МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Программной инженерии Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 Программное обеспечение информационных технологий (программирование интернет-приложений)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

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

Тема «Интернет-магазин мобильных телефонов»

Исполнитель

студент 2 курса группы 5 Круглик Алексей Викторович

(Ф.И.О.)

Руководитель работы ст.п. Сухорукова И.Г.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Пацей Н.В.

(подпись)

Минск 2022

**Оглавление**

[Введение 1](#_Toc104307364)

[1.Аналитический обзор литературы 3](#_Toc104307365)

[1.1. Аналитический обзор источников 3](#_Toc104307366)

[1.2. Аналитический обзор аналогов 3](#_Toc104307367)

[2.Анализ требований к программному средству и разработка функциональных требований 4](#_Toc104307368)

[2.1. Спецификация функциональных требований 4](#_Toc104307369)

[2.1.1. Требования к параметрам технических и программных средств 4](#_Toc104307370)

[2.1.2. Требования к организации входных данных 5](#_Toc104307371)

[2.1.3. Требования к организации выходных данных 5](#_Toc104307372)

[2.1.4. Требования к надёжности 5](#_Toc104307373)

[2.2. Описание функциональности программного средства 5](#_Toc104307374)

[3.Проектирование программного средства 6](#_Toc104307375)

[3.1. Структура проекта 6](#_Toc104307376)

[3.2. Реализация архитектуры MVVM 7](#_Toc104307377)

[3.3. Проектирование базы данных 9](#_Toc104307378)

[3.4. Проектирование архитектуры проекта 12](#_Toc104307379)

[3.5. Классы Users, Supports, Images, Orders, Goods 14](#_Toc104307380)

[3.6. Класс AplicationContext 15](#_Toc104307381)

[3.7. Классы ViewModel 15](#_Toc104307382)

[3.8. Класс ObservableObject 15](#_Toc104307383)

[3.9. Класс RelayCommand 15](#_Toc104307384)

[3.10. Валидация данных 15](#_Toc104307385)

[4. Реализация программного средства 16](#_Toc104307386)

[4.1. Выполняемые функции 16](#_Toc104307387)

[4.1.1. Регистрация и авторизация 16](#_Toc104307388)

[4.1.2. Добавление телефонов 16](#_Toc104307389)

[4.1.3. Удаление телефонов 17](#_Toc104307390)

[4.1.4. Редактирование телефонов 18](#_Toc104307391)

[4.1.5. Поиск телефонов по названию 18](#_Toc104307392)

[4.1.6. Сортировка телефонов по цене 18](#_Toc104307393)

[4.1.7. Возможность обращения в поддержку 19](#_Toc104307394)

[4.1.8. Покупка телефонов 20](#_Toc104307395)

[4.1.9. Отмена покупки телефонов 20](#_Toc104307396)

[5.Тестирование 22](#_Toc104307397)

[6.Методика использования программного средства 24](#_Toc104307398)

[ЗАКЛЮЧЕНИЕ 29](#_Toc104307399)

[Список литературы 30](#_Toc104307400)

[Приложение 31](#_Toc104307401)

[Приложение А 31](#_Toc104307402)

[Приложение Б 32](#_Toc104307403)

[Приложение С 33](#_Toc104307404)

## **Введение**

Интернет магазин мобильных телефонов – это программа на ПК, позволяющая удобно выбирать девайс под себя. Интернет-магазин — не корпоративный сайт, который можно рекомендовать практически всем компаниям. Интернет-магазин в первую очередь полезен владельцам стандартизированных товаров и услуг. Например, продавать с виртуальных полок сотовые телефоны и автозапчасти значительно проще, чем, скажем, одежду или обувь, которые требуется примерять. Вторая группа потенциальных владельцев Интернет-магазинов — компании, занимающиеся традиционной розничной коммерцией. В этом случае онлайновый бизнес тесно взаимосвязан с обычными продажами и выступает в качестве дополнительного к основному — розничной торговле. Такая схема, пожалуй, наиболее удачна.

Цель моего курсового проекта - разработать приложение для взаимодействия продавца и покупателя, которое происходит в онлайне. Многие этапы приобретения и продажи довольно успешно проходят дистанционно. Это является большим преимуществом. Но в тоже время, именно в этом и заключается одна из трудностей, которую надо преодолеть любому интернет-магазину - внушить доверие посетителю и расположить его к совершению покупки.

Язык разработки проекта – C#. При выполнении курсового проекта будут использованы принципы и приемы ООП. Также будут использоваться технологии Windows Presentation Foundation(WPF).

## **1.Аналитический обзор литературы**

## **1.1. Аналитический обзор источников**

В ходе подготовки пояснительной записки была изучена специальная техническая, учебно-методическая и справочная литература, статьи и материалы, опубликованные в сети интернет.

Принцип создания окон для регистрации и авторизации был взят из статьи «WPF – система авторизации и регистрации». В статье были рассмотрены алгоритм работы системы авторизации и регистрации и пример создания окон регистрации и авторизации.

Алгоритм работы окна авторизации:

1. выполняется проверка заполняемости всех полей;
2. выполняется проверка существования записи в базе данных;
3. если запись существует, пользователь авторизуется.

Алгоритм работы окна регистрации:

1. выполняется проверка заполняемости всех полей;
2. выполняется проверка существования записи с таким же логином в базе данных;
3. если записи не существует, добавляется новая запись.

Принцип работы с SQL были получены из статьи «Подключение к базе данных». В статье было рассмотрено подключение необходимых библиотек, работа с SQL.

Концепция создания отчета была получена из статьи «Report-Driven Design». В статье была рассмотрена общая информация о создании отчёта и пример создания отчёта.

Дополнительная информации о принципах работы с WPF была получена из интернет-источника «Metanit», содержащего практические советы по работе с технологией.

## **1.2. Аналитический обзор аналогов**

Описание аналогов Интернет-магазинов мобильных телефонов, представлена на Рисунке 1.1 и Рисунке 1.2

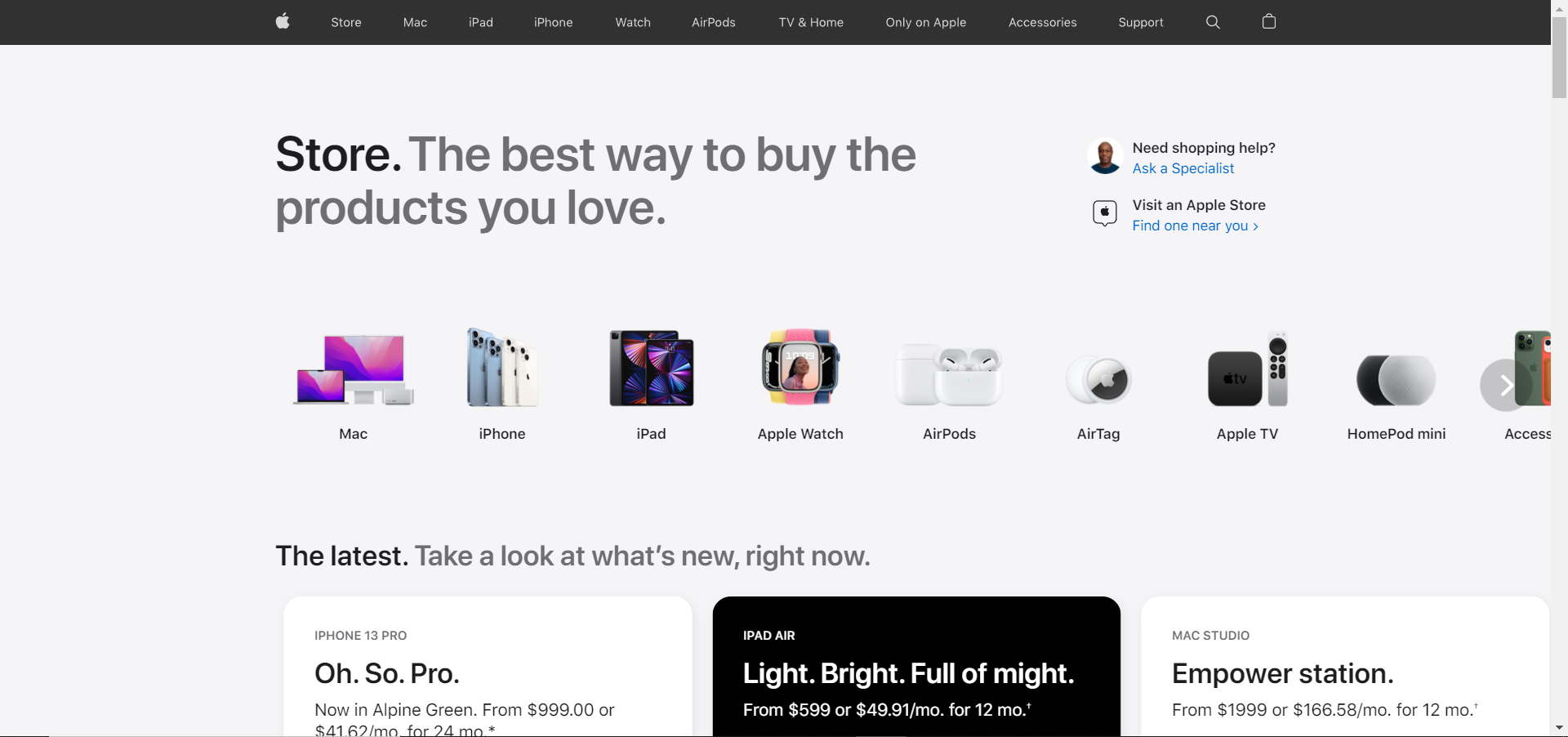


Рисунок 1.1 Аналог Интернет-магазина “App Store”.

Плюсы Интернет-магазина “App store”:

1)Отсортированный поиск по гаджетам.

