|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ ИМЕНИ К.Г. РАЗУМОВСКОГО (ПЕРВЫЙ КАЗАЧИЙ УНИВЕРСИТЕТ)»**  **(ФГБОУ ВО «МГУТУ ИМ. К.Г. РАЗУМОВСКОГО (ПКУ)»)** |
| **УНИВЕРСИТЕТСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ** |
|  |

|  |
| --- |
|  |
| **КУРСОВОЙ ПРОЕКТ** |
| по междисциплинарному курсу: МДК.02.02. Технология разработки и защиты баз данных. |
|  |
| на тему: Разработка системы автоматизации работы интернет-магазина |
|  |
| Студента группы 090203-9о-20/1  специальности 09.02.03 Программирование в компьютерных системах |
| Попова Михаил Вячеславовича |

|  |  |  |
| --- | --- | --- |
| Студент |  | М.В. Попов |
| Руководитель курсового проекта |  | Е.А. Ларионова |
| Председатель ПЦК специальности 09.02.03 Программирование в компьютерных системах |  | А.И. Глускер |

|  |  |  |
| --- | --- | --- |
| Дата защиты «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2023 г. | | |
| Оценка:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | |
| Заведующий отделением № 1 |  | И.А. Миланова |

Москва

2023

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 3

1 ОБЩАЯ ЧАСТЬ 5

1.1 Назначение и цели создания системы 5

1.1.1 Объект исследования 5

1.1.2 Предмет исследования 5

1.1.3 Основная цель проекта 5

1.2 Обзор и анализ предметной области 5

1.3 Жизненный цикл базы данных 7

1.3.1 Анализ предметной области 7

1.3.2 Проектирование 8

1.3.3 Реализация 8

1.3.4 Тестирование 9

1.3.5 Эксплуатация и сопровождение 9

1.4 Выбор и характеристика СУБД 9

1.5 Выбор и характеристика среды разработки приложения 10

2 СПЕЦИАЛЬНАЯ ЧАСТЬ 12

2.1 Постановка задачи 12

2.2 Архитектура информационной системы 12

2.3 Логическая модель базы данных 13

2.4 Нормализация таблиц 14

2.5 Описание таблиц 17

2.6 Другие объекты базы данных 21

2.7 Разработка приложения 24

2.7.1 Диаграмма вариантов использования 24

2.7.2 Решение главной задачи проекта 24

2.7.3 Тестирование приложения 48

2.7.4 Защита информационной системы 49

2.8 Инструкция пользователю 49

2.8.1 Общие сведения об информационной системе 50

2.8.2 Требования к техническим средствам 51

2.8.3 Требования к программным средствам 51

2.8.4 Формы ввода 51

2.8.5 Прокат автомобиля 51

ЗАКЛЮЧЕНИЕ 52

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 53

ПРИЛОЖЕНИЕ А. SQL скрипты на создание и заполнение базы данных 54

ПРИЛОЖЕНИЕ Б. КОД ПРОГРАММЫ 72

Окно «Входа» 72

Окно «Выбора авто» 78

Окно «Аренда автомобиля» 85

Окно «Поездка» 88

# ВВЕДЕНИЕ

Курсовой проект посвящен разработке информационной системы автоматизации интернет-магазина.

Основная цель проекта – создание приложения средствами языка программирования C#, которое позволяет клиенту оформлять заказы, администраторам следить за корректностью содержания товаров. Также данное приложение должно поддерживать подключение базы данных (MS SQL).

База данных создается для информационного обслуживания клиентов, желающих заказать товары, а также администраторов для поддержки актуальности базы данных. БД должна содержать данные о товарах, пользователях, категориях и подкатегория товаров, личные данные пользователей такие как: имя, дата рождения, номер телефона и т.д., позволять выбирать пользователям товары исходя из их характеристик, стоимости и категорий, оформлять заказы.

Система автоматизации работы интернет-магазина предназначена для пользователей, администраторов и продавцов. Пользователям предоставлена возможность сортировать товары, по определенным характеристикам, параметрам и категориям, складывать товары в корзину, оформлять заказы, изменять данные об учетной записи в профиле. Администраторы имеют право удалять, изменять, а так же добавлять товары, не исключая возможностей пользователей. Продавцы могут добавлять товары и совершать те же действия, что и пользователи.

Система автоматизации работы интернет-магазина является актуальной и необходимой на текущий момент времени так как, большое количество людей предпочитаю совершать покупки удаленно в различных сервисах, нежели выбирать товары в розничных магазинах.

Информационная система позволяет упростить и ускорить бизнес-процессы, такие как регистрация пользователей, управление данными пользователей и оформление товаров. Система предоставляет пользователям простую возможность выбрать и купить товар, без необходимость посещать различные розничные магазины. Администраторам система предоставляет создавать, удалять, изменять характеристики товаров и т.д. Курсовой проект направлен на достижение этих целей.

При разработке приложения были рассмотрены уже существующие приложения в данной предметной области, такие как «Ozon», «DNS» и т.д.

# Общая часть

## Назначение и цели создания системы

### Объект исследования

Информационная система для автоматизации работы интернет-магазина.

### Предмет исследования

Настольное приложение, выполняющее функции системы автоматизации работы интернет-магазина, взаимодействующее с базой данных MSSQL Server.

### Основная цель проекта

Разработка информационной системы для автоматизации работы интернет-магазина, которая позволяет пользователю выбрать и оформить товары.

## Обзор и анализ предметной области

Приложение создаётся для удалённого обслуживания пользователей, а также для администрирования информационного ресурса. В базе данных должна храниться следующая информация: товары, роли пользователей, дату и время добавления товара в корзину, свойства товара и т.д.

В соответствии с предметной областью система строится с учётом следующих особенностей:

* Пользователь может добавить и оформить товары доступные для продажи.
* Администраторы вносят изменения в информационный ресурс, путём изменения, удаления, добавления записей.
* Администраторы могут использовать систему как пользователи.

Выделим базовые сущности этой предметной области:

**Классы**. Атрибуты классов – код класса.

**Категории**. Атрибуты категорий – код категории, название категории.

**Подкатегории**. Атрибуты подкатегории – код подкатегории, код категории, свойства подкатегории

**Товары**. Атрибуты товаров – код товара, код подкатегории, свойства товара.

**Корзина.** Код пользователя, код товара, дата добавления товара, количество единиц товара.

**Пользователь.** Атрибуты пользователя – код пользователя, роль пользователя, электронная почта, пароль.

**Роли.** Атрибуты Роли – код роли, название роли.

**Личная информация пользователя.** Атрибуты личной информации пользователя – код пользователя, имя пользователя, фамилия, отчество, номер телефона, фото пользователя.

**Заказы.** Атрибуты заказа – код пользователя, код заказа, дата и время составления заказа, статус оплаты, дата и время оплаты

**Детали заказа.** код заказа, код товара.

## Жизненный цикл базы данных

Жизненный цикл базы данных включает следующие этапы:

* Анализ предметной области,
* Проектирование,
* Реализация,
* Тестирование,
* Эксплуатация и сопровождение.

### Анализ предметной области

Для разработки и построения логической модели базы данных будет использоваться ER Assistant. Для реализации физической базы данных проекта будет использован Microsoft SQL Server где будет храниться вся информация. Для написания программного продукта требуется Visual Studio 2022.

Разработчик располагает всеми необходимыми средствами для разработки баз данных: ПО построения логических моделей БД (ER Asistant), СУБД (MS SQL SERVER 2019), среда разработки ПО (Visual Studio 2022 Community).

Необходимо создать настольное приложение для системы автоматизации работы интернет-магазина, взаимодействующее с базой данных SQL.

