ИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 «Программное обеспечение информационных

технологий»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
КУРСОВОГО ПРОЕКТА:**

по дисциплине             «Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема «Система Управления Складом»

Исполнитель

студент 2 курса группы 4                                                  Кравченко Н.В.

(Ф.И.О.)

Руководитель работы                                                 ассистент         Северинчик Н. А.

(учен. степень, звание, должность, подпись, Ф.И.О.)  
  
Курсовой проект защищен с оценкой

Председатель Пацей Н.В. (подпись)

Минск 2021

Содержание

[Введение 3](#_Toc42775075)

[1 Постановка задачи 4](#_Toc42775076)

[1.1 Средства разработки 4](#_Toc42775077)

[1.2 Обзор прототипов и аналогов 5](#_Toc42775078)

[2 Разработка архитектуры проекта 7](#_Toc42775079)

[2.1 Обобщенная структура 7](#_Toc42775080)

[2.2 Диаграмма UML 8](#_Toc42775081)

[2.3 Диаграмма классов 8](#_Toc42775082)

[2.4 Проектирование базы данных 11](#_Toc42775083)

[3 Разработка функциональной модели 14](#_Toc42775084)

[3.1 Описание функций приложения и MVVM 14](#_Toc42775085)

[3.2 Основные окна приложения](#_Toc42775087) 21

[3.2.1 Главное окно приложения](#_Toc42775088) 22

[3.3 Основные страницы приложения](#_Toc42775090) 23

[3.3.1 Страница авторизации](#_Toc42775091) 24

[3.3.2 Страница регистрации](#_Toc42775092) 25

[3.3.3 Страница клиентов](#_Toc42775093) 27

[3.3.4 Страница товаров 28](#_Toc42775094)

[3.3.5 Страница заказов](#_Toc42775095) 29

[3.3.6 Страница дополнительной информации о заказе 30](#_Toc42775096)

[5 Тестирование, проверка работоспособности и анализ полученных ре-зультатов 21](#_Toc42775100)

[6 Руководство по установке и использования 23](#_Toc42775101)

[Заключение 25](#_Toc42775102)

[Список используемых источников 26](#_Toc42775103)

[Приложение A. UML-диаграмма проекта «WarehouseManagementSystem» 27](#_Toc42775104)

Введение

С каждым годом различные отрасли промышленности растут в достаточно быстрой прогрессии. В целом система управления товарами используется в различных отраслях промышленности, от производства до коммунальных услуг, здравоохранения, образования, государственного управления и т.д.

Моё приложение инвентаризации будет процессом организации и хранения достаточного количества продуктов на складе для удовлетворения спроса, а также контроля потока и поддержания запасов, чтобы гарантировать, что нужное количество инвентаря доступно в нужное время и нужного качества.

Это приложение инвентаризации позволяет пользователю управлять подсчетом товаров и их движением, заказами на их покупку и продажу, импортом и экспортом, управлять поставщиками и клиентами в любом бизнесе.

Это поможет пользователю сделать многие из этих процессов автоматическими без необходимости запоминать детали, связанные с заказом, обработкой, комплектованием и упаковкой.

Целью курсового проекта является: развитие практических навыков вёрстки приложений на базе WPF с использованием языка разметки — XAML, и языка программирования — С#, показать навыки разработки дизайна, макета для приложения и умение грамотно разделить информацию для пользователя с использованием интересных цветовых сочетаний.

# Постановка задачи

Основной задачей курсового проекта является разработка приложения с возможностью управлять подсчетом товаров и их движением, заказами на их покупку и продажу, импортом и экспортом, управлять поставщиками и клиентами, что поможет пользователям сделать многие из этих процессов автоматическими без необходимости запоминать детали, связанные с заказом.

Функционально данное программное средство должно выполнять следующие задачи:

* выполнять регистрацию и авторизацию;
* поддерживать работу с базой данных;
* управлять клиентами, товарами, заказами, пользователями
* отслеживать количество товаров на складе, их названия, описание, фотографий, категорий, стоимости;
* добавлять новые товары, клиентов, пользователей
* обновлять товары и их данные
* осуществлять удаление товаров
* осуществлять поиск товаров и клиентов

## Средства разработки

При разработке приложения были использованы:

* среда разработки Microsoft Visual Studio 2019;
* технология WPF;
* объектно-ориентированная технология Entity Framework на базе фреймворка .NET
* система управления Microsoft SQL Server.

Microsoft Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Интегрированная среда разработки (IDE) представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения. Помимо стандартного редактора и отладчика, которые существуют в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для упрощения процесса разработки.

Windows Presentation Foundation (WPF) — это система следующего поколения для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем. C WPF можно создавать широкий спектр как автономных приложений, так и приложений, размещенных в веб-обозревателе. в основе WPF лежит векторная система визуализации, не зависящая от разрешения и созданная с расчетом на возможности современного графического оборудования. WPF расширяет базовую систему полным набором функций разработки приложений, элементами управления, привязкой данных, макетом, графикой, анимацией, стилями, шаблонами, документами, мультимедиа, текстом и оформлением. Именно использование WPF позволило гибко управлять дизайном интерфейса, также стало возможным подключение различных сторонних пакетов и использование паттернов.

Entity Framework представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работает с объектами.

Microsoft SQL Server — система управления реляционными БД. SQL Server Management Studio – утилита из Microsoft SQL Server, необходимая для управления и администрирования всех компонентов Microsoft SQL Server. Она в себя включает скриптовый редактор и графическую программу, которая работает с объектами и настройками сервера. SQL является, прежде всего, информационно-логическим языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных.

## Обзор прототипов и аналогов

В интернете есть десятки программ, которые позволяют предпринимателям вести учет в рознице, но большинство из них являются «сырыми» или мало функциональными.

Покупая случайную учетную программу для склада, есть риск, что она:

* неудобна для освоения;
* не поддерживает работу с имеющимся на складе оборудованием;
* не содержит критичного для работы функционала;
* не имеет круглосуточной службы поддержки
* раскрывает достаточные возможности только на максимальном тарифе.

Ниже перечислены критерии выбора программ учёта, которые нужно учесть предпринимателю:

1. Перечень поддерживаемых операций. Кому-то нужно знать просто приход/расход, а для кого-то важен дополнительно ценовой учет и аналитика продаж.

2. Стоимость внедрения и сопровождения. Нет смысла подробно вникать в обзор программы складского учета, если предприниматель не готов платить минимальный ежемесячный платеж.

3. Круглосуточная техническая поддержка.

4. Наличие дополнительных опциональных модулей (CMS, бухгалтерских, логистических). При планировании дисконтной системы CMS просто необходима.

5. Сетевые возможности. Например, для распределенных в пространстве складов будет актуальна только облачная программа складского учета.

6. Простота освоения. Новый сотрудник должен за несколько часов освоить основные возможности программы.

7. Стабильность работы. Программа не должна подвисать и перезагружаться, потому что это может приводить к потере последних введенных данных.

8. Наличие полнофункциональной демо-версии. Гораздо проще выбрать программу, загрузив ее полнофункциональную версию и испытав возможности

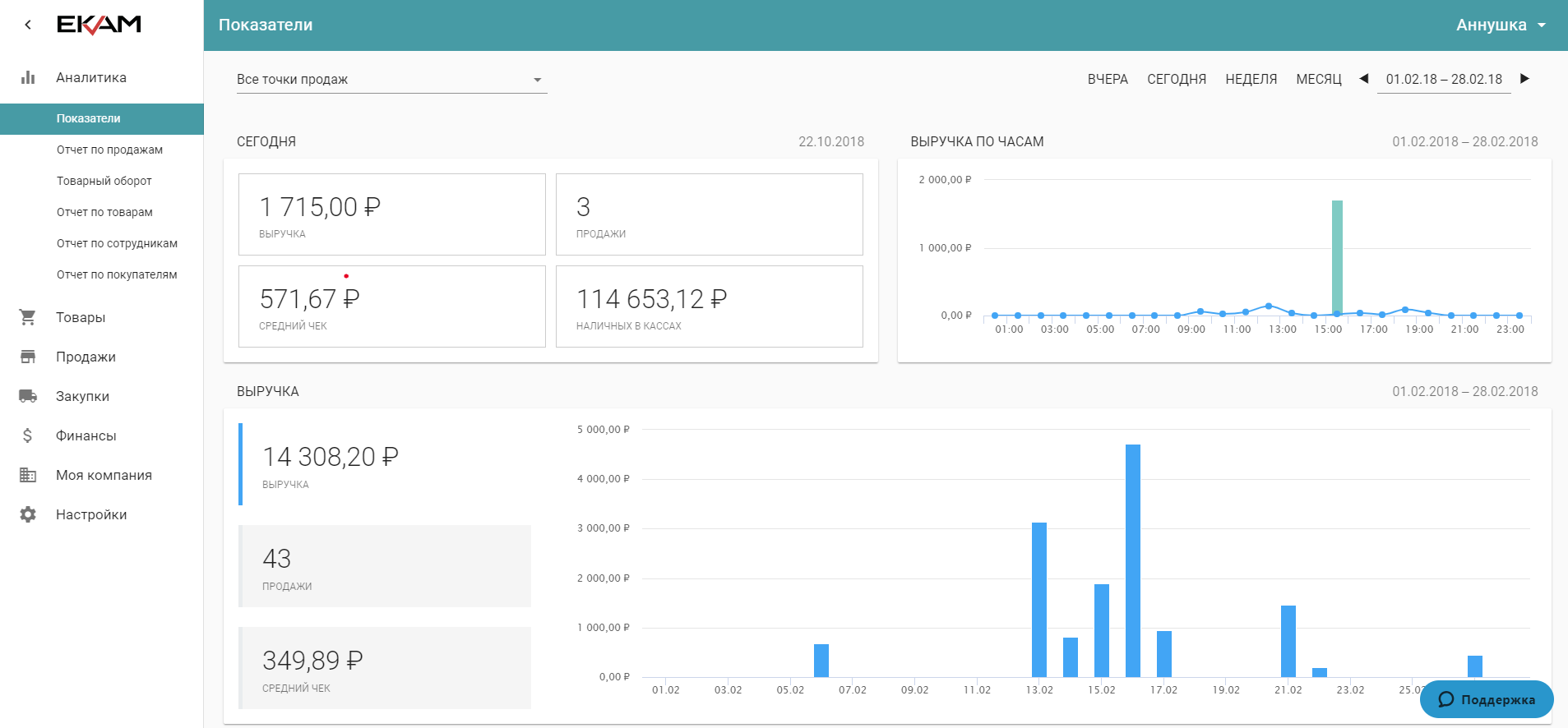
9. Открытый API, позволяющий дорабатывать программу под индивидуальные потребности клиента.

10.Удобный интерфейс. Переключение между меню во время работы должно занимать у персонала минимум времени.

Далее я приведу примеры программ для розничной торговли в разрезе соответствия их перечисленным критериям.

**«ЕКАМ»**

Товароучетная система «ЕКАМ» является одним из лидеров на СНГ рынке в своей нише и занимает высокие позиции в рейтингах.



Интерфейс программы складского учета «ЕКАМ» (рис. 1.1)

Программа работает через «облако» и предназначается для комплексной автоматизации малого и среднего бизнеса в сфере торговли и оказания услуг. [«ЕКАМ»](https://www.ekam.ru/page/sklad-programma) позволяет автоматизировать следующие операции:

* складской учет;
* продажи;
* закупки;
* управленческая отчетность;
* создание программ лояльности и дисконтных программ;
* прием и выполнение заказов в кафе и ресторанах;
* движение денег.

Также программа складского учета «ЕКАМ» интегрируется с бухгалтерскими приложениями, упрощая составление налоговой отчетности.

**Плюсы программы складского учета «ЕКАМ»:**

1. Оперативный и точный контроль за складскими остатками.
2. Автоматическая установка продажной цены, исходя из заданной наценки.
3. Удобное и быстрое проведение инвентаризации.
4. Интеллектуальная система закупок.
5. Удаленная работа с программой.
6. Большое разнообразие настраиваемых аналитических, финансовых, товарных и управленческих отчетов.
7. Возможность одновременного учета товаров в нескольких структурных подразделениях.
8. Загрузка прайсов поставщиков в программу, облегчающее создание номенклатуры.
9. Мультиплатформенность: программой можно пользоваться на компьютерах через браузер и на мобильных гаджетах через специальное приложение.
10. Приемлемые для малого бизнеса тарифные планы.
11. Подключение интернет-магазина.
12. Дружелюбный, понятный интерфейс, позволяющий самостоятельно проводить настройку программы и быстро обучать работе с программой новых сотрудников..
13. Наличие практичного модуля клиентской базы с возможностью привязки дисконтных карт.
14. Наличие полнофункциональной демо-версии, позволяющей 14 дней пользоваться всеми возможностями программы.
15. Наличие телефона круглосуточной технической поддержки.
16. Автоматическая передача данных о продажах в 1С.
17. Составление технологических карт для общепита.
18. Высокая стабильность работы
19. Открытый API для индивидуальной настройки под требования клиента.

**Минусы «ЕКАМ»:**

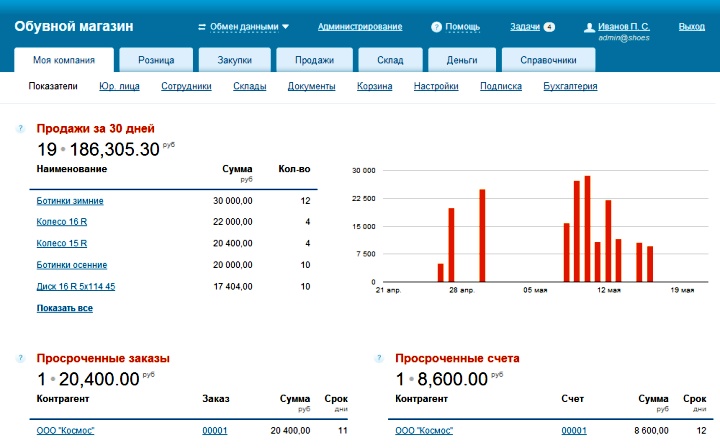
1. Отсутствует функционал по управлению доставкой.
2. Нет блока календаря и задач.

Одним из достоинств «ЕКАМ» является оперативная и профессиональная техническая поддержка. Её специалисты решают проблему как подсказками, так и при помощи удаленного подключения к компьютеру пользователя.

Настройка «ЕКАМ» производится сотрудниками компании. После этого предприниматели могут пользоваться всеми преимуществами приложения.