2)Встроенная функция поддержки для пользователей которые забыли пароль от системы, поломка девайса. Также есть поиск по часто задаваемым вопросам.

3)Есть дополнительный ассортимент аксессуаров.

4)Есть корзина для девайсов, которые вы хотите купить.

5)Можно делать сравнение между девайсами по их характеристикам.

Минусов в этом Интернет-магазине я не нашёл.

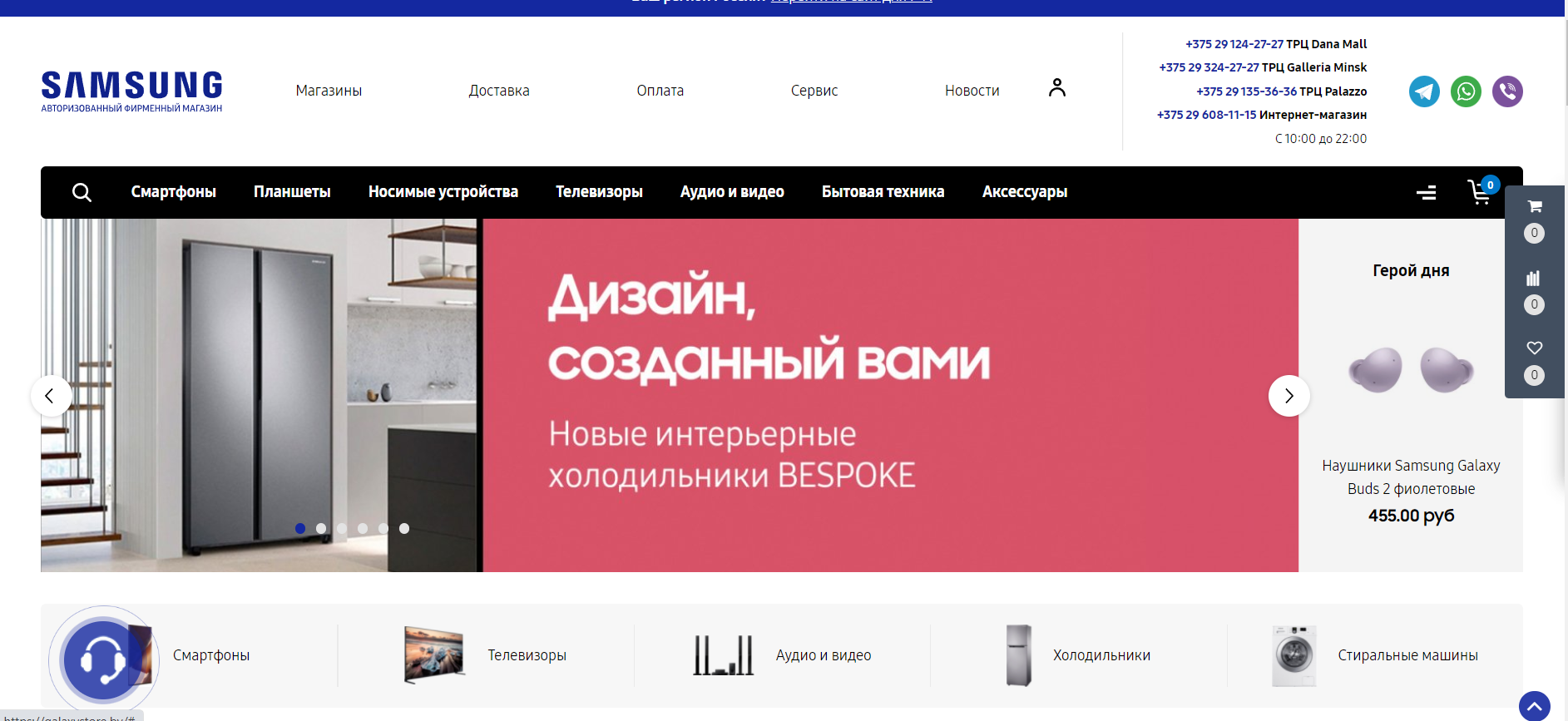


Рисунок 1.2. Аналог Интернет-магазина “Samsung”.

Плюсы Интернет-магазина “Sumsung”:

1)Отсортированный поиск по гаджетам.

2) Есть дополнительный ассортимент аксессуаров.

3) Есть корзина для девайсов, которые вы хотите купить.

Минусы Интернет-магазина “Sumsung”:

1)Нет встроенной функции поддержки для пользователей которые забыли пароль от системы. Нет помощи с ремонтом поломки девайса. Также нет поиска по часто задаваемым вопросам.

2) Нельзя делать сравнение между девайсами по их характеристикам.

## **2.Анализ требований к программному средству и разработка функциональных требований**

## **2.1. Спецификация функциональных требований**

**2.1.1. Требования к параметрам технических и программных средств**

Программное средство может функционировать на ЭВМ со следующими минимальными характеристиками:

1. процессор Intel® Core™ i5 с тактовой частотой 2.4 ГГц и лучше;
2. оперативная память 256 МБ и более;
3. операционная система: Windows 10.

**2.1.2. Требования к организации входных данных**

Входные данные для программного средства должны быть представлены в виде вводимого пользователем с клавиатуры текста: логин, пароль. После аутентификации пользователю предоставляется возможность работы с системой.

Данные, вводимые пользователем, должны проверяться на корректность в процессе аутентификации.

## **2.1.3. Требования к организации выходных данных**

В качестве выходных данных будут выступать файлы, отображающие данные и результаты их обработки в виде отчёта.

## **2.1.4. Требования к надёжности**

Надежное (устойчивое) функционирование программы должно быть обеспечено выполнением совокупности организационно-технических мероприятий, перечень которых приведен ниже:

1. надёжное (устойчивое) функционирование программы должно быть обеспечено выполнением совокупности организационно-технических мероприятий, перечень которых приведен ниже:
2. организацией бесперебойного питания технических средств,   
   выполнением требований «ГОСТ 31078-2002. Защита информации. Испытания программных средств на наличие компьютерных вирусов. Типовое руководство»;
3. необходимым уровнем квалификации сотрудников профильных подразделений.

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать времени, необходимого на перезагрузку операционной системы и запуск программы, при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

## **2.2. Описание функциональности программного средства**

На основании анализа исходных данных для проектируемого программного средства можем выделить, что основной целью является создание качественного программного продукта, позволяющего решить существующие проблемы таких как:

* обеспечение удобного процесса приобретения мобильных телефонов и гарантированное качество в их использовании;

Диаграмма классов представлена в Приложении А.

## **3.Проектирование программного средства**

## **3.1. Структура проекта**

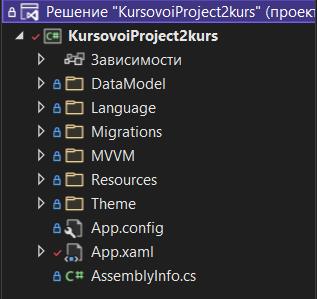


Рисунок 3.1 – Структура проекта

Описание всех пакетов проекта описаны в таблице 3.1.

Таблица 3.1 – Описание структурных пакетов проекта

|  |  |
| --- | --- |
| Имя пакета | Описание |
| DataModel | Здесь описаны модели, с которыми  происходит вся работа в приложении. |
| Resources | Содержит пакеты, в которых находятся словари ресурсов, необходимые графические элементы  и другие файлы. |
| View | Содержит все представления, которые позволяют пользователю работать с приложением. Они описывают графическую  составляющую приложения. |
| ViewModel | Содержит логику, которая позволяет получить данные при помощи View, обработать их, использую при этом Model, после чего передать в базу данных. |

Окончание таблицы 3.1

|  |  |
| --- | --- |
| App.config | Файл конфигурации приложения. |
| Language | Содержит пакеты, в которых находятся словари ресурсов, необходимые для локализации. |
| Migrations | Содержит мигрированную базу данных. |

## **3.2. Реализация архитектуры MVVM**

При создании приложения использовался паттерн проектирование MVVM. Он заключается разделении представления от бизнес-логики. Структура шаблона MVVM представлена на рисунке 3.2.

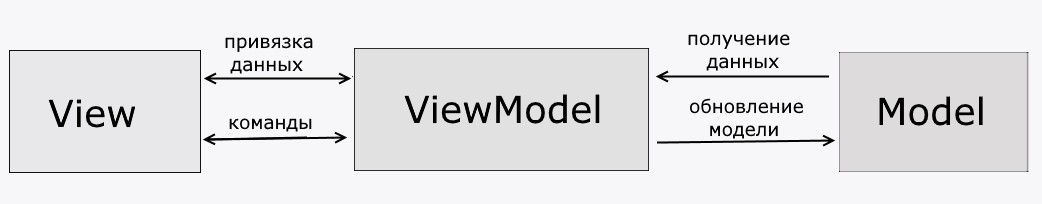


Рисунок 3.2 – Структура шаблона MVVM

Подробная структура проекта представлена на рисунке 3.3.

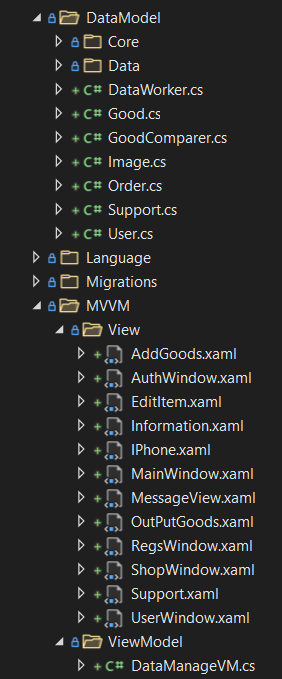


Рисунок 3.3 – Структура проекта

Models описывает используемые в приложении данные, а также содержит логику, непосредственно связанную этими данными.

Views определяет визуальный интерфейс, через который пользователь взаимодействует с приложением.

