Министерство образования Республики Беларусь

Учреждение образования

Белорусский государственный университет информатики и радиоэлектроники

Институт информационных технологий

Факультет повышения квалификации и переподготовки

Кафедра «Микропроцессорные системы и сети»

КУРСОВОЙ ПРОЕКТ

на тему: Программное средство учета и внутреннего перемещения товаров

Выполнила:

Слушатель гр. 10331

Абрамчук О.Г.

Проверила:

Желакович И. М.

МИНСК 2023

Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Институт информационных технологий

Кафедра «Микропроцессорные системы и сети»

|  | УТВЕРЖДАЮ | |
| --- | --- | --- |
|  | Заведующий кафедрой МПСС | |
|  | *И.В.Кашникова* | |
|  |  | |
|  | (подпись) | |
|  |  | 2023 г. |

ЗАДАНИЕ

по курсовому проектированию

| Слушателю | *Абрамчук Ольге Геннадьевне* | | | |
| --- | --- | --- | --- | --- |
| Группа | *10331* | | | |
| 1. Тема работы | *Программное средство учета и внутреннего перемещения товаров* | | | |
| 2. Срок сдачи студентом законченной работы | | | |  |
| 3. Исходные данные к работе | |  | | |
| *Приложение должно предоставлять пользователю возможность выполнять следующие действия: добавление, редактирование и удаление пунктов хранения; добавление и удаление моделей одежды и отдельных товаров; создание заявок на внутреннее перемещение товаров, просмотр списка заявок и удаление выбранных заявок; авторизация сотрудников в приложении. Приложение должно обеспечить вывод в виде документа созданную заявку на перемещение товаров, отчет по остаткам товаров по пунктам хранения.* | | | | |
| Язык программирования | | | *С#* | |
| Среда разработки | | | *Microsoft Visual Studio2022* | |
| Вид приложения | | | *desktop (WPF)* | |
| Парадигма программирования | | | *ООП* | |
| Способ организации данных | | | *СУБД MSSQL* | |
| Технология доступа к данным | | | *Entity Framework* | |
| Архитектурный паттерн | | | *MVVM* | |

| 4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке) | | | | | |
| --- | --- | --- | --- | --- | --- |
| *1)* | | *Введение* | | | |
| *2)* | | *Аналитический обзор и постановка задачи* | | | |
| *3)* | | *Разработка методов и моделей* | | | |
| *4)* | | *Разработка проекта программного обеспечения* | | | |
| *5)* | | *Разработка алгоритмов и их описание* | | | |
| *6)* | | *Тестирование полученного программного продукта* | | | |
| *7)* | | *Заключение* | | | |
| *8)* | | *Приложение: Листинг кода с комментариями* | | | |
| 5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков) | | | | | |
| *1)* | | *трассировочная матрица* | | | |
| *2)* | | *USE CASE диаграммы* | | | |
| *3)* | | *схема ресурсов системы* | | | |
| *4)* | | *схема базы данных* | | | |
| *5)* | | *диаграмма классов* | | | |
| *6)* | | *макеты окон* | | | |
| *7)* | | *3 схемы алгоритмов* | | | |
| 6. Консультант по работе | | | | *И.М.Желакович* | |
| 7. Дата выдачи задания | | | | *09.12.2022* | |
| 8. Календарный график работы на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов): | | | | | |
| *Разделы 1,2* | | | *28.12.2022 г.* | | *25%* |
| *Раздел 3,4* | | | *12.02.2023 г.* | | *35%* |
| *Раздел 5,6* | | | *01.03.2023 г.* | | *30%* |
| *Раздел 7* | | | *20.04.2023 г.* | | *10%* |
|  | | |  | |  |
| РУКОВОДИТЕЛЬ | | |  | | *И.М.Желакович* |
|  | | | (подпись) | |  |
|  | | |  | |  |
| Задание принял к исполнению | | | *09.12.2022* | |  |
|  | | | (дата) | | (подпись слушателя) |
|  |  | | | | |

**СОДЕРЖАНИЕ**

| Введение……………………………………………………………………... | 5 |
| --- | --- |
| 1  Аналитический обзор и постановка задачи…………………………….. | 6 |
| 1.1  Обзор существующих программных решений для складского учета……………………………………………………….…………………. | 6 |
| 1.2  Анализ предметной области и постановка задач……………..………. | 10 |
| 2  Разработка методов и моделей…………………………….…………….. | 12 |
| 3  Разработка проекта программного обеспечения……………………...... | 14 |
| 3.1  Функциональный анализ предметной области….……………………. | 14 |
| 3.2  Разработка схема базы данных..………………….……………………. | 16 |
| 3.3  Разработка диаграммы классов…….…………….……………………. | 21 |
| 3.4  Разработка графического интерфейса пользователя…………………. | 22 |
| 4  Разработка алгоритмов и их описание….……………………………...... | 29 |
| 5  Тестирование полученного программного продукта…….…………….. | 32 |
| Заключение…………………………………………………………………... | 37 |
| Список использованных источников………………………………………. | 38 |
| Приложение А  (обязательное) Программный код алгоритмов расчета остатков товаров на пунктах хранения ………...………. | 39 |
| Приложение Б (обязательное) Листинг кода разработанного приложения | 40 |

**ВВЕДЕНИЕ**

При ведении бизнеса, связанного с продажами, нужно уделять особое внимание организации учета перемещения товаров между пунктами их хранения и продаж.

 Для успешной работы необходимо постоянно владеть информацией о виде и количестве товара, который находится в наличии. С этим и помогают справляться системы учета перемещения товаров [1].

В настоящее время в распоряжении бизнеса имеются программные средства, позволяющие вести учет всех процессов на предприятии, в том числе складской. Существующие программные средства являются достаточно затратными по внедрению и сопровождению для индивидуальных предпринимателей (далее ИП), обладают избыточным функционалом, требуют дополнительного оборудования и обучения. Чаще всего ИП не имеют специально выделенного склада, товары хранятся в местах продаж.

ИП не уделяется особого внимания по ведению учета перемещения товаров. Преимущественным способом ведения такого учета служит ручной учет в журналах, либо ведение таблиц Excel, что очень трудоёмко и не исключает большого количества ошибок из-за человеческого фактора.

Учитывая вышесказанное, можно сделать вывод, что автоматизация учета перемещения товаров для ИП является актуальной проблемой и требует поиска путей решения. Выходом можем стать разработка автоматизированной системы, которая позволит вести учет всех перемещений товаров между пунктами хранения и продаж в режиме реального времени, отслеживать остатки по каждому товару и в разрезе по каждому пункту продаж.

Актуальность работы: данное исследование направлено на решение одной из важнейших проблем ИП – учет перемещения товаров, который является залогом точного и своевременного отражения остатков товаров.

Целью курсового проекта является проектирование и разработка программного средства, предназначенного для учета перемещения товаров между пунктами хранения и продаж для ведения оперативного учета остатков товаров.

Для достижения поставленной цели используется следующий инструментарий: Microsoft SQL Server 2018, Visual Studio 2022 Community, Entity Framework.

**1 АНАЛИТИЧЕСКИЙ ОБЗОР И ПОСТАНОВКА ЗАДАЧИ**

**1.1** **Обзор существующих программных решений для складского учета**

В настоящее время на рынке программного обеспечения (далее ПО) существует множество программных продуктов для автоматизации работы склада.

Они отличаются друг от друга набором функций, стабильностью, удобством работы, стоимостью [2].

Приступая к анализу основных из них, следует отметить, что на рынке отсутствуют решения, предназначенные исключительно для учета внутреннего перемещения товаров.

Для анализа выбраны следующие системы: «1C: Управление небольшой фирмой для Беларуси», «Мой Склад», «Учет: услуги, торговля и склад». Примеры программных окон представлены на рисунке 1-6.

* + 1. «1C: Управление небольшой фирмой для Беларуси»

Комплексное готовое решение для управления и учета на предприятиях малого и среднего бизнеса. На рисунках 1-2 представлены примеры окон программы.

Рисунок 1 – Интерфейс программы «1C: Управление небольшой фирмой для Беларуси»: перемещение товараИзображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Интерфейс приложения «1C: Управление небольшой фирмой для Беларуси»: просмотр остатковИзображение выглядит как стол

Автоматически созданное описание

Данная программа позволяет работать одновременно большому количеству пользователей, осуществлять подключение сканеров штрих-кода, терминалов сбора данных и прочего оборудования, организовать единую информационную систему для управления различными аспектами деятельности предприятия. Основные функции: заказы клиентов, оптовые продажи, резервирование товаров, цены, скидки, бонусы, банк, касса, кадровый учет и другие[3].

