**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

***Факультет Информационных технологий***

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**КУРСОВОЙ ПРОЕКТ**

**Дисциплина:** \_\_\_\_\_\_\_\_Объектно-ориентированное программирование (С#)\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Тема:** \_\_\_\_Информационная система управления товарами на складе\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Выполнил(а): студент(ка) группы \_\_221-371\_\_**

\_\_\_\_\_\_\_\_\_\_Куренков Алексей Олегович\_\_\_\_\_\_\_\_\_\_

(Фамилия И.О.)

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Проверил: \_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

(Фамилия И.О., степень, звание)

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Москва**

**2023**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

***Факультет Информационных технологий***

***Кафедра Информатики и информационных технологий***

**\_\_\_\_\_\_\_\_\_\_\_\_Утверждаю**

**Зав. кафедрой Е.В. Булатников**

**« \_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.**

**ЗАДАНИЕ**

**на курсовое проектирование**

**Направление 09.03.02 «Информационные системы и технологии»**

**Группа: 221-371**

**Дисциплина:**\_\_\_\_\_\_ Объектно-ориентированное программирование (С#)\_\_\_\_\_\_

**Студент: \_\_\_**Куренков Алексей Олегович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**1. Тема проекта: \_\_\_\_\_** Информационная система управления товарами на складе\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2. Срок сдачи студентом законченного проекта: \_\_22** Декабря 2023 года**\_\_\_\_\_\_**

**3. Исходные данные к проекту:\_** Язык программирования: C# , Microsoft Visual Studio,Фреймворк: .NET Framework \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**4. Содержание расчетно-пояснительной записки (перечень подлежащих**

**разработке вопросов): Разработка модели данных, Описание функциональности и основных требований к приложению \_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**5. Перечень графического материала (с точным указанием обязательных**

**чертежей):\_\_\_\_\_**Листинг программы, скриншоты этапов программирования**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**6. Литература и прочие материалы, рекомендуемые студенту для изучения:**

**Джозев, Ал. and Aл. Бен, 2021. C# 9.0. Карманный справочник. Диалектика, стр: 256; Марк, Пр., 2023. C# 10 и.NET 6. Современная кросс-платформенная разработка. Питер, стр: 848; Рихтер, Дж., 2023. CLR via C#. Программирование на платформе Microsoft. NET Framework 4.5 на языке C#. 4-е изд.. Питер, стр: 896; Болье, А., 2021. Изучаем SQL. Генерация, выборка и обработка данных. Диалектика, стр: 400; Маркин, А. В. Программирование на sql в 2 ч. Часть 2 : учебник и практикум для бакалавриата и магистратуры / А. В. Маркин. — М. : Издательство Юрайт, 2019. — 292 с; Литвиненко, Н.А., 2022. Декларативное программирование на языке XAML + C#. WPF проекты. Горячая линия - Телеком, стр: 320.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**7. Промежуточные сроки исполнения \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**8. Дополнительные указания \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**9. Дата выдачи задания \_\_\_7\_Сентября 2023\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*Руководитель проекта: \_\_* *Полубояринова Анастасия Сергеевна \_\_\_\_\_\_\_\_\_\_\_\_*

*Задание принял студент:\_Куренков Алексей Олегович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 5](#_Toc153801911)

[**1** **АНАЛИТИЧЕСКАЯ ЧАСТЬ** 6](#_Toc153801912)

[**1.1** **Описание функциональности и основных требований к приложению** 6](#_Toc153801913)

[**1.2** **Разработка модели данных** 7](#_Toc153801914)

[**1.3** **Выбранные технологии** 11](#_Toc153801915)

[**2** **ПРОЕКТИРОВАНИЕ** 14](#_Toc153801916)

[**2.1** **Разработка функций запроса к бд** 14](#_Toc153801918)

[**2.2** **Реализация вывода формы ввода данных** 17](#_Toc153801919)

[**3** **ПРАКТИЧЕСКАЯ ЧАСТЬ** 20](#_Toc153801921)

[**3.1** **Разработка баз данных** 20](#_Toc153801922)

[**3.2** **Создание окна Авторизации** 21](#_Toc153801923)

[**3.3** **Создание основного функционала приложения** 22](#_Toc153801924)

[**4** **ТЕСТИРОВАНИЕ** 25](#_Toc153801925)

[**ЗАКЛЮЧЕНИЕ** 28](#_Toc153801926)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 30](#_Toc153801927)

# **ВВЕДЕНИЕ**

**Система управления складом** – это мощный программный инструментарий, предназначенный для автоматизации управления процессами склада. Система управления складом помогает уменьшить трудозатраты, сократить ошибки при отборе и отгрузке заказов, упростить проведение инвентаризации и повысить ее точность, улучшить качество обслуживания клиентов.

Актуальность данной темы заключается в разработке простой, но структурированной системой управления товаров на складе. В ней должно четко прописана логическая структура данных, поскольку такая модель позволяет легко и быстро получать, обрабатывать и выдавать нужные нам результаты запросов из склада к сотруднику предприятия.

Для этого необходимо выявить следующие задачи:

1. Разработать логику модели данных под товары, склад и сотрудников
2. Управление данных склада предприятия, а конкретно: добавление, обновление и удаление.
3. Добавление роли. Пользователи разных должностей могут работать с разными типами данных, поэтому должно быть четкое распределение доступа к данным.
4. Безопасность. Любое приложение ограничивается доступом к данным предприятия при помощи входа в систему через зарегистрированные аккаунты. Это делается в целях защиты от кражи данных, которые могут повлиять на работу предприятия в целом.