Кратное описание содержания пакета Views описаны в таблице 3.2.

Таблица 3.2 – Описание содержания пакета Views

|  |  |
| --- | --- |
| Название | Описание |
| AddGoods.xaml | Окно добавления телефона. |
| EditGoods.xaml | Окно редактирования телефона. |
| AuthWindow.xaml | Окно логина. |
| ShopWindow.xaml | Окно администратора, в котором можно просматривать всю информацию. |
| MessageView.xaml | Окно для вывода успешности операции. |
| MainWindow.xaml | Окно пользователя, в котором можно совершать покупки, фильтрацию, поиск. |
| RegsWindow.xaml | Окно регистрации. |
| SupportWindow.xaml | Окно обращения в поддержку |
| UserWindow.xaml | Пользовательское окно |

ViewModels связывает модель и представление через механизм привязки данных.

## **3.3. Проектирование базы данных**

Для данного курсового проекта была спроектирована база данных, которая получила название «ManageAppDBLAST1». Для управления базой данных используется ПО Microsoft SQ Server 2019. Логическая схема данных представлена на рисунке 3.1.

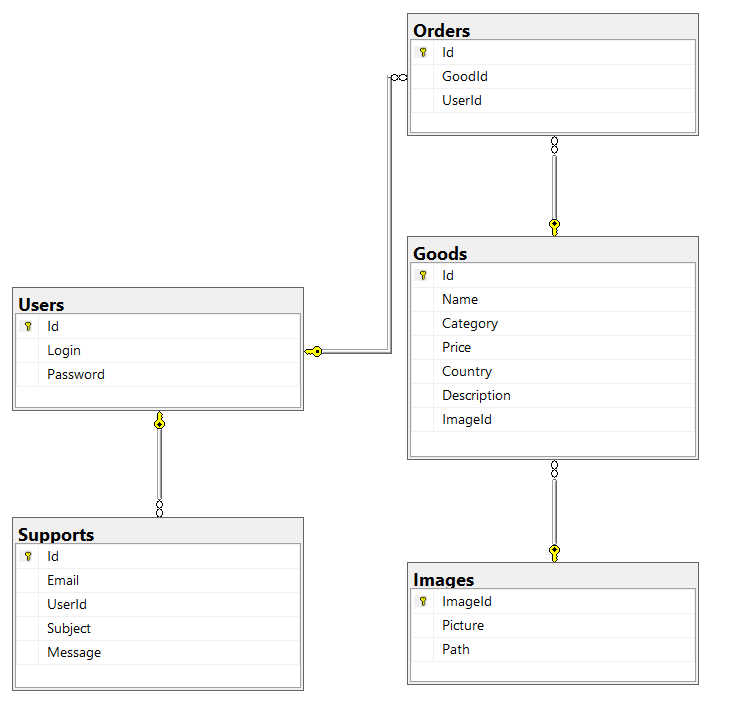


Рисунок 3.1 – Логическая схема данных

В базе данных находится 5 таблиц. В таблице Users хранятся данные для идентификации пользователей, зарегистрированные в приложении, Orders – личные заказы пользователей, Supports – хранит данные пользователя, при отправке сообщения для связи с магазином, Images – хранит данные про изображения наших товаров, Goods – список товаров, добавленных в коллекцию товаров пользователями, с их характеристиками.

Таблица Users состоит из следующих столбцов:

* Id;
* Login;
* Password;

Столбец Login содержит логин пользователя, Password – содержит пароль пользователя.

Таблица Goods состоит из столбцов:

* Id;
* Name;
* Category;
* Price;
* Country;
* Description;
* ImageId;

Столбец Name содержит наименование товара, Category – содержит категорию товара, Price – содержит цену товара, Country – представляет страну изготовителя, Description – указывает на описание товара.

Таблица Supports состоит из столбцов:

* Id;
* Email;
* UserId;
* Subject;
* Message.

В столбце Email хранится электронная почта пользователя, Subject – хранится указанная тема при обращении за помощью к интернет-магазину, Message – хранит сообщение пользователя.

Таблица Orders включает столбцы:

* Id;
* GoodId;
* UserId;

Таблица Images включает столбцы:

* ImageId;
* Picture;
* Path;

Столбец ImageId содержит идентификатор изображения гаджета, Picture – Хранит само изображение, Path – путь к изображению.

Скрипты создания базы данных приложения представлены в Приложении Б.

## **3.4. Проектирование архитектуры проекта**

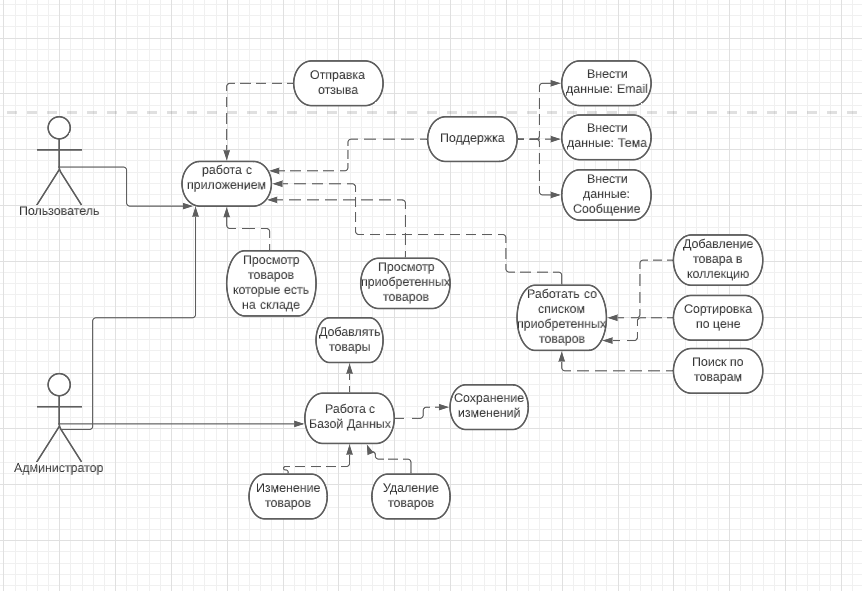
На рисунке 3.2 изображена диаграмма вариантов использования приложения. 

Рисунок 3.2 – Диаграмма использования

На рисунке 3.3 была отображена диаграмма последовательности для авторизации. На диаграмме последовательности отображаются только те объекты, которые непосредственно принимают участие во взаимодействии.



Рисунок 3.3 – Диаграмма последовательности для авторизации

На рисунке 3.4 изображена диаграмма компонентов приложения.

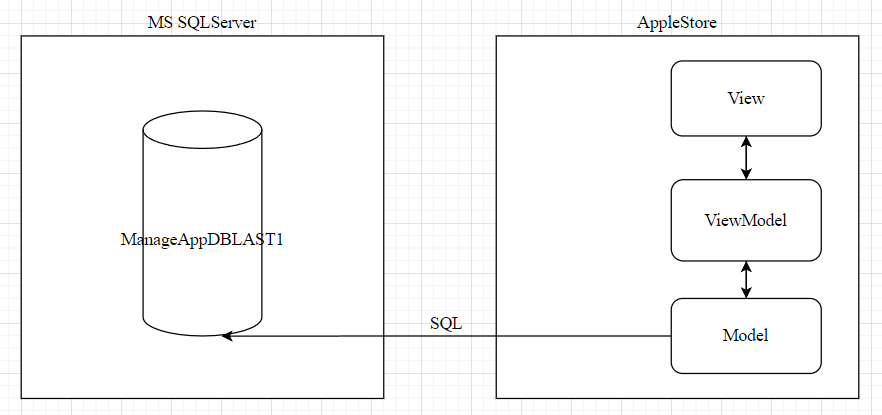


Рисунок 3.4 – Диаграмма компонентов

На рисунке 3.5 представлена блок-схема для алгоритма регистрации пользователя.



Рисунок 3.4 – Схема алгоритма регистрации

Логика перехода между страницами реализована в классе MainWindow. Для получения и записи данных в БД используются классы AplicationContext и DataWorker , которые создают достаточный уровень абстракции для быстро доступа к БД. Для создания подключения и работы с БД используется технология Entity Framework Core. Для администрирования приложения путём доступа к БД был создан пользователь с логином admin, обладающий правами администратора.

При разработке приложения был реализован архитектурный паттерн MVVM. Он используется для разделения модели и её представления, что необходимо для их изменения отдельно друг от друга.

## **3.5. Классы Users, Supports, Images, Orders, Goods**

Данные классы представляют из себя модели сущностей, содержащие поля, аналогичные тем, которые располагаются в базе данных приложения.

## **3.6. Класс AplicationContext**

В любом приложении, работающим с БД через Entity Framework Core, необходимо использовать контекст (класс производный от DbContext) и набор данных DbSet, через который можно взаимодействовать с таблицами из БД. В данном случае таким контекстом является класс AplicationContext.

## **3.7. Классы ViewModel**

Классы с суффиксом ViewModel содержат: закрытые поля, открытые свойства и команды, привязываемые к элементам управления на страницах View. ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление. И также VewModel определяет логику по обновлению данных в модели.

## **3.8. Класс ObservableObject**

Он обеспечивает базовую реализацию для INotifyPropertyChanged и INotifyPropertyChanging, раскрывая события PropertyChanged и PropertyChanging, он предоставляет методы OnPropertyChangedи OnPropertyChanging, которые можно переопределить в производных типах, чтобы настроить способ создания событий уведомления, предоставляет ряд SetPropertyметодов, которые можно использовать для простой установки значений свойств из типов, наследуемых от ObservableObject, и для автоматического создания соответствующих событий.

## **3.9. Класс RelayCommand**

RelayCommand - это ICommandреализации, которые могут предоставлять метод или делегировать представлению. Эти типы действуют как способ связывания команд между моделью представления и элементами пользовательского интерфейса.

