное меню состоит из трех частей:

1. Вкладки.
2. Панель инструментов.
3. Таблица данных.

Количество вкладок соответствует количеству таблиц в базе данных. Общее их число равно 12. Некоторые вкладки могут быть недоступны в зависимости от группы прав пользователя.

Набор кнопок в панели инструментов зависит от выбранной вкладки. Это будет рассмотрено в данной главе.

Таблица данных выбирается через нажатие на определенную вкладку. Например, при нажатии на вкладку «Комплектующие» выведется таблица комплектующих и соответствующая панель инструментов.



***Рисунок 3.14 – Главное меню***

Все таблицы в программе реализованы по единому принципу. В фреймворке WPF имеется элемент управления – DataGrid, позволяющий создавать таблицы данных. С помощью DataGrid можно быстро и эффективно организовать выполнение CRUD-операций, и также связать его с базой данных. Для этого с помощью Entity Framework выполняется загрузка определенной таблицы в представление. Далее с помощью свойства DataGrid «ItemsSource» можно указать ссылку на представление. После этого таблица будет связана с базой данных.

* + - 1. **Вкладка «Пользователи»**

На рисунке 3.15 изображена таблица пользователей. Для взаимодействия с ней доступны следующие инструменты:

1. Обновить (загрузить данные из базы данных заново).
2. Сохранить изменения, сделанные в таблице.
3. Создать нового пользователя.
4. Изменить данные существующего пользователя.
5. Удалить полностью информацию о пользователе.
6. Экспортировать данные обо всех пользователях в файл формата .xlsx.



***Рисунок 3.15 – Вкладка «Пользователи»***

При нажатии на кнопку «Создать» открывается окно «HSMS – Добавить пользователя» (рисунок 3.16). Для добавления пользователя необходимо указать пароль и его группу прав. Его идентификатор будет показан после нажатия на кнопку «ОК».



***Рисунок 3.16 – Окно добавления пользователя***

При нажатии на кнопку «Изменить» открывается окно «HSMS – Изменить данные пользователя» (рисунок 3.17). Для изменения данных необходимо ввести ИД пользователя и заполнить нужные поля. Если пароль не требуется изменять, то поле следует оставить пустым.



***Рисунок 3.17 – Окно изменения данных пользователя***

При нажатии на кнопку «Удалить» открывается окно «HSMS – Удалить пользователя», в котором необходимо указать ИД пользователя и нажать на кнопку «ОК» для выполнения удаления.



***Рисунок 3.18 – Окно удаления пользователя***

* + - 1. **Вкладка «Склады»**

На рисунке 3.19 изображена таблица складов. Для взаимодействия с ней доступны следующие инструменты:

1. Обновить (загрузить данные из базы данных заново).
2. Сохранить изменения, сделанные в таблице.
3. Отфильтровать данные по разным параметрам.
4. Экспортировать данные обо всех складах в файл формата .xlsx.

Выполнение CRUD-операций происходит только через таблицу.



***Рисунок 3.19 – Вкладка «Склады»***

При нажатии на кнопку «Фильтр» открывается окно «HSMS – Фильтр складов» (рисунок 3.20). Необходимо заполнить те поля, по которым должен осуществляться фильтр данных. Остальные нужно оставить пустыми.



***Рисунок 3.20 – Окно фильтра складов***

* + - 1. **Вкладка «Поставщики»**

На рисунке 3.21 изображена таблица поставщиков. Для взаимодействия с ней доступны следующие инструменты:

1. Обновить (загрузить данные из базы данных заново).
2. Сохранить изменения, сделанные в таблице.
3. Экспортировать данные обо всех поставщиках в файл формата .xlsx.

Выполнение CRUD-операций происходит только через таблицу.



***Рисунок 3.21 – Вкладка «Поставщики»***

* + - 1. **Вкладка «Категории комплектующих»**

На рисунке 3.22 изображена таблица категорий комплектующих. Для взаимодействия с ней доступны следующие инструменты:

1. Обновить (загрузить данные из базы данных заново).
2. Сохранить изменения, сделанные в таблице.
3. Экспортировать данные обо всех категориях комплектующих в файл формата .xlsx.