# **АНАЛИТИЧЕСКАЯ ЧАСТЬ**

## **Описание функциональности и основных требований к приложению**

В рамках данной работы, основной целью курсовой работы заключается в *разработки системы управления товарами на складе*. Необходимо будет управлять данными разных типов товаров, а конкретно – техника и книги. Кроме того, требуется все эти товары где-то хранить в некотором кол-ве. Не стоит забывать, что за всем этим отвечают сотрудники, у которых есть определенный уровень доступа управлению данных. Кроме того, может быть вариант того, что сотрудник только начал исполнять обязанности в организации и для создания нового аккаунта потребуется определенное время. Поэтому должна быть возможность просмотра некоторой информации в роли «Гость». Все данные должны храниться в некотором месте, с возможностью в любой момент из данного хранилища взять, обработать, записать данные. С точки зрения функциональности работы, необходимо не только разработать связь с неким хранилищем данных, но и спроектировать все функции программы посредством Объектно-Ориентированного программирования (ООП), с описанием каждой модели данных системы управления товарами. Далее нужно продумать проверку безопасности входа пользователя в систему. Необходимо будет проверять данные на их наличия в хранилище данных и просматривать правильность ввода пароля в систему.

В сумме все это можно вкратце выделить основные требования к приложению:

* Создать общую концепцию модели данных
* Создания базы данных
* Управление данными: Запись, просмотр, изменение.
* Разработка авторизации
* Проектирование функционала на основе принципа ООП
* Безопасность входа
* Роли

Описанием функциональности будет исходить из задач. Например, под каждую модель необходимо будет продумать свои свойства и методы работы с данными. Кроме этого, также потребуется написать некоторые функции и передачи запросов обращения из приложения в хранилище данных для последующей манипуляции с ними. Разработать отдельные компоненты под безопасность приложения, которая по большей части будет направлена на проверку корректности ввода данных. Все эти компоненты необходимо будет реализовать по четкому, структурированному принципу работы с кодом клинской части приложения.

## **Разработка модели данных**

Перед тем, как начать разрабатывать приложение и базу данных, потребуется продумать логическую модель данных, которую будет использовать в последующем. Это достаточно важный момент, поскольку в случае возникновения ошибок на поздних этапах разработки, которые будут связаны с фундаментом, придется всю разработку свернуть и вернуться к начальному этапу, что сильно скажется на предприятии в худшую сторону. Для этого приложения потребуется продумать несколько небольших моделей данных под каждый конкретный объект, а именно:

* Склад. В него будут храниться все наши товары на складе. Причем они могут храниться в нескольких секторах. В каждом секторе может быть некоторое кол-во товаров. Ко всему этому, за каждый склад отвечает конкретный сотрудник предприятия.
* Сотрудники. В любом предприятии всегда имеется сотрудники, который отвечает за ту или иную задачу. В его параметры входят: ФИО, дата рождения, зарплата, должность. Кроме этого, сотрудники имеют персональные аккаунты соответствующими со своими должностями управлением складом.
* Товары. На текущий момент товары имеют два основных типов: Книги и Техника. У каждого типа есть свои особенности и представления данных.

Теперь следует остановиться на каждом из них. Для начала стоит разобраться с сотрудниками. Выше уже были описаны их основные свойства, однако их надо стоит разбить до атомарных данных, где каждый отдельный вид информации описывает конкретное назначение. В итоги мы будем иметь:

* Фамилия
* Имя
* Отчество
* Дата рождения,
* Зарплата

Должность (или роль) будет выноситься под другую таблицу. Это из-за того, что у нас данные должны быть не только атомарными и простыми, но и уникальными. Сотрудников в предприятии могут быть несколько, имеющий одну и ту же должность, в следствии чего, данные будут повторяться. Хоть это не сильно будет влиять на модель, однако без разбиения, наша база данных будет занимать больше места. Поэтому необходимо один раз прописать все должности сотрудников отдельно и привязать их к нашим сотрудникам. В итоге у нас получается две модели: сотрудники и их должности. Не стоит забывать, что у них есть аккаунты, нацелены на безопасность от чужого проникновения в систему. Поэтому третья модель будет нацелена на описание аккаунтов сотрудников, состоящий из двух основных полей: Логин и Пароль. Вот и получается у нас три модели для сотрудников: Описательная, аккаунты и должности. Нам потребуется лишь создать таблицу со связями всех этих моделей. Это шаг нам будет важен во время разработки баз данных, ибо вся информация в моделях более общая. Данная таблица будет у нас лишь хранить номера строк, который подойдет под каждого сотрудника.

Следующим шагом будет разработка модели товаров. Товары имеют двух типов: **Книги, Техника**. Причем, у этих товаров есть общие параметры, такие как:

* Имя товара
* Артикул
* Описание
* Цена

Поэтому во время разработки проекта потребуется создать целых 3 модели. Одна из них будет описывать более общие свойства всех товаров, а следом уже будут идти типы этих параметров со своими уникальными чертами.

Для Книг будут уникальные свойства: автор и страницы. Для Техники: серия, система, количество ядер процессора, память, год производства.