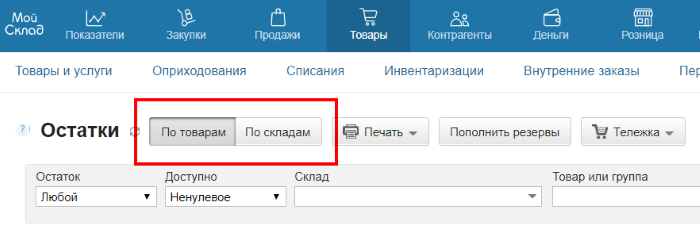
1.1.2 «Мой Склад»

Программа «Мой Склад» — специализированное программное оснащение для торгового бизнеса. На рисунках 3-4 представлены примеры окон программы.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – Интерфейс программы «Мой склад»: перемещение товара

Рисунок 4 – Интерфейс приложения «Мой склад»: просмотр остатков

Данное решение предназначено для учета продаж, управления складом или магазином. Основные пользователи - предприятия торговли малого и среднего бизнеса. Сервис реализует функции обработки заказов, управления продажами через интернет магазин, закупками товара и контроля финансовых расчетов с контрагентами [4]. Основные функции: выставление счетов, договоров, печать ТТН, управление ценами, ведение расчета прибылей и убытков и другие.

1.1.3«Учет: услуги, торговля и склад»

Программа для индивидуальных предпринимателей, организаций малого и среднего бизнеса, занимающихся оказанием услуг и/или осуществляющих торговую, а также производственную деятельность[5]. На рисунке 5 представлен пример главного окна программы.

Рисунок 5 – Интерфейс приложения «Учет: услуги, торговля и склад»Изображение выглядит как стол

Автоматически созданное описание

Программа предоставляет следующий функционал: учет и анализ состояния расчетов,документооборот, учет услуг, складской учет, учет в розничной торговле.

В таблице 1 проанализированы основные сравнительные характеристики данных программ. Сравнение проведено по минимальным пакетам, представленным разработчиками данных программ. В минимальный пакет должно входить: работа 1-2 сотрудников, 2 и более точки продаж, возможность подключения дополнительных опций, возможность работать в сети.

Таблица 1 – Сравнительные характеристики программ-аналогов

|  | 1C: для небольшой фирмы | Мой склад | Учет: услуги, торговля и склад |
| --- | --- | --- | --- |
| Количество сотрудников | До 5 | 2 | любое (в рамках 1 юр. лица) |
| Стоимость подключения, руб. | 702 | - | 900 |
| Ежемесячная оплата, руб | 30 | 37 | - |
| Подключение доп. мест | - | + | + |
| Излишний функционал | + | + | + |
| Необходимость обучения | + | Есть инструкция | + |
| Подключение доп. опций | + | + | + |

Продолжение таблицы 1

| Дополнительно: | 4 часа в месяц бесплатных консультаций, подключение доп. мест только в других пакетах. | Доп. место (17 руб/мес),  доп. точка (17 руб/мес). | Возможность доработки (цена договорная), консультирование – от 40 руб/час. |
| --- | --- | --- | --- |

Как видим из таблицы - эти программы обладают существенными недостатками для начинающих ИП (ограниченность в финансовых ресурсах): достаточно высокая стоимость, необходимость обучения, платное сопровождение, оплата подключения дополнительного рабочего места и приобретение в пакете излишнего функционала.

Таким образом, принято решение в ходе данной работы реализовать собственное ПС по учету внутреннего перемещения товаров, лишенную указанных недостатков. ПС позволит владельцу бизнеса получить необходимый функционал с минимальными затратами.

**1.2 Анализ предметной области и постановка задач**

В настоящий время большинство индивидуальных предпринимателей (далее ИП), ведет учет внутреннего перемещения товаров в неавтоматизированном формате - файлы Excel или бумажные носители.

Чтобы сделать работу по организации внутреннего перемещения товаров организованной, упростить их учет и контроль за их движением необходимо использовать специализированное программное обеспечение[6].

Рассмотрим деятельность ИП, занимающегося розничной торговлей женской одеждой. Планируется разработать ПС учета внутреннего перемещения товаров для ИП. В разрабатываемой программе должны быть реализованы следующие функции:

- добавление и удаление моделей одежды;

- добавление и удаление новых единиц товара;

- добавление, удаление и редактирование пунктов хранения и продаж товара;

- оформление заявок на перемещение товаров;

- ведение и просмотр остатков товаров в разрезе пунктов хранения;

- добавление и удаление учетных записей.

ПС должно обладать простым, удобным и легко осваиваемым интерфейсом. Также должна быть разработана база данных (далее БД), в которой предусмотрено хранение информации о моделях одежды, размерных рядах, пунктах хранения и остатках. Также в БД должна храниться информация о пользователях и пользовательских ролях. БД должна обеспечивать надежное хранение и доступ к информации. Вход в приложение должен осуществляться по логину и паролю.

Приложение не должно быть перегружено излишним функционалом.

Программа должна быть написана с использованием языка программирования С# с использованием Entity Framework. Графический интерфейс реализуется с использованием технологии Windows Presentation Foundation Microsoft.NET Framework (WPF). В качестве СУБД будет использоваться MS SQL SERVER.

**2 РАЗРАБОТКА МЕТОДОВ И МОДЕЛЕЙ**

Разрабатываемое приложение должно включать возможность регистрации и авторизации пользователей, добавления моделей и товаров, пунктов хранения и заявок на перемещение, а также удаление добавленных элементов. Но кроме этого программа должна производить расчеты остатков товаров в разрезе пунктов хранения и сумму по каждому товару в заявке на перемещение.

Для выполнения расчетов применяются следующие математические модели:

- Расчет суммы товаров производится по формуле (1):

**Sum =** Quantity \* Product.Model.Price(1)

где Sum — сумма;

Quantity — количество единиц товара;

Product.Model.Price — закупочная цена единицы товара.

В предыдущей главе был проведен анализ предметной области. В результате этого анализа принято решение о выделении следующих сущностей:

1. модель: сущность содержит информацию о моделях одежды;
2. товар: сущность содержит информацию о моделях одежды в разрезе размеров;
3. товар для перемещения: сущность содержит информацию о перемещаемом товаре;
4. заявка на перемещение: сущность содержит информацию о заявке на перемещение товаров;
5. пункт хранения: сущность содержит информацию о пункте хранения товара;
6. остатки: сущность содержит информацию об остатках товара;
7. пользователь: сущность содержит информацию о пользователях системы;
8. роль: сущность содержит информацию о пользовательских ролях.

А также 2 справочника:

1. цвет: сущность содержит информацию об используемых цветах;
2. размер: сущность содержит информацию о размерах одежды.

Исходя из приведенных выше сущностей, построена концептуальная модель предметной области, которая представлена на рисунке 6.

Рисунок 6 – Концептуальная модель предметной областиИзображение выглядит как диаграмма

Автоматически созданное описание

**3 РАЗРАБОТКА ПРОЕКТА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

**3.1** **Функциональный анализ предметной области**

Анализ предметной области позволяет выделить ее сущности, определить первоначальные требования к функциональности и определить границы проекта. Модель предметной области должна быть документирована, храниться и поддерживаться в актуальном состоянии до этапа реализации.

Для визуализации связей между элементами системы разработаем трассировочную матрицу. По определению матрица трассируемости — двумерная таблица, содержащая соответствие функциональных требований продукта и подготовленных тестовых сценариев. Трассировочная матрица проекта представлена в таблице 2.

Таблица 2 – Трассировочная матрица

| № | Пожелания заказчика | № функции | Функциональное требование |
| --- | --- | --- | --- |
| 1. | Программа должна обеспечить вход в систему каждому пользователю по личному логину и паролю | 1.1  1.2 | Ввод логина  Ввод пароля |
| 2. | Программа должна обеспечивать владельцу работу со списком моделей одежды. | 2.1  2.2  2.3  2.4  2.5 | Ввод названия модели  Ввод артикула  Выбор цвета из списка  Ввод закупочной цены  Выбор фото модели |
| 3. | Программа должна обеспечивать владельцу работу с товарами по размерам в разрезе моделей. | 3.1  3.2 | Выбор модели из списка моделей  Выбор размера из списка размеров |
| 4. | Программа должна обеспечивать владельцу работу с пунктами хранения товаров (ПХТ). | 4.1  4.2 | Ввод названия ПХТ  Ввод адреса ПХТ |

Продолжение таблицы 2

| 5. | Программа должна обеспечивать владельцу работу с заявками на перемещение товаров. | 5.1  5.2  5.3  5.4  5.5 | Выбор даты в календаре  Выбор пункта отправления  товаров  Выбор пункта назначения товара  Выбор названия товара из перечня  Ввод кол-ва товара для  перемещения |
| --- | --- | --- | --- |
| 6. | Программа должна выводить остатки в разрезе ПХТ. | 6.1 | Выбор ПХТ в списке |
| 7. | Программа должна быть простой и интуитивно понятной пользователю | - | Абстрактное пожелание |
| 8. | Программа должна обеспечивать администратору работу с учетными записями. | 8.1  8.2  8.3  8.4 | Ввод логина  Ввод пароля  Ввод ФИО  Выбор роли |

Функциональные требования хорошо иллюстрирует диаграмма вариантов использования. Модели вариантов использования изображены на рисунках 6-7

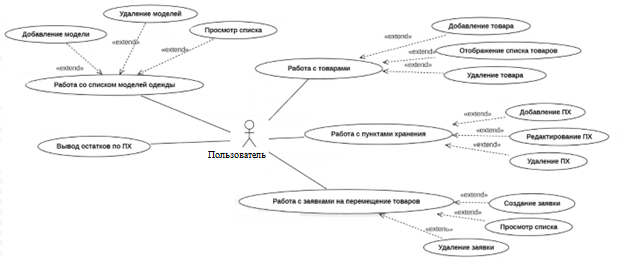


Рисунок 7 – Диаграмма вариантов использования программы пользователем

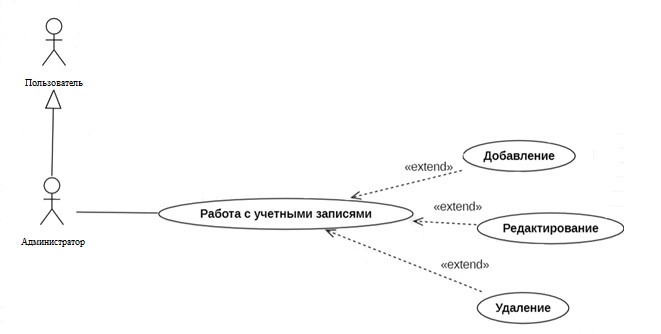


Рисунок 8 – Диаграмма вариантов использования программы администратором

Для отображения конфигурации блоков данных и обрабатывающих блоков используется схема ресурсов системы, которая представлена на рисунке 9.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 9 – Схема ресурсов системы

**3.2 Разработка схемы базы данных**

Схема БД должна включать набор всех схем ее таблиц, т.е. описание всех колонок этих таблиц (их типов, допустимых значений, связей между таблицами, типа внешних ключей и т.д.), без учета конкретных данных.