**«МойСклад»**

Рейтинг облачной программы складского учета «МойСклад» среди бизнесменов довольно высокий. Это обусловлено стабильностью работы приложения и достаточностью его функций для мелких предпринимателей.



Интерфейс программы складского учета «МойСклад» (рис. 1.2)

Кроме того, разработчики приложения не стали ограничиваться поддержкой торговли и добавили в функционал производственные операции.

Плюсы программы «МойСклад»:

1. Широкий функционал, подходящий для розничной, оптовой торговли, сферы общепита и небольших производств.

2. Поддержка дисконтных карт, создание клиентской базы, формирование воронки продаж.

3. Стабильная работа.

4. Дружелюбный интерфейс и легкость освоения программы новыми сотрудниками.

5.Наличие демо-версии с полным функционалом.

6.Мультиплатформенность: программой можно пользоваться на ОС Windows, macOS, Android, Linux, iOS.

7. Открытый API.

Минусы «МойСклад»:

1. Стоимость ежемесячной абонплаты выше среднерыночной.

2. Отсутствие шаблонов для продаж.

3. Тарифные планы программы складского учета МойСклад

4. Тарифные планы программы складского учета «МойСклад»

Можно сказать, что программа «МойСклад» создавалась для небольших магазинов и компаний, а приобретают ее и представители более крупного бизнеса. Но учитывать пожелания и замечания клиентов разработчики не спешат, что приводит к недовольству технической поддержкой и программой в целом.

Проанализировав плюсы и минусы данных товароучетных систем, которые являются одними из лидеров на СНГ рынке в своей нише и занимает высокие позиции в рейтингах я попытался реализовать плюсы и избежать минусы этих приложений.

# Разработка архитектуры проекта

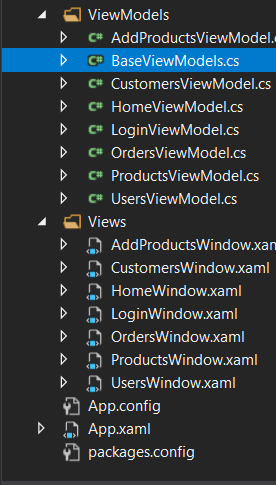
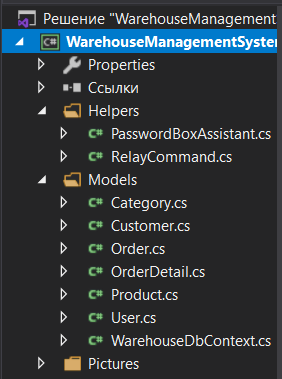
## Обобщенная структура

Решение представлено проектом «WarehouseManagementSystem» с имеющей структурой (Рисунок 2.1.).

Рисунок 2.1 – Обобщенная структура проекта

Описание обобщенной структуры проекта «WarehouseManagementSystem» представлено в таблице 2.1.

Таблица 2.1 – Обобщенная структура проекта



|  |  |
| --- | --- |
| Объект | Описание |
| Properties | свойства проекта, содержит информацию о сборке, используемых ресурсах и настройках |
| Ссылки | перечень сборок, используемых в проекте |
| Папка Helpers | привязка(соединение) команды с имеющей к ней прикладной логикой, отвечающей за обслуживание интерфейса |
| Папка Models | описывает используемые в приложении данные |
| Папка Pictures | содержит основные картинки проекта |
| Папка ViewModels | Связывает модель и представление, определяет логику по обновлению данных в модели |
| Папка Views | определяет визуальный интерфейс, через который пользователь взаимодействует с приложением |
| App.config | файл с параметрами проекта |
| App.xaml | класс Application |
| packages.config | файл для поддержки списка пакетов, на которые ссылается проект |

## Диаграмма UML

UML — унифицированный язык моделирования (Unified Modeling Language) — язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования системного проектирования и отображения организационных структур.

В данной программе существует две разновидности ролей пользователя. На диаграмме вариантов использования, представлены роли администратора и пользователя (Рисунок 2.2.).

Пользователю предоставляется выполнять регистрацию и авторизацию (с помощью адреса электронной почты и пароля), управление клиентами, товарами, заказами, пользователями, отслеживание количества товаров на складе, их названия, категорий, стоимости, описания, фотографий, добавление нового товара, категории, клиента, создание нового пользователя, обновление товара, удаление товара, поиск товара.

Администратору предоставляется поддерживать работу c базой данных и регистрировать пользователей приложения.

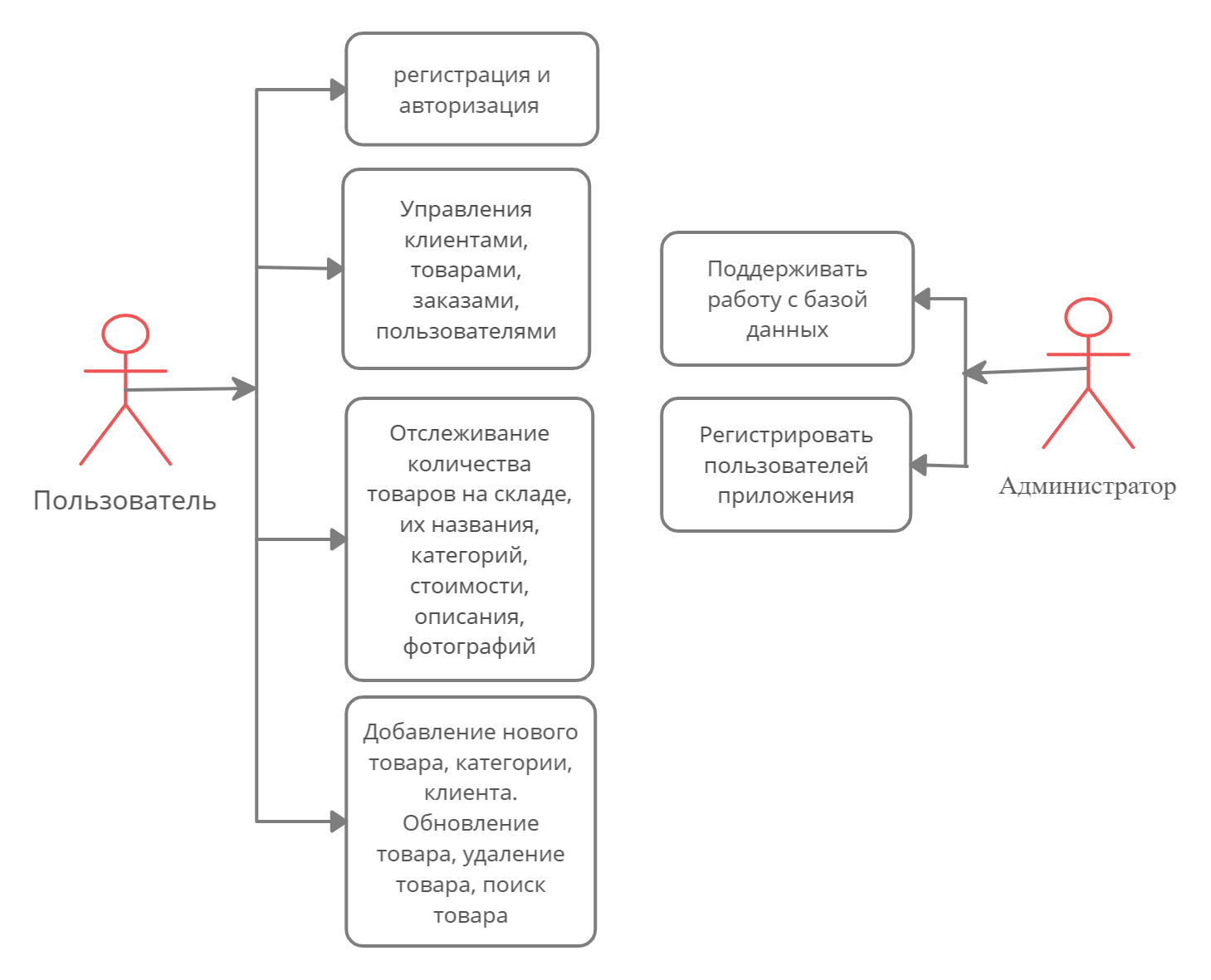


Рисунок 2.2 – Диаграмма вариантов использования

## Диаграмма классов

Структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов, методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

На данной диаграмме классов изображены классы программы, методы, поля и свойства классов. Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру (поля, методы) и типы отношений (наследование, реализация интерфейсов).

Диаграмма классов моделей представления (связывает модель и представление через механизм привязки данных) представлена на рисунке 2.3.

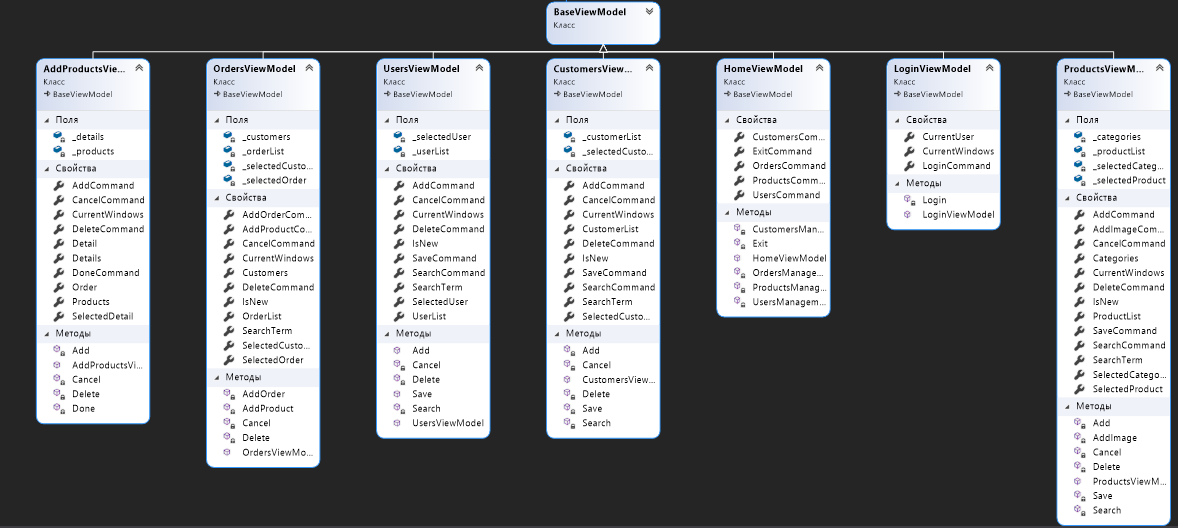


Рисунок 2.3 – Диаграмма классов модели представления.

Диаграмма классов представления (определяют визуальный интерфейс) представлена на рисунке 2.4.

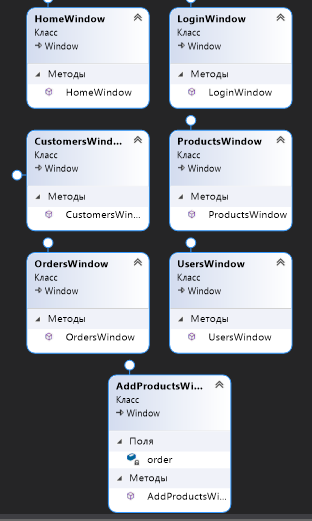


Рисунок 2.4 – Диаграмма классов представления.

Диаграмма классов моделей (описывает используемые в приложении данные) представлена на рисунке 2.5.

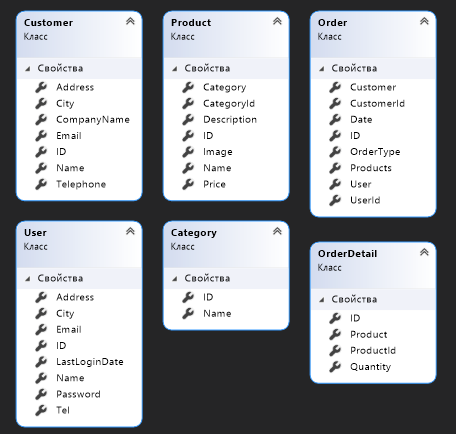


Рисунок 2.5 – Диаграмма классов моделей.

Диаграмма классов ассистирования пароля и привязки команды с имеющей к ней прикладной логикой, отвечающей за обслуживание интерфейса представлена на рисунке 2.6.

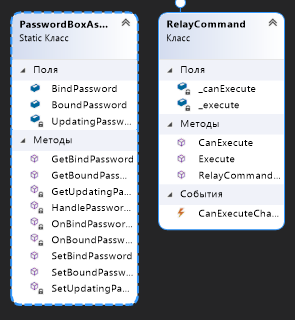


Рисунок 2.6 – Диаграмма классов ассистирования пароля и привязки команды с прикладной логикой.Диаграмма контекста базы данных 2.7.

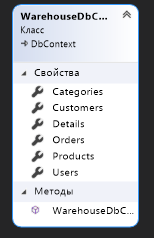


Рисунок 2.7 – Диаграмма контекста базы данных.

Диаграмма классов ресурсов и настроек рисунке 2.8.

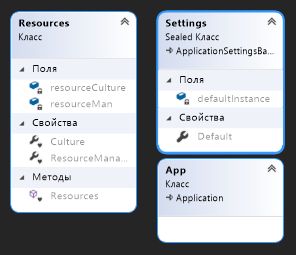


Рисунок 2.8 – Диаграмма классов ресурсов и настрое.

## Проектирование базы данных

Данные для работы будут храниться в базе данных. Её создание происходит с помощью Microsoft SQL Server — это система управления реляционными базами данных, которая подключается к приложению.

При разработке приложения сначала разрабатывалась база данных, которая должна содержать всю нужную информацию для приложения, после программное обеспечение. База данных состоит из 6 таблиц, между которыми установлены связи один ко многим. (Рисунок 2.9.)