Требования к проекту:

* База данных должна быть расширяемой для добавления новых пользователей, администраторов, товаров, подкатегорий, и т.д.,
* Авторизированный пользователь должен иметь возможность просматривать список всех доступных товаров.

### Проектирование

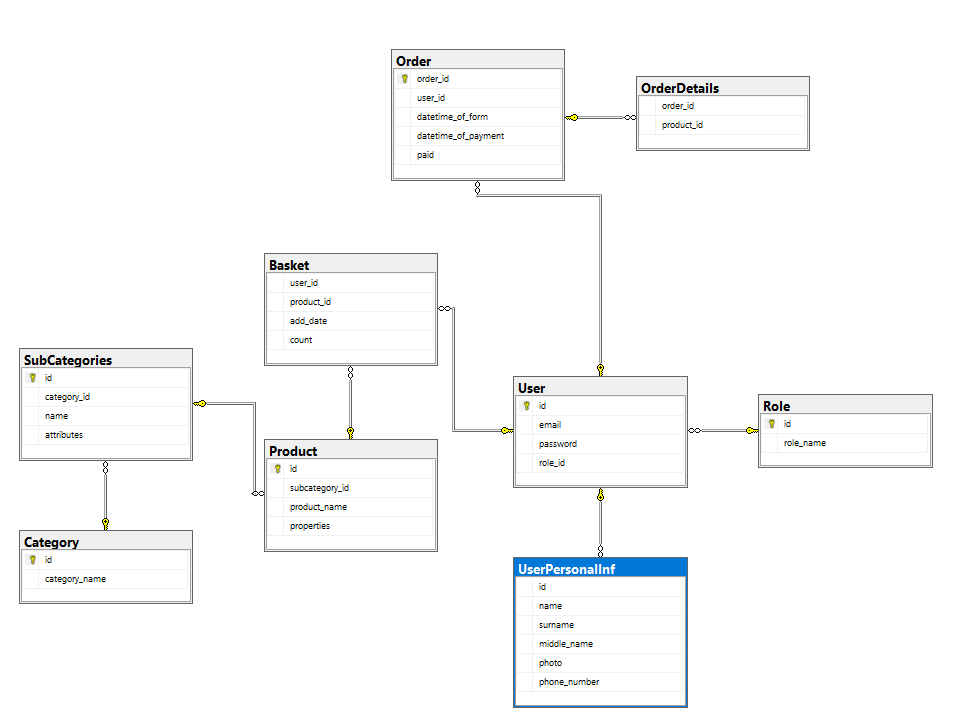


Рисунок 2 – Физическая модель базы данных

### Реализация

Разработка программы, позволяющей работать с БД производилась в Visual Studio 2022 на языке C#, на платформе Windows Presentation Foundation (WPF)

### Тестирование

Тестирование и отладка программного обеспечения выполнялись во время реализации. Все ошибки, выявленные при запуске или работе с приложением – устранены. Тестирование прошло успешно.

### Эксплуатация и сопровождение

Программный продукт готов к использованию и дальнейшему сопровождению.

## Выбор и характеристика СУБД

Для разработки информационной системы была выбрана реляционная СУБД «MS SQL SERVER 2019». Так как в системе предусматривалась концепция клиент-серверного взаимодействия, которая включает в себя сервер с базой данных и клиентов которых обслуживает этот сервер.

Данная архитектура была выбрана по причине того, что клиенты и сервер могут работать на разных устройствах в сети.

Выбранная архитектура позволяет распределить нагрузку на устройствах и обеспечить высокую производительность приложений, масштабировать приложения в случае необходимости. Также данная архитектура позволяет сделать части информационной системы независимыми, что делает возможным независимое изменение или обновление конкретной части.

Выбор СУБД MS SQL SERVER 2019 был обусловлен тем, что – эта СУБД обладает огромным сообществом пользователей, имеет высокую производительность и скорость работы с базами данных, бесплатное обслуживание, обеспечивает безопасность и защиту данных, имеет обширный функционал и гибкость настройки, а также имеет отличную совместимость с другими программными продуктами компании Microsoft.

Преимущества платформы:

* Высокая производительность,
* Безопасность,
* Надёжность.
* Масштабируемость

Из недостатков можно выделить то, что, MSSQL SERVER работает только на операционной системе Windows.

## Выбор и характеристика среды разработки приложения

Для разработки приложения была выбрана интегрированная среда разработки «Visual Studio 2022», включающая [интегрированную среду разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) программного обеспечения и ряд других инструментальных средств, язык программирования c#, и модуль Entity Framework.

C# — это высокоуровневый объектно-ориентированный язык программирования общего назначения, разработанный компанией Microsoft в рамках платформы .NET Framework, который может использоваться для разработки разнообразных типов приложений для операционных систем Windows, macOS, Android, iOS, Linux, а также для веб-приложений и сервисов.

Entity Framework — это современный модуль сопоставления отношений объектов, который позволяет создавать чистый, переносимый и высокоуровневый уровень доступа к данным с помощью .NET (C#) в различных базах данных. Он поддерживает запросы LINQ, отслеживание изменений, обновления и миграции схемы.

Entity Framework позволяет ускорить разработку приложений связанных с базами данных, а также упростить написание кода разработчику.

Visual Studio одной из лучших IDE для C#, так как она также является продуктом корпорации Microsoft, как и C#. Microsoft старается создавать продукты с расчётом на то что они будут максимально взаимодействовать между собой.

Visual Studio 2022 позволяет создавать не только [консольные](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) [приложения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), но и, приложения с [графическим интерфейсом](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F), в том числе с поддержкой технологий Windows Presentation Foundation и [Windows Forms](https://ru.wikipedia.org/wiki/Windows_Forms), а также [веб-сайты](https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%B9%D1%82), [веб-приложения](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5), [веб-службы](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D0%B0) для всех платформ, поддерживаемых [Windows](https://ru.wikipedia.org/wiki/Windows), [Windows Mobile](https://ru.wikipedia.org/wiki/Windows_Mobile), и т.д.

В число преимуществ Visual Studio входит следующее:

* Наличие бесплатной версии;
* Множество встроенных инструментов, позволяющих комфортно работать на C#;
* Поддержка .Net платформы;
* Поддержка git.

# **СПЕЦИАЛЬНАЯ ЧАСТЬ**

## Постановка задачи

Средствами MS SQL SERVER спроектировать базу данных и создать приложение средствами C# и WPF, позволяющее, реализовать систему для проката автомобилей.

Система должна позволять

* Зарегистрировать пользователя;
* Осуществлять поиск автомобиля на карте, с использованием карты Google Maps;
* Выбор автомобиля по заданным критериям из списка имеющихся автомобилей;
* Арендовать выбранный автомобиль;
* Поддержку базы данных в актуальном состоянии;

Обеспечить защиту базы данных и приложения.

## Архитектура информационной системы

База данных

Сервер

Клиент

Рисунок 3 – Клиент-серверная модель

Архитектура информационной системы представляет собой клиент-серверную модель взаимодействия с базой данных. На сервере хранится база данных, в свою очередь пользователь клиент-приложения может взаимодействовать с сервером. Сначала пользователь должен пройти аутентификацию или регистрацию, Далее пользователь может проходить авторизацию, просматривать данные об авто, выбирать авто, арендавать авто и т.д. Сервер в данном случае - это компьютер пользователя. Клиент-приложение сначала обращается к серверу, только после этого сервер уже обращается к базе данных и получает все необходимые данные, которые нужны пользователю.