При составлении схемы БД разрабатываемой программы создано необходимое количество таблиц для хранения нужной информации – Balanses, Movements, ItemMovs, Models, Products, Warehouses, Colors, Sizes, Users, UserRoles.

Изображение выглядит как диаграмма

Автоматически созданное описаниеИзображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 10 – Схема базы данных

Данные таблицы соответствуют одноименным сущностям. Значения атрибутов сущностей представлены в следующих таблицах.

Сущность «Модели» предназначена для хранения информации о поступающих моделях одежды. Описание атрибутов представлено в таблице 3.

 Таблица 3 − Модели (Models)

| Атрибут | Тип данных | Описание | Обязательный атрибут |
| --- | --- | --- | --- |

Продолжение таблицы 3

| Article(РК) | int | Артикул | not null |
| --- | --- | --- | --- |
| Title | nvarchar | Название модели | not null |
| Price | decimal | Цена закупки | not null |
| Photo | varbinary | Фото модели | not null |
| ColorId(FK) | int | Код цвета | null |

Сущность «Товары» предназначена для хранения информации о моделях в разрезе размеров. Описание атрибутов представлено в таблице 3.

Таблица 4 − Товары (Products)

| Атрибут | Тип данных | Описание | Обязательный атрибут |
| --- | --- | --- | --- |
| Id (РК) | int | Код товара | not null |
| Article(FK) | int | Артикул | not null |
| SizeId (FK) | int | Код размера | not null |

Сущность «Пункты хранения» предназначена для хранения информации о пунктах хранения и продаж товаров. Описание атрибутов представлено в таблице 5.

Таблица 5 − Пункты хранения (Warehouses)

| Атрибут | Тип данных | Описание | Обязательный атрибут |
| --- | --- | --- | --- |
| Number (РК) | int | Номер пункта хранения | not null |
| Name | nvarchar | Название пункта хранения | not null |
| Address | nvarchar | Адрес пункта хранения | not null |

Сущность «Перемещения» предназначена для хранения информации о заявках на перемещение товаров между пунктами хранения. Описание атрибутов представлено в таблице 6.

Таблица 6 − Перемещения (Movements)

| Атрибут | Тип данных | Описание | Обязательный атрибут |
| --- | --- | --- | --- |
| Id (РК) | int | Код перемещения | not null |

Продолжение таблицы 6

| ShoppingPointId (FK) | int | Код пункта отправления | null |
| --- | --- | --- | --- |
| ReceivingPointId (FK) | int | Код пункта назначения | null |
| ShippingDate | datetime | Дата оформления заявки | not null |
| TotalAmount | decimal | Общая сумма | not null |

Сущность «Элементы перемещения» предназначена для хранения информации о наименовании и количестве перемещаемых товаров в привязке к конкретной заявке на перемещение. Описание атрибутов представлено в таблице 7.

 Таблица 7 − Элементы перемещения (ItemMovs)

| Атрибут | Тип данных | Описание | Обязательный атрибут |
| --- | --- | --- | --- |
| Number (РК) | int | Номер элемента | not null |
| MovementId (FK) | int | Код перемещения | not null |
| ProductId (FK) | int | Код товара | not null |
| Quantity | int | Количество элементов | not null |

Сущность «Остатки» предназначена для хранения информации об остатках товаров по пунктам хранения. Описание атрибутов представлено в таблице 8.

Таблица 8 − Остатки (Balances)

| Атрибут | Тип данных | Описание | Обязательный атрибут |
| --- | --- | --- | --- |
| Id (РК) | int | Код остатков | not null |
| WarehouseNumber (FK) | int | Код пункта хранения | not null |
| ProductId (FK) | int | Код товара | not null |
| Quantity | int | Количество товара | not null |

Сущность «Цвета» предназначена для хранения списка доступных цветов. Описание атрибутов представлено в таблице 9.

Таблица 9 − Цвета (Colors)

| Атрибут | Тип данных | Описание | Обязательный атрибут |
| --- | --- | --- | --- |
| Id (РК) | int | Код цвета | not null |
| Name | nvarchar | Название цвета | not null |

Сущность «Размеры» предназначена для хранения информации о доступных размерах. Описание атрибутов представлено в таблице 10.

 Таблица 10 − Размеры (Sizes)

| Атрибут | Тип данных | Описание | Обязательный атрибут |
| --- | --- | --- | --- |
| Id (РК) | int | Код размера | not null |
| Name | nvarchar | Название размера | not null |

Сущность «Пользователи» предназначена для хранения информации о пользователях, имеющих доступ к программе. Описание атрибутов представлено в таблице 11.

Таблица 11 − Пользователи (Users)

| Атрибут | Тип данных | Описание | Обязательный атрибут |
| --- | --- | --- | --- |
| Id (РК) | int | Код пользователя | not null |
| Login | nvarchar | Логин | not null |
| Password | nvarchar | Пароль | not null |
| FirstName | nvarchar | Имя | not null |
| LastName | nvarchar | Фамилия | not null |
| SecondName | nvarchar | Отчество | null |

Сущность «Пользовательские роли» предназначена для хранения информации о ролях пользователей. Описание атрибутов представлено в таблице 12.

Таблица 12 − Пользовательские роли (UserRole)

| Атрибут | Тип данных | Описание | Обязательный атрибут |
| --- | --- | --- | --- |
| Id (РК) | int | Код роли | not null |
| Name | nvarchar | Название роли | not null |

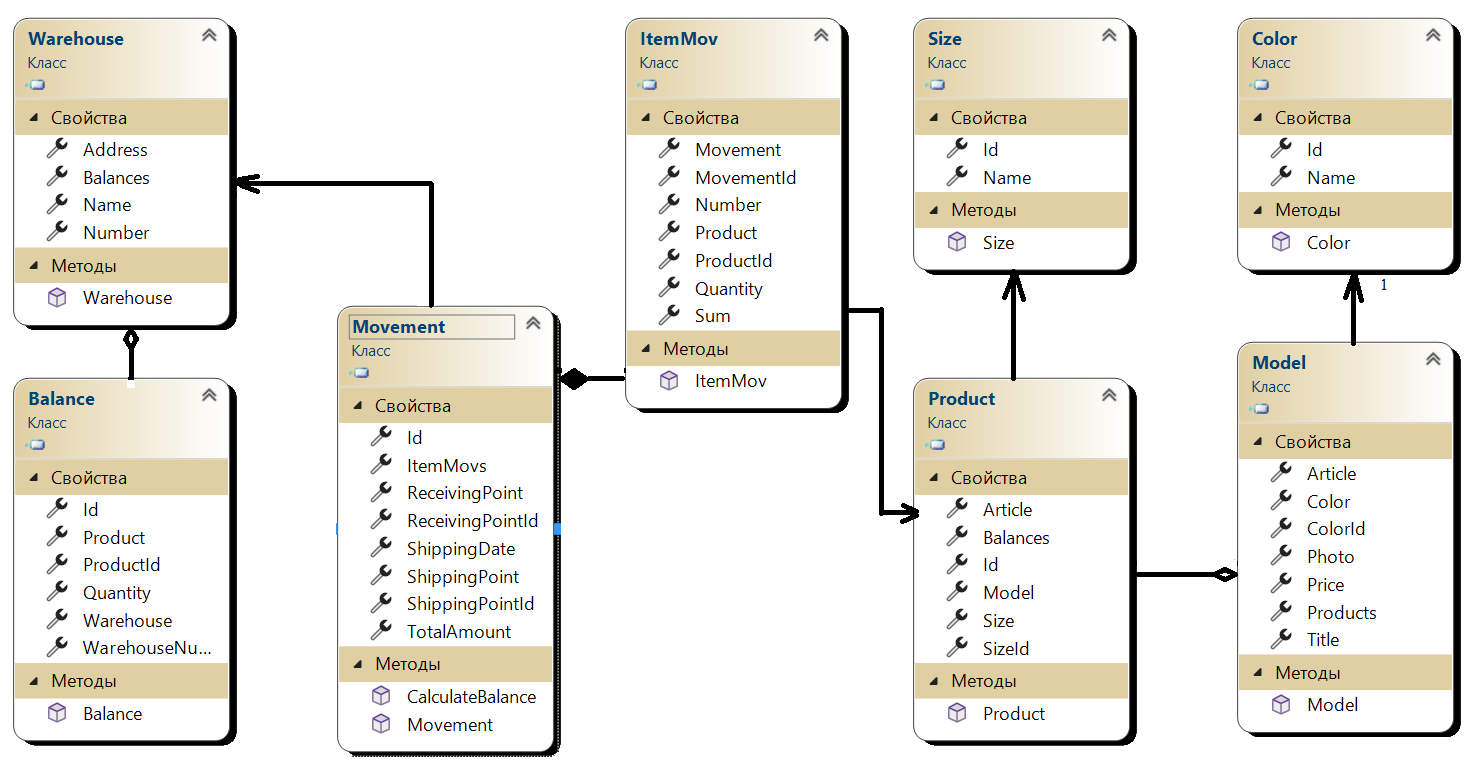
При переходе от инфологической модели к реляционной модели была раскрыта связь М:М между отношениями «Пользователи» и «Пользовательские роли». Отношением-связкой стало отношение «UserUserRole». В нем в качестве FK были созданы атрибуты «UsersId» и «UserRolesId». Они вместе образуют уникальный идентифицирующий составной (композитный) PK. Описание атрибутов представлено в таблице 13.