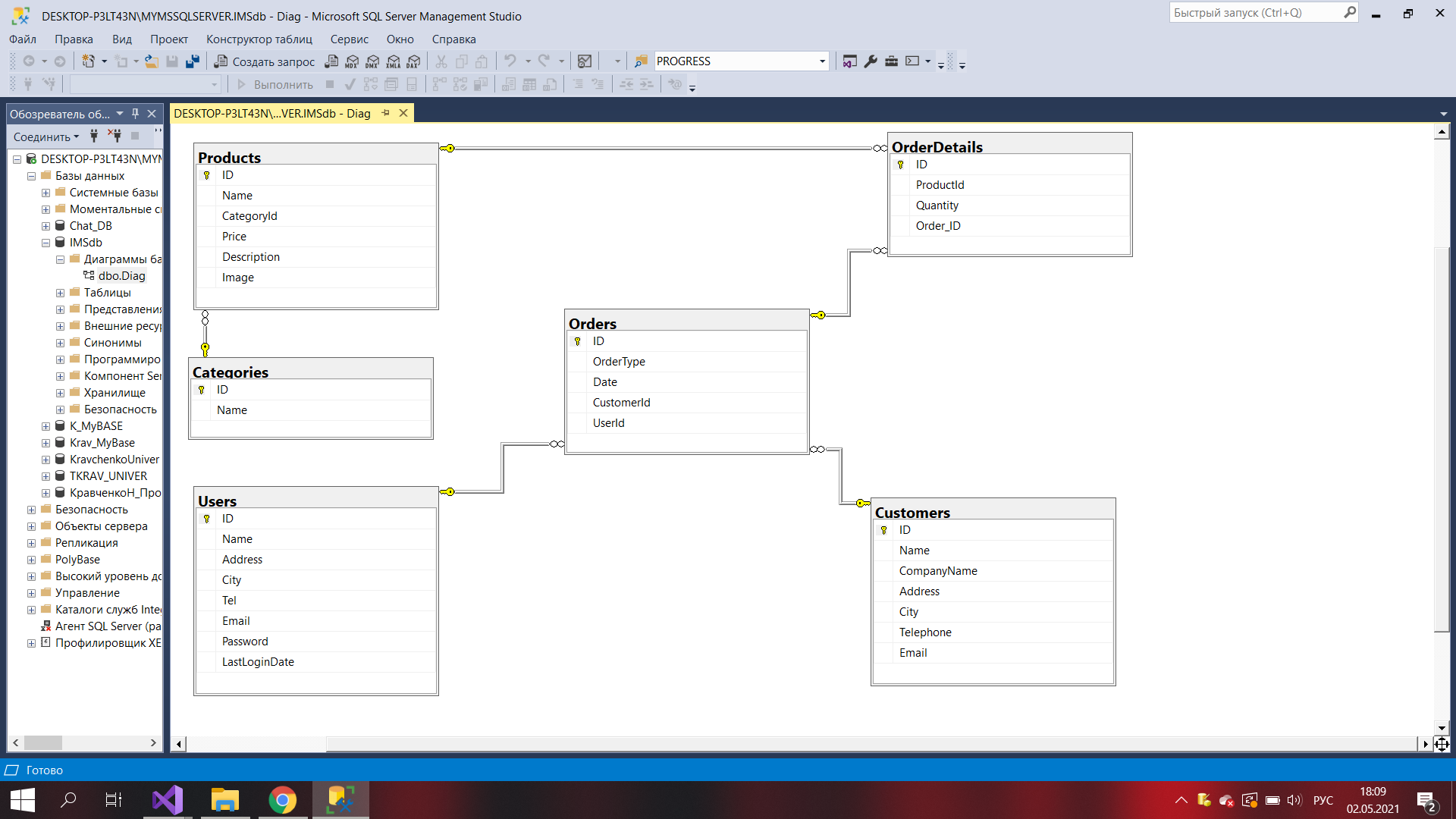


Рисунок 2.9 – Диаграмма базы данных.

Таблица Products предназначена для хранения информации о добавленных на склад продуктов, состоит из 6 столбцов:

­– ID - уникальный идентификатор продукта;

– Name - название продукта;

– Category - категория продукта;

– Price - цена за единицу продукта;

– Description - описание продукта;

– Image - изображение продукта;

Таблица Category предназначена для хранения информации о категориях добавленных на склад продуктов, состоит из 2 столбцов:

­– ID - уникальный идентификатор категории продукта;

– Name - название категории продукта;

Таблица Orders предназначена для хранения информации о добавленных заказах продуктов на (вход/выход) со склада, состоит из 5 столбцов:

­– ID - уникальный идентификатор заказа продукта;

– OrderType - тип заказа (вход/выход продукта);

– Date - дата заказа на вход/выход продукта;

– CustomerId - уникальный идентификатор заказчика;

– UserId - уникальный идентификатор пользователя;

Таблица OrderDetails предназначена для хранения дополнительной информации заказов продуктов на (вход/выход) со склада, состоит из 4 столбцов:

­– ID - уникальный идентификатор дополнительной информации о заказах продуктов на (вход/выход) со склада

– ProductId - уникальный идентификатор продукта

– Quantity - количество продуктов для заказов на (вход/выход) со склада;

– Order\_ID - уникальный идентификатор заказа;

Таблица Customers предназначена для хранения информации о заказчиках, состоит из 7 столбцов:

­– ID - уникальный идентификатор заказчика;

– Name - имя заказчика;

– CompanyName - название компании заказчика;

– Address - адрес заказчика или офиса его компании;

– City - город заказчика;

– Telephone - мобильный телефон заказчика;

– Email - почтовый адрес заказчика (имэйл);

Таблица Users предназначена для хранения информации о добавленных на склад продуктов, состоит из 8 столбцов:

­– ID - уникальный идентификатор пользователя;

– Name – Имя пользователя;

– Address – адрес пользователя;

– City – город пользователя;

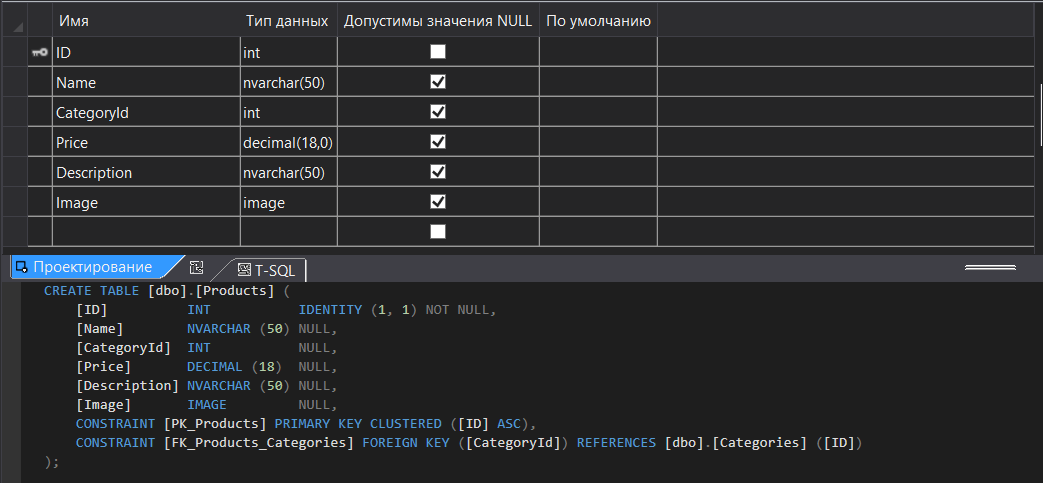
– Tel – мобильный телефон пользователя;

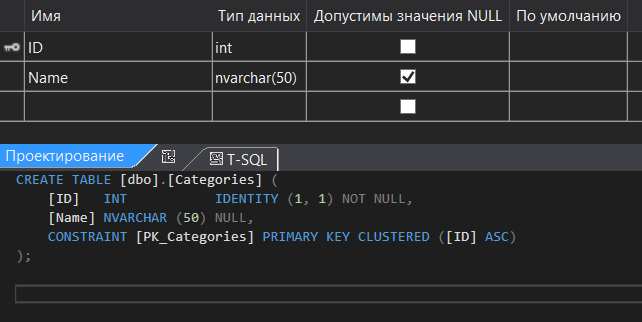
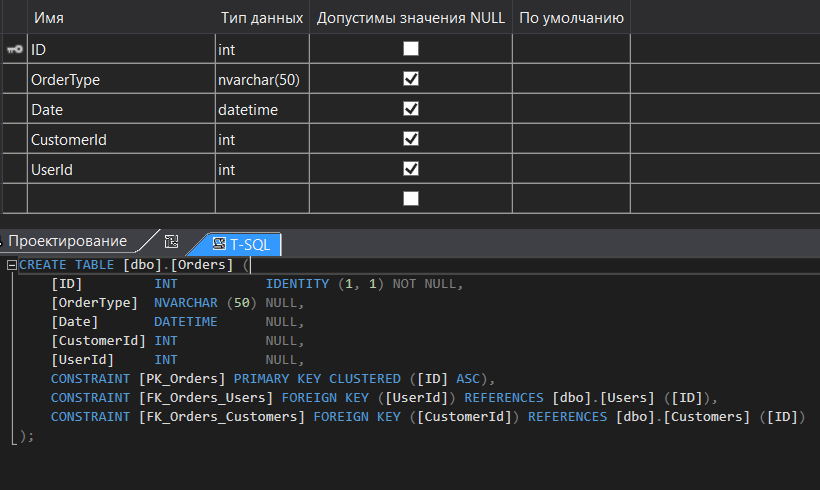
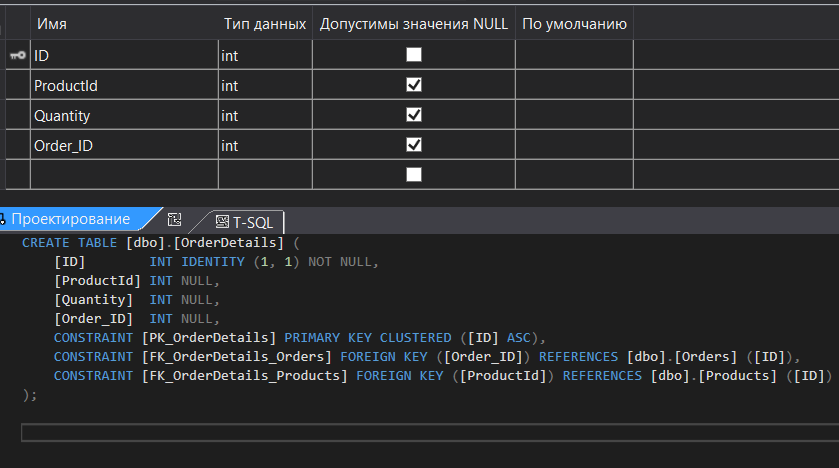
– Email – почтовый адрес пользователя(имэйл) для входа в приложение;

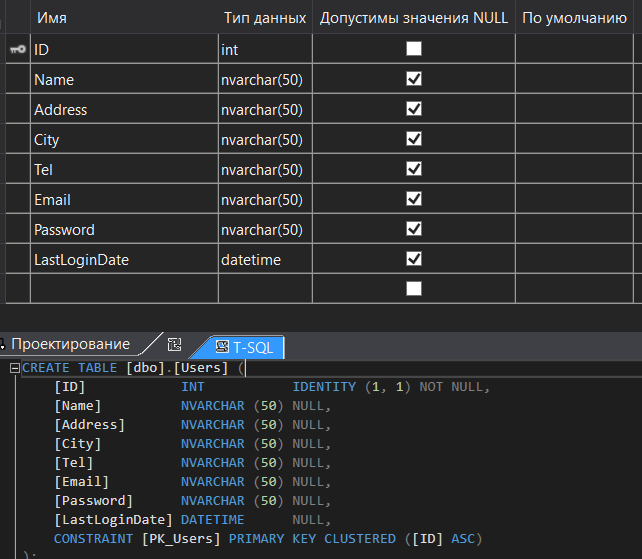
– Password – пароль пользователя для входа в приложение;

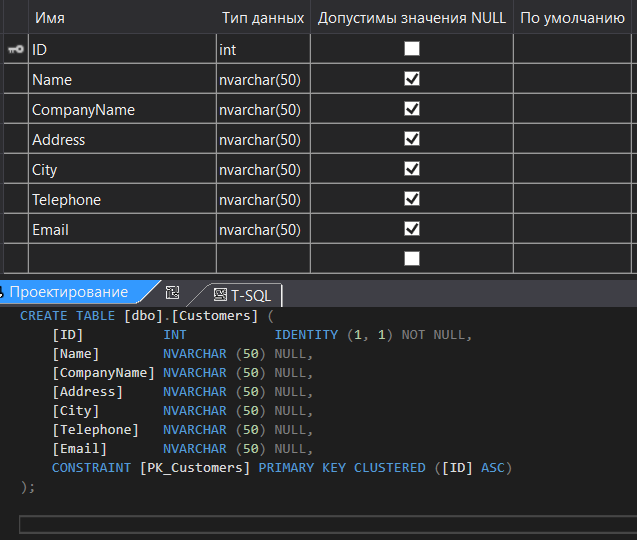
– LastLoginDate – дата последнего входа в приложение;

Также в каждой таблице для поля ID прописан атрибут Identity, который позволяет сделать столбец идентификатором, значение которого будет автоматически увеличиваться на единицу для каждого следующего значения(строки).









# Разработка функциональной модели

## Описание функций приложения и MVVM

* Выполнять авторизацию (регистрация пользователя происходит под аккаунтом админа уже в самом приложении);
* отслеживать количество товаров на складе, их названия, описание, фотографий, категорий, стоимости;
* добавлять новые товары, клиентов, пользователей, ордера(заказы) на вход/выход продукта со склада и его количество
* осуществлять обновление данных о товаре, заказчике, ордере(заказе), пользователе
* осуществлять поиск товаров, клиентов и пользователей
* осуществлять удаление товаров, клиентов и пользователей

В процессе создания приложения был реализован паттерн MVVM. Паттерн MVVM (Model-View-ViewModel) позволяет отделить логику приложения от визуальной части (представления). Данный паттерн является архитектурным, то есть он задает общую архитектуру приложения.

MVVM состоит из трех компонентов: модели (Model), модели представления (ViewModel) и представления (View).

Паттерн MVVM в WPF:

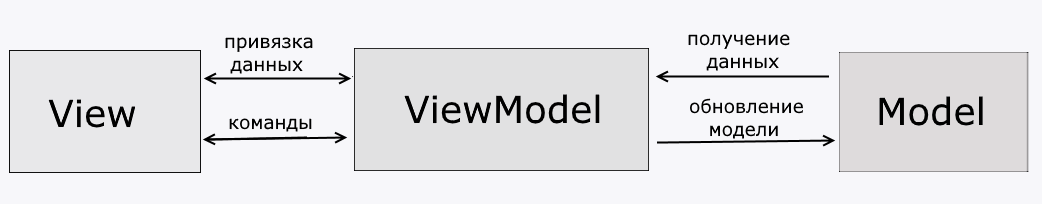


Рисунок 3.1 – схема паттерна MVVM

**Model**

Модель описывает используемые в приложении данные. Модели могут содержать логику, непосредственно связанную этими данными, например, логику валидации свойств модели. В то же время модель не должна содержать никакой логики, связанной с отображением данных и взаимодействием с визуальными элементами управления.