Преимущества:

* Все данные хранятся на сервере, который защищён лучше большинства клиентов;
* Распределение функций вычислительной системы между несколькими независимыми компьютерами в сети;
* Высокая надёжность;

Недостатки:

* Высокая стоимость оборудования.
* Информационная система полностью зависит от сервера;
* Сопровождение данной информационной системы требует отдельного специалиста — системного администратора,

## Логическая модель базы данных

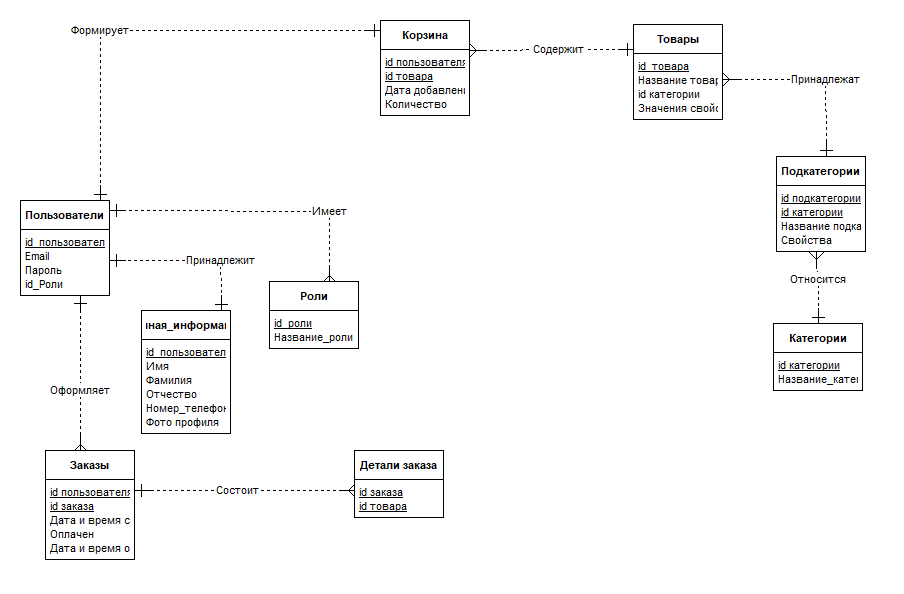


Рисунок 1 – Логическая модель базы данных

## Нормализация таблиц

Таблицы были приведены к первой нормальной форме, все атрибуты просты, далее неделимы.

Таблицы приведены ко второй нормальной форме так как каждый не ключевой атрибут функционально полно зависит от ключа.

Таблицы были приведены к третьей нормальной форме, отсутствует транзитивная зависимость.

Код класса\*

Рисунок 4 – «Классы»

//Доделать

## Разработка приложения

### Диаграмма вариантов использования

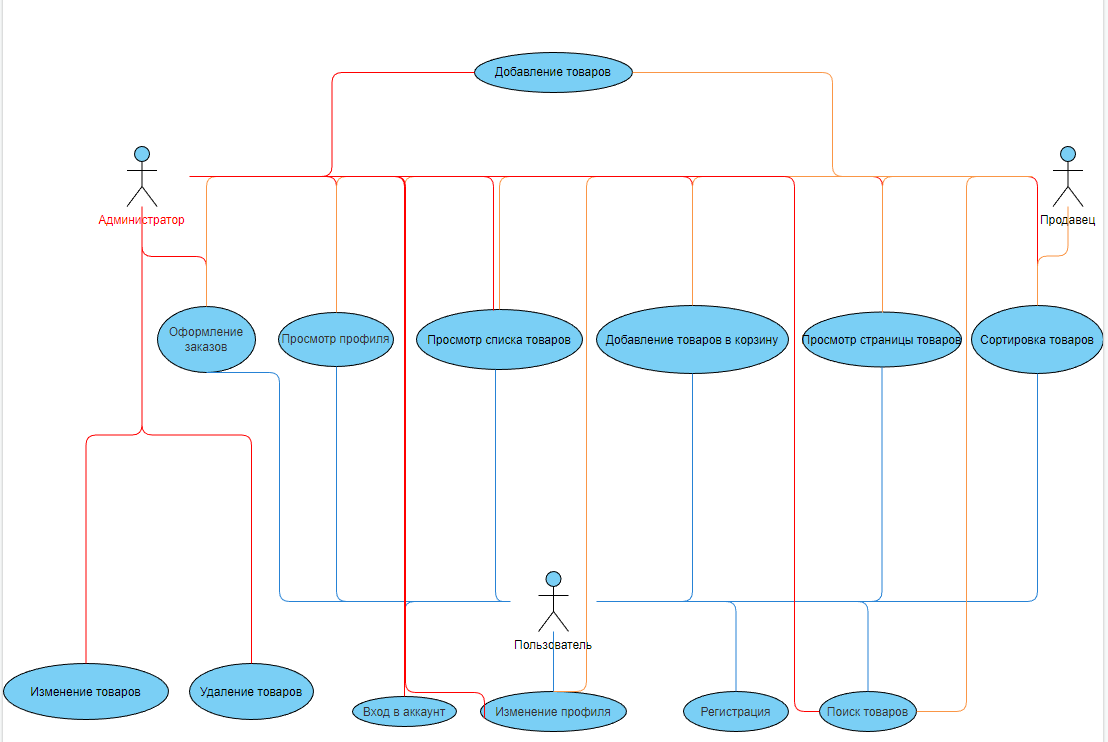


Рисунок 16 – Диаграмма вариантов использования

### Решение главной задачи проекта

Главной задачей проекта по созданию информационной системы для системы автоматизации работы интернет-магазина, состоит из создания приложения, которое будет предоставлять функционал для выборки товаров и эффективно управлять всеми бизнес процессами оформления заказов в интернет-магазине: сортировка товара по определенным характеристикам, поиск товаров, выбор определенного товара, выбор количества товара, оплата и предоставление пользователю информации в удобной форме.

Подзадачи, которые необходимо решить для реализации главной задачи проекта:

* Пользователь регистрируется в системе
* Пользователь добавляет товары в корзину
* Пользователь формирует заказ
* Пользователь оплачивает заказ

В данном фрагменте кода регистрация и авторизация пользователя (смотреть рисунок – 17, 18).

GO

CREATE PROCEDURE AddUser

@email varchar(256),

@password varchar(max),

@name varchar(max),

@surname varchar(max) = '',

@middle\_name varchar(max) = '',

@photo varbinary(max) = NULL,

@phone\_number varchar(12) = NULL,

@role char(1) = 'U'

AS

INSERT INTO [User](email, [password], role\_id) VALUES(@email, @password, @role);

INSERT INTO UserPersonalInf(id, name, surname, middle\_name, photo, phone\_number)

VALUES(

(

SELECT id FROM [User] WHERE email=@email

),

@name, @surname, @middle\_name, @photo, @phone\_number);

GO

namespace InternetStore.Pages

{

public partial class SignUp : Page

{

private static SignUp \_context = null!;

private SignUp()

{

InitializeComponent();

}

public static SignUp GetInstance()

{

if (\_context == null)

{

\_context = new SignUp();

}

return \_context;

}

private void OnSignUp(object sender, RoutedEventArgs e)

{

var login = (WatermarkTextBox?)UIHelper.FindUid(this, "LoginField");

var password = (WatermarkPasswordBox?)UIHelper.FindUid(this, "PasswordField");

var User = BaseProvider.DbContext.Users

.ToList()

.Where(user => (login.Text == user.Email) && (password.Password == user.Password.Decrypt()))

.FirstOrDefault();

if (User != null)

{

var userDTO = BaseProvider.DbContext.UserViewDtos.ToList().Where(user => user.Id == User.Id).First();

NavigationService.Navigate(new StoreMain(userDTO));

}

else

Xceed.Wpf.Toolkit.MessageBox.Show("Неверный логин или пароль");

}

private void ReturnToMainPage(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(null);