Таблица 13 − UserUserRole

| Атрибут | Тип данных | Описание | Обязательный атрибут |
| --- | --- | --- | --- |
| UsersId (РК) | int | Код пользователя | not null |
| UserRolesId (РК) | int | Код роли | not null |

**3.3 Разработка диаграммы классов**

В соответствии с концептуальной моделью предметной области были спроектированы следующие классы, представленные на диаграмме классов (рисунок 11).



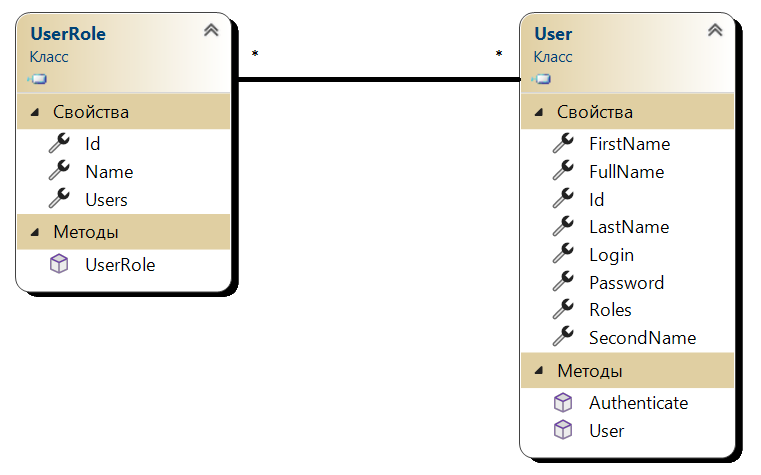
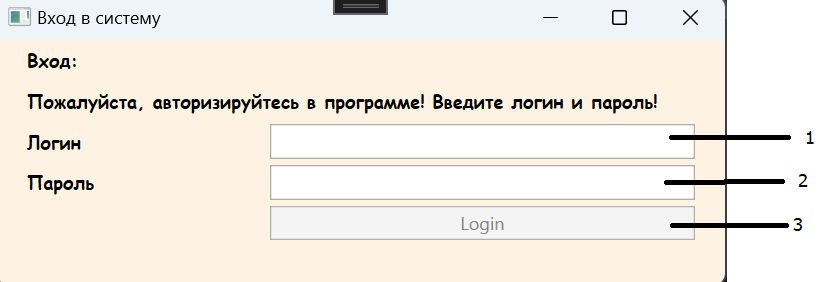


Рисунок 11 – Диаграмма классов

**3.4 Разработка графического интерфейса пользователя**

Исходя из диаграммы вариантов использования и этапа анализа требований был спроектирован простой и интуитивно понятный графический пользовательский интерфейс. Макеты разрабатываемых окон и описание полей приведены на рисунках 12 – 21.

Макет окна входа в систему представлен на рисунке 12.



1 – текстовое поле для ввода логина;

2 – поле для ввода пароля;

3 – кнопка для подтверждения входа.

Рисунок 12 – Макет окна авторизации

Макет главного окна приложения представлен на рисунке 13.

Изображение выглядит как текст, в помещении

Автоматически созданное описание

1. – пункт меню для открытия окна для работы с моделями;

2 – пункт меню для открытия окна работы с товарами;

3 – пункт меню для открытия меню операций;

1. – пункт меню для открытия окна работы администратора;
2. – таблица для отображения списка пунктов хранения;
3. – кнопка добавления пункта хранения;
4. – кнопка удаления выделенного пункта хранения;
5. – кнопка редактирования информации о пунктах хранения;
6. – обновление информации;

10 – таблица отображения остатков по пунктам хранения.

Рисунок 13 – Макет главного окна приложения

Макет окна для создания новых пунктов хранения и редактирования существующих пунктов хранения представлен на рисунке 14.

Изображение выглядит как текст, в помещении

Автоматически созданное описание

1 – текстовое поле для ввода названия пункта хранения;

2 – текстовое поле для ввода адреса;

3 – кнопка для сохранения введенной информации.

Рисунок 14 – Макет окна добавления/редактирования пунктов хранения

Макет окна, предназначенного для отображения списка моделей одежды и работы с указанным списком, представлен на рисунке 15.

Изображение выглядит как текст, спортивная игра, снимок экрана

Автоматически созданное описание

1. – таблица для отображения списка моделей одежды;

2 – кнопка удаления выделенной модели;

3 – кнопка добавления новой модели.

Рисунок 15 – Макет окна для работы с моделями одежды

Макет окна для добавления новых моделей одежды представлен на рисунке 16.

Изображение выглядит как диаграмма

Автоматически созданное описание

1 – текстовое поле для ввода названия модели;

2 – поле для ввода артикула модели;

3 – раскрывающийся список для выбора цвета модели;

4 – кнопка выбора фото;

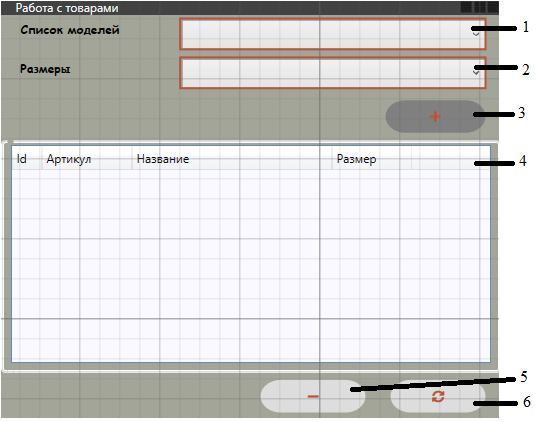
5 – поле ввода закупочной цены;

6 – поле отображения выбранного фото;

7 – кнопка для сохранения информации о модели.

Рисунок 16 – Макет окна для добавления новой модели

Макет окна, предназначенного для добавления новых товаров, отображения списка товаров и удаления товаров из списка, представлен на рисунке 17.



1 – раскрывающийся список для выбора модели;

2 – раскрывающийся список для выбора размера;

3 – кнопка добавления товара в список;

4 – таблица для отображения добавленных товаров;

5 – кнопка для удаления выбранного товара из списка;

6 – кнопка для обновления данных.

Рисунок 17 – Макет окна работы с товарами

Макет окна для отображения списка заявок на перемещение товаров представлен на рисунке 18.



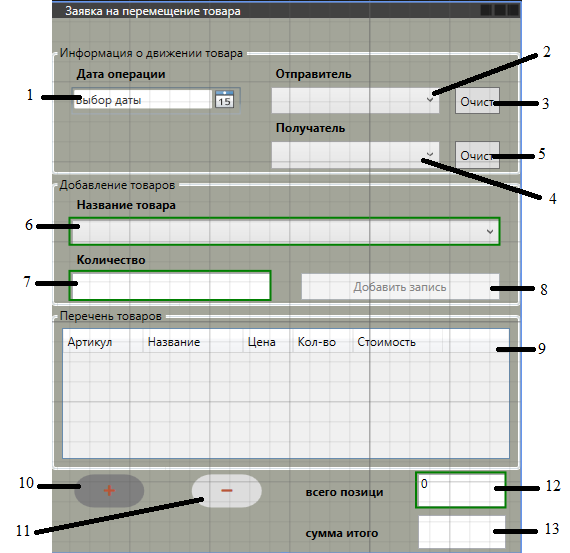
1 – таблица для отображения списка заявок на перемещение;

2 – кнопка для удаления выбранной заявки;

3 – кнопка для печати заявки.

Рисунок 18 – Макет окна просмотра списка заявок на перемещение

Макет окна для создания заявок на внутреннее перемещение товаров представлен на рисунке 19.