**View**

View или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Применительно к WPF представление - это код в xaml, который определяет интерфейс в виде кнопок, текстовых полей и прочих визуальных элементов.

Хотя окно (класс Window) в WPF может содержать как интерфейс в xaml, так и привязанный к нему код C#, однако в идеале код C# не должен содержать какой-то логики, кроме разве что конструктора, который вызывает метод InitializeComponent и выполняет начальную инициализацию окна. Вся же основная логика приложения выносится в компонент ViewModel.

Однако иногда в файле связанного кода все может находиться некоторая логика, которую трудно реализовать в рамках паттерна MVVM во ViewModel.

Представление не обрабатывает события за редким исключением, а выполняет действия в основном посредством команд.

**ViewModel**

ViewModel или модель представления связывает модель и представление через механизм привязки данных. Если в модели изменяются значения свойств, при реализации моделью интерфейса INotifyPropertyChanged автоматически идет изменение отображаемых данных в представлении, хотя напрямую модель и представление не связаны.

ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление. И также VewModel определяет логику по обновлению данных в модели.

Поскольку элементы представления, то есть визуальные компоненты типа кнопок, не используют события, то представление взаимодействует с ViewModel посредством команд.

Например, пользователь хочет сохранить введенные в текстовое поле данные. Он нажимает на кнопку и тем самым отправляет команду во ViewModel. А ViewModel уже получает переданные данные и в соответствии с ними обновляет модель.

Итогом применения паттерна MVVM является функциональное разделение приложения на три компонента, которые проще разрабатывать и тестировать, а также в дальнейшем модифицировать и поддерживать.

## Основные окна приложения

### Главное окно приложения

HomeWindow — основное окно игры. Оно содержит кнопки навигации, а также объект отображения страниц (Рисунок 3.3).

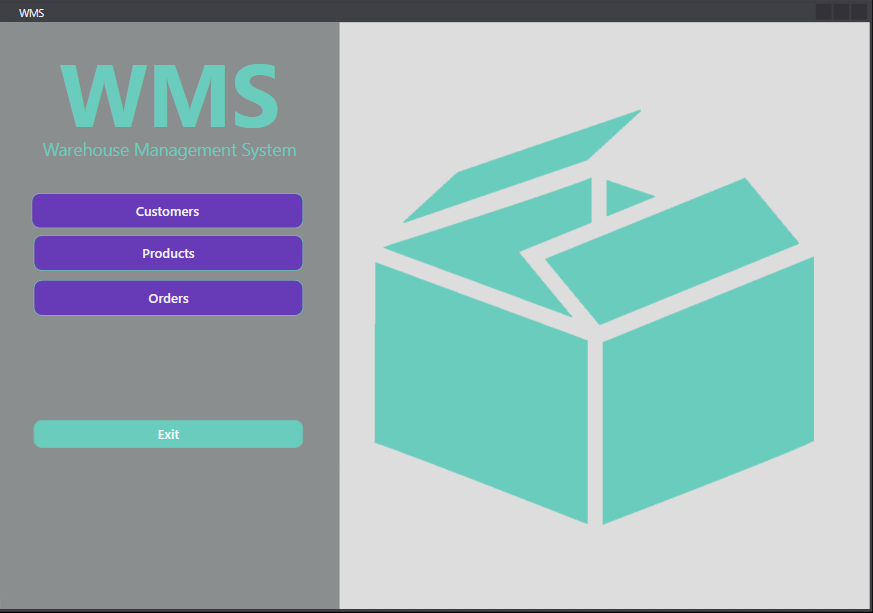


Рисунок 3.3 – главное окно приложения.

## Основные страницы приложения

Навигация по приложению осуществляется с помощью окна HomeWindow. При использовании определённой кнопки меняется страница, реализованный с помощью команд (RelayCommand):

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

using WarehouseManagementSystem.Helpers;

using WarehouseManagementSystem.Views;

namespace WarehouseManagementSystem.ViewModels

{

public class HomeViewModel : BaseViewModel

{

public ICommand UsersCommand { get; set; }

public ICommand CustomersCommand { get; set; }

public ICommand ProductsCommand { get; set; }

public ICommand OrdersCommand { get; set; }

public ICommand ExitCommand { get; set; }

public HomeViewModel()

{

this.CustomersCommand = new RelayCommand(param => this.CustomersManagement(), param => true);

this.ProductsCommand = new RelayCommand(param => this.ProductsManagement(), param => true);

this.OrdersCommand = new RelayCommand(param => this.OrdersManagement(), param => true);

this.ExitCommand = new RelayCommand(param => this.Exit(), param => true);

}

private void Exit()

{

MessageBoxResult rsltMessageBox = MessageBox.Show("Do you want to exit?", "Exit", MessageBoxButton.YesNo, MessageBoxImage.Warning);

switch (rsltMessageBox)

{

case MessageBoxResult.Yes:

Application.Current.Shutdown();

break;

case MessageBoxResult.No:

break;

}

}

private void CustomersManagement()

{

CustomersWindow myWindow = new CustomersWindow();

myWindow.Show();

}

private void ProductsManagement()

{

ProductsWindow myWindow = new ProductsWindow();

myWindow.Show();

}

private void OrdersManagement()

{

OrdersWindow myWindow = new OrdersWindow();

myWindow.Show();

}

private void UsersManagement()

{

UsersWindow myWindow = new UsersWindow();

myWindow.Show();

}

}

}

### Страница авторизации

Страница авторизации содержит поля Email и Password. Кнопка авторизации не сработает до тех пор, пока не будут введены корректные данные в поля. Если при авторизации введены данные админа, то мы попадаем на страницу UserWindow, где далее админ может добавить различных пользователей, которые имеют данные (имя, телефон, адрес, город, email, пароль), используя которые пользователь может потом зайти в приложение.

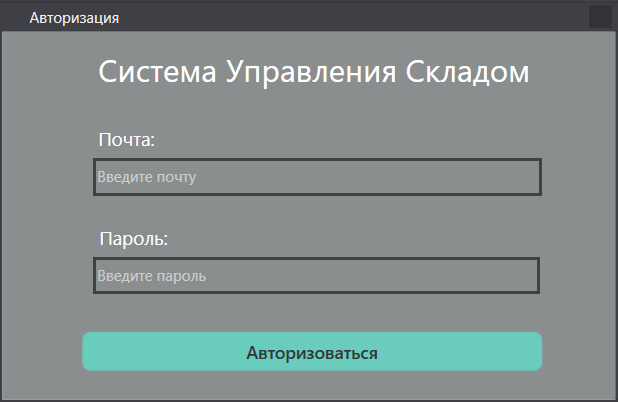


Рисунок 3.4 – Окно авторизации LoginWindow.

### Страница регистрации

Как таковая видимая регистрация в приложении отсутствует. Для регистрации пользователя мы заходим под учётной записью админа, регистрируем пользователя, и далее передаём ему данные для входа в его аккаунт. Данные пользователя могут содержать имя, телефон, адрес, почту, город, пароль. Админ может добавлять и удалять пользователей, а также изменять их данные.



Рисунок 3.5 – Окно пользователей UsersWindow.

### Страница клиентов

Страница клиентов позволяет пользователю управлять клиентами. Клиент имеет такие данные: уникальный идентификатор, имя, название компании, адрес, телефон, город, почта. Пользователь может осуществлять поиск клиентов по имени, добавлять клиентов, обновлять данные клиентов и так же удалять клиентов. Для удаления клиента изначально нужно удалить все его заказанные товары на вход/выход со склада и сам заказ.

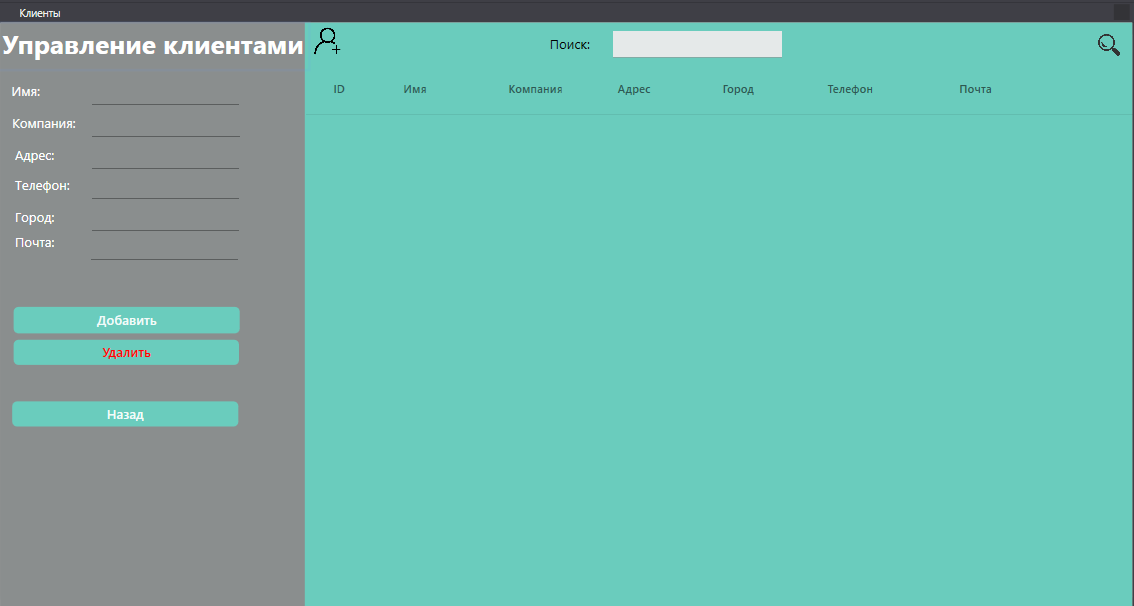
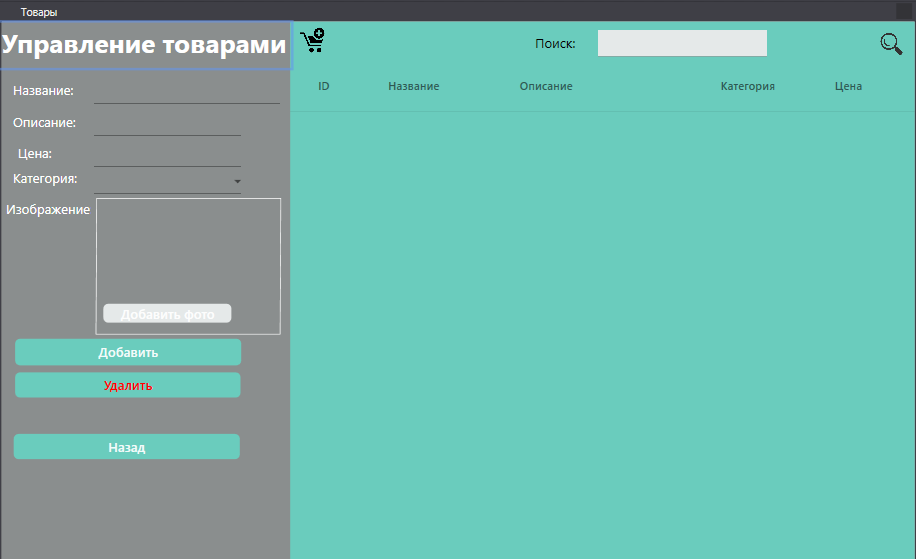


Рисунок 3.6 – Окно клиентов СustomersWindow.

### Страница товаров

Страница товаров позволяет пользователю управлять товарами. Продукт имеет такие данные: уникальный идентификатор, название, описание, категория товара, цена, а также его фотография. Пользователь может осуществлять поиск товаров по названию и описанию, добавлять товары, обновлять данные о товарах и так же удалять товары. Для удаления товара изначально нужно удалить все заказы товара на вход/выход со склада и сам заказ.

 Рисунок 3.7 – Окно товаров ProductsWindow.

### Страница заказов

Страница заказов позволяет пользователю добавлять заказы на определённые товары и выбирать их количество. Заказ имеет такие данные: уникальный идентификатор, дата заказа, имя клиента, сделавшего заказ. Пользователь может добавлять заказы и также удалять заказы. Для удаления заказа изначально нужно удалить все заказанные товары на вход/выход со склада этого заказа.

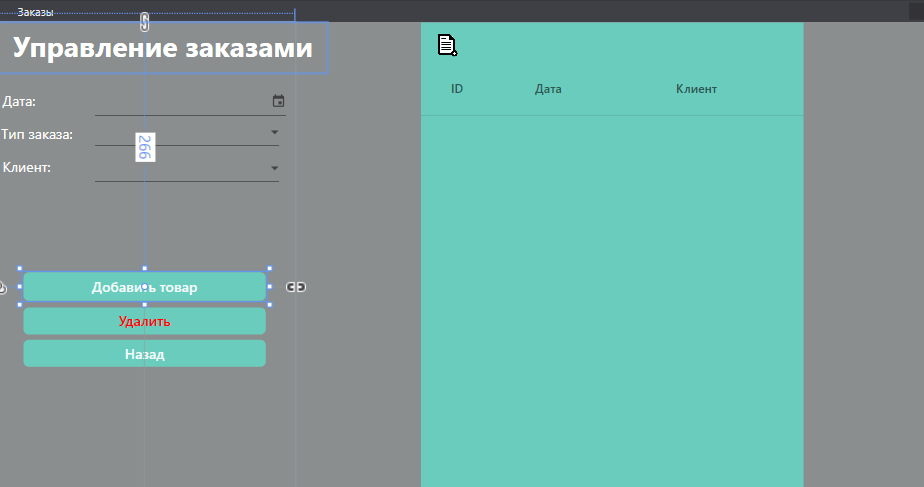


Рисунок 3.8 – Окно заказов OrdersWindow.

### 3.3.6 Страница дополнительной информации о заказе

Страница дополнительной информации о заказе, где указывается товар и его количество для заказа определённым заказчиком. В окне отображаются такие данные о заказе: название товара, его количество. После добавления товаров и их количества на заказ, при последующей попытке добавления товаров в заказ они добавляться не будут. Для этого нужно создать уже новый заказ.



Рисунок 3.9 – Окно дополнительной информации о заказе AddProductsWindow (выбор продукта и его количество)

# Тестирование, проверка работоспособности и анализ полученных результатов

Программное средство проходило тестирование. Сначала происходила отладка программы в процессе написания кода, после завершения основного кода, во время тестирования на работоспособность, и при наличии ошибок, исправлялся. Таким образом добивалась максимальная работоспособность программы. Пользователю нужно указать на неправильно введенные данные или недочеты, чтобы он мог их исправить.