}

}

}

namespace InternetStore.Pages

{

public partial class Registration : Page

{

#region [ Binding Fields]

private DependencyProperty userNameField =

DependencyProperty.Register("UserName", typeof(string), typeof(Registration));

private DependencyProperty emailField =

DependencyProperty.Register("Email", typeof(string), typeof(Registration));

private DependencyProperty passwordField =

DependencyProperty.Register("Password", typeof(string), typeof(Registration));

private DependencyProperty repeatPasswordField =

DependencyProperty.Register("RepeatPassword", typeof(string), typeof(Registration));

#endregion

#region [ Binding Properties ]

[Required(AllowEmptyStrings = false)]

public string UserName

{

get => (string)GetValue(userNameField);

set => SetValue(userNameField, value);

}

[Required(AllowEmptyStrings = false)]

[EmailAddress(ErrorMessage = "Невалидная почта!")]

public string Email

{

get => (string)GetValue(emailField);

set => SetValue(emailField, value);

}

//[Required(AllowEmptyStrings = false)]

[PasswordPropertyText]

public string Password

{

get => (string)GetValue(passwordField);

set => SetValue(passwordField, value);

}

[Required(AllowEmptyStrings = false)]

[Compare("Password")]

public string RepeatedPassword

{

get => (string)GetValue(repeatPasswordField);

set => SetValue(repeatPasswordField, value);

}

#endregion

private static Registration \_context = null!;

private Registration()

{

InitializeComponent();

}

public static Registration GetInstance()

{

if (\_context == null)

{

\_context = new Registration();

}

return \_context;

}

private bool isValid(object property, string propertyName)

{

var errors = new List<System.ComponentModel.DataAnnotations.ValidationResult>();

Validator.TryValidateProperty(property, new ValidationContext(this, null, null) { MemberName = propertyName }, errors);

return errors.IsNullOrEmpty();

}

private bool IsValidEmail()

{

bool result = isValid(Email, nameof(Email));

if (!result)

{

Xceed.Wpf.Toolkit.MessageBox.Show("Некорректный адрес электронной почты!");

}

return result;

}

private bool IsValiUserName()

{

bool result = isValid(UserName, nameof(UserName));

if (!result)

{

Xceed.Wpf.Toolkit.MessageBox.Show("Некорректное имя пользователя!");

}

return result;

}

private bool IsValidPassword()

{

if (!isValid(Password, nameof(Password)))

{

Xceed.Wpf.Toolkit.MessageBox.Show("Слишком простой пароль!"); return false;

}

else if (!isValid(RepeatedPassword, nameof(RepeatedPassword)))

{

Xceed.Wpf.Toolkit.MessageBox.Show("Пароли не совпадают!"); return false;

}

return true;

}

private bool Validate()

{

return (IsValiUserName() && IsValidEmail() && IsValidPassword());

}

private void ReturnToMainPage(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(null);

}

private bool UserNotExists()

{

if (Email != null)

{

Xceed.Wpf.Toolkit.MessageBox.Show("Пользователь с такой почтой уже существует");

return BaseProvider.DbContext.Users.Where(user => user.Email == Email).IsNullOrEmpty();

}

return false;

}

private void Registrastration(object sender, RoutedEventArgs e)

{

Password = ((WatermarkPasswordBox?)UIHelper.FindUid(this, "Password"))!.Password;

RepeatedPassword = ((WatermarkPasswordBox?)UIHelper.FindUid(this, "Password"))!.Password;

if (Validate() && UserNotExists())

{

SqlParameter email = new SqlParameter("email", Email);

SqlParameter name = new SqlParameter("name", UserName);

SqlParameter password = new SqlParameter("password", Password.Encrypt());

BaseProvider.CallStoredProcedureByName("AddUser", email, password, name);

var userDTO = BaseProvider.DbContext.UserViewDtos.ToList().Where(user => user.Email == Email).First();

userDTO.Photo = ImageManager.ImageSourceToBytes(new TiffBitmapEncoder(), UserIcon.Source);

NavigationService.Navigate(new StoreMain(userDTO));

}

}

private void LoadImage(object sender, RoutedEventArgs e)

{

UserIcon.Source = ImageManager.LoadImageFromFileDialog();

}

}

}

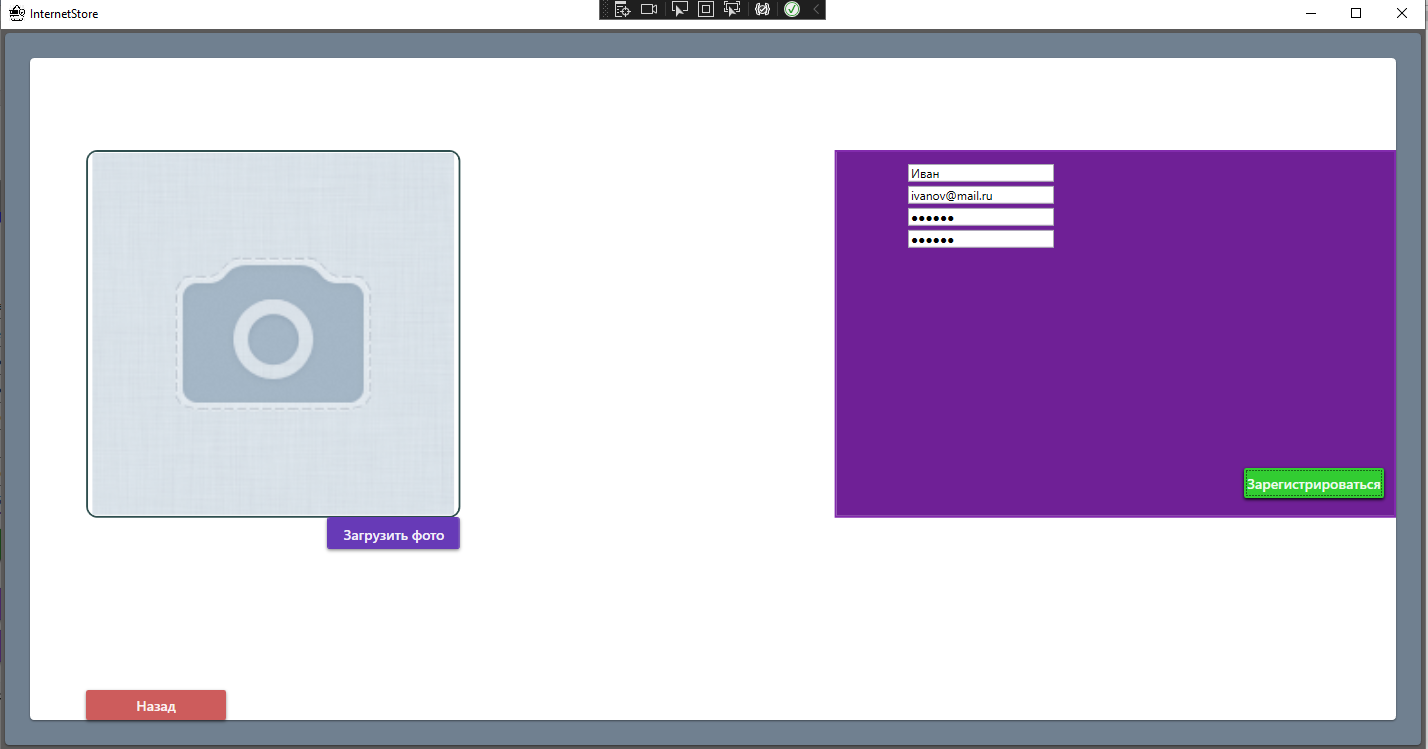


Рисунок 17 – скриншот «Регистрация пользователя»