1 – элемент «Календарь» для выбора даты отгрузки;

2 – раскрывающийся список для выбора пункта отгрузки;

3 – кнопка для очистки поля 2;

4 - раскрывающийся список для выбора пункта назначения;

5 – кнопка для очистки поля 4;

6 - раскрывающийся список для выбора товара;

7 – поле для ввода количества единиц товара;

8 – кнопка для добавления записи о перемещаемом товаре;

9 – таблица для отображения записей о перемещаемых товарах;

10 – кнопка для сохранения заявки на перемещение;

11 – кнопка для удаления выделенной записи о товаре из таблицы;

12 – поле для отображения о количестве позиций в заявке;

13 – поле для отображения общей суммы товаров в заявке.

Рисунок 19 – Макет окна оформления заявки на перемещение товаров

Макет окна для работы администратора представлен на рисунке 20.

Изображение выглядит как диаграмма

Автоматически созданное описание

1 - таблица для отображения списка пользователей;

2 – кнопка для добавления нового пользователя;

3 – кнопка для редактирования информации о пользователе;

4 – кнопка для удаления пользователя.

Рисунок 20 – Макет окна для работы администратора

Макет окна для добавления новых пользователей и редактирования уже существующих представлен на рисунке 21.

Изображение выглядит как текст, небо, снимок экрана, день

Автоматически созданное описание

1 - поле для ввода фамилии пользователя;

2 – поле для ввода имени пользователя;

3 – поле для ввода отчества пользователя;

4 – поле для ввода логина пользователя;

5 - поле для ввода пароля пользователя;

6 - раскрывающийся список для выбора ролей;

7 – кнопка для сохранения введенной роли;

8 - кнопка для сохранения введенной информации.

Рисунок 21 – Макет окна для создания пользователя

**4 РАЗРАБОТКА АЛГОРИТМОВ И ИХ ОПИСАНИЕ**

Одним из требований к разрабатываемому программному средству является расчет остатков товаров в разрезе пунктов хранения.

Алгоритм перерасчета остатков товаров представляет собой последовательность действий, которая выполняется при осуществлении пользователем операций по перемещению товаров. Для каждой записи в заявке на перемещение выполняются алгоритмы по пересчету остатков товаров в пункте отгрузки и в пункте назначения. Алгоритм представлен на рисунке 22.

Рисунок 22 – Алгоритм перерасчета остатков товаровИзображение выглядит как диаграмма

Автоматически созданное описание

Алгоритм перерасчета остатков товаров в пункте отгрузки представляет собой последовательность действий, которая выполняется при расчете остатков в пункте отгрузки. В зависимости от того, указан ли пункт отгрузки, осуществляется пересчет остатков товаров. Если товаров для отгрузки не достаточно, программа отображает сообщение об ошибке. Алгоритм пересчета остатков товаров в пункте отгрузки представлен на рисунке 23.

Рисунок 23 – Алгоритм пересчета остатков товара в пункте отгрузки

На рисунке 24 представлен алгоритм пересчета остатков товаров в пункте назначения. Он представляет собой последовательность действий, которая выполняется при расчете остатков в пункте назначения. В зависимости от того, указан ли пункт назначения, осуществляется пересчет остатков товаров. Если данной категории товаров нет в остатках по пункту назначения, то пересчет остатков происходит после добавления товара в остатки.Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 24 – Алгоритм пересчета остатков товара в пункте назначенияИзображение выглядит как диаграмма

Автоматически созданное описание

Исходный код алгоритмов приведен в приложении А.

**5 ТЕСТИРОВАНИЕ ПОЛУЧЕННОГО ПРОГРАММНОГО ПРОДУКТА**

Тестирование программного обеспечения является очень важным и в то же время трудоемким видом деятельности. Тестирование – это прежде всего оценка промежуточных продуктов, созданных в процессе разработки программного обеспечения. Это гораздо более широкое поле деятельности, нежели просто проверка некоторых частей или программной системы в целом с целью определения степени ее соответствия техническим требованиям [7].

В данной работе была выбрана методология функционального тестирования.

Функциональное тестирование – это тестирование программного обеспечения в целях проверки реализуемости функциональных требований, то есть способности программного обеспечения в определенных условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает программное обеспечение, какие задачи оно решает [8].

Функциональное тестирование предполагает составление плана тестирования, который строится на основе вариантов использования и включает в себя описание тестов и их результаты.

План тестирования приведен в таблице 13.

Таблица 14 – План тестирования

| **Вариант**  **использования** | **Описание теста** | **Результат** |
| --- | --- | --- |
| Добавление модели | Запускаем приложение. Выбираем пункт меню «Модели» => «Список моделей». Нажимаем кнопку «+». Появляется окно «Новая модель». Заполняем необходимую информацию. Нажимаем кнопку «Сохранить».  **Ожидаемый результат:** в список моделей должна добавиться новая модель. | Тест пройден. Новая модель добавлена в список моделей. |
| Удаление модели | Запускаем приложение. Выбираем пункт меню «Модели» => «Список моделей». Из списка выбираем модель для удаления. Нажимаем кнопку «-». Появляется окно-подтверждение. Нажимаем кнопку «OK».  **Ожидаемый результат:** выбранная из списка модель должна быть удалена из списка. | Тест пройден, выбранная модель удалена из списка. |

Продолжение таблицы 14

| Добавление товара | Запускаем приложение. Выбираем пункт меню «Товары» => «Добавить новый товар». Появляется окно «Перечень товаров». Выбираем необходимую информацию. Нажимаем кнопку «+».  **Ожидаемый результат:** в список товаров должен быть добавлен новый товар. | Тест пройден, новый товар добавлен в перечень товаров. |
| --- | --- | --- |
| Удаление товара | Запускаем приложение. Выбираем пункт меню «Товары» => «Добавить новый товар». Появляется окно «Перечень товаров». Из списка выбираем товар для удаления. Нажимаем кнопку «-».  **Ожидаемый результат:** выбранный товар должен быть удален из перечня товаров. | Тест пройден, выделенный товар удален из перечня. |
| Создание новой операции перемещения | Запускаем приложение. Выбираем пункт меню «Операции» => «Новая операция». Появляется окно «Заявка на перемещение товара». Заполняем сектор «Информация о движении товара», далее сектор «Добавление товаров». Нажимаем кнопку «Добавить запись». Нажимаем кнопку «+».  **Ожидаемый результат:** в список заявок на перемещение товаров добавлена новая заявка. | Тест пройден, заявка на перемещение создана. |
| Удаление заявки на перемещение | Запускаем приложение. Выбираем пункт меню «Операции» => «Все операции». Появляется окно «Список перемещений». Из списка выбираем заявку для удаления. Нажимаем кнопку «-». Появляется окно-подтверждение. Нажимаем кнопку «OK».  **Ожидаемый результат:** выбранная заявка должна быть удалена из списка заявок на перемещение. | Тест пройден, заявка удалена из списка. |
| Добавление нового ПХ | Запускаем приложение. В главном окне нажимаем кнопку «+». Открывается окно «Пункт хранения». Заполняем необходимые поля и нажимаем кнопку «Save».  **Ожидаемый результат:** в список пунктов хранения должен быть добавлен новый пункт хранения. | Тест пройден, новый ПХ добавлен в список ПХ. |
| Редактирование ПХ | Запускаем приложение. В главном окне в списке ПХ выбираем пункт для редактирования. Нажимаем кнопку «Редактировать». Открывается окно «Пункт хранения». Изменяем данные и нажимаем кнопку «Save».  **Ожидаемый результат:** в информации о пункте хранения должны быть отображены внесенные изменения. | Тест пройден, данные о ПХ отредактированы. |

Продолжение таблицы 14

| Удаление ПХ | Запускаем приложение. В главном окне в списке ПХ выбираем пункт для удаления. Нажимаем кнопку «-». Открывается окно подтверждения удаления. Нажимаем кнопку «ОК».  **Ожидаемый результат:** выбранный ПХ должен быть удален из списка пунктов хранения. | Тест пройден, выбранный ПХ удален из списка. |
| --- | --- | --- |
| Просмотр остатков в разрезе ПХ | Запускаем приложение. В главном окне в списке ПХ выбираем ПХ, по которому нужно просмотреть остатки.  **Ожидаемый результат:** в таблице отображения остатков должны отобразиться остатки по выбранному ПХ. | Тест пройден, в таблице выведены остатки по выбранному ПХ. |
| Добавление пользователя | Запускаем приложение. В главном окне выбираем пункт меню «Администратор». Открывается окно «Окно администратора». Нажимаем кнопку «+ пользователя». В открывшемся окне вводим данные. Нажимаем кнопку «Save».  **Ожидаемый результат:** в списокпользователей должна быть добавлена информация о новом пользователе. | Тест пройден, новый пользователь добавлен. |
| Редактирование информации о пользователе | Запускаем приложение. В главном окне выбираем пункт меню «Администратор». Открывается окно «Окно администратора». В списке выбираем пользователя для редактирования. Нажимаем кнопку «Редактировать». В появившемся окне меняем данные. Нажимаем кнопку «Save».  **Ожидаемый результат:** в информации о пользователе должны быть отображены внесенные изменения. | Тест пройден, запись отредактирована. |
| Авторизация в системе | Запускаем приложение. Вводим в поле «Логин» - логин, в поле «Пароль» - пароль. Нажимаем кнопку «Login».  **Ожидаемый результат:** пользователь авторизован, выполнен вход в приложение, отображено главное окно. | Тест пройден, отображено главное окно приложения. |