Результат обработки исключения при авторизации, если не введена почта или пароль, или данные являются неверными (Рисунок 5.1).

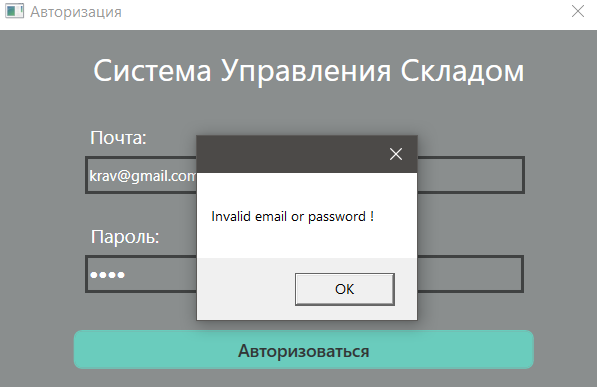


Рисунок 5.1 – Пример проверки авторизации.

Ошибка при регистрации, если введена почта пользователя, уже зарегистрированного в системе ранее исключена, т.к регистрация осуществляется администратором, который имеет доступ к базе данных всех пользователей и проверяет данные на соответствие нормам.

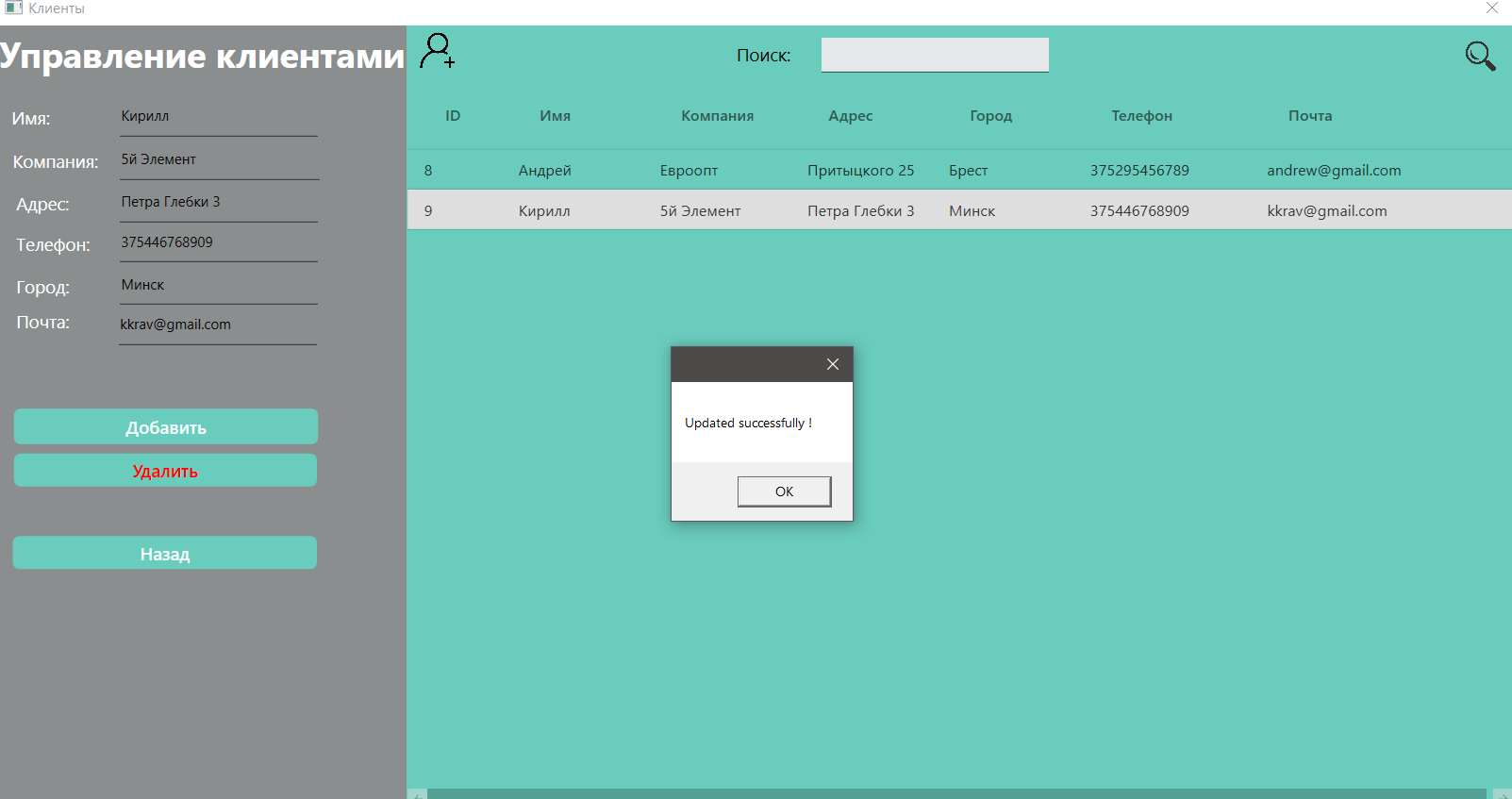
Результат при добавлении клиента (Рисунок 5.2)

Рисунок 5.2 – Пример теста добавления клиента.

Результат обработки исключения при попытке удалить клиента, если клиент не выбран (Рисунок 5.3).

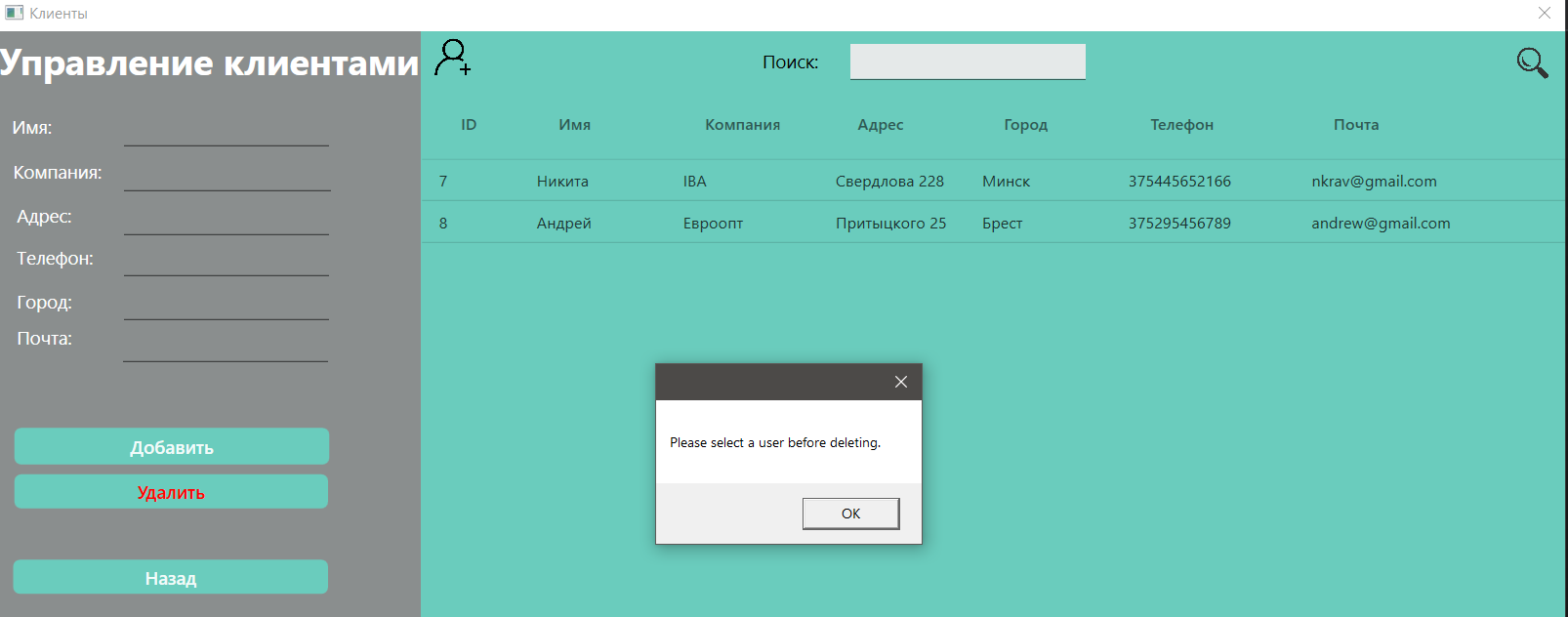


Рисунок 5.3 – Пример теста удаления невыбранного клиента.

Пример теста при попытке удаления клиента (Рисунок 5.4).

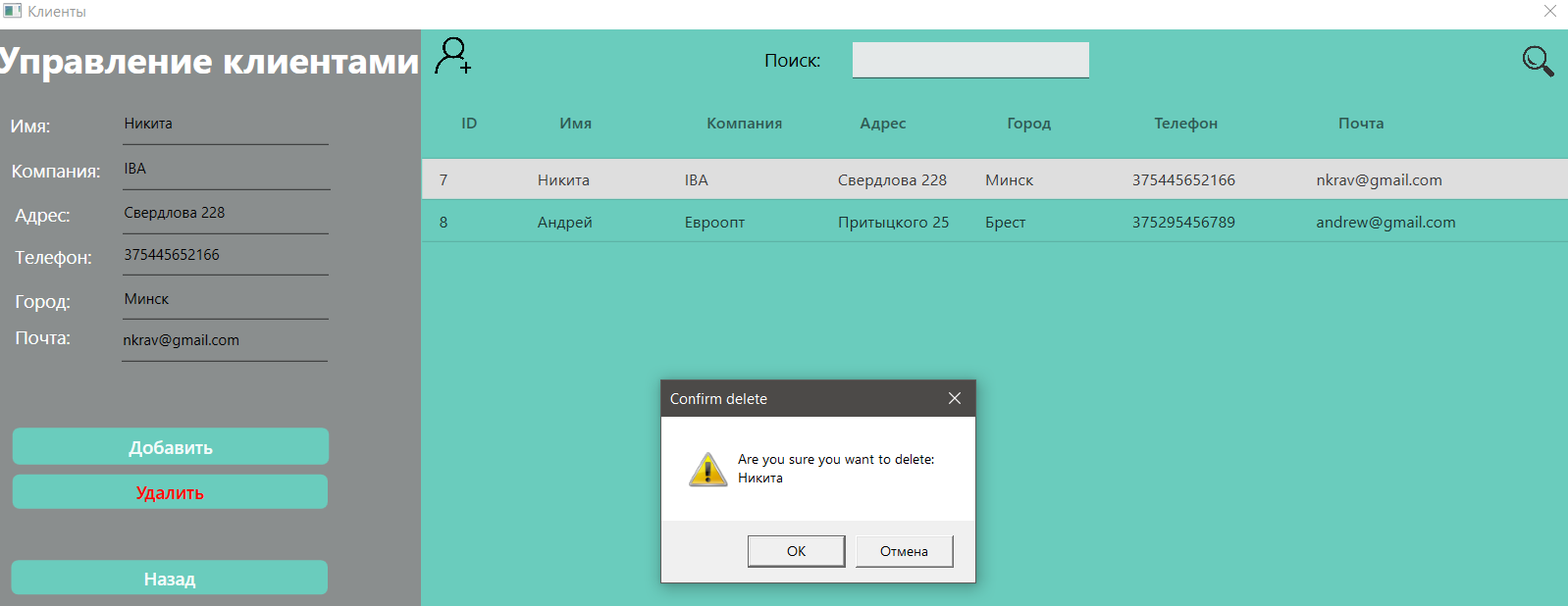


Рисунок 5.4 – Пример теста при удалении клиента.

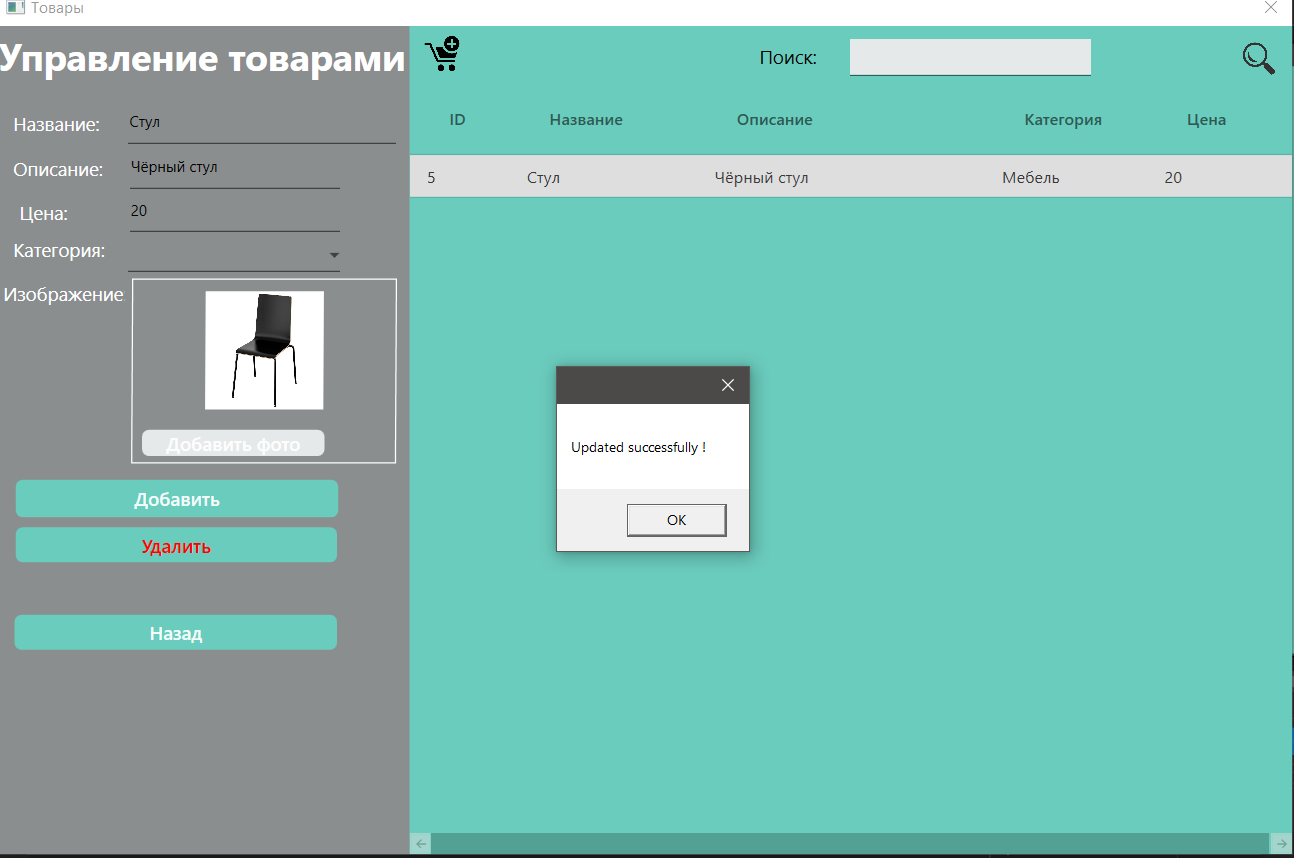
Результат при добавлении товара (Рисунок 5.5)

Рисунок 5.5 – Пример теста добавления товара.