Последним штрихом будет модели склада. Их будет две, поскольку у нас товары будут разделяться под два разных типа, для удобства хранения и быстрого получения доступа к ним. Каждый склад разделен на определённые зоны и их количество товаров в этих зонах может различаться. Более того, под каждый склад выделен свой сотрудник для работы с данными товарами в этих секторах. Вот и получается, что свойства моделей будет состоять из:

* Товар
* Сотрудник, который отвечает за склад
* Кол-во товаров
* Сектора(зоны)

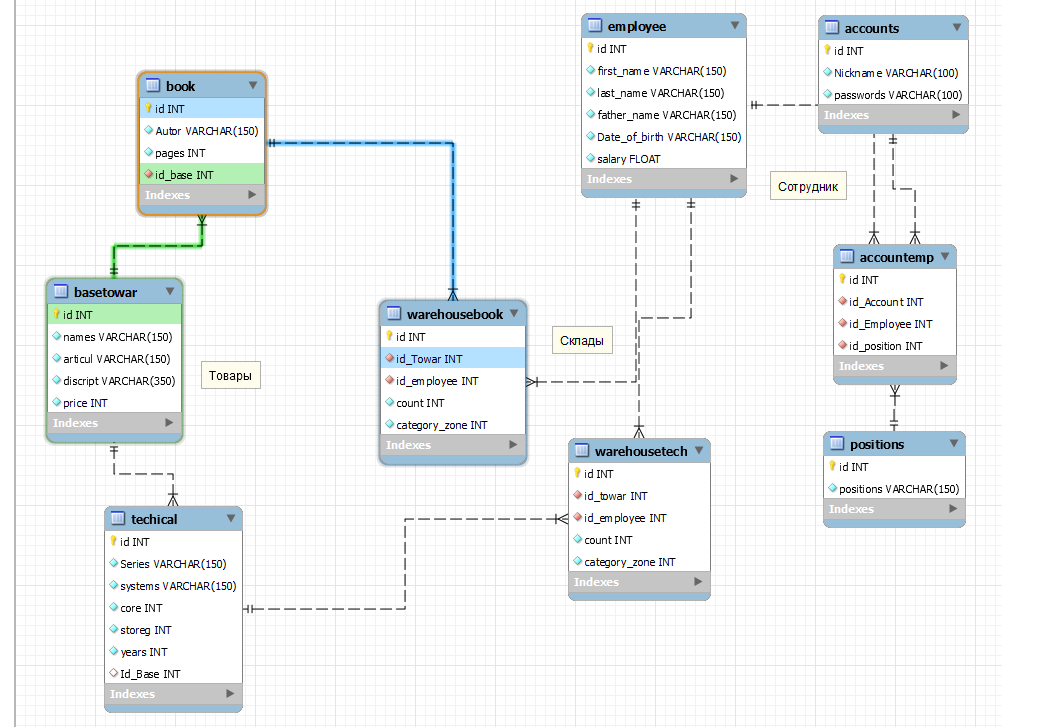


Рис.1.2.1 Общая схема представления моделей данных

## **Выбранные технологии**

В рамках этой курсовой работы была задача разработать информационную систему на основе представленных технологий - *Язык программирования: C#, Интегрированная среда разработки: Microsoft Visual Studio, Фреймворк: .NET Framework или более новая версия.*

**С# (“Си Шарп”)** - один из наиболее быстро растущих и востребованных, а также в какой-то степени «удобный» язык программирования. Он является модифицированным языком С/C++ от компании Microsoft, позволяющее создать наиболее универсальное средство для разработки программного обеспечения большого количества устройств и операционных систем[[1]](#footnote-1).

*Основные плюсы C#:*

* Независимость от аппаратного функционала. Программу не нужно адаптировать или внедрять проект под многочисленные платформы.
* Управление памятью. Если требуется, чтобы программа работала как можно стабильнее, необходимо часто выполнять очистку от различного мусора. Язык программирования C# выполняет эту задачу в автоматическом режиме.
* Строгость типизации. Если вы объявляете переменную в C#, необходимо сначала указать, что в ней находится (например – строка, число или массив).

**Visual Studio** - это мощное средство разработчика, которое можно использовать для выполнения всего цикла разработки в одном месте. Это комплексная интегрированная среда разработки (IDE), которую можно использовать для записи, редактирования, отладки и сборки кода, а затем развертывания приложения.

**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;
* исключаются проблемы с производительностью сред выполнения скриптов или интерпретируемого кода;
* обеспечивается взаимодействие на основе промышленных стандартов, которое гарантирует интеграцию кода платформы .NET Framework с любым другим кодом[[2]](#footnote-2).

**Windows Presentation Foundation (WPF)** - это библиотека (ещё его называют фреймворком) для создания пользовательских интерфейсов для интеллектуальных клиентских приложений. Данный инструмент предлагает базовый код для представления основных параметров приложения с графическим интерфейсом, включая:

* Главные окна,
* Диалоговые окна,
* Элементы управления,
* Системы меню и т.п.

Начиная с версии .NET 3.0, также предлагается альтернативный API-интерфейс под названием Windows Presentation Foundation (WPF). WPF может дать намного более мощные средства, которые позволяют изменить визуальные элементы в любой части экранного содержимого. Кроме этого, есть возможность использовать встроенную систему анимации, чтобы делать все это прямо на глазах пользователя. И поскольку механизм WPF визуализирует содержимое окна как часть одной операции, он может обрабатывать неограниченное количество слоев перекрывающихся элементов, даже имеющих нерегулярную форму и частичную прозрачность.