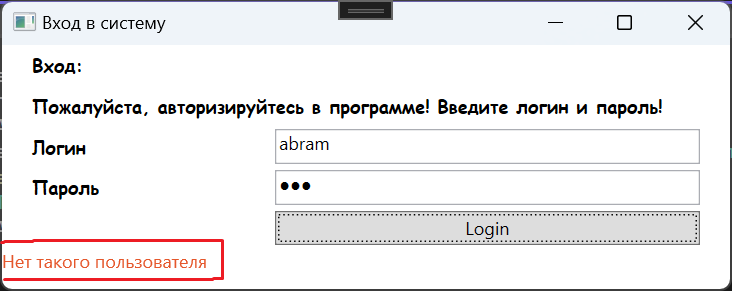
Уведомление системы при вводе неверного логина или пароля пользователя представлено на рисунке 25:

Рисунок 25 – Уведомление системы о вводе неверного логина или пароля

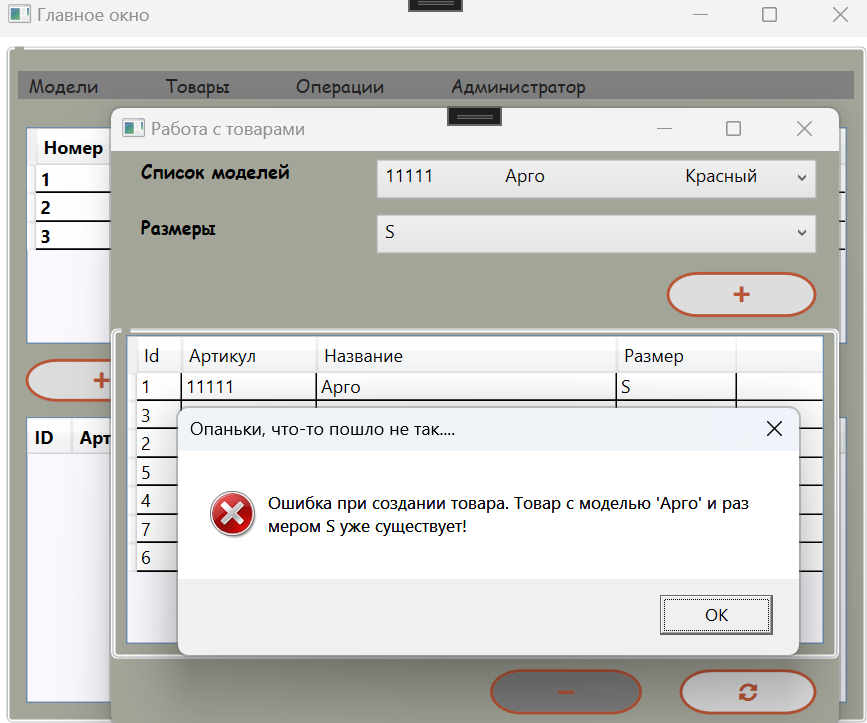
Уведомление системы при попытке добавления уже существующего товара представлено на рисунке 26:

Рисунок 26 – Уведомление системы при вводе уже существующего товара

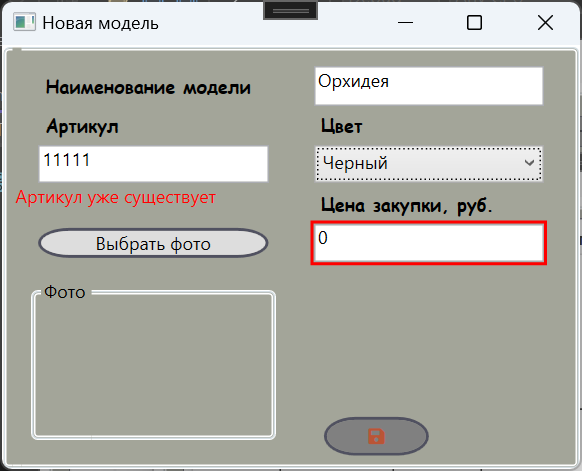
Уведомление системы при попытке добавления уже существующей модели представлено на рисунке 27:

Рисунок 27 – Уведомление системы при попытке добавления уже существующей модели

На основе анализа результатов тестирования можно сделать вывод, что разработанные программные средства информационной системы учета внутреннего перемещения товаров работают корректно.

**ЗАКЛЮЧЕНИЕ**

В данном курсовом проекте было создано программное средство, которое служит для автоматизации процесса внутреннего перемещения товаров между пунктами хранения и продаж. Цель разработки: минимизация временных затрат и повышение точности в учете внутреннего перемещения товаров.

Программа предоставляет пользователю возможность добавлять новые модели одежды, модели одежды в разрезе размеров, создавать заявки на перемещение товаров, быстро рассчитывать остатки товаров по пунктам хранения, создавать новые пункты хранения. Разработанное программное средство позволяет пользователю отслеживать остатки товаров с целью своевременного пополнения ассортимента и недопущения чрезмерного количества залежавшихся товаров в пунктах продаж.

Приложение реализовано посредством языка программирования С# (среда Visual Studio 2022 Community) с использованием следующего инструментария: Lucidchart, Microsoft SQL Server Management Studio 2018.

В результате выполнения данной работы разработано программное средство, отвечающее поставленным задачам. Несмотря на выполнение поставленных задач, в настоящее время намечены работы, которые помогут расширить функциональные возможности программного средства: добавление новой функциональности «Продавец», разработка функциональности по работе с поставщиками и покупателями.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Автоматизация складского учета – с чего начать? [Электронный ресурс]. – 2022. – Режим доступа: <https://skyg.ru/blog/avtomatizacziya-skladskogo-ucheta-s-chego-nachat/>. – Дата доступа: 29.12.2022.
2. Экспертный рейтинг программ складского учета товаров [Электронный ресурс]. - 2021. - Режим доступа: <https://www.ekam.ru/blogs/pos/ekspertnyy-reyting-programm-skladskogo-ucheta> . – Дата доступа: 09.01.2023.
3. 1C: Управление небольшой фирмой для Беларуси [Электронный ресурс]. - Режим доступа: <https://justsystems.by/upravlenie-nebolshoj-firmoj/> . – Дата доступа: 11.02.2023.
4. [Электронный ресурс]. - Режим доступа: <https://proresult.by/mojsklad/> . – Дата доступа: 02.02.2023.
5. Складской учет [Электронный ресурс]. - Режим доступа: <http://golubsoft.by/products/gs-uchet-uslugi-torgovlya-i-sklad-pri-usn>? . – Дата доступа: 14.02.2023.

# Постолаки, Н. Автоматизация складского учета: что нужно знать перед внедрением системы на складе [Электронный ресурс] / Н. Постолаки // Port. – 2021. – Режим доступа: <https://portkkm.ru/overview/avtomatizatsiya-skladskogo-ucheta-chto-nuzhno-znat-pered-vnedreniem-sistemy-na-sklade/> . – Дата доступа: 22.02.2023.

# Управление проектами // Википедия. 2001. [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Управление\_проектами . - Дата доступа: 03.03.2023.

# Функциональное тестирование // Википедия. 2001. [Электронный ресурс]. – Режим доступа: [https://ru.wikipedia.org/wiki/Функциональное\_тестирование](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5). – Дата доступа: 03.03.2023.

Приложение А

Программный код алгоритмов расчета остатков товаров на пунктах хранения

public void CalculateBalance()

{

foreach (var item in ItemMovs)

{

if (ShippingPoint != null)

{

var currentItem = ShippingPoint.Balances.First(y => y.Product == item.Product);

if (currentItem.Quantity>=item.Quantity)

{

ShippingPoint.Balances.First(y => y.Product == item.Product).Quantity -= item.Quantity;

}

else

{

throw new InsufficientAmountException($"Нет столько единиц товара '{item.Product.Model.Title}' на складе");

}

}

if (ReceivingPoint != null)

{

var b = ReceivingPoint.Balances.FirstOrDefault(y => y.Product == item.Product);

if (b == null)

{

ReceivingPoint.Balances.Add(new Balance()

{

Product = item.Product,

Quantity = 0

});

}

ReceivingPoint.Balances.First(y => y.Product == item.Product).Quantity += item.Quantity;

}

}

}

}

Приложение Б

Листинг кода разработанного приложения

// остатки товаров

namespace TermPaper.Domain.Entities

{

public class Balance

{

public int Id { get; set; }

public int WarehouseNumber { get; set; }

public int ProductId { get; set; }

public int Quantity { get; set; }

public virtual Warehouse Warehouse { get; set; }

public virtual Product Product { get; set; }

}

}

//Класс-справочник цветов

using System.Collections;

using System.Collections.Generic;

namespace TermPaper.Domain.Entities

{

public class Color

{

public int Id { get; set; }

public string Name { get; set; }

}

}

// отдельная запись в движении товара

namespace TermPaper.Domain.Entities

{

public class ItemMov

{

public int Number { get; set; }

public int MovementId { get; set; }

public int ProductId { get; set; }

public int Quantity { get; set; } // кол-во, штук

public decimal Sum { get => Quantity \* Product.Model.Price; }

public virtual Product Product { get; set; }

public virtual Movement Movement { get; set; }

}

}

using System.Collections.Generic;