## **3.10. Валидация данных**

Чтобы избежать ошибок при вводе данных пользователем в поля приложения, используются ограничения, накладываемые на команды, выполняемые при нажатии кнопок, подтверждающие введённые в поля данные: кнопки не доступны до того момента, пока в поля не будут внесены верные данные. В некоторых окнах предусмотрен вывод предупреждающих сообщений.

## **4. Реализация программного средства**

В данной главе детальнее разбираются некоторые классы приложения, которые используются для выполнения основных операций.

# **4.1. Выполняемые функции**

Приложение выполняет следующие функции:

* регистрация и авторизация;
* добавление телефонов;
* удаление телефонов;
* редактирование телефонов
* покупка и отмена покупки телефонов
* просмотр телефонов;
* просмотр отсортированных по цене телефонов;
* поиск телефонов по названию;
* возможность обращения в поддержку.

## **4.1.1. Регистрация и авторизация**

Регистрация и авторизация пользователей происходит в классе DataManageVM.

Для регистрации в классе находится команда RegisterUser, в которой вызывается метод класса DataWorker GetLogins() (Приложение C), в котором проверяется существует ли данный логин в базе данных, далее сравнивается происходит сравнение паролей, и, если успешно, происходит регистрация пользователя. Для авторизации в классе находится команда LoginUser(Приложение C), в которой вызывается метод класса DataWorker CheckUserData()(Приложение C), в котором проверяется существует ли данный пароль и логин на самом деле и соответствует ли пароль тому что хранится в базе данных, если все успешно – происходит авторизация пользователя и в зависимости от значения логина демонстрирует соответствующую страницу.

## **4.1.2. Добавление телефонов**

Добавление ноутбуков реализовано в команде AddNewGoods(Приложение C) класса DataManageVM.

При добавлении нового ноутбука идет проверка введенных значений на пустоту, если проверка пройдена, то происходит добавление телефона в базу данных посредством метода CreateGoods() класса DataWorker, иначе система уведомляет об ошибке.

public static string CreateGoods(int id, string name, string category, int price, string country, string description, Image image)

{

string result = "Уже существует";

using (ApplicationContext db = new ApplicationContext())

{

//проверяем существует ли ноутбук

bool checkIsExist = db.Goods.Any(el => el.Name == name && el.Category == category && el.Price == price && el.Country == country && el.Description == description);

if (!checkIsExist)

{

Good newTelephone = new Good

{

Id = id,

Name = name,

Country = country,

Price = price,

Description = description,

Category = category,

IPhoneImage = image

};

db.Goods.Add(newTelephone);

db.SaveChanges();

result = "Сделано";

}

return result;

} }

Рисунок 1 – Листинг CreateGoods();

## **4.1.3. Удаление телефонов**

Удаление телефонов реализовано в команде DeleteItem(Приложение C) класса DataManageVM.

При нажатии на кнопку удаления, в команде вызывается метод DeleteGoods() класса DataWorker, в котором происходит удаление соответствующего телефона из базы данных.

public static string DeleteGoods(Good good)

{

string result = "Такого товара не существует";

using (ApplicationContext db = new ApplicationContext())

{

db.Goods.Remove(good);

db.SaveChanges();

DeleteImage(GetImageById(good.ImageId));

result = "Сделано! Товар " + good.Name + " удален из меню";

}

return result; }

Рисунок 2 – Листинг DeleteGoods();

## **4.1.4. Редактирование телефонов**

Редактирование телефонов реализовано в команде EditGoods(Приложение класса DataManageVM.

При нажатии на кнопку редактирования, в команде вызывается метод открытия окна редактирования OpenEditItemWindowMethod(), в который передается выбранный телефон, в открытом окне при нажатии кнопки вызывается команда EditItem, где идет проверка введенных значений на пустоту и на соответствие возможным значениям, если проверка пройдена, то происходит редактирование телефона, иначе система уведомляет об ошибке.

## **4.1.5. Поиск телефонов по названию**

Поиск телефонов по названию реализован в команде SearchGoods класса DataManageVM.

public RelayCommand SearchGood

{

get

{

return searchGood ?? new RelayCommand(obj =>

{

List<Good> goods = new List<Good>();

goods = AllGoods.FindAll(x => x.Name.Contains(IPhones));

AllGoods = goods;

UpdateAllGoodsMyView();

AllGoods = DataWorker.GetAllGoods();

});

} }

Рисунок 3 – Листинг SearchGoods;

## **4.1.6. Сортировка телефонов по цене**

Сортировка телефонов по цене реализована в команде SortGoods класса DataManageVM.

public RelayCommand SortGood

{

get

{

return sortGood ?? new RelayCommand(obj =>

{

AllGoods.Sort((x, y) => x.Price.CompareTo(y.Price));

UpdateAllGoodsMyView();

});

}}

Рисунок 4 – Листинг SortGood;

## **4.1.7. Возможность обращения в поддержку**

Обращение в поддержку реализовано в команде SendMessage класса DataManageVM

При нажатии на кнопку отправки сообщения в команде вызывается CreateSupport() ( Приложение С), и, при успешной отправке, происходит отображение успешной отправки, иначе система уведомляет об ошибке.

public RelayCommand SendMessage

{

get

{

return sendMessage ?? new RelayCommand(obj =>

{

try

{

Window wnd = obj as Window;

if (Mail == null || Mail.Replace(" ", "").Length == 0)

{

SetRedBlockControll(wnd, "MailBlock");

}

if (Subject == null || Subject.Replace(" ", "").Length == 0)

{

SetRedBlockControll(wnd, "SubjectBlock");

}

if (Message == null || Message.Replace(" ", "").Length == 0)

{

SetRedBlockControll(wnd, "MessageBlock");

}

else

{

string resultStr = "";

var smtpClient = new System.Net.Mail.SmtpClient("smtp.mail.ru", 587);

smtpClient.Credentials = new System.Net.NetworkCredential("lilkrug\_2003@mail.ru", "XWcfqqJUPGvTSV7Uw20C");

smtpClient.EnableSsl = true;

smtpClient.Send(new System.Net.Mail.MailMessage("lilkrug\_2003@mail.ru", $"{Mail}", "Интернет-магазин мобильных телефонов",

"Здравствуйте, ваше обращение доставлено, в ближайшее время мы вам ответим"));

resultStr = DataWorker.CreateSupport(Mail, Subject, Message);

ShowMessageToUser(resultStr);

wnd.Close();

SetNullValuesToProperties();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

);

}

}

Рисунок 5 – Листинг SendMessage;

## **4.1.8. Покупка телефонов**

Покупка телефона реализована в команде BuyGood класса DataManageVM

В ней вызывается метод CreateOrder(Приложение С) класса DataWorker.

public RelayCommand BuyGood

{

get{

return buyGood ?? new RelayCommand(obj =>{

string resultStr = "Не выбран товар";

UserWindow usr = new UserWindow();

if (SelectedGoods != null)

{

resultStr = DataWorker.CreateOrder(SelectedGoods, DataWorker.FindIdByLogin(Login));

}

UpdateAllOrdersUserWindow();

ShowMessageToUser(resultStr);

});

}}

Рисунок 6 – Листинг BuyGood;

## **4.1.9. Отмена покупки телефонов**

Покупка телефона реализована в команде CancelOrder класса DataManageVM

В ней вызывается метод DeleteOrder(Приложение С) класса DataWorker.

public RelayCommand CancelOrder

{

get

{

return cancelOrder ?? new RelayCommand(obj =>

{

string resultStr = "Ничего не выбрано";

if (SelectedOrder != null)

{

resultStr = DataWorker.DeleteOrder(SelectedOrder);

UpdateAllOrdersUserWindow();

}

ShowMessageToUser(resultStr);

SetNullValuesToProperties();

});

}

}

Рисунок 7 – Листинг CancelOrder;

## **5.Тестирование**

Основной целью тестирования приложения было стремление доказать невозможность введения пользователем данных, которые бы могли привести приложение в неработоспособное состояние. Были проведены:

* тесты на валидацию данных при авторизации и регистрации;
* тесты на введение данных (наименование, категория, цена, страна, описание) при добавлении товара администратором;

В момент регистрации возможна ситуация, когда пользователь вводит уже существующий логин и пароль. Обработка данного исключения продемонстрирована на рисунке 5.1.

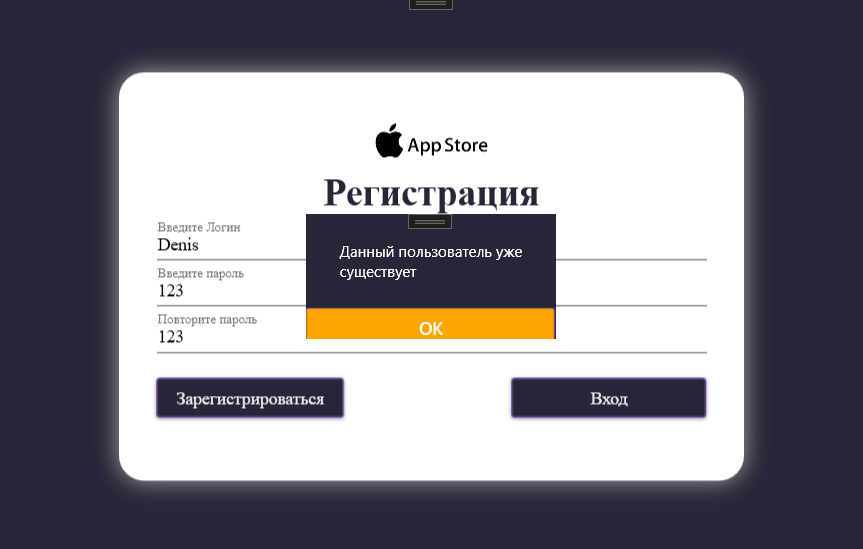


Рисунок 5.1 ­– Данный пользователь уже существует

Предупреждение о неверно введённом пароле или логине продемонстрировано на рисунке 5.2.