Результат обработки исключения при попытке удалить товар, если товар не выбран (Рисунок 5.6).



Рисунок 5.6 – Пример теста удаления невыбранного товара.

Пример теста при попытке удаления товара (Рисунок 5.7).

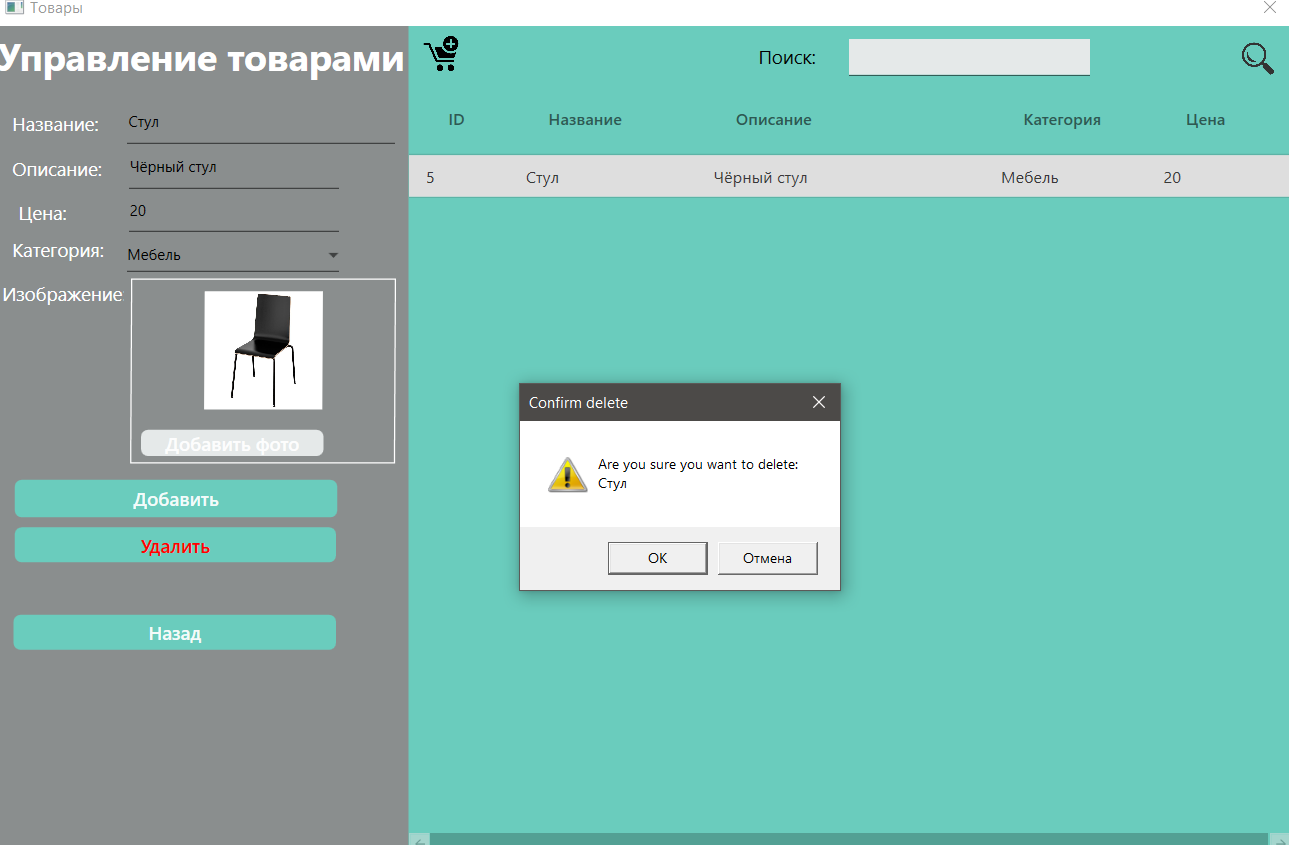


Рисунок 5.7 – Пример теста при удалении товара.

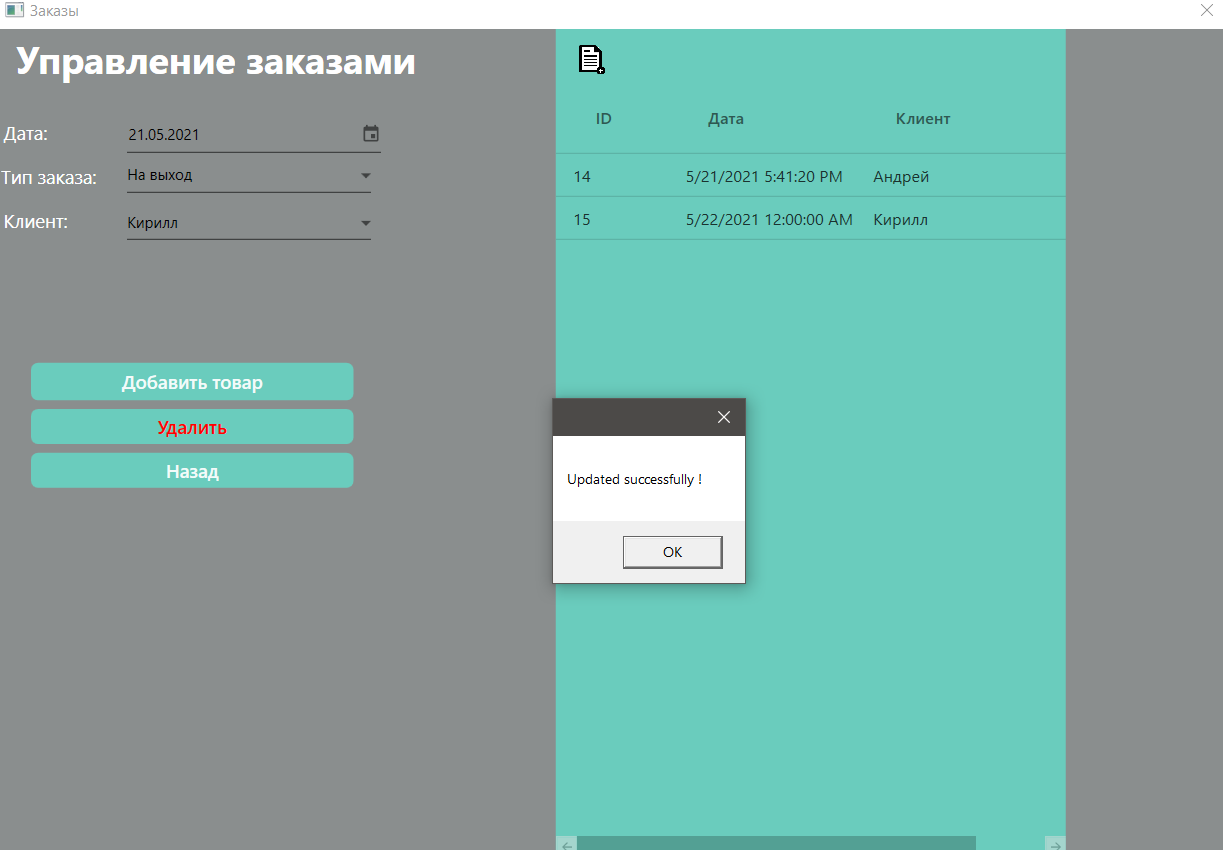
Результат при добавлении заказа (Рисунок 5.8)

Рисунок 5.8 – Пример теста добавления заказа.

Результат обработки исключения при попытке удалить заказ, если заказ не выбран (Рисунок 5.9).

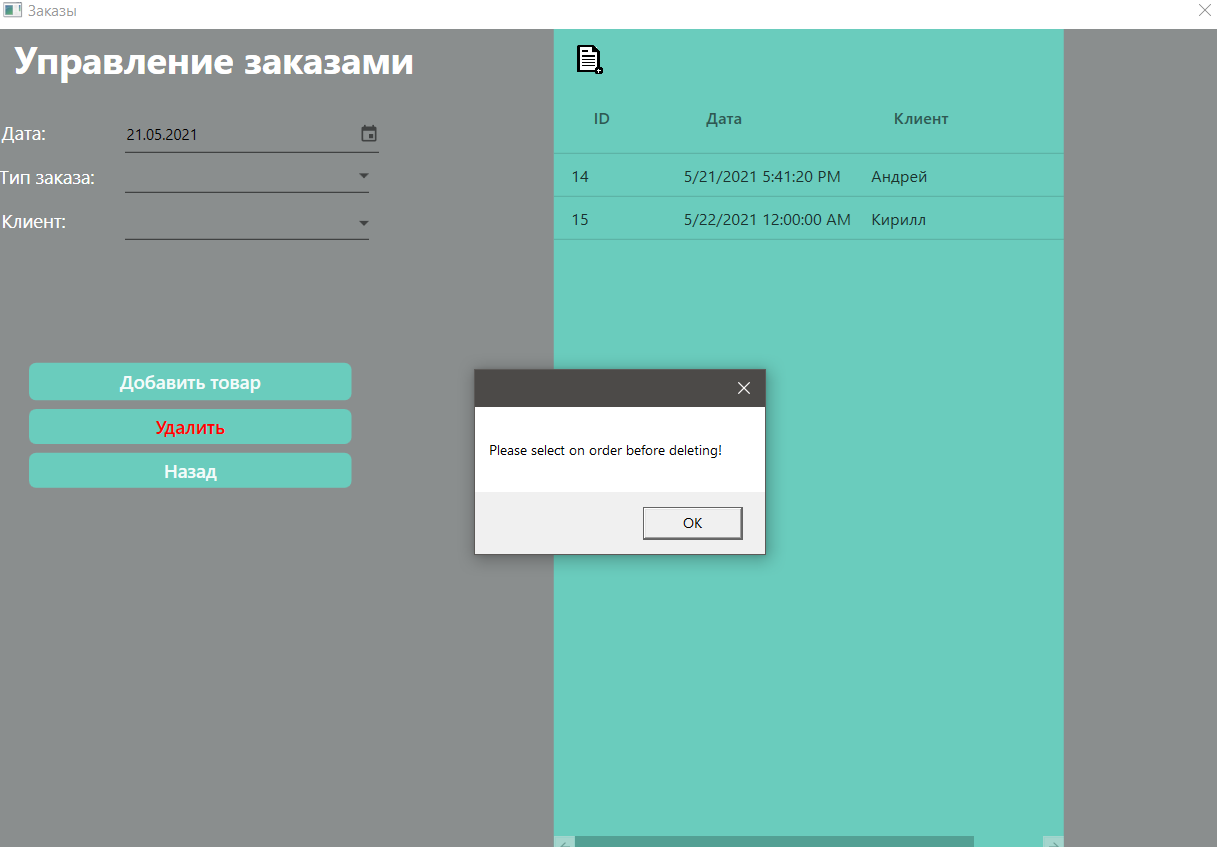


Рисунок 5.9 – Пример теста удаления невыбранного заказа.

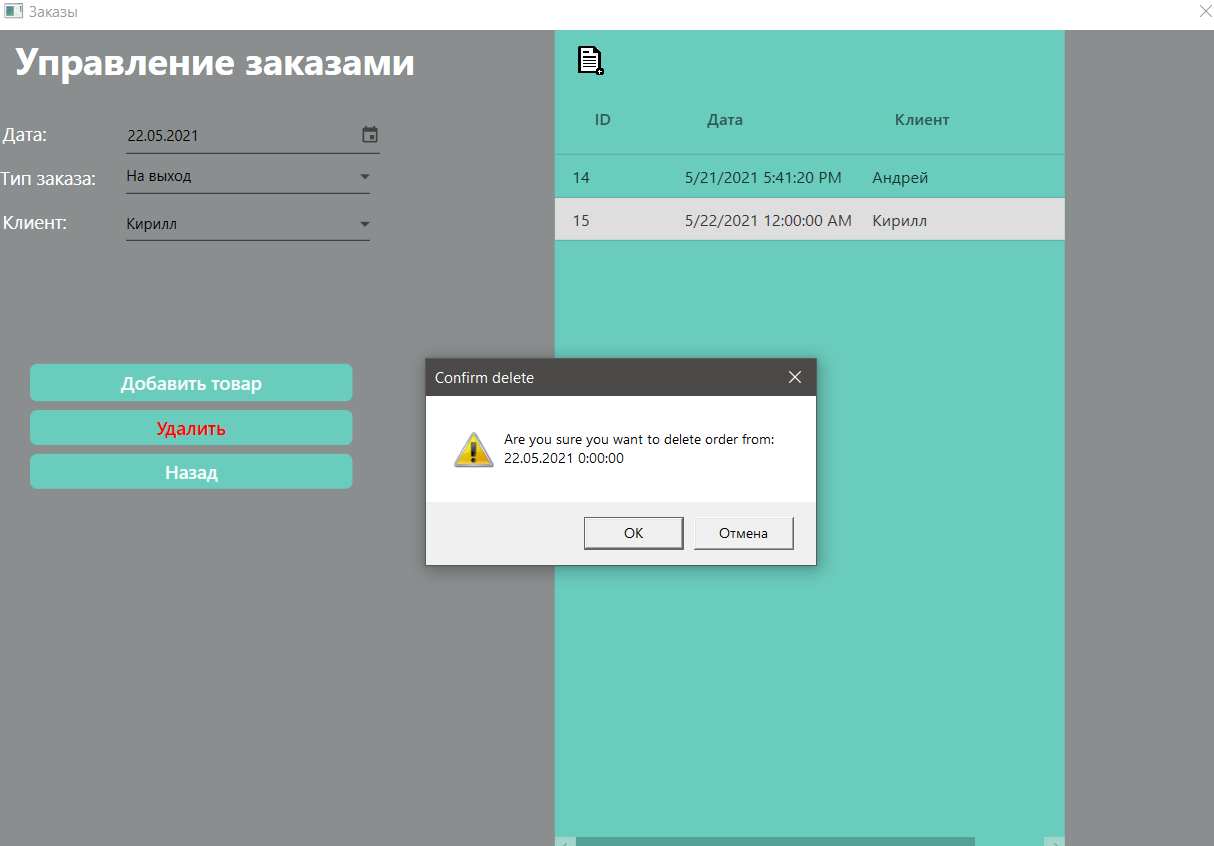
Пример теста при попытке удаления заказа (Рисунок 5.10).