# **ПРОЕКТИРОВАНИЕ**



## **Разработка функций запроса к бд**

Как было описано ранее, были проанализированы основные требования и описаны основные модели. Прежде всего эти данные должны будут обрабатываться непосредственно в приложении. Причем структурность кода наших моделей должна быть описана четко по принципу. Прежде всего стоит написать класс, который будет отвечать за подключение к нашей базе данных. Она будет описана свойствами базы данных, а методы реализации будет в основном лишь открытие и закрытие потока, как показано в листинге 2.1.1

*Листинг 2.1.1*

|  |
| --- |
| class BdConnect  {  protected string conns;  protected MySqlConnection conn;  public BdConnect() {  try  {  conns = "Server=" + "localhost" + ";Database=" + "Warehouse" + ";port=" + "3306" + ";User=" + "root" + ";password=" + "SaraParker206";  conn = new MySqlConnection(conns);  }  catch (Exception e)  {  Console.WriteLine("Ошибка подключение к Базе Данных к основному ПК - " + e.Message);  Console.WriteLine("Попытка подключения к базе данных второму ПК");  }  }  public BdConnect(string host, string database, string port, string user, string password)  {  conns = "Server=" + host + ";Database=" + database + ";port=" + port + ";User=" + user + ";password=" + password;  conn = new MySqlConnection(conns);  }  public void BdConnsOpen()  {  conn.Open();  }  public void BdConnsExit()  {  conn.Close();  }  } |

От него будут исходить несколько других классов, которые будут уже связаны непосредственно с запросами к базе. Данных классов несколько и делятся под два общих вида: Ввод данных и вывод данных.

К вводу данных приведены следующие классы:

* BDInsertBook – запись книг
* BDInsertTechical – запись техники
* BDInsertWareBook – запись книг в склад
* BDInsertWareTech – запись техники в склад

К выводу данных:

* BDSelectBook – вывод книг
* BDSelectTechical – вывод техники
* BDSelectEmployee – вывод сотрудников (предназначен для работы с безопасности)

У каждого класса по 1-2 метода, отвечающий за запись/вывода данных, в соответствии с моделями данных. На листинге 2.1.2 и 2.1.3 приведены примеры того, как выглядит функционал.

*Листинг 2.1.2*

|  |
| --- |
| class BDInsertBook: BdConnect, InterfasesBDBook  {  MySqlCommand? commands;  public void InsertBase(string name, string articul, string discript, int price)  {  string MySqlCode = "INSERT INTO basetowar (names,articul,discript,price) " +  "VALUES (@n,@a,@d,@p)";  commands = new MySqlCommand(MySqlCode, conn);  commands.Parameters.Add("@n", MySqlDbType.VarChar).Value = name;  commands.Parameters.Add("@a", MySqlDbType.VarChar).Value = articul;  commands.Parameters.Add("@d", MySqlDbType.VarChar).Value = discript;  commands.Parameters.Add("@p", MySqlDbType.Int32).Value = price;  commands.ExecuteNonQuery();  }  public void InsertBook(string author, int pages, string name)  {  string MySqlCode0 = "SELECT id FROM basetowar WHERE names=@n";  int Key = 0;  MySqlCommand commands2 = new MySqlCommand(MySqlCode0, conn);  commands2.Parameters.Add("@n", MySqlDbType.VarChar).Value = name;  using (MySqlDataReader reading = commands2.ExecuteReader())  {  while (reading.Read())  {  Key = (int)reading.GetValue(0);  }  }  string MySqlCode = "INSERT INTO book (Autor,pages,id\_base) " +  "VALUES (@a,@p,@id)";  commands = new MySqlCommand(MySqlCode, conn);  commands.Parameters.Add("@a", MySqlDbType.VarChar).Value = author;  commands.Parameters.Add("@p", MySqlDbType.Int32).Value = pages;  commands.Parameters.Add("@id", MySqlDbType.Int32).Value = Key;  commands.ExecuteNonQuery();  } |

Причем стоит заметить, что данные модели наследуются от родителя, связанный лишь с подключением к базе данных. Соответственно они будут иметь все свойства и методы родителя.

*Листинг 2.1.3*

|  |
| --- |
| class BDSelectBook: BdConnect, InterfasesBDSelect  {  MySqlCommand? commands;  public DataTable SelectTowar()  {  string MySqlCode = "select names,articul,discript,price,autor,pages from basetowar join book on basetowar.id = book.Id\_Base;";  MySqlCommand commands = new MySqlCommand(MySqlCode, conn);  DataTable table = new DataTable();  MySqlDataAdapter adapter = new MySqlDataAdapter(commands);  adapter.Fill(table);  return table;  }  } |

Поскольку эти классы наследуются от родителя BdConnect и могут реализовывать одни и те же методы, был сделан класс интерфейсов под каждый из них, общий вид показан на листинге 2.4. Это связано с тем, что каждый класс наследник может дополнять существующие свойства и метода класса родителя и работать немного иначе. Данный набор нам помогает сделать их более гибкими и не дает возможность срабатыванию другого метода другого наследника, который может не использоваться.

*Листинг 2.1.4*

|  |
| --- |
| interface InterfasesBDSelect  {  public void BdConnsOpen();  public void BdConnsExit();  }  interface InterfasesBDBook  {  public void BdConnsOpen();  public void InsertBase(string name, string articul, string discript, int price);  public void InsertBook(string author, int pages, string name);  public void BdConnsExit();  }  interface InterfasesBDWareBook  {  public void BdConnsOpen();  public void InsertBook(int count, int zone, string emp, string towar);  public void BdConnsExit();  }  interface InterfasesBDTech  {  public void BdConnsOpen();  public void InsertBase(string name, string articul, string discript, int price);  public void InsertTechical(string series, string systems, int core, int store, int year, string name);  public void BdConnsExit();  }  interface InterfasesBDWareTech  {  public void BdConnsOpen();  public void InsertTechical(int count, int zone, string emp, string towar);  public void BdConnsExit();  } |