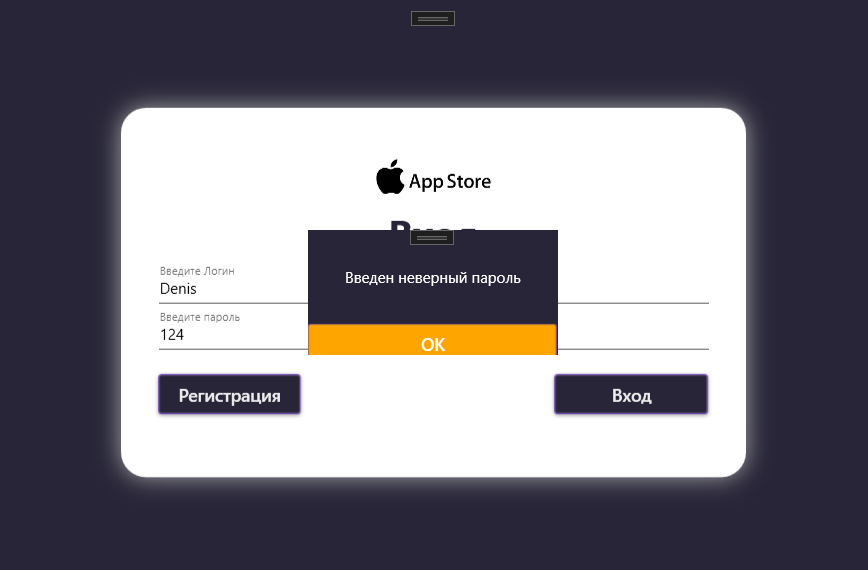


Рисунок 5.2 – Неверный пароль

Предупреждение о не введенных данными некоторых полей ,будет подсвечиваться красным цветом, это продемонстрированно на рисунке 5.3.

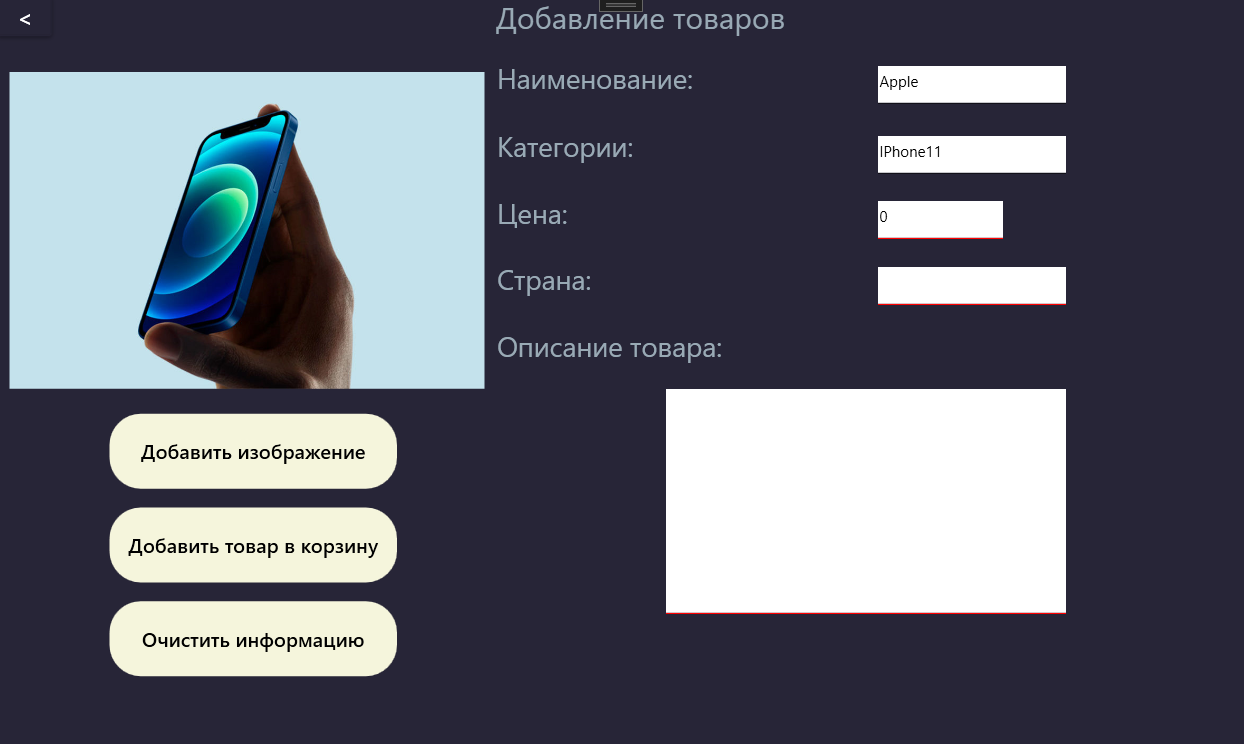


Рисунок 5.3 – Не заполненные данными поля

## **6.Методика использования программного средства**

Окно входа содержит два поля для ввода логина и пароля. После ввода верных данных, необходимо нажать на кнопку вход. Окно входа в аккаунт представлено на рисунке 6.1.

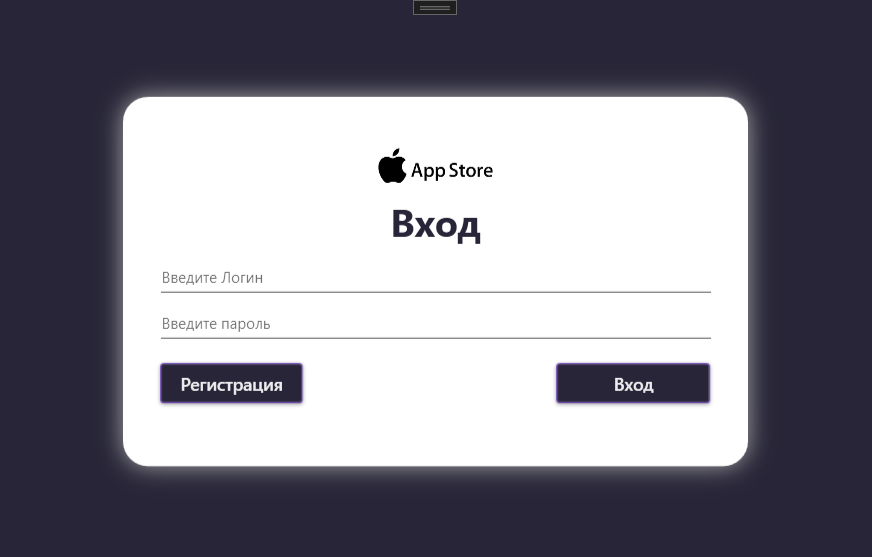


Рисунок 6.1 – Окно входа в аккаунт

Окно регистрации содержит поля для ввода логина, пароля и повторного пароля. Окно регистрации представлено на рисунке 6.2.

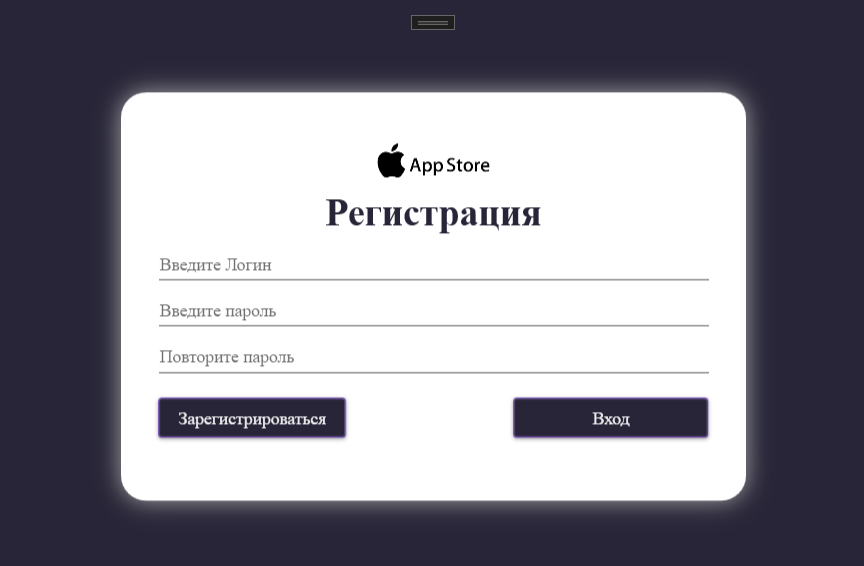


Рисунок 6.2 – Окно регистрации

После авторизации пользователь попадает на домашнюю страницу, которая представляет из себя страницу в которой будут показаны товары, которые можно приобрести, окно домашней страницы пользователя представлено на рисунке 6.3.