Рисунок 5.10 – Пример теста при удалении заказа.

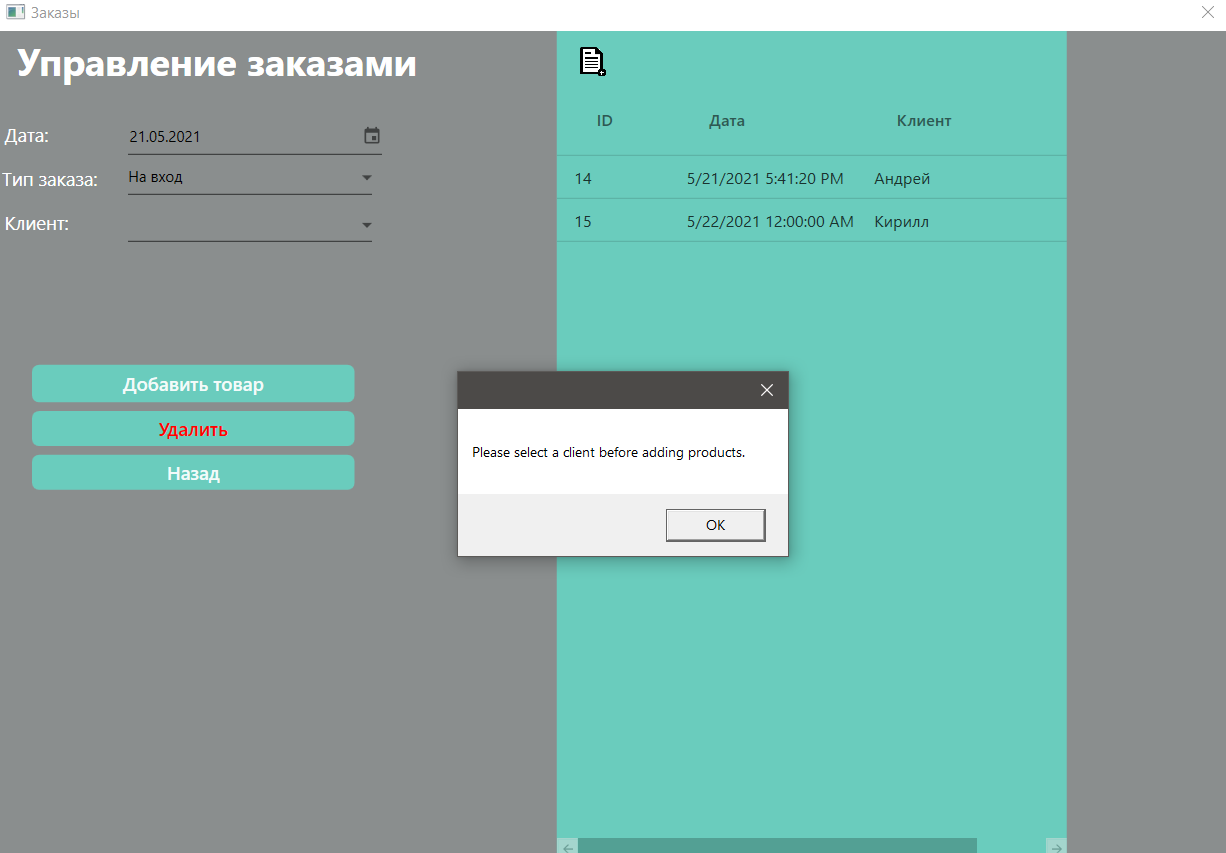
Результат обработки исключения при попытке добавить заказ, если клиент не выбран (Рисунок 5.11).

Рисунок 5.11 – Пример теста добавления заказа при невыбранном клиенте.

Результат при добавлении товара и его количества на заказа не выбрав товар или количество товара (Рисунок 5.12)

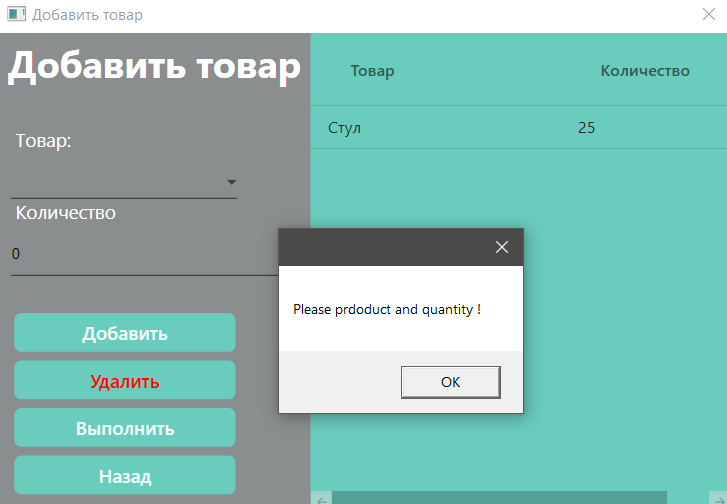


Рисунок 5.12 – Пример теста добавления товара на заказ не выбрав товар и к-во.

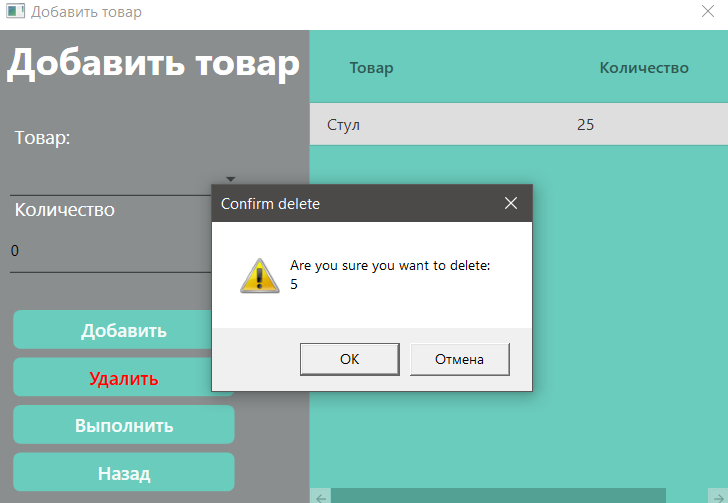
Результат при удалении товара и его количества на заказа (Рисунок 5.13)

Рисунок 5.13 – Пример теста удалении товара на заказ.

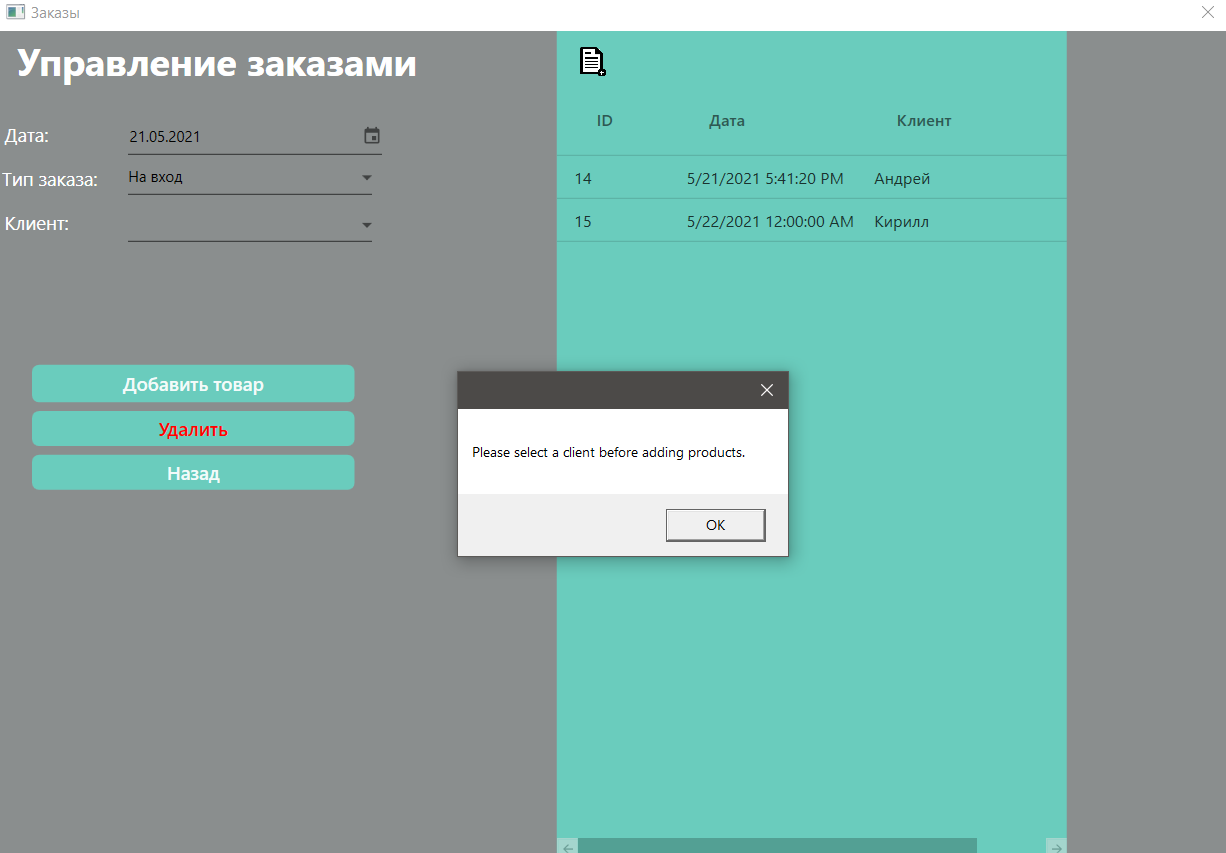
Результат обработки исключения при попытке удалить заказ, если не удалены товары на заказ (Рисунок 5.14).

Рисунок 5.14 – Пример теста удаления заказа, не удалив заказанные товары.

Результат при попытке удалить пользователя при входе в приложение со стороны админа (Рисунок 5.15).

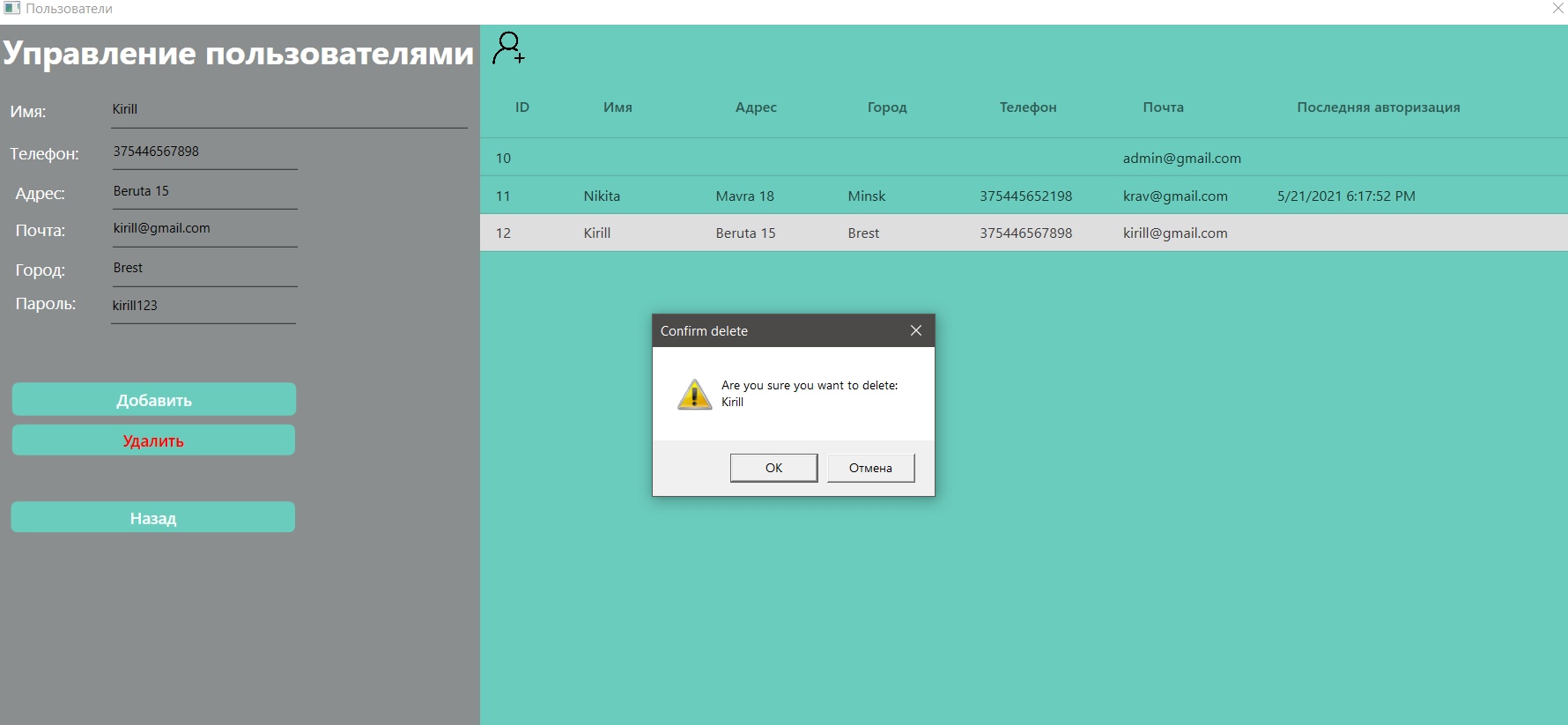


Рисунок 5.15 – Пример теста удаления пользователя админом.

Результат обработки исключения при попытке удалить пользователя, если пользователь не выбран (Рисунок 5.16).