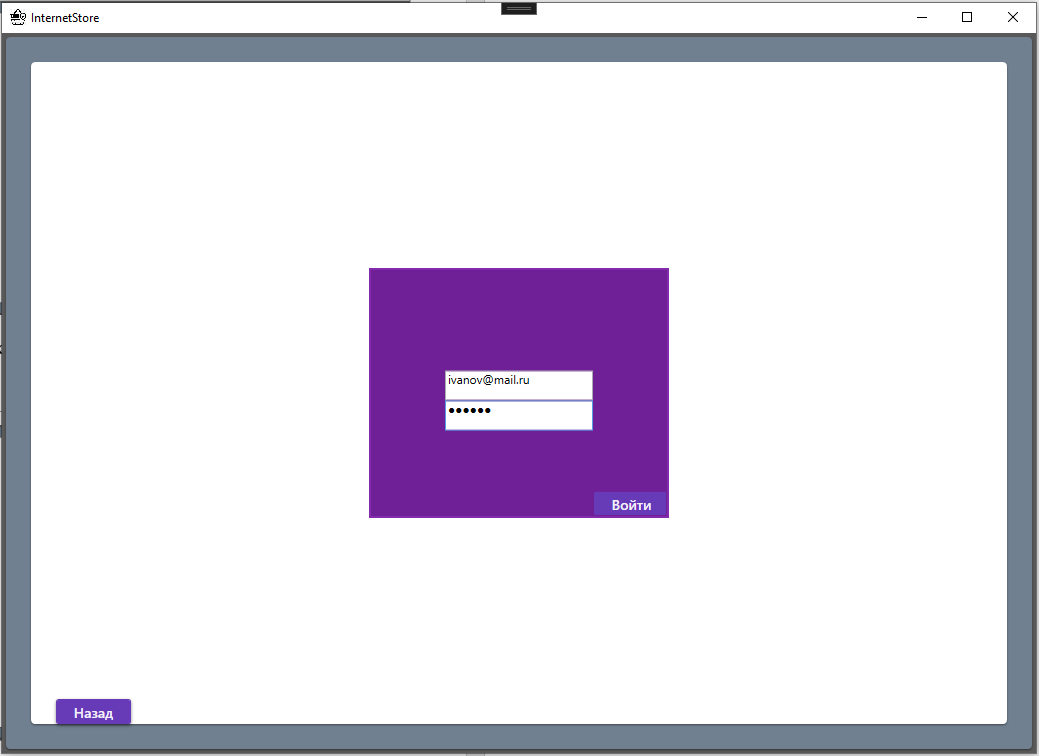


Рисунок 18 – скриншот «Авторизация пользователя»

В данном фрагменте кода создание списка товаров интернет-магазина (смотреть рисунок – 19).

namespace InternetStore.Controls.XAMLControls

{

public partial class ProductArea : UserControl

{

private LoadDecorator LoadDecorator = new LoadDecorator();

public bool AddingAccept = false;

public bool EditionAccept = false;

public RoutedEventHandler ItemClickHandler = null!;

public RoutedEventHandler ItemDoubleClickHandler = null!;

public List<AbsProductView> ItemList = new();

public ProductArea()

{

InitializeComponent();

}

/// <summary>

///

/// </summary>

/// <param name="x"></param>

///

public void Load()

{

LoadDecorator.Load<Grid>(grid, LoadProducts);

}

private void LoadProducts()

{

ItemList.Clear();

if (AddingAccept)

{

CreateAdditionalCard();

}

foreach (var product in BaseProvider.DbContext.Products.ToList())

{

ItemBuilder builder = new(product);

builder.isChangeable().isSortable();

if (EditionAccept)

{

builder.isEdittable();

}

if (builder.ItemCount == 0)

{

builder.SetImage(Path.Combine(Environment.GetEnvironmentVariable("Images")!, "outOfStock.png"));

builder.SetVisibility("HandledButton", Visibility.Collapsed);

}

if (ItemClickHandler != null) NotifyChangeClickHandler(ItemClickHandler);

if (ItemDoubleClickHandler != null) NotifyChangeDoubleClickHandler(ItemDoubleClickHandler);

ItemList.Add(builder.Build());

}

ListBox.ItemsSource = ItemList;

}

public void CreateAdditionalCard()

{

Product model = new();

model.ProductName = "Добавить товар";

ItemBuilder CardBuilder = new(model, false);

CardBuilder.SetImage(Path.Combine(Environment.GetEnvironmentVariable("Images")!, "additem.png"));

CardBuilder.SetFontSize("DescriptionText", 16).SetFontWidth("DescriptionText", FontWeights.Bold);

CardBuilder.SetVisibility("HandledButton", Visibility.Collapsed);

CardBuilder.SetVisibility("CostText", Visibility.Collapsed);

ItemList.Add(CardBuilder.Build());

}

private void SortParam(Func<AbsProductView, bool> x = null!)

{

if (x != null) ItemList = ItemList

.Where(x)

.Union(ItemList.Where(item => !item.isSortable))

.ToList();

ItemList.Sort((previousProduct, currentProduct) => -currentProduct.Cost.CompareTo(previousProduct.Cost));

ListBox.ItemsSource = ItemList;

}

public void Sort(string searchString)

{

LoadDecorator.Load<Grid>(grid, SearchByName, searchString);

}

public void Sort(int minCost, int maxCost)

{

LoadDecorator.Load<Grid>(grid, SortByCost, minCost, maxCost);

}

public void Sort(int SubCategoryID)

{

LoadDecorator.Load<Grid>(grid, SelectSubCategory, SubCategoryID);

}

private void SearchByName(string searchText)

{

Load();

if (!searchText.IsNullOrEmpty())

SortParam(

product => product.ItemName

.Split(" ")

.Select(x => x.Trim().ToLower())

.Contains(searchText.ToLower())

);

}

private void SortByCost(int minCost, int maxCost)

{

SortParam(product => Enumerable.Range(minCost, maxCost).Contains((int)(product.Cost)));

}

private void SelectSubCategory(int SubCategoryID)

{

Load();

SortParam(product => product.ProductModel.SubcategoryId == SubCategoryID);

}

public void NotifyChangeClickHandler(RoutedEventHandler handler)

{

ItemClickHandler = handler;

foreach (var item in ItemList)

{

item.UpdateClickHandler(handler);

}

}

public void NotifyChangeDoubleClickHandler(RoutedEventHandler handler)

{

ItemDoubleClickHandler = handler;

foreach (var item in ItemList)

{

item.UpdateDoubleClickHandler(handler);

}

}

}

}

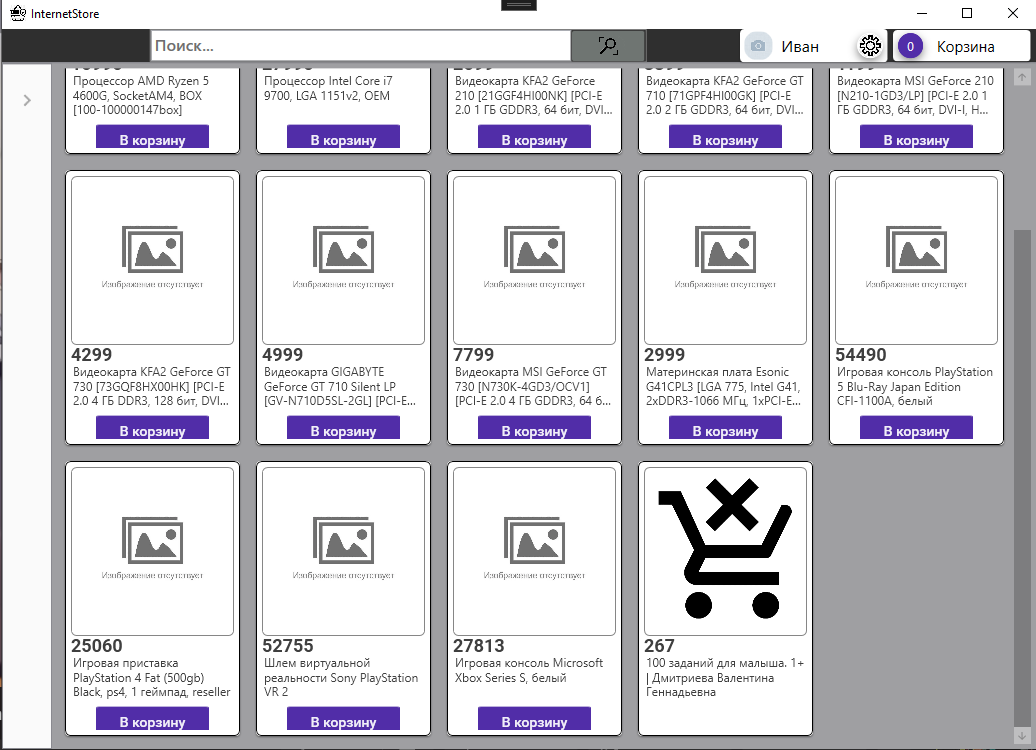


Рисунок 19 – скриншот «Список товаров»