## **Реализация вывода формы ввода данных**

Из предыдущей главы было наглядно показан принцип описания наших моделей с принципом работы классов наследников через интерфейсы, но есть иной подход разработке подобных классов. Мы можем, как на примере листинга 2.5 увидеть, что класс родителя можно описать, как абстрактный класс. Абстрактный класс считается лишь шаблоном или черновиком для наследников и никакой реализации оно не имеет. Все преимущественные моменты уже показывают в деле именно наследники. При помощи приставки override, они дополняют шаблон и реализовывают свойства и методы родителя, описывая уже собственные функции и принцип работы. Благодаря такому способу, можно не создавать отдельно интерфейсы под каждого наследника. Они уже делают это в процессе реализации, потому что, как ранее было сказано, они взяли с одного и того же черновика, но описав их каждый индивидуально.

*Листинг 2.2.1*

|  |
| --- |
| abstract class Container2  {  protected StackPanel mainContainer;  protected TextBlock name;  protected TextBox input;  abstract public string Output { get; }  public Container2()  {  }  abstract public void SettingName(string n);  abstract public void SettingInput();  abstract public void SettingStack();  abstract public StackPanel Result();  } |

*Листинг 2.2.2*

|  |
| --- |
| class BlockInput: Container2  {  public BlockInput()  {  mainContainer = new StackPanel();  name = new TextBlock();  input = new TextBox();  }  public override string Output { get { return input.Text; } }  public override void SettingName(string n)  {  name.Text = n;  name.Width = 150;  name.Height = 100;  name.FontSize = 18;  name.TextWrapping = TextWrapping.Wrap;  input.Margin = new Thickness(10, 0, 10, 0);  }  public override void SettingInput()  {  input.Width = 100;  input.Height = 100;  input.FontSize = 18;  input.TextWrapping = TextWrapping.Wrap;  input.Margin = new Thickness(5,0,5,0);  }  public override void SettingStack()  {  input.Width = 600;  //---------------------------------  mainContainer.Orientation = Orientation.Horizontal;  mainContainer.Margin = new Thickness(5);  //----------------------------------  mainContainer.Children.Add(name);  mainContainer.Children.Add(input);  }  public override StackPanel Result()  {  return mainContainer;  } |



# **ПРАКТИЧЕСКАЯ ЧАСТЬ**

## **Разработка баз данных**

Следующим шагом будет разработка баз данных, на основе того, что было описано в аналитической части. Для этого потребуется создать саму базу данных в СУБД MySql. Сама база данных будет иметь название - warehouse. Далее необходимо создать таблицы под наши описанные модели из аналитической части. В общей сложности должно выйти такое кол-во таблиц, как на рисунке 3.1.1.

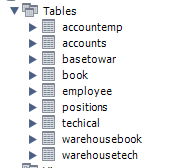


Рис.3.1.1 Список таблиц базы данных warehouse

Основными типами данных наших параметров описаны следующими:

* Int – Числовой тип данных, хранящий целое число.
* Nvarchar(размер строки в байтах) – строка текста. Определить тип текста можно иными способами (иногда ещё может зависит от СУБД): text, char, varchar. Стоит различать varchar от nvarchar.  Varchar позволяет хранить строки в формате ASCII, где один символ занимает 1 байт, а nvarchar хранит строки в формате Unicode, где каждый символ занимает 2 байта.
* Date – описывает дату и время.

Кроме этого использовалось такое понятие, как foreign key – внешний ключ. В отличии от первичного, он может повторяться. Однако его основная работа заключается в установки связи между данными в двух таблицах с целью контроля данных, которые могут храниться в таблице внешнего ключа.

## **Создание окна Авторизации**

Наше окно состоит из пару полей и кнопок. Поля будут предназначены для ввода данных логина и пароля сотрудника. Когда пользователя введет, ему потребуется нажать на кнопку входа, чтобы начать процедуру проверки.

Вся процедура проверки описана двумя классами CorrectLogins и CorrectPass. Первый класс отвечает за проверку логина в хранилище данных на наличие такого аккаунта в системе. Второй отвечает за проверку пароля. Пароль будет проверяться по трем параметра:

* Сколько элементов было введено
* Существует ли Заглавная буква?
* Существует ли число?

От каждого правильного варианта будет отмечаться специальный флажок, пропускающий следующую процедуру. Если все флажки были переключены – это говорит о том, что все в порядке. В случае ошибок, приложение создаст сообщений об данной проблеме и выведет на экран. Подобный функционал описан в листинге 3.2.1. Существуют ещё два глобальных флажка, которые проверяют, корректно ли все введено, для входа в этот аккаунт. Если в одном из скриптов возникнут проблемы, будет вылезать ещё одно сообщение о ошибке входа и в таком варианте потребуется ввести другой логин. К счастью, есть вариант зайти под ролью «Гость». Он дает возможность просмотреть только часть функционала нашего приложения. Какую-либо работу и манипуляцию с данными такой аккаунт не дает возможности.

*Листинг 3.2.1*

|  |
| --- |
| class CorrectLogins  {  private string login;  public string Login { get { return login; } }  public CorrectLogins(string str) {  login = str;  }  public bool CheckAccount()  {  bool isBool = false;  BDSelectEmployee select = new BDSelectEmployee();  select.BdConnsOpen();  try  {  isBool = select.SelectEmployeeAccount(login);  if (isBool)  {  MessageBox.Show("Логин совпадает с данными в базе данных." + isBool);  }  else  {  MessageBox.Show("Логин не совпадает с данными из базы данных. Введите другой логин." + isBool);  }  }  catch (Exception ex)  {  MessageBox.Show("Произошла ошибка SQL команды." + ex.Message);  }  select.BdConnsExit();  return isBool;    }  } |

Во время работы этих классов используется модель класса BDSelectEmployee, для получения данных запроса о сотруднике. По большой части, класс проверяет на количество строк после запроса. В случае, если аккаунт действительно есть в базе, он возвращает булевое значение.