Выполнение CRUD-операций происходит только через таблицу.



***Рисунок 3.22 – Вкладка «Категории комплектующих»***

* + - 1. **Вкладка «Комплектующие»**

На рисунке 3.23 изображена таблица комплектующих. Для взаимодействия с ней доступны следующие инструменты:

1. Обновить (загрузить данные из базы данных заново).
2. Сохранить изменения, сделанные в таблице.
3. Отфильтровать данные по разным параметрам.
4. Построить график.
5. Экспортировать данные обо всех комплектующих в файл формата .xlsx.

Выполнение CRUD-операций происходит только через таблицу.



***Рисунок 3.23 – Вкладка «Комплектующие»***

При нажатии на кнопку «Фильтр» открывается окно «HSMS – Фильтр комплектующих» (рисунок 3.24). Необходимо заполнить те поля, по которым должен осуществляться фильтр данных. Остальные нужно оставить пустыми.



***Рисунок 3.24 – Окно фильтра комплектующих***

При нажатии на кнопку «График» открывается окно с круговой диаграммой, на которой изображены доли категорий комплектующих (рисунок 3.25).



***Рисунок 3.25 – График доли категорий комплектующих***

* + - 1. **Вкладка «Типы характеристик комплектующих»**

На рисунке 3.26 изображена таблица типов характеристик комплектующих. Для взаимодействия с ней доступны следующие инструменты:

1. Обновить (загрузить данные из базы данных заново).
2. Сохранить изменения, сделанные в таблице.
3. Экспортировать данные обо всех типах характеристик в файл формата .xlsx.

Выполнение CRUD-операций происходит только через таблицу.



***Рисунок 3.26 – Вкладка «Типы характеристик комплектующих»***

* + - 1. **Вкладка «Характеристики комплектующих»**

На рисунке 3.27 изображена таблица характеристик комплектующих. Для взаимодействия с ней доступны следующие инструменты:

1. Обновить (загрузить данные из базы данных заново).
2. Сохранить изменения, сделанные в таблице.
3. Отфильтровать данные по разным параметрам.
4. Экспортировать данные обо всех характеристиках комплектующих в файл формата .xlsx.

Выполнение CRUD-операций происходит только через таблицу.



***Рисунок 3.27 – Вкладка «Характеристики комплектующих»***

При нажатии на кнопку «Фильтр» открывается окно «HSMS – Фильтр характеристик комплектующих» (рисунок 3.28). Необходимо заполнить те поля, по которым должен осуществляться фильтр данных. Остальные нужно оставить пустыми.



***Рисунок 3.28 – Окно фильтра характеристик комплектующих***

* + - 1. **Вкладка «Учет комплектующих»**

На рисунке 3.29 изображена таблица учета комплектующих. Для взаимодействия с ней доступны следующие инструменты:

1. Обновить (загрузить данные из базы данных заново).
2. Сохранить изменения, сделанные в таблице.
3. Отфильтровать данные по разным параметрам.
4. Экспортировать данные об учете комплектующих в файл формата .xlsx.

Выполнение CRUD-операций происходит только через таблицу.



***Рисунок 3.29 – Вкладка «Учет комплектующих»***

При нажатии на кнопку «Фильтр» открывается окно «HSMS – Фильтр учета комплектующих на складах» (рисунок 3.30). Необходимо заполнить те поля, по которым должен осуществляться фильтр данных. Остальные нужно оставить пустыми.



***Рисунок 3.30 – Окно фильтра учета комплектующих на складах***

* + - 1. **Вкладка «Поставки»**

На рисунке 3.31 изображена таблица поставок. Для взаимодействия с ней доступны следующие инструменты:

1. Обновить (загрузить данные из базы данных заново).
2. Сохранить изменения, сделанные в таблице.
3. Отфильтровать данные по разным параметрам.
4. Построить график.
5. Экспортировать данные обо всех поставках в файл формата .xlsx.

Выполнение CRUD-операций происходит только через таблицу.