В данном фрагменте кода создание главной страницы интернет-магазина (смотреть рисунок – 20).

namespace InternetStore.Pages

{

/// <summary>

/// Логика взаимодействия для StoreMain.xaml

/// </summary>

public partial class StoreMain : Page

{

private UserViewDto User = null!;

public static ProductBasket Basket = null!;

public ProductArea ProductList = null!;

private ToolBox ToolPanel = null!;

public UserViewDto CurrentUser => User;

public StoreMain(UserViewDto userModel)

{

User = userModel;

InitializeComponent();

InitToolPanel();

CreateBasket();

LoadBasketIcon();

LoadProfileIcon();

LoadProductArea();

ToolPanel.SearchBox.SetSearchHandler(SearchProduct);

//ProductList.SelectSubCategory(2);

//ProductList.SortByCost(0, 3400);

}

private void CreateBasket()

{

Basket = new ProductBasket(User.Id);

}

private void InitToolPanel()

{

ToolPanel = ToolBox.GetInstance();

Grid.SetRow(ToolPanel, 0);

Grid.SetColumn(ToolPanel, 0);

Grid.SetColumnSpan(ToolPanel, 4);

ToolPanel.Margin = new Thickness(0, 0, 0, 1);

ToolPanel.Background = (SolidColorBrush)new BrushConverter().ConvertFrom("#FF2F2F2F")!;

ToolPanel.BorderBrush = (SolidColorBrush)new BrushConverter().ConvertFrom("#FF1F1E1E")!;

grid.Children.Add(ToolPanel);

}

private void LoadProfileIcon()

{

//ProfileIcon ProfileIcon = (ProfileIcon)FindName("ProfileIcon");

ToolPanel.ProfileIcon.Click += ProfileNavigate;

if (CurrentUser.Photo != null)

ToolPanel.ProfileIcon.UserIcon.Source = ImageManager.LoadImage(CurrentUser.Photo);

else

ToolPanel.ProfileIcon.UserIcon.Source = ImageManager.LoadImage(Path.Combine(Environment.GetEnvironmentVariable("Images")!, "camera\_200.png"));

ToolPanel.ProfileIcon.UserName = BaseProvider.DbContext.UserPersonalInfs.ToList()

.Where(row => row.Id == User.Id).First().Name;

ToolPanel.BasketIcon.Count = Basket.ProductCount;

}

private void LoadProductArea()

{

ProductContainerBuilder ContainerBuilder = new ProductContainerBuilder();

ContainerBuilder.SetProductClickHandler(AddToBasket);

ContainerBuilder.SetProductDoubleClickHandler(ToProductPage);

switch (CurrentUser.RoleId)

{

case UserType.Rentail:

ContainerBuilder.RentalPermissions();

break;

case UserType.Adminstrator:

ContainerBuilder.AdministratorPermissions();

break;

}

ProductList = ContainerBuilder.Build();

Grid.SetColumn(ProductList, 1);

Grid.SetRow(ProductList, 1);

Grid.SetColumnSpan(ProductList, 3);

Grid.SetRowSpan(ProductList, 3);

grid.Children.Add(ProductList);

grid.UpdateLayout();

ProductList.Load();

}

private void LoadBasketIcon()

{

ToolPanel.BasketIcon.DoubleClick += ToBasket;

}

private void ProfileNavigate(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(Profile.getInstance(CurrentUser));

}

private void AddToBasket(object sender, RoutedEventArgs e)

{

BasketItem BasketEl = new BasketItem(((Item)sender).ProductModel);

BasketEl.Width = 525;

Basket.Add(BasketEl);

ToolPanel.BasketIcon.Count = Basket.ProductCount;

}

private void ToProductPage(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new ProductPage((AbsProductView)sender));

}

private void SearchProduct(object sender, RoutedEventArgs e)

{

ProductList.Sort(ToolPanel.SearchBox.InputText.Text);

}

private void ShowSubCategory(object sender, RoutedEventArgs e)

{

}

private void ToBasket(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(Basket);

}

}

}

# 

Рисунок 20 – скриншот «Главная страница»

В данном фрагменте кода отображен результат добавления товара в корзину (смотреть рисунок – 21).

namespace InternetStore.Pages

{

/// <summary>

/// Логика взаимодействия для ProductBasket.xaml

/// </summary>

public partial class ProductBasket : Page

{

private int UserId;

private List<IBasketViewItem> Products = new List<IBasketViewItem>();

public int ProductCount => Products.Count;

public ProductBasket(int UID)

{

UserId = UID;

InitializeComponent();

foreach (var basketProduct in BaseProvider.DbContext.Baskets

.ToList()

.Where(userBasket => userBasket.UserId == UserId))

{

// Добавить настройку вида продукта в билдере корзины

Product model = BaseProvider.DbContext.Products

.ToList()

.Where(product => product.Id == basketProduct.ProductId)

.First();

BasketItem basketItem = new(model);

basketItem.Width = 525;

//basketItem.DeleteBtn.Click += RemoveItem;

Products.Add(basketItem);

NotifyBasketChange();

}

}

private void NotifyBasketChange()

{

if (Products.Count > 0)

BasketList.ItemsSource = Products;

}

public bool inBasket(int ID)

{

return !Products.Where(product => product.ProductModel.Id == ID).IsNullOrEmpty();

}

private void CreateBasketEntitry()

{

}

public void Add(IBasketViewItem product)

{

SqlParameter uid = new SqlParameter("user\_id", UserId);

SqlParameter productId = new SqlParameter("product\_id", product.ProductModel.Id);

SqlParameter count = new SqlParameter("count", product.Count);

if (inBasket(product.ProductModel.Id))

{

var ProductInBasket = Products.Where(Bproduct => Bproduct.ProductModel.Id == product.ProductModel.Id).First();

ProductInBasket.Count++;

count.Value = ProductInBasket.Count;

BaseProvider.CallStoredProcedureByName("UpdateProductCountInBasket", uid, productId, count);

}

else

{

SqlParameter AddingDateTime = new SqlParameter("addDate", DateTime.Now);

BaseProvider.CallStoredProcedureByName("AddProductToBasket", uid, productId, count, AddingDateTime);

Products.Add(product);

}

//BaseControl.DbContext.Baskets.Add();

NotifyBasketChange();

}

public void Remove(IBasketViewItem product)

{

product.Count--;

SqlParameter uid = new SqlParameter("user\_id", UserId);

SqlParameter productId = new SqlParameter("product\_id", product.ProductModel.Id);

SqlParameter count = new SqlParameter("count", product.Count);

if (product.Count <= 0)

{

BaseProvider.DbContext.Baskets.Remove(BaseProvider.DbContext.Baskets.Single(basketProduct => (basketProduct.UserId == UserId)