//Класс модель

namespace TermPaper.Domain.Entities

{

public class Model

{

public int Article { get; set; } //PK

public string Title { get; set; }

public decimal Price { get; set; }

public int ColorId { get; set; }

public byte[]? Photo { get; set; }

public virtual Color Color { get; set; }

public virtual ICollection<Product> Products { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using TermPaper.Domain.Entities.Exceptions;

//движение товаров

namespace TermPaper.Domain.Entities

{

public class Movement

{

public int Id { get; set; }

public int? ShippingPointId { get; set; } // пункт отгрузки

public int? ReceivingPointId { get; set; } //пункт приемки

public DateTime ShippingDate { get; set; } // дата отгрузки

public decimal TotalAmount { get; set; } // общая сумма отгрузки

public virtual ICollection<ItemMov> ItemMovs { get; set; }

public virtual Warehouse ShippingPoint { get; set; }

public virtual Warehouse ReceivingPoint { get; set; }

public void CalculateBalance()

{

foreach (var item in ItemMovs)

{

if (ShippingPoint != null)

{

var currentItem = ShippingPoint.Balances.First(y => y.Product == item.Product);

if (currentItem.Quantity>=item.Quantity)

{

ShippingPoint.Balances.First(y => y.Product == item.Product).Quantity -= item.Quantity;

}

else

{

throw new InsufficientAmountException($"Нет столько единиц товара '{item.Product.Model.Title}' на складе");

}

}

if (ReceivingPoint != null)

{

var b = ReceivingPoint.Balances.FirstOrDefault(y => y.Product == item.Product);

if (b == null)

{

ReceivingPoint.Balances.Add(new Balance()

{

Product = item.Product,

Quantity = 0

});

}

ReceivingPoint.Balances.First(y => y.Product == item.Product).Quantity += item.Quantity;

}

}

}

}

}

// отдельный товар

using System.Collections.Generic;

namespace TermPaper.Domain.Entities

{

public class Product

{

public int Id { get; set; }

public int Article { get; set; } //артикул модели

public int SizeId { get; set; }

public virtual Size Size { get; set; }

public virtual Model Model { get; set; }

public virtual IEnumerable<Balance> Balances { get; set; }

}

}

//Класс-справочник размеров

namespace TermPaper.Domain.Entities

{

public class Size

{

public int Id { get; set; }

public string Name { get; set; }

}

}

//Пункты хранения

using System.Collections.Generic;

namespace TermPaper.Domain.Entities

{

public class Warehouse

{

public int Number { get; set; } //номер ПХ

public string Name { get; set; }

public string Address { get; set; }

public virtual ICollection<Balance> Balances { get; set; }

}

}

using System.Collections.Generic;

// класс для хранения инф-ии о пользователе

namespace TermPaper.Domain.Entities

{

public class User

{

public int Id { get; set; }

public string Login { get; set; }

public string Password { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public string? SecondName { get; set; }

public virtual IEnumerable<UserRole> Roles { get; set; }

public string FullName => $"{LastName} {FirstName} {SecondName}".Trim();

/// <summary>

/// Возвращает значение, верный ли пароль для этого пользователя

/// </summary>

/// <param name="pwd"></param>

/// <returns>

/// Если pwd равен паролю из Password - вернет true

/// </returns>

public bool Authenticate(string pwd) => Password == pwd;

}

}

using System.Collections.Generic;

//класс для хранения инф-ии о польз-их ролях

namespace TermPaper.Domain.Entities

{

public class UserRole

{

public int Id { get; set; }

public string Name { get; set; }

public virtual ICollection<User>? Users { get; set; }

}

}

//Ошибка поле не может быть пустым

using System.Globalization;

using System.Windows.Controls;

namespace TermPaper.ViewModels.Validations

{

public class NotEmptyValidation : ValidationRule

{

public override ValidationResult Validate(object value, CultureInfo cultureInfo)

{

var r = value?.ToString()?.Length > 0;

return new ValidationResult(r, "Поле не может быть пустым");

}

}

}

using System.Collections.ObjectModel;

using System.Windows.Input;

using TermPaper.Commands;

using TermPaper.Domain.Entities;

using TermPaper.UseCases;

using TermPaper.UseCases.Interfaces;

namespace TermPaper.ViewModels

{

public class MainWindowViewModel : BaseViewModel

{

IGenericRepository<Warehouse> repository;

private IReportPrinter printer;

public MainWindowViewModel(IGenericRepository<Warehouse> repository, IReportPrinter printer)

{

this.repository = repository;

this.printer = printer;

Warehouses = new ObservableCollection<Warehouse>(repository.GetAll());

}

private ObservableCollection<Warehouse> warehouses;

public ObservableCollection<Warehouse> Warehouses

{

get => warehouses;

private set

{

Set(ref warehouses, value);

}

}

private Warehouse selectedItem;//храним выбранный пользователем склад, тк весь объект хранится

public Warehouse SelectedItem {

get

{

return selectedItem;

}

set

{

Set(ref selectedItem, value);

}

}

#region Добавить новый или редактировать склад

private ICommand \_addOrEditWarehouseCommand;

public ICommand AddOrEditWarehouseCommand => \_addOrEditWarehouseCommand ?? new RelayCommand(OnAddOrEdit);

private void OnAddOrEdit(object p)

{

if (SelectedItem.Number == 0)

{

repository.Create(SelectedItem);

}

else

{

repository.Update(SelectedItem);

}

OnRefresh(null);

}

#endregion

#region Удаление ПХ

private RelayCommand \_deleteCommand;

public ICommand DeleteCommand => \_deleteCommand ?? new RelayCommand(OnDeleteWarehouseEx, (obj) => SelectedItem != null);

private void OnDeleteWarehouseEx(object p)

{

repository.Delete(SelectedItem);

OnRefresh(null);

}

#endregion

// команда для обновления данных

public override ICommand RefreshCommand => refreshCommand ??= new RelayCommand(OnRefresh);

private ICommand refreshCommand;

private void OnRefresh(object param)

{

repository.Cancel(SelectedItem); //при редактировании склада, если польз отказался, для отмены изменений в окне

Warehouses = new ObservableCollection<Warehouse>(repository.GetAll());

SelectedItem = null;

}

#region Печать отчета по остаткам

private ICommand \_printReportCommand;

public ICommand PrintReportCommand => \_printReportCommand ?? new RelayCommand(OnPrintReport, (p) =>

{

return SelectedItem != null;

});

private void OnPrintReport(object p)

{

printer.PrintReport(SelectedItem);

}

#endregion

}

}

<Window x:Class="TermPaper.Views.MainWindow"

x:Name="\_MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:TermPaper"

xmlns:vm="clr-namespace:TermPaper.ViewModels"

xmlns:fa5="http://schemas.fontawesome.com/icons/"

mc:Ignorable="d"

Title="Главное окно" Height="500" Width="600"

d:DataContext="{d:DesignInstance Type=vm:MainWindowViewModel}">

<Window.InputBindings>

<KeyBinding Command="{Binding AddNewCommand, ElementName=\_MainWindow}" Gesture="Insert"/>

</Window.InputBindings>

<Window.Resources>

<Style TargetType="DataGrid" BasedOn="{StaticResource BorderData}"/>

<Style TargetType="GroupBox">

<Setter Property="Padding" Value="3"/>

<Setter Property="Margin" Value="5"/>

<Setter Property="FontWeight" Value="Bold"/>

</Style>

</Window.Resources>

<GroupBox Background="#A3A599">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="0.3\*"/>

<RowDefinition/>

<RowDefinition Height="0.25\*"/>

<RowDefinition/>

<RowDefinition Height="0.3\*"/>

</Grid.RowDefinitions>

<Menu Background="Gray" Grid.Row="0" VerticalAlignment="Center" FontFamily="Comic Sans MS" BorderBrush="#505160" BorderThickness="1">

<MenuItem Header="Модели">

<MenuItem Header="Список моделей" Command="{Binding ShowListCommand, ElementName=\_MainWindow}"/>

</MenuItem>

<Separator/>

<MenuItem Header="Товары">

<MenuItem Header="Добавить новый товар" Command="{Binding ShowListProductsCommand, ElementName=\_MainWindow}"/>

</MenuItem>

<Separator/>

<MenuItem Header="Операции">

<MenuItem Header="Новая операция" Command="{Binding ShowMovementCommand, ElementName=\_MainWindow}"/>

<MenuItem Header="Все операции" Command="{Binding ShowMovementsList, ElementName=\_MainWindow}"/>

</MenuItem>

<Separator/>

<MenuItem Header="Администратор" Command="{Binding ShowAdminWindowCommand, ElementName=\_MainWindow}"/>

<Separator/>

</Menu>

<!--<ToolBar Background="Gray" Grid.Row="1">

</ToolBar>-->

<DataGrid IsReadOnly="True" SelectedItem="{Binding Path=SelectedItem}" AutoGenerateColumns="False" Grid.Row="1" x:Name="WarehouseGrid" Margin="5" Background="#F9F9FF" ItemsSource="{Binding Warehouses}">