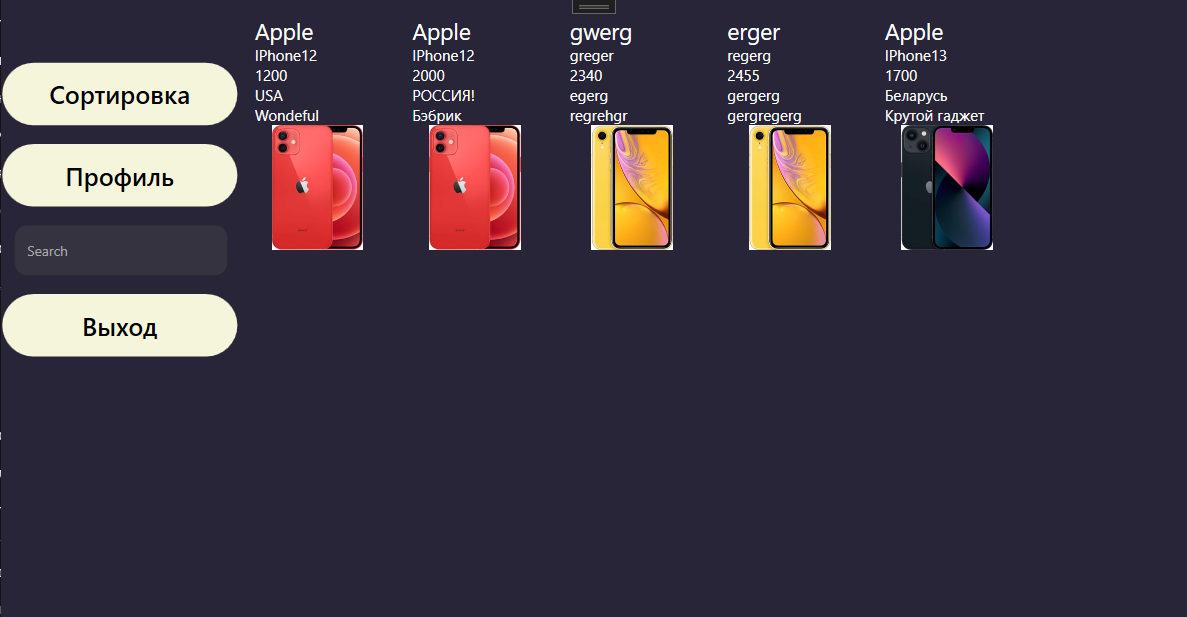
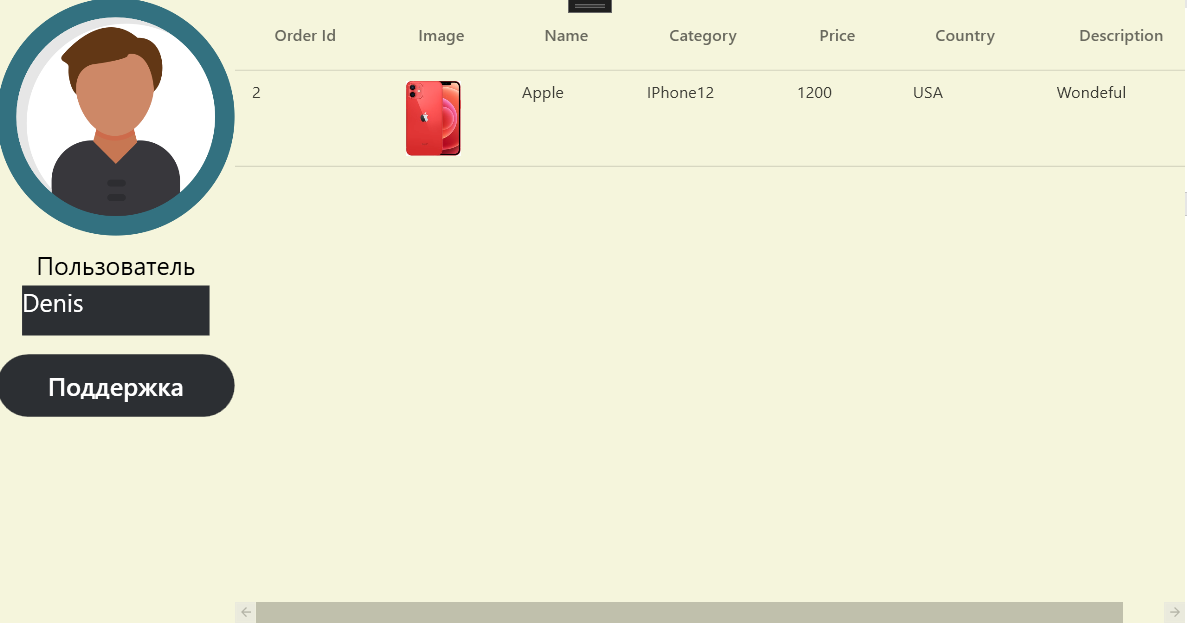


Рисунок 6.3 – Окно показывающее товары для продажи

При нажатии правой кнопкой мыши и нажатии – купить, мы приобретаем товар, после этого переходим в профиль и видим имя нашего пользователя и наш приобретенный товар, окно профиля представлено на рисунке 6.4.

Рисунок 6.4 – Профиль пользователя

Также предусмотрено то, что пользователь может обратиться за помощью в поддержку, при наличии неисправностей в приобретённом товаре. Для этого надо нажать на кнопку поддержка и перейти на следующее окно для заполнение информацией, окно представлено на рисунке 6.5.

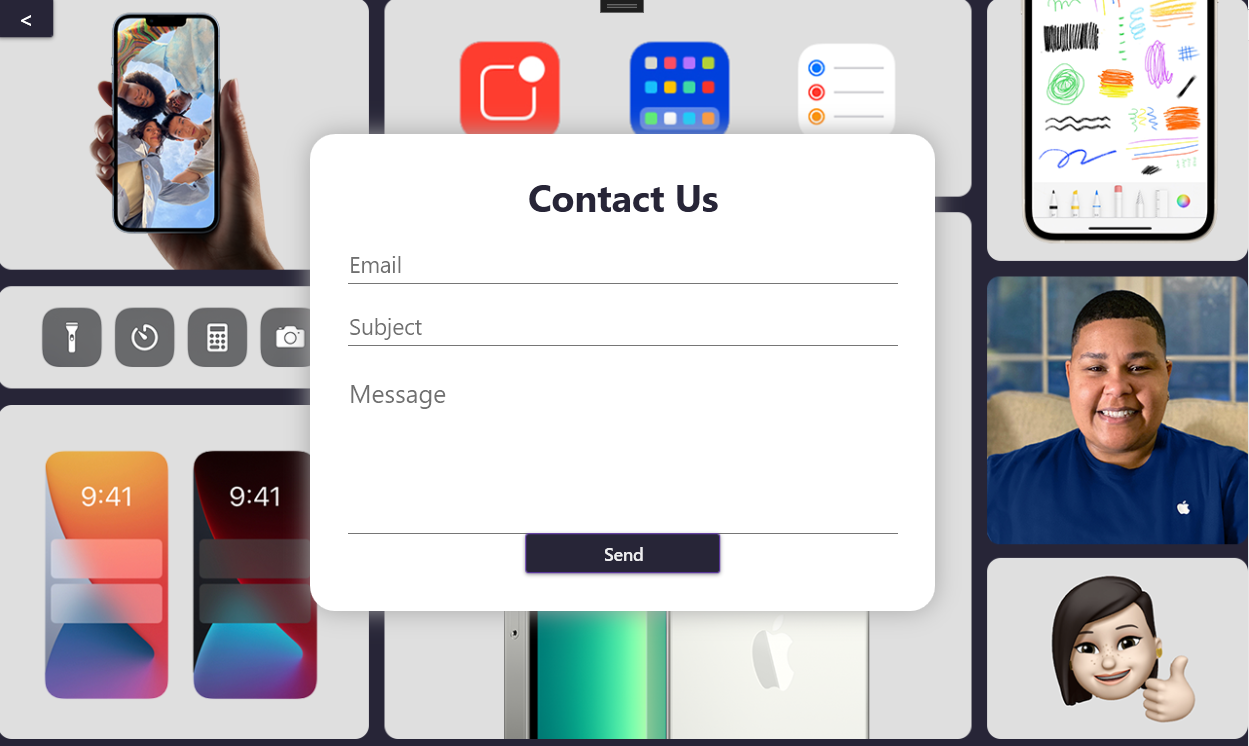


Рисунок 6.5 – Поддержка для пользователя

При входе в приложение под аккаунтом администратора мы попадаем на домашнюю страницу администратора, которая представлена на рисунке 6.6.

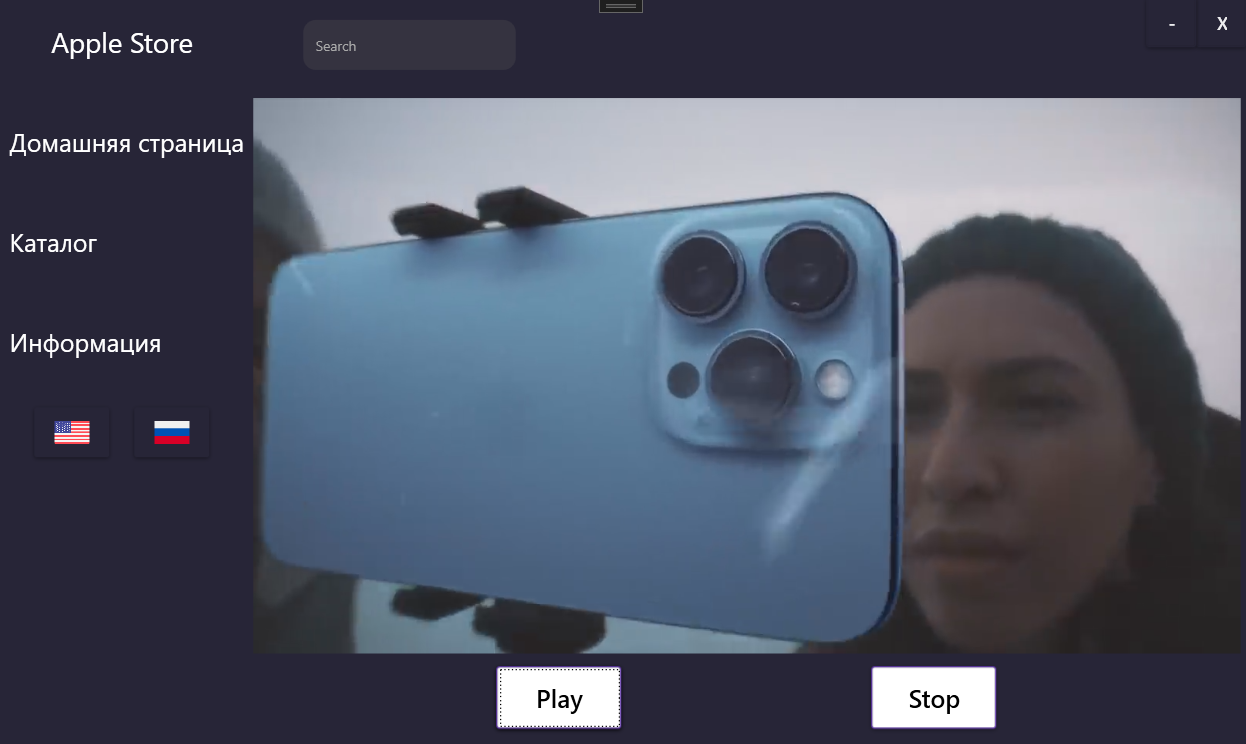


Рисунок 6.6 – Домашняя страница администратора

Страница «Информация» содержит информацию о разработчике, номер версии приложения и небольшое его описание. Данная страница представлена на рисунке 6.7.