&& (basketProduct.ProductId == product.ProductModel.Id)));

Products.Remove(product);

}

else

BaseProvider.CallStoredProcedureByName("UpdateProductCountInBasket", uid, productId, count);

NotifyBasketChange();

}

public void RemoveItem(object sender, RoutedEventArgs e)

{

}

private void ToMainPage(object sender, RoutedEventArgs e)

{

NavigationService.GoBack();

}

}

}

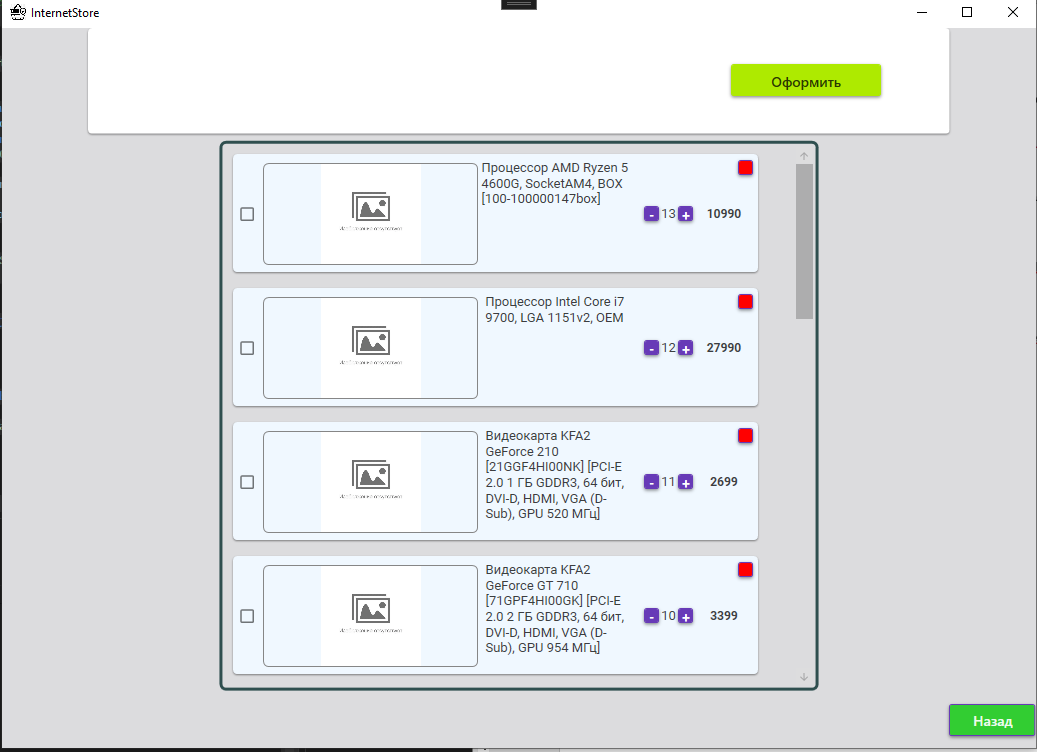


Рисунок 21 – скриншот «Список товаров в корзине»

В данном фрагменте кода отображен результат поиска товара (смотреть рисунок – 22-23).

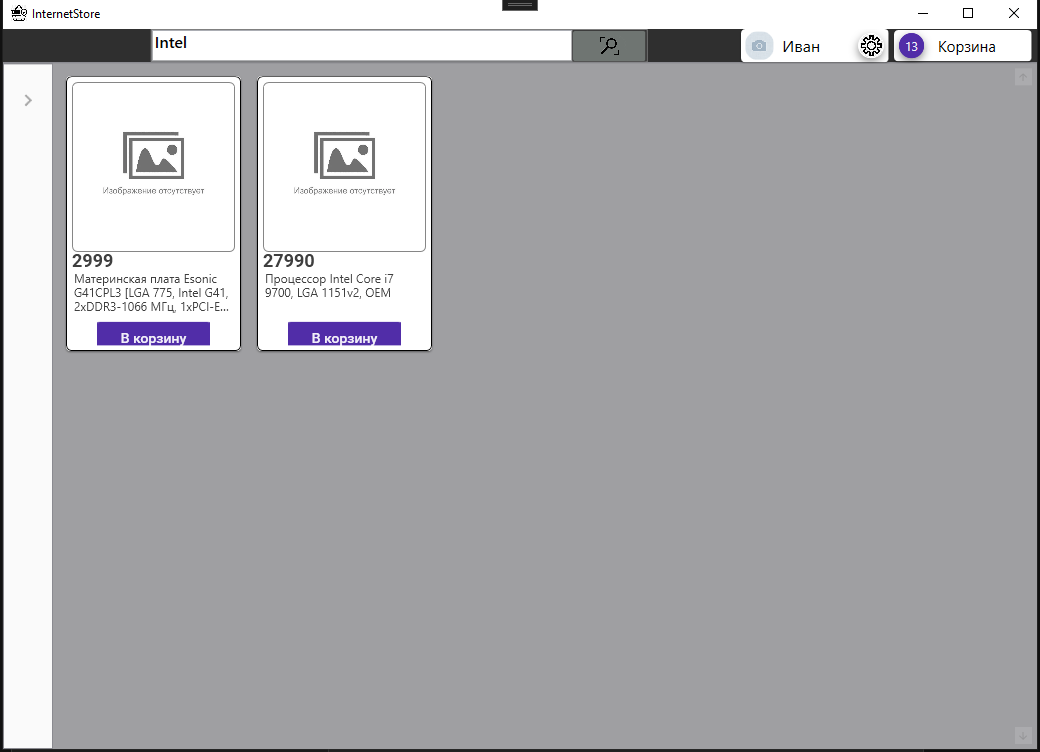


Рисунок 22 – скриншот «Поиск товара»

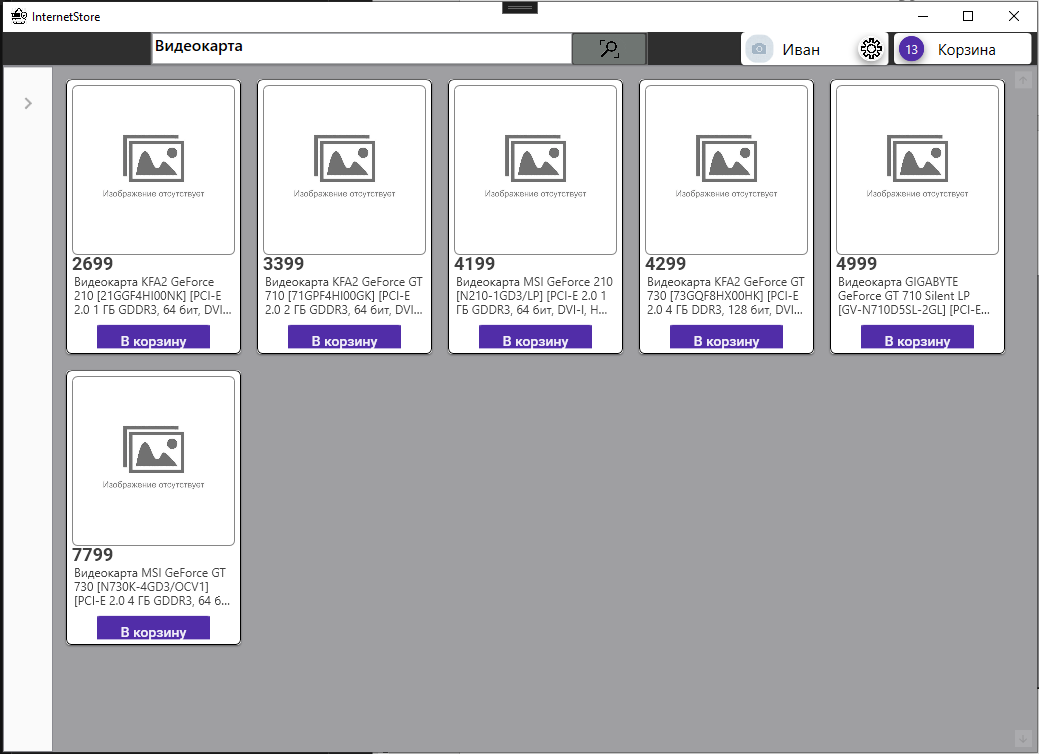


Рисунок 23 – скриншот «Поиск товара»