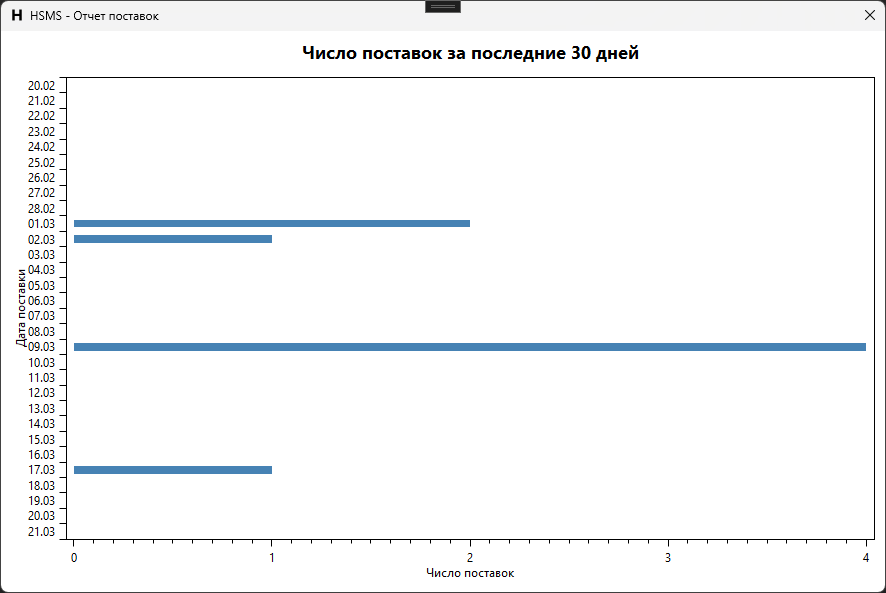
***Рисунок 3.31 – Вкладка «Поставки»***

При нажатии на кнопку «Фильтр» открывается окно «HSMS – Фильтр поставок» (рисунок 3.32). Необходимо заполнить те поля, по которым должен осуществляться фильтр данных. Остальные нужно оставить пустыми.



***Рисунок 3.32 – Окно фильтра поставок***

При нажатии на кнопку «График» открывается окно со столбчатой диаграммой, на которой изображено число поставок за последние 30 дней (рисунок 3.33).



***Рисунок 3.33 – График числа поставок за последние 30 дней***

* + - 1. **Вкладка «Заказы»**

На рисунке 3.34 изображена таблица заказов. Для взаимодействия с ней доступны следующие инструменты:

1. Обновить (загрузить данные из базы данных заново).
2. Сохранить изменения, сделанные в таблице.
3. Отфильтровать данные по разным параметрам.
4. Построить график.
5. Экспортировать данные обо всех заказах в файл формата .xlsx.

Выполнение CRUD-операций происходит только через таблицу.