## **Создание основного функционала приложения**

После авторизации сотрудника, ему будет выведено небольшое окно с меню инструментами и кнопкой. Меню инструментов позволяет сотрудникам работать с данными таблиц из баз данных. В основном встречается две кнопки. Insert и Select. Они позволяют вводить данные и записывать в базу, а также выводить содержимое товаров на экран. Функционал этих двух инструментов может различаться, в зависимости от роли сотрудника. Для этого, перед стартом основного окна, запускается небольшая проверка данных. Через класс BDSelectEmployeeпоступает запрос о должности сотрудника. Итоговым результатом запроса становится само её название. Более подробно описано в листинге 3.3.1.

*Листинг 3.3.1*

|  |
| --- |
| public string SelectRole(string log)  {  string MySqlCode = "select positions from positions where id in (select id\_position from accountemp where id\_Account in (select id from accounts where nickname = @l));";  MySqlCommand commands = new MySqlCommand(MySqlCode, conn);  DataTable table = new DataTable();  MySqlDataAdapter adapter = new MySqlDataAdapter(commands);  commands.Parameters.Add("@l", MySqlDbType.VarChar).Value = log;  adapter.Fill(table);  string? result = table.Rows[0][0].ToString();  return result;  } |

Далее запускается код, который проверяет полученную должность и на основе этого, открывает функционал сотрудника. Функциональность каждый из инструментов ориентирован на работу с товарами, описанные из аналитической части: Книги и Техника. Каждый из сотрудников отвечает за работу того или иного товара. Для того, чтобы записать данные, необходимо будет выбрать подкатегорию этих инструментов. Затем, вместо пустого фона, который находится ниже меню, будет всплывать результат. В случае Insert, будут выведены специальные поля с их обозначениями, соответствующий каждому параметру модели товаров. В них необходимо будут ввести новые данные. Как только все будет заполнено необходимо будет нажать на кнопку «Записать», дабы наши результаты отразились уже в базе данных. В следствии чего, мы можем увидеть наши результаты уже при помощи инструмента Select, выбрав нужную подкатегорию. Данный инструмент позволит нам просмотреть список товаров на складе. Стоит учесть момент, что в клиентской части запускается сначала метод, который расписывает обозначение каждого столбца, который будет в последующим выведен, а уже после будет выводиться итоговые результаты запроса при помощи классов BDSelectBook и BDSelectTechical.

Кроме всего прочего, имеется возможность просмотреть свой профиль. Это дает нам краткую сводку о том, кто он и чем занимается. Все данные можно будет просмотреть, нажав на кнопку, возле панели инструментов, который будет представлен после авторизации. Имя кнопки, представляется именем нашего аккаунта. В зависимости от имени, клиент будет подгружать соответствующие данные сотрудника и записывать их в соответствующие поля. Однако стоит вспомнить, что есть такой аккаунт, как «Гость». У него роль будет ограничен лишь одним инструментом – вывод данных. Это позволить просмотреть то, как будет примерно работать приложение и как с ним можно будет взаимодействовать. Так такого этого аккаунта не существует. Он считается временным сотрудником. В профиле можно узнать, когда данный сотрудник с такой ролью был создан в системе.

# **ТЕСТИРОВАНИЕ**

У нас есть два аккаунта сотрудников. Первый аккаунт отвечает за работу техники товаров, а второй за книги. Представим ситуацию, что нам нужно записать новый товар. Нам потребуется авторизироваться в системе. Для этого были предоставлены логины и пароли для входа в систему. Допустим, нам выдали неверный логин, который в системе не существует. Мы вводим в поля соответствующие данные, которые дала администрация. Что же произойдет?

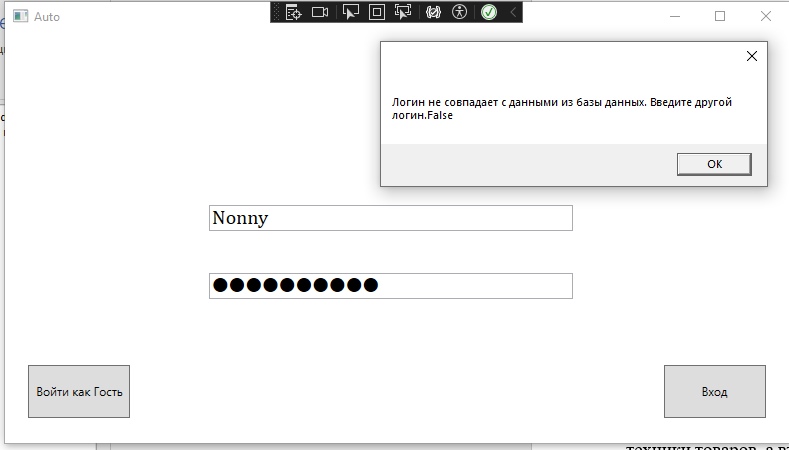


Рис. 4.1 – Ошибка входа. Неверный логин.