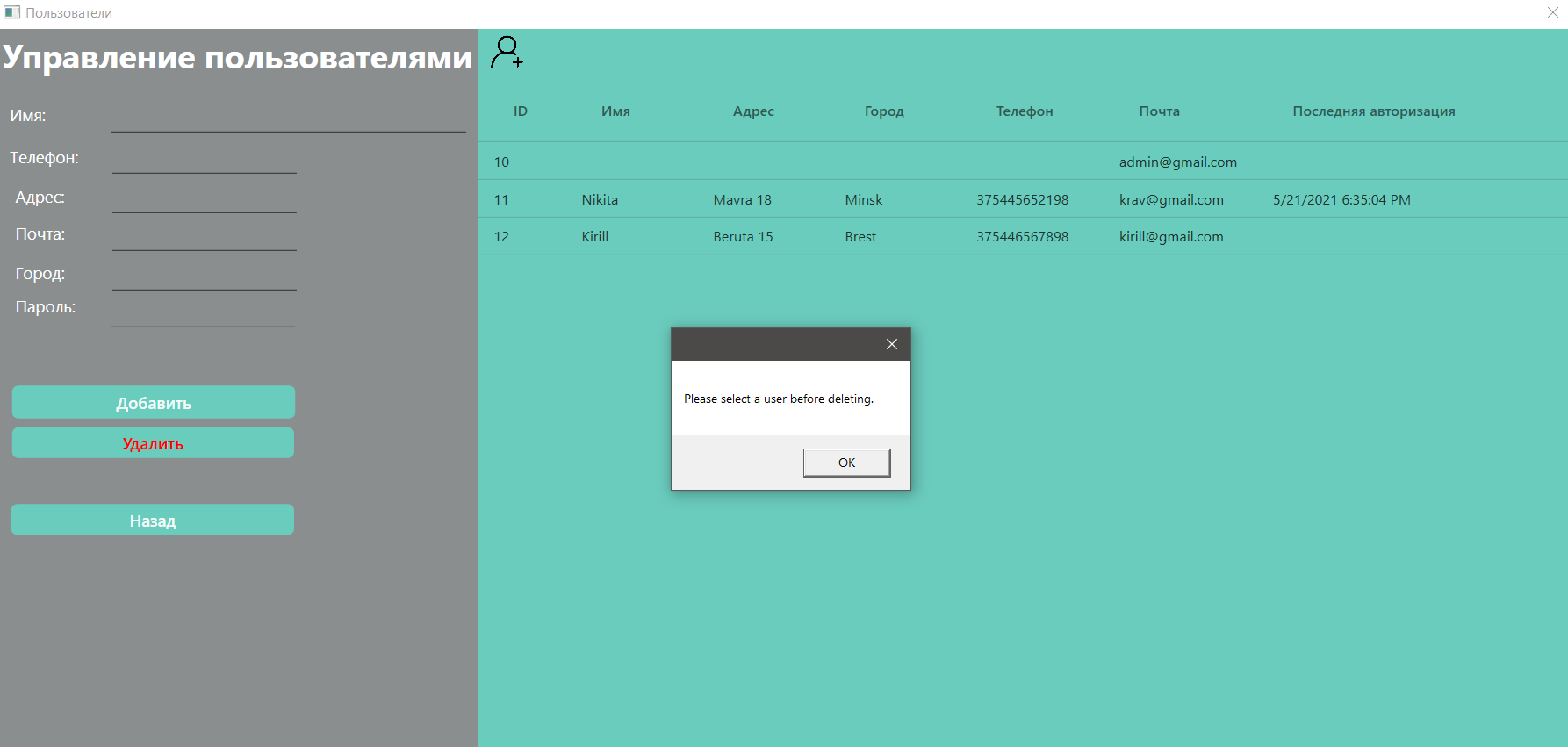


Рисунок 5.16 – Пример теста удаления невыбранного пользователя.

# Руководство по установке и использования

Начальная страница для пользователя и админа одинаковые (Рисунок 4.2). На начальной странице необходимо ввести учетные данные, для дальнейшего входа в программу. С начальной страницы пользователь может перейти на страницу регистрации для создание учетной записи.

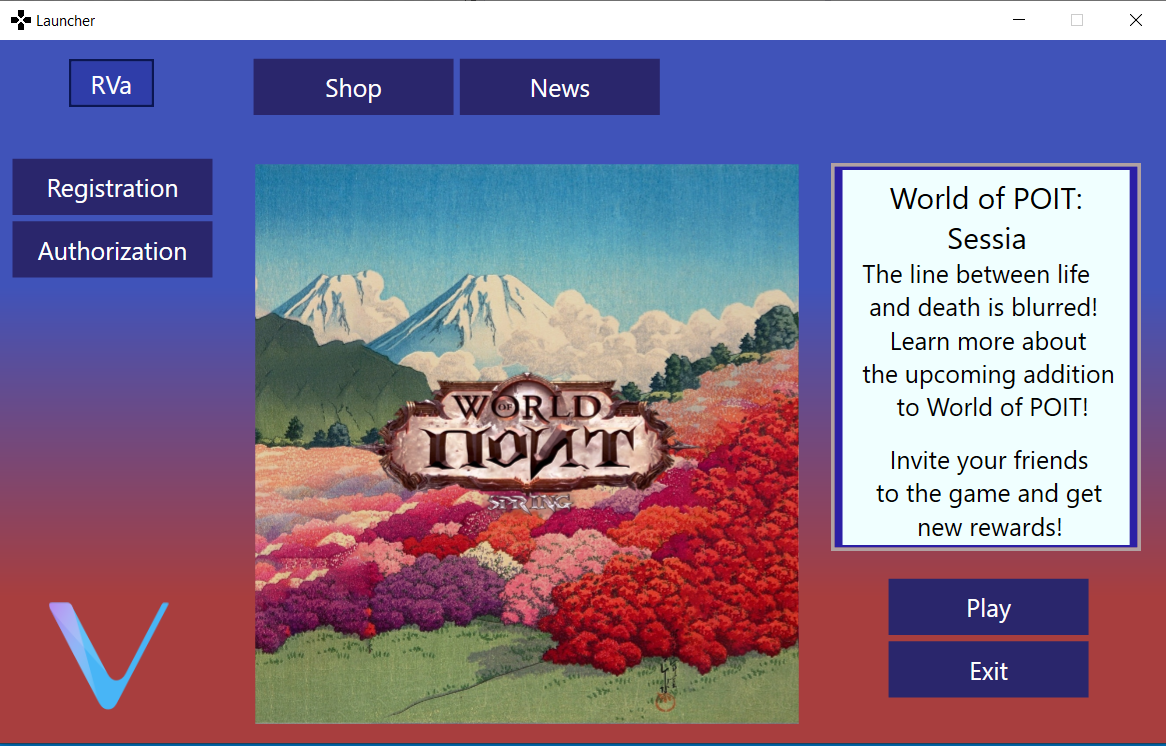


Рисунок 4.1 – Начальная страница.

На странице регистрации необходимо ввести общую информации для регистрации (Рисунок 4.3) и дальнейшей авторизации в программе.

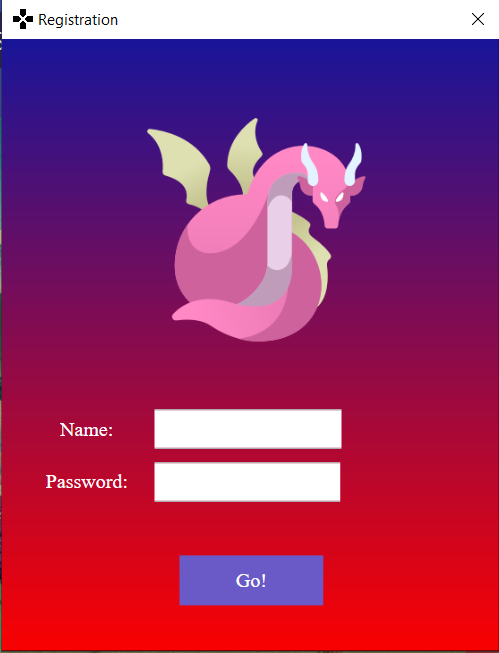


Рисунок 4.2 – Страница регистрации.

Страница профиля в зависимость от уровня доступа для различных пользователей разная: пользователь может запустить игру (при условии, что игра куплена) (Рисунок 4.4), а для администратора появляется таблица с запросами.

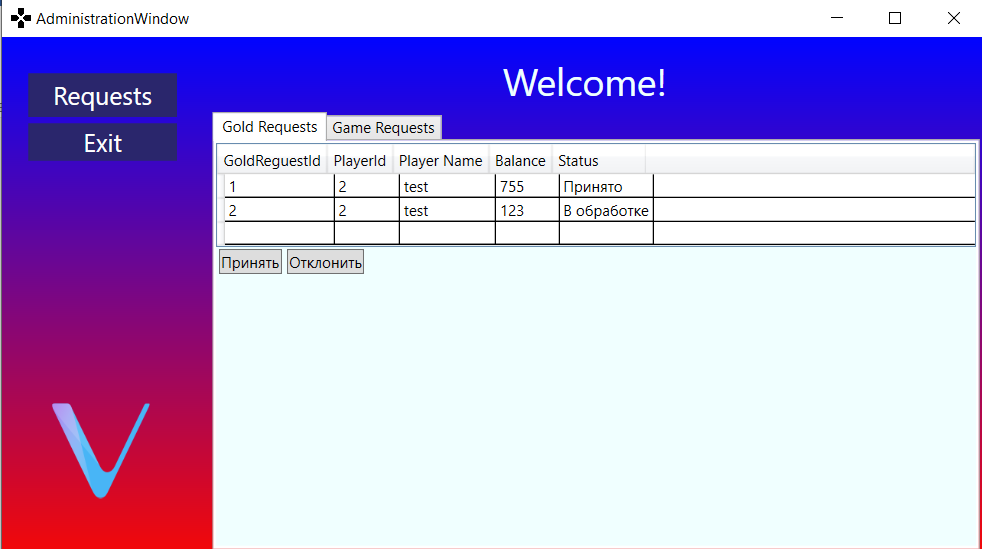
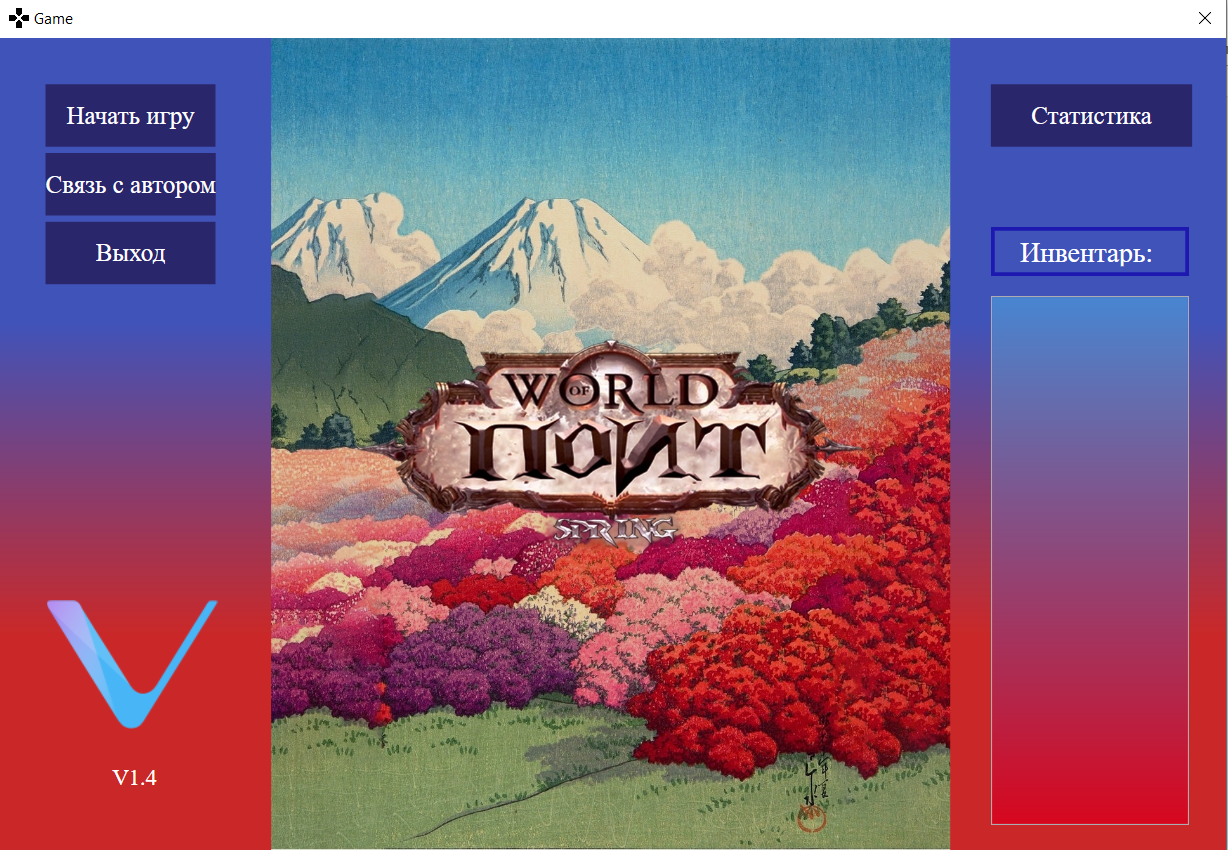


Рисунок 4.3 – Страница профиля для пользователя и администратора.

# Заключение

Итогом курсового проекта является рабочее приложение типа «Система Управления Складом» с возможностью управлять подсчетом товаров и их движением, заказами на их покупку и продажу, импортом и экспортом, управлять поставщиками и клиентами, что поможет пользователям сделать многие из этих процессов автоматическими без необходимости запоминать детали, связанные с заказом.

В ходе выполнения курсового проекта было выполнено следующее:

* создание базы данных для хранения всей информации;
* разработка архитектуры приложения;
* разработка функциональной части приложения;
* разработка пользовательского интерфейса;
* написание исходного кода приложения;
* тестирование приложения.

В соответствии с полученным результатом работы программы можно сделать вывод, что требования технического задания выполнены, интерфейс получился достаточно простой и понятный, в нём сможет разобраться любой, даже далёкий от информационных технологий пользователь.

За время выполнения курсового проекта по созданию приложения были изучены способы работы с «SQL Server», принципы создания приложения с использованием .NET и WPF, изучен новый язык верстки XAML для построения интерфейса и работы с ним. Использовался Entity Framework для работы с базой данный, а само приложение написано на языке С#.

# Список используемых источников

1. Общие сведения о Microsoft Visual Studio [Электронный ресурс]. / Режим доступа <https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019>/
2. Windows Presentation Foundation [Электронный ресурс]. / Режим доступа: <https://ru.wikipedia.org/wiki/Windows_Presentation_Foundation>
3. Учебник по языку SQL [Электронный ресурс]. / Режим доступа: <https://habr.com/ru/post/255361/>