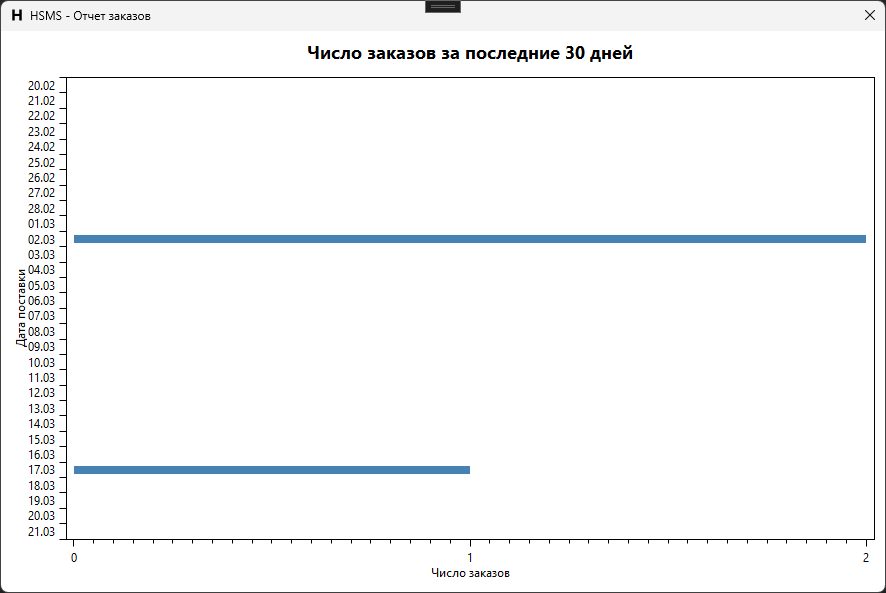
***Рисунок 3.34 – Вкладка «Заказы»***

При нажатии на кнопку «Фильтр» открывается окно «HSMS – Фильтр заказов» (рисунок 3.35). Необходимо заполнить те поля, по которым должен осуществляться фильтр данных. Остальные нужно оставить пустыми.



***Рисунок 3.35 – Окно фильтра заказов***

При нажатии на кнопку «График» открывается окно со столбчатой диаграммой, на которой изображено число заказов за последние 30 дней (рисунок 3.36).



***Рисунок 3.36 – График числа заказов за последние 30 дней***

* + - 1. **Вкладка «Содержимое заказов»**

На рисунке 3.37 изображена таблица содержимого заказов. Для взаимодействия с ней доступны следующие инструменты:

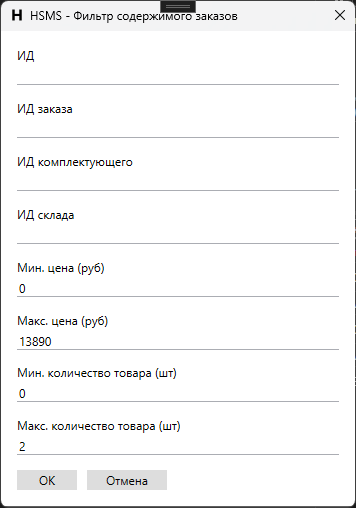
1. Обновить (загрузить данные из базы данных заново).
2. Сохранить изменения, сделанные в таблице.
3. Отфильтровать данные по разным параметрам.
4. Экспортировать данные о содержимом всех заказов в файл формата .xlsx.

Выполнение CRUD-операций происходит только через таблицу.



***Рисунок 3.37 – Вкладка «Содержимое заказов»***

При нажатии на кнопку «Фильтр» открывается окно «HSMS – Фильтр содержимого заказов» (рисунок 3.38). Необходимо заполнить те поля, по которым должен осуществляться фильтр данных. Остальные нужно оставить пустыми.



***Рисунок 3.38 – Окно фильтра содержимого заказов***

* + - 1. **Вкладка «Конфигурации»**

На рисунке 3.39 изображена таблица конфигураций. Для взаимодействия с ней доступны следующие инструменты:

1. Обновить (загрузить данные из базы данных заново).
2. Сохранить изменения, сделанные в таблице.
3. Отфильтровать данные по разным параметрам.
4. Экспортировать данные обо всех конфигурациях в файл формата .xlsx.

Выполнение CRUD-операций происходит только через таблицу.



***Рисунок 3.39 – Вкладка «Конфигурации»***

При нажатии на кнопку «Фильтр» открывается окно «HSMS – Фильтр конфигураций» (рисунок 3.40). Необходимо заполнить те поля, по которым должен осуществляться фильтр данных. Остальные нужно оставить пустыми.



***Рисунок 3.40 – Окно фильтра конфигураций***

* 1. **Выводы по третьей главе**

По итогам выполнения этапа реализации были получены следующие результаты:

1. Выбраны инструментальные средства: приложение на языке программирования C# с использованием фреймворков WPF и Entity Framework, СУБД – PostgreSQL.
2. Созданы модели для взаимодействия с таблицами из базы данных.
3. Выполнена настройка подключения к СУБД.
4. Разработаны запросы для работы с базой данных.
5. Реализован интерфейс.

**Заключение**

Здесь будет текст…

**Библиографический список**

Здесь будет библиографический список…