Как представлено на рисунке 4.1, у нас будет выводиться сообщение об ошибке и запросит ввести другой логин от аккаунта. В этом случае администрация нам выдает верный логин. Однако, как оказалось, пароль был тоже неправильным. Приключается следующая история, как показано на рисунке 4.2, наш данный пароль не имеет один из следующих параметров, которые были описаны ранее, касаемо заглавных букв или если у нас нету числа в строке пароля. В таком варианте, снова обращаемся к администрации. После успешной авторизации, нам будет всплывать окно, что наш логин был введен корректно и можем войти в систему.

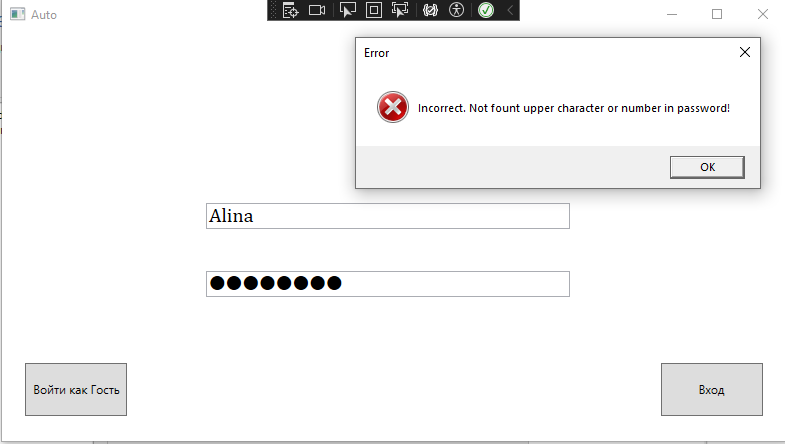


Рис.4.2 – Ошибка ввода пароля

В главном окне приложения нам предоставлены инструменты для работы с товарами техники. Для добавления техники необходимо будет навести на Insert и выбрать подкатегорию «Записать технику». Выведется список полей, который надо будет заполнить. Допустим ввели необходимые значения, как показано на рисунке 4.3, нажали на «Записать» и все прошло успешно. Теперь потребуется просмотреть список товаров.



Рис.4.3 Заполнение полей данными

На рисунке 4.4 представлен вариант, где все будут выведены на экран. Среди них будет представлен товар, который был недавно добавлен в базу данных.

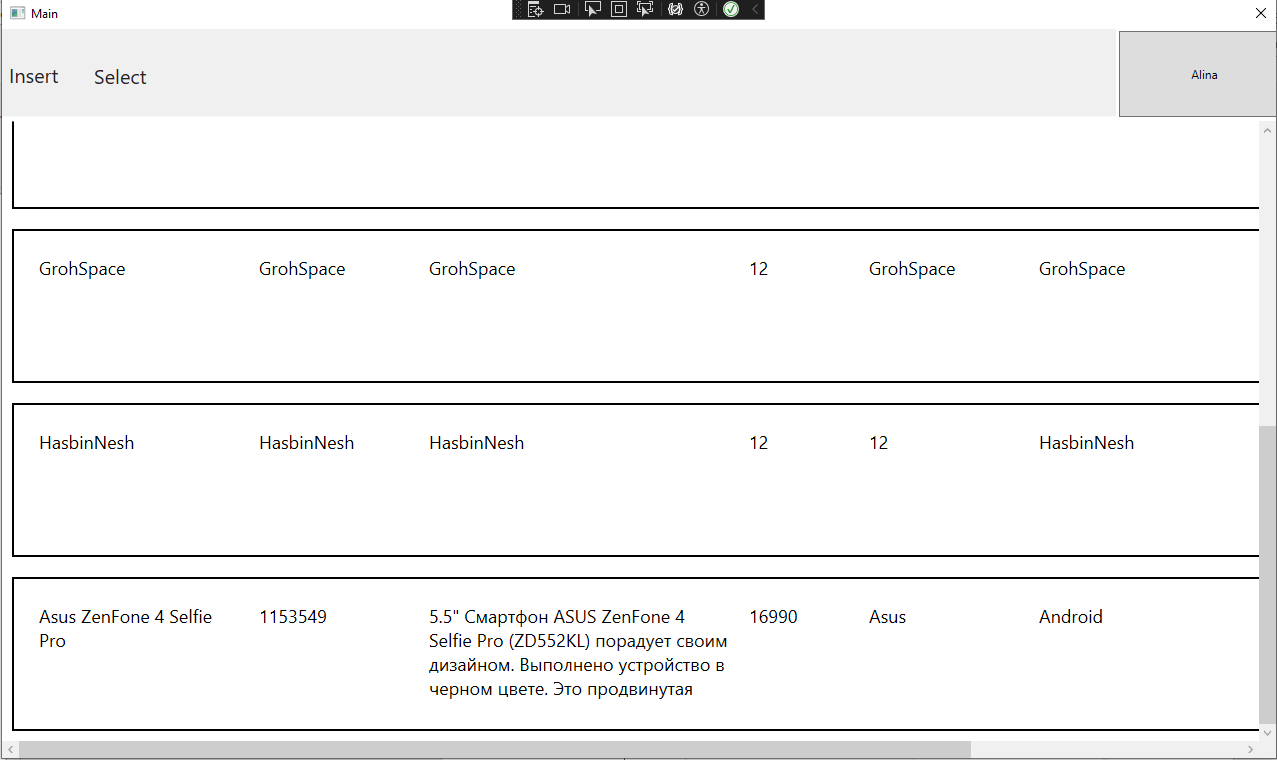


Рис. 4.4 – Вывод список техники

С другим пользователем будет аналогичный случай, единственное отличие будет, это работа с книгами. На рисунке 4.5 представлен альтернативный вариант выдачи списка книг из базы.