<DataGrid.InputBindings>

<KeyBinding Command="{Binding DeleteWarehousetWithConfirmationCommand, ElementName=\_MainWindow}" Gesture="Delete"/>

<KeyBinding Command="{Binding EditWarehouseCommand, ElementName=\_MainWindow}" Gesture="CTRL+E"/>

</DataGrid.InputBindings>

<DataGrid.Columns>

<DataGridTextColumn Header="Номер" Width="70" Binding="{Binding Path=Number}"/>

<DataGridTextColumn Header="Название" Width="140" Binding="{Binding Path=Name}"/>

<DataGridTextColumn Header="Адрес" Width="310" Binding="{Binding Path=Address}"/>

</DataGrid.Columns>

</DataGrid>

<Grid Grid.Row="2">

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Button Command="{Binding AddNewCommand, ElementName=\_MainWindow}" Style="{StaticResource ButtonAdd}"/>

<Button Grid.Column="1" Command="{Binding DeleteWarehousetWithConfirmationCommand, ElementName=\_MainWindow}" Style="{StaticResource ButtonDelete}"/>

<Button Grid.Column="2" Command="{Binding EditWarehouseCommand, ElementName=\_MainWindow}" Style="{StaticResource ButtonEdit}"/>

<Button Grid.Column="3" Command="{Binding RefreshCommand}" Style="{StaticResource ButtonRefresh}"/>

<Button Grid.Column="4" Command="{Binding PrintReportCommand}" Style="{StaticResource ButtonPrint}"/>

</Grid>

<DataGrid IsReadOnly="True" AutoGenerateColumns="False" Grid.Row="3" Grid.RowSpan="2" x:Name="NaklGrid" ItemsSource="{Binding SelectedItem.Balances}" Margin="5" Background="#F9F9FF">

<DataGrid.Columns>

<DataGridTextColumn Header="ID" Width="30" Binding="{Binding Path=Id}"/>

<DataGridTextColumn Header="Артикул" Width="80" Binding="{Binding Path=Product.Article}"/>

<DataGridTextColumn Header="Модель" Width="100" Binding="{Binding Path=Product.Model.Title}"/>

<DataGridTextColumn Header="Размер" Width="70" Binding="{Binding Path=Product.Size.Name}"/>

<DataGridTextColumn Header="Цвет" Width="90" Binding="{Binding Path=Product.Model.Color.Name}"/>

<DataGridTextColumn Header="Цена" Width="60" Binding="{Binding Path=Product.Model.Price}"/>

<DataGridTextColumn Header="Кол-во" Width="90" Binding="{Binding Path=Quantity}"/>

</DataGrid.Columns>

</DataGrid>

</Grid>

</GroupBox>

</Window>

Для работы с авторизацией

using Microsoft.IdentityModel.Tokens;

using System;

using System.Linq;

using System.Windows.Input;

using TermPaper.Commands;

using TermPaper.Domain.Entities;

using TermPaper.UseCases;

namespace TermPaper.ViewModels

{

public class HarvestPasswordEventArgs : EventArgs

{

public string Password;

}

public class LoginViewModel : BaseViewModel

{

User? authUser;

IGenericRepository<User> userRepository;

public event EventHandler<HarvestPasswordEventArgs> HarvestPassword;

public event Action LoginSuccess;

public LoginViewModel(IGenericRepository<User> userRepository)

{

this.userRepository = userRepository;

}

public User? AuthUser

{

get => authUser;

set

{

authUser = value;

Set(ref authUser, value);

}

}

private string loginError=String.Empty;

public string LoginError {

get => loginError;

private set

{

Set(ref loginError, value);

}

}

private string? login;

public string? Login

{

get => login;

set

{

Set(ref login, value);

}

}

private ICommand \_loginCommand;

public ICommand LoginCommand => \_loginCommand ?? new RelayCommand((p) =>

{

LoginError = String.Empty;

var password = GetPassword();

User dbUser =userRepository.FindBy(user=>user.Login==Login).FirstOrDefault();

if (dbUser == null)

{

LoginError = "Нет такого пользователя";

return;

}

if (dbUser.Authenticate(password))

{

authUser = null;

AuthHelper.Me = dbUser;

LoginSuccess?.Invoke();

}

else

{

LoginError = "Неверный пароль";

}

},

(p) => !Login.IsNullOrEmpty());

public override ICommand RefreshCommand => throw new System.NotImplementedException();

private string GetPassword()

{

var pwargs = new HarvestPasswordEventArgs();

HarvestPassword?.Invoke(this, pwargs);

return pwargs.Password;

}

}

}

<Window x:Class="TermPaper.Views.WorkWithUser"

Name="\_window"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:validations="clr-namespace:TermPaper.ViewModels.Validations"

xmlns:local="clr-namespace:TermPaper.Views"

xmlns:vm="clr-namespace:TermPaper.ViewModels"

mc:Ignorable="d"

d:DataContext="{d:DesignInstance Type=vm:AdminViewModel}"

Title="Окно администратора" Height="370" Width="610" >

<Window.Resources>

<Style TargetType="Label" BasedOn="{StaticResource LabelStyle}"/>

<Style TargetType="TextBox">

<Setter Property="ToolTip" Value="{Binding RelativeSource={RelativeSource Self}, Path=(Validation.Errors)[0].ErrorContent}"/>

<Setter Property="Margin" Value="15 0"/>

<Setter Property="Height" Value="30"/>

</Style>

<ControlTemplate x:Key="validationFailed">

<Border BorderBrush="#BA5536" BorderThickness="2">

<AdornedElementPlaceholder />

</Border>

</ControlTemplate>

</Window.Resources>

<Grid Background="AliceBlue">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition Height="0.8\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<StackPanel Orientation="Vertical">

<Label Content="Фамилия"/>

<TextBox x:Name="lastName">

<TextBox.Text>

<Binding Path="SelectedItem.LastName" UpdateSourceTrigger="PropertyChanged">

<Binding.ValidationRules>

<ExceptionValidationRule/>

<validations:NotEmptyValidation ValidatesOnTargetUpdated="True"/>

</Binding.ValidationRules>

</Binding>

</TextBox.Text>

</TextBox>

</StackPanel>

<StackPanel Grid.Column="1" Orientation="Vertical">

<Label Content="Имя"/>

<TextBox x:Name="name">

<TextBox.Text>

<Binding Path="SelectedItem.FirstName" UpdateSourceTrigger="PropertyChanged">

<Binding.ValidationRules>

<ExceptionValidationRule/>

<validations:NotEmptyValidation ValidatesOnTargetUpdated="True"/>

</Binding.ValidationRules>

</Binding>

</TextBox.Text>

</TextBox>

</StackPanel>

<StackPanel Grid.Row="1" Orientation="Vertical">

<Label Content="Отчество"/>

<TextBox x:Name="secName" Text="{Binding SelectedItem.SecondName}"/>

</StackPanel>

<StackPanel Grid.Row="2" Grid.Column="1" Orientation="Vertical">

<Label Content="Выбрать роли"/>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<ListBox x:Name="role" Margin="15 0" Height="30" SelectionMode="Extended" ItemsSource="{Binding Roles}" DisplayMemberPath="Name" />

<Button Grid.Column="1" Margin="15 0" Command="{Binding SaveCommand, ElementName=\_adminWin}">

<Button.Style>

<Style TargetType="Button" BasedOn="{StaticResource ButtonSave}"/>

</Button.Style>

</Button>

</Grid>

</StackPanel>

<StackPanel Grid.Row="1" Grid.Column="1" Orientation="Vertical">

<Label Content="Ввод логина"/>

<TextBox x:Name="login">

<TextBox.Text>

<Binding Path="SelectedItem.Login" UpdateSourceTrigger="PropertyChanged">

<Binding.ValidationRules>

<ExceptionValidationRule/>

<validations:NotEmptyValidation ValidatesOnTargetUpdated="True"/>

</Binding.ValidationRules>

</Binding>

</TextBox.Text>

</TextBox>

</StackPanel>

<StackPanel Grid.Row="2" Orientation="Vertical">

<Label Content="Ввод пароля"/>

<PasswordBox x:Name="pWord" Margin="15 0" Height="30"/>

</StackPanel>

<Button Grid.Row="3" Margin="25 15" Command="{Binding SaveUserCommand, ElementName=\_window}">

<Button.Style>

<Style TargetType="Button" BasedOn="{StaticResource ButtonSave}">

<Setter Property="IsEnabled" Value="False"/>

<Style.Triggers>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding Path=(Validation.HasError),ElementName=lastName}" Value="False"/>

<Condition Binding="{Binding Path=(Validation.HasError),ElementName=name}" Value="False"/>

<Condition Binding="{Binding Path=(Validation.HasError),ElementName=login}" Value="False"/>

<Condition Binding="{Binding Path=(Validation.HasError),ElementName=pWord}" Value="False"/>

<Condition Binding="{Binding Path=(Validation.HasError),ElementName=role}" Value="False"/>

</MultiDataTrigger.Conditions>

<Setter Property="IsEnabled" Value="True"/>

</MultiDataTrigger>

</Style.Triggers>

</Style>

</Button.Style>

</Button>

</Grid>

</Window>