Рисунок 6.7 – Страница с информацией о приложении

Страница для добавления новых товаров администратором в магазин представлена на рисунке 6.8.

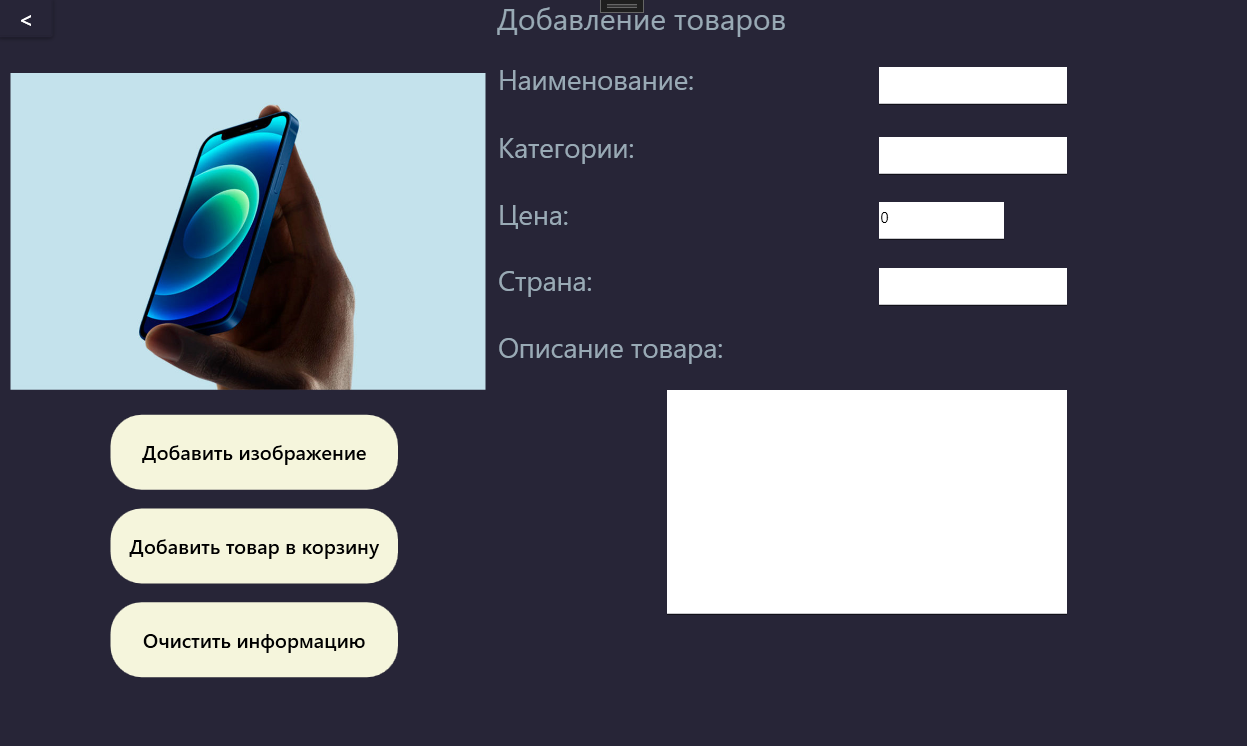


Рисунок 6.8 – Страница для добавления товаров в магазин

На странице администрирование есть возможность редактирования всех таблиц базы данных приложения, в чём и заключается особенность работы с приложением в роли администратора. На странице расположен элемент управления, позволяющий перемещаться между всеми таблицами базы данных, а также текстовые поля, используемые для изменения, удаления или добавления записей. Стоит отметить, что при работе с данной страницей имеет место каскадное удаление записей, что позволяет не нарушать ограничений целостности базы данных. Страница администрирования представлена на рисунке 6.9.

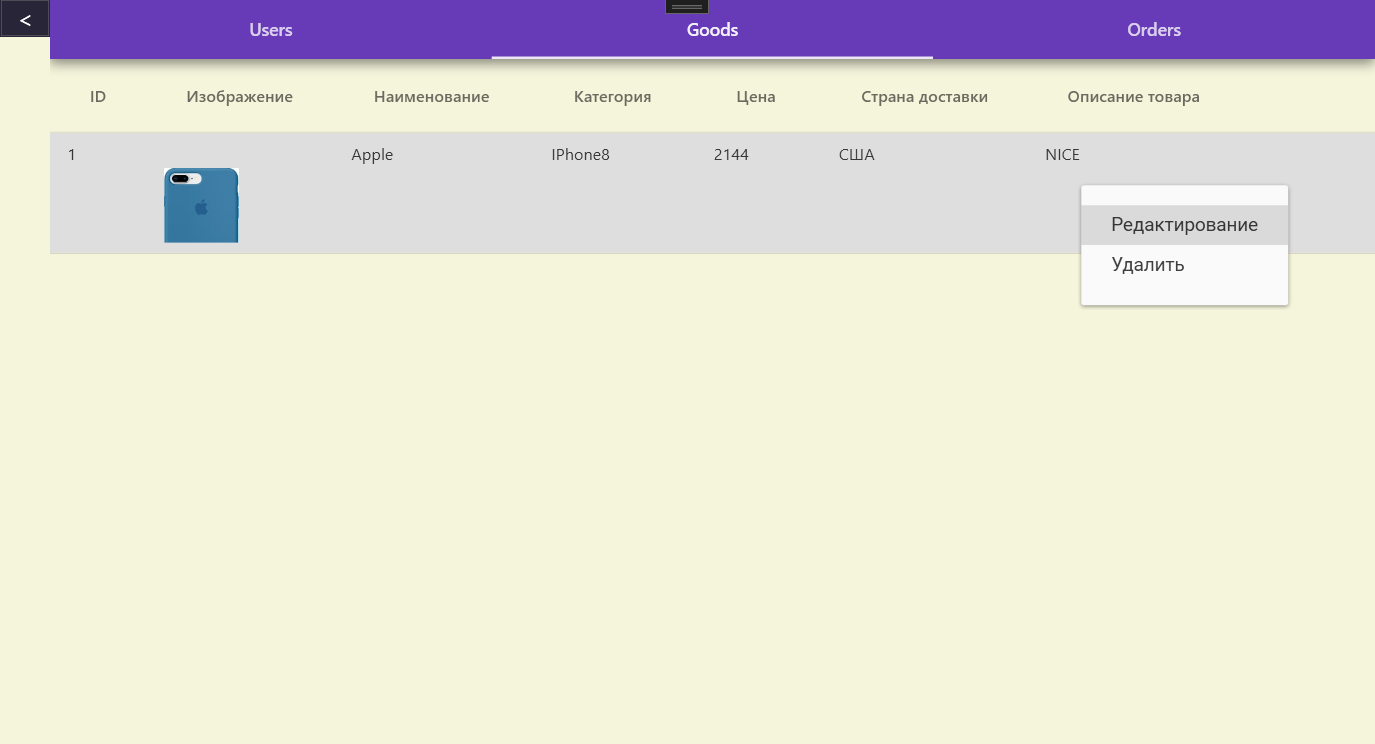


Рисунок 6.9 – Страница администрирования приложения

# **ЗАКЛЮЧЕНИЕ**

Основной задачей данного проекта было усвоение основных технологий разработки десктопных приложений. Разработанное программное средство послужило практической реализацией знаний, полученных в результате изучения теоретических знаний работы с базами данных, технологиями WPF и Entity Framework Core.

В процессе решения поставленной задачи была достигнута поставленная цель по созданию приложения «Интернет-магазин мобильных телефонов». Данное приложение может служить не только примером практического применения полученных знаний, но также представляет из себя готовый продукт, который может иметь практическую пользу при использовании его по назначению.

В программном средстве были реализованы следующие функции:

* авторизация и регистрация;
* вывод телефонов;
* покупка, отмена покупки телефонов;
* поиск и фильтрация телефонов;
* обращение в поддержку для решения каких-либо проблем;
* добавление, изменение и удаление товаров в коллекцию товаров для приобретения;
* администрирование приложения.

Разработанная программа имеет завершённый вид, работает верно и готова к использованию.

## **Список литературы**

[1] Миха Мартин, Роберт Мартин «Agile Principles, Patterns, and Practices in C# » – 2006 г.