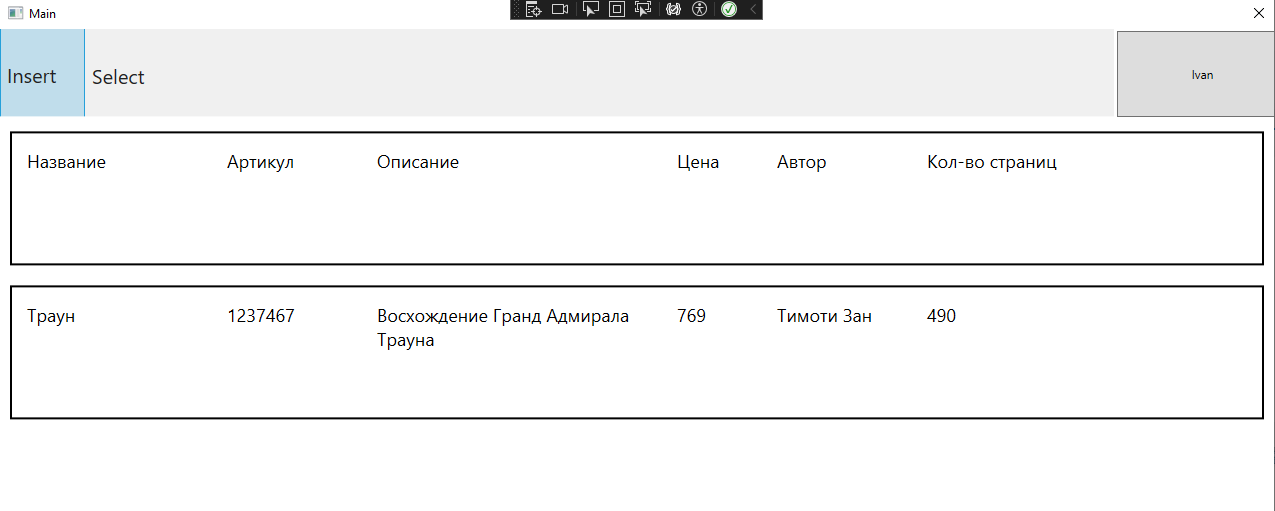


Рис. 4.5 – Вывод списка книг

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения данной работы было произведено ряд исследований технологий и устройства предприятия, в котором происходит управление складом различной продукции. На основе небольшой сводки информации были определены цель и задачи работы. Далее были описаны модели для удобства работы с данными для простоты её обработки и обращения к ней. Особенностями было распределения основных типов информации каждого реального объекта и отсортирована под соответствующие типы моделей. Вскоре было ознакомление с технологиями, благодаря которым было реализован данный проект.

Во время разработки приложения, данные модели были спроектированы по основным принципам Объектно-Ориентированного программирования. В них подробно было прописано, как работают методы наших моделей уже в самой программе. Более того, описана реализация того, как будет подключено приложение к хранилищу данных, а также выявили альтернативный способ реализации подобных классов.

Далее была реализация уже следующих компонентов:

* База данных – место, где будут храниться все наши данные. Они были реализованы на основе описанных моделей, что были представлены ещё в аналитической части.
* Классы для корректности ввода аккаунты – позволяет проверить аккаунт на его наличие и корректность ввода правильного пароля.
* Реализована возможность работа с данными в основном меню приложении

Также, для проверки, были произведены некоторые тесты с разных аакаунтов, которые имею в свою очередь разные роли и разные функции работы с приложением.

В итоге исходя из вышесказанного, можно сделать вывод, что основные задачи работы были успешно выполнены, а в следствии чего, цель курсовой работы была успешно достигнута.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Джозев, Ал. and Aл. Бен, 2021. C# 9.0. Карманный справочник. Диалектика, стр: 256.
2. Марк, Пр., 2023. C# 10 и.NET 6. Современная кросс-платформенная разработка. Питер, стр: 848.
3. Рихтер, Дж., 2023. CLR via C#. Программирование на платформе Microsoft. NET Framework 4.5 на языке C#. 4-е изд.. Питер, стр: 896.
4. Болье, А., 2021. Изучаем SQL. Генерация, выборка и обработка данных. Диалектика, стр: 400.
5. Учебник по языку SQL (DDL, DML) на примере диалекта MS SQL Server. Date Views 2023 год www.habr.com.
6. Нормализация отношений. Шесть нормальных форм. Date Views 2023 год www.habr.com.
7. Максименко, Д., 2023. Разработка индекса для системы управления базами данных MySQL на основе B-link деревьев для интенсивной многопоточной вставки. Разработка индекса для системы управления базами данных MySQL на основе B-link деревьев для интенсивной многопоточной вставки, Санкт-Петербургский политехнический университет Петра Великого Date Views 2023 год www.elib.spbstu.ru.
8. Маркин, А. В. Программирование на sql в 2 ч. Часть 2 : учебник и практикум для бакалавриата и магистратуры / А. В. Маркин. — М. : Издательство Юрайт, 2019. — 292 с
9. Литвиненко, Н.А., 2022. Декларативное программирование на языке XAML + C#. WPF проекты. Горячая линия - Телеком, стр: 320.

1. Джозев, Ал. and Aл. Бен, 2021. C# 9.0. Карманный справочник. Диалектика [↑](#footnote-ref-1)
2. Рихтер, Дж., 2023. CLR via C#. Программирование на платформе Microsoft [↑](#footnote-ref-2)