В данном фрагменте кода показана страница товара (смотреть рисунок – 24-25).

namespace InternetStore.Pages

{

public partial class ProductPage : Page

{

private AbsProductView Product = null!;

public ProductPage(AbsProductView product)

{

InitializeComponent();

Product = product;

if (Product.Image != null)

ProductImage.Source = ImageManager.LoadImage(product.Image);

else

ProductImage.Source = ImageManager.LoadImage(Path.Combine(Environment.GetEnvironmentVariable("Images")!, "emptyProduct.png"));

ProductName.Text = product.ProductModel.ProductName;

CategoryName.Text = BaseProvider.DbContext.SubCategories.Single(category => category.Id == product.ProductModel.SubcategoryId).Name;

ProductCount.Text = product.Count.ToString();

GenerateProperties();

}

private void GenerateProperties()

{

StringBuilder stringBuilder = new StringBuilder();

var SubCategory = BaseProvider.DbContext.SubCategories.Single(subcat => subcat.Id == Product.ProductModel.SubcategoryId);

Dictionary<string, object> SubCategoryAttributes = SubCategory.Attributes.Parse<string, object>();

if (SubCategoryAttributes != null && Product.Properties != null)

{

var prop = Product.Properties.ToList().Where(dict => SubCategoryAttributes.ContainsKey(dict.Key));

PropertyList.ItemsSource = prop

.Select(dict => CreateProperty(dict.Key, dict.Value));

}

}

private TextBlock CreateProperty(string key, object value)

{

TextBlock Property = new TextBlock();

Property.Inlines.Add($"{key}: ");

Property.Inlines.Add($"{value}");

Property.Inlines.FirstInline.FontWeight = FontWeights.Bold;

return Property;

}

private void ToMainPage(object sender, System.Windows.RoutedEventArgs e)

{

NavigationService.GoBack();

}

}

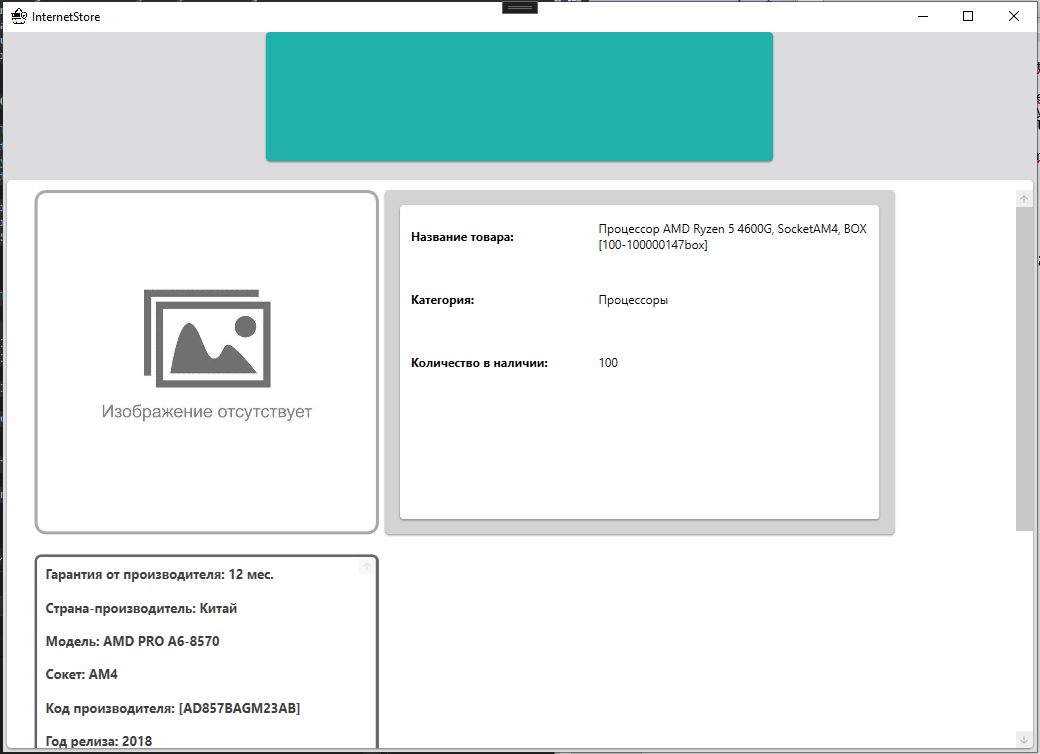


Рисунок 24 – скриншот «Страница товара»

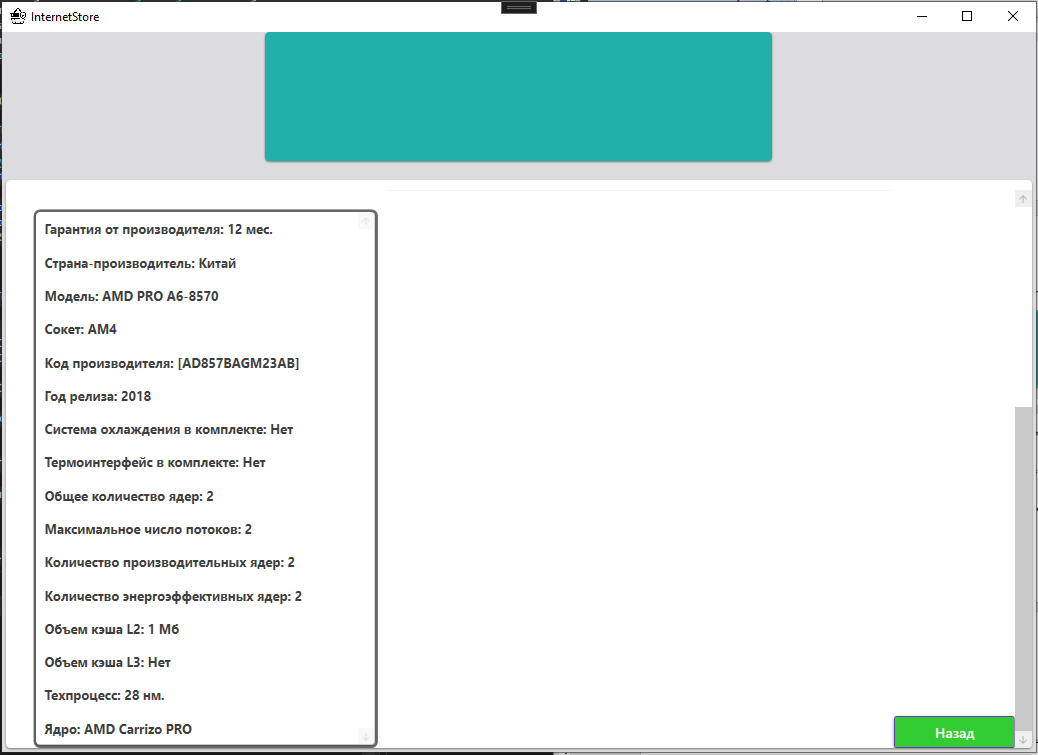


Рисунок 25 – скриншот «Страница товара»