[2] Статья “WPF – система авторизации и регистрации” [Электронный ресурс]. – Электронные данные. – https://you-hands.ru/2018/08/31/wpf-sistema-avtorizacii-i-registracii/

[3] Статья “WPF” [Электронный ресурс]. – Электронные данные. – https://metanit.com/sharp/wpf/1.php

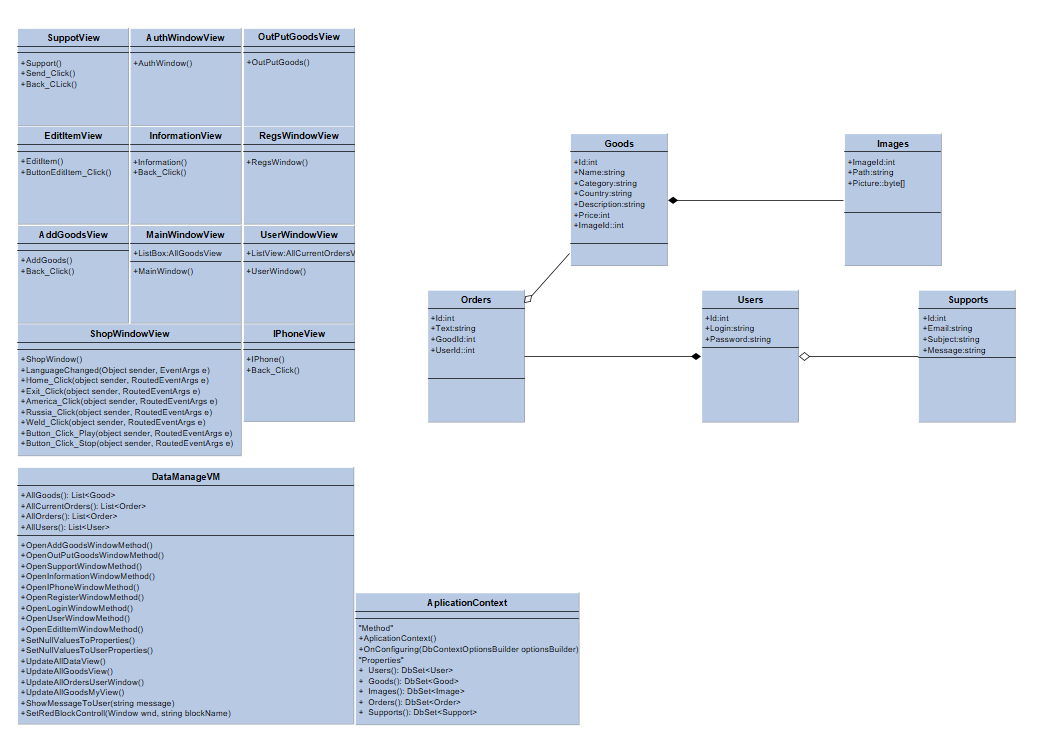
[4] Статья “Generating model from an existing database” [Электронный ресурс]. –Электронные данные. – <https://www.learnentityframeworkcore.com/walkthroughs/existing-database>

[5] Статья “Паттерн MVVM” [Электронный ресурс]. – Электронные данные. – https://metanit.com/sharp/wpf/22.1.php

## **Приложение**

## **Приложение А**

**Диаграмма классов**

****

## **Приложение Б**

**Скрипт для создания таблиц базы данных приложения**

Use master;

Create Database Kursovoi;

Use Kursovoi;

Create Table Users

(

Id int constraint PK\_USERS primary key(Id) identity(1,1),

Login nvarchar(50) NOT NULL Unique,

Password nvarchar(50) NOT NULL Unique

)

Create Table Goods

(

Id int constraint PK\_GOODS primary key(Id) identity(1,1),

Name nvarchar(50) NOT NULL,

Category nvarchar(50) NOT NULL,

Price int default '0' NOT NULL,

Country nvarchar(50) NOT NULL,

Description nvarchar(50) NOT NULL,

ImageId int NOT NULL constraint FK\_GOODS\_DATA\_GOODS foreign key (ImageId) references Images(ImageId)

)

Create Table Images

(

ImageId int constraint PK\_IMAGES primary key(ImageId) identity(1,1),

Picture varbinary NOT NULL,

Path nvarchar(50) NOT NULL

)

Create Table Orders

(

Id int constraint PK\_ORDERS primary key(Id) identity(1,1),

GoodId int NOT NULL constraint FK\_ORDERS\_DATA\_ORDERS foreign key (GoodId) references Goods(Id),

UserId int NOT NULL constraint FK\_ORDERS\_DATA\_ORDERS foreign key (UserId) references Users(Id)

)

Create Table Supports

(

Id int constraint PK\_ORDERS primary key(Id) identity(1,1),

Email nvarchar(50) NOT NULL,

Subject nvarchar(50) NOT NULL,

Message nvarchar(50) NOT NULL

)

## **Приложение С**

public RelayCommand RegisterUser

{

get

{

return registerUser ?? new RelayCommand(obj =>

{

Window wnd = obj as Window;

string PasswNoMatch = "Пароли не сходятся";

string LoginExists = "Данный пользователь уже существует";

if (!DataWorker.GetLogins(Login))

{

if (Password == RePassword && Password != null && RePassword != null)

{

DataWorker.RegisterUser(Login, Password);

SetNullValuesToProperties();

SetNullValuesToUserProperties();

wnd.Hide();

OpenLoginWindowMethod();

wnd.Close();

}

else

{

ShowMessageToUser(PasswNoMatch);

}

}

else

{

ShowMessageToUser(LoginExists);

}

}

);

}

}

Рисунок 1 – Листинг RegisterUser;

public static bool GetLogins(string login)

{

bool result = false;

using (ApplicationContext db = new ApplicationContext())

{

result = db.Users.Any(el => el.Login == login);

}

return result;

}

Рисунок 2 – Листинг GetLogins();

public static string CheckUserData(string login, string password)

{

string result = "Отлично";

using (ApplicationContext db = new ApplicationContext())

{

//проверяем существует ли пользователь

bool checkIsLoginExist = db.Users.Any(el => el.Login == login);

bool checkIsPasswordCorrect = db.Users.Any(el => el.Login == login && el.Password == password);

if (checkIsLoginExist)

{

if (checkIsPasswordCorrect)

{

result = "Отлично";

}

else

{

result = "Введен неверный пароль";

}

}

else

{

result = "Данного пользователя не существует";

}

return result;

}

Рисунок 3 – Листинг CheckUserData();

public static string LoginUser(string login, string password)

{

DataManageVM.Login = login;

DataManageVM.Password = password;

DataManageVM.Id = FindIdByLogin(login);

return "День добры " + login;

}

Рисунок 4 – Листинг LoginUser();

public RelayCommand AddNewIPhone

{

get

{

return addNewIPhone ?? new RelayCommand(obj =>

{

Window wnd = obj as Window;

string resultStr = "";

if (IPhoneName == null || IPhoneName.Replace(" ", "").Length == 0)

{

SetRedBlockControll(wnd, "TextBoxNameGood");

}

if (IPhoneCategory == null || IPhoneCategory.Replace(" ", "").Length == 0)

{

SetRedBlockControll(wnd, "TextBoxCategory");

}

if (IPhoneCountry == null || IPhoneCountry.Replace(" ", "").Length == 0)

{

SetRedBlockControll(wnd, "TextBoxCountry");

}

if (IPhoneDescription == null || IPhoneDescription.Replace(" ", "").Length == 0)

{

SetRedBlockControll(wnd, "TextBoxDescription");

}

if (IPhonePrice == 0)

{

SetRedBlockControll(wnd, "TextBoxPrice");

}

else

{

resultStr = DataWorker.CreateGoods(0, IPhoneName, IPhoneCategory, IPhonePrice, IPhoneCountry, IPhoneDescription, IPhoneImage);

ShowMessageToUser(resultStr);

OutPutGoods wd = new OutPutGoods();

UpdateAllGoodsView();

SetNullValuesToProperties();

wnd.Close();

}

}

);

}

}

Рисунок 5 – Листинг AddNewIPhone;

public RelayCommand DeleteItem

{

get

{

return deleteItem ?? new RelayCommand(obj =>

{

string resultStr = "Ничего не выбрано";

//если телефон

if (SelectedTabItem.Name == "GoodsTab" && SelectedGoods != null)

{

resultStr = DataWorker.DeleteGoods(SelectedGoods);

UpdateAllDataView();

}

//обновление

SetNullValuesToProperties();

OutPutGoods outPutGoods = new OutPutGoods();

ShowMessageToUser(resultStr);

});

} }

Рисунок 6 – Листинг DeleteItem;

public RelayCommand EditGoods

{

get

{

return editGoods ?? new RelayCommand(obj =>

{

Window window = obj as Window;

string resultStr = "Не выбран телефон";

string noCategoryStr = "Не выбрана категория";

string noPriceStr = "Не выбрана цена";

string noCountryStr = "Не выбрана страна изготовитель";

string noDescriptionStr = "Нет Описания товара";

if (SelectedGoods != null)

{

if (IPhoneCategory != null)

{

if (IPhonePrice != 0)

{

if (IPhoneCountry != null)

{

if (IPhoneDescription != null)

{

resultStr = DataWorker.EditGoods(SelectedGoods, IPhoneCategory, IPhonePrice, IPhoneCountry, IPhoneDescription);

UpdateAllDataView();

SetNullValuesToProperties();

ShowMessageToUser(resultStr);

window.Close();

}

else

{

ShowMessageToUser(noDescriptionStr);

}

}

else

{

ShowMessageToUser(noCountryStr);

}

}

else

{

ShowMessageToUser(noPriceStr);

}

}

else

{

ShowMessageToUser(noCategoryStr);

}

}

else

{

ShowMessageToUser(resultStr);

}

});

}

}

Рисунок 7 – Листинг EditGoods;

public static string CreateOrder(Good good, int userId)

{

string result = "Не получилось";

using (ApplicationContext db = new ApplicationContext())

{

Order newOrder = new Order

{

Id = 0,

GoodId = good.Id,

UserId = userId

};

db.Orders.Add(newOrder);

db.SaveChanges();

result = "Сделано";

return result;

}

}

Рисунок 8 – Листинг CreateOrder();