МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение 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 Программное средство «Интернет-магазин часов “Золотое время”»

Исполнитель

студент (ка) 2 курса группы 6 Панчук Илья Эдуардович

(Ф.И.О.)

Руководитель работы Ассистент Чистякова Ю.А.

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

Курсовой проект защищен с оценкой

Председатель Пацей Н.В.

(подпись)

Минск 2023

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc135211798)

[1. Аналитический обзор прототипов и литературных источников 4](#_Toc135211799)

[1.1 Анализ прототипов 4](#_Toc135211800)

[1.1.1 Интернет-ресурс «WatchWorld.com» 4](#_Toc135211801)

[1.1.2 Интернет-ресурс «TimeWearshop.by» 5](#_Toc135211802)

[1.1.3 Интернет-ресурс «TimeCity.by» 6](#_Toc135211803)

[1.2 Постановка задачи 7](#_Toc135211804)

[2. Анализ требований к программному средству и разработка функциональных требований 8](#_Toc135211805)

[2.1 Описание средств разработки 8](#_Toc135211806)

[2.2 Определение требований к программному средству 10](#_Toc135211807)

[3. Проектирование программного средства 11](#_Toc135211808)

[3.1. Архитектура системы 11](#_Toc135211809)

[3.2. Взаимоотношения между классами 12](#_Toc135211810)

[3.3. Модель базы данных 12](#_Toc135211811)

[3.4. Проектирование последовательности взаимодействия 15](#_Toc135211812)

[4. Реализация программного средства 16](#_Toc135211813)

[4.1 Реализация шаблона MVVM и другие паттерны 16](#_Toc135211814)

[4.2 Реализация хеширования паролей 20](#_Toc135211815)

[5. Тестирование, проверка работоспособности и анализ полученных результатов 21](#_Toc135211816)

[5.1. Тестирование авторизации и регистрации 21](#_Toc135211817)

[5.3. Тестирование панели администратора 22](#_Toc135211818)

[6. Руководство по установке и использованию 23](#_Toc135211819)

[Заключение 30](#_Toc135211820)

[Список литературы 31](#_Toc135211821)

[Приложение А 32](#_Toc135211822)

[Приложение Б 33](#_Toc135211823)

[Приложение В 34](#_Toc135211824)

[Приложение Г 35](#_Toc135211825)

ВВЕДЕНИЕ

В современном обществе онлайн-шопинг стал неотъемлемой частью нашей жизни, и электронная коммерция продолжает активно развиваться. Одной из важных отраслей в этой сфере является торговля часами, которая пользуется постоянным спросом со стороны потребителей.

Цель данного курсового проекта - разработать и создать приложение интернет-магазина часов, которое предоставит пользователям удобный и надежный способ приобретения часов с помощью современных технологий. Проект нацелен на улучшение опыта покупателей, предлагая им широкий выбор часов различных марок, стилей и ценовых категорий.

Основные задачи, которые должен выполнять создаваемый проект включают в себя:

* удобное оформление заказов клиентов;
* регистрация новых клиентов и хранение информации о них;
* сделать удобным выбор наиболее подходящей модели часов.

Для реализации приложения будет использоваться язык программирования C# и технология WPF (Windows Presentation Foundation. Архитектура приложения будет построена на основе паттерна Model-View-ViewModel (MVVM). В качестве СУБД для хранения данных будет использоваться Microsoft SQL Server.

1. Аналитический обзор прототипов и литературных источников

* 1. Анализ прототипов

Курсовой проект представляет из себя разработку приложения для интернет-магазина часов. В сфере продажи часов существует множество аналогичных решений, которые предоставляют покупателям возможность выбирать и заказывать качественные и стильные часы прямо из комфорта своего дома. В данном разделе рассмотрим несколько интернет-ресурсов, а также настольных и веб-приложений, которые успешно функционируют и позволяют любителям часов наслаждаться широким выбором моделей и удобным способом покупки.

* + 1. Интернет-ресурс «WatchWorld.com»

WatchWorld.com предлагает огромный выбор часов различных брендов, включая элитные и спортивные модели. "WatchWorld" известен своими конкурентоспособными ценами и регулярными акциями, которые делают покупки еще более привлекательными для клиентов. Ресурс предлагает удобный интерфейс с возможностью фильтрации по бренду, функциональности, материалам и ценовому диапазону. Они также предоставляют подробную информацию о доставке и возврате товара.. Пример интерфейса приложения представлен на рисунке 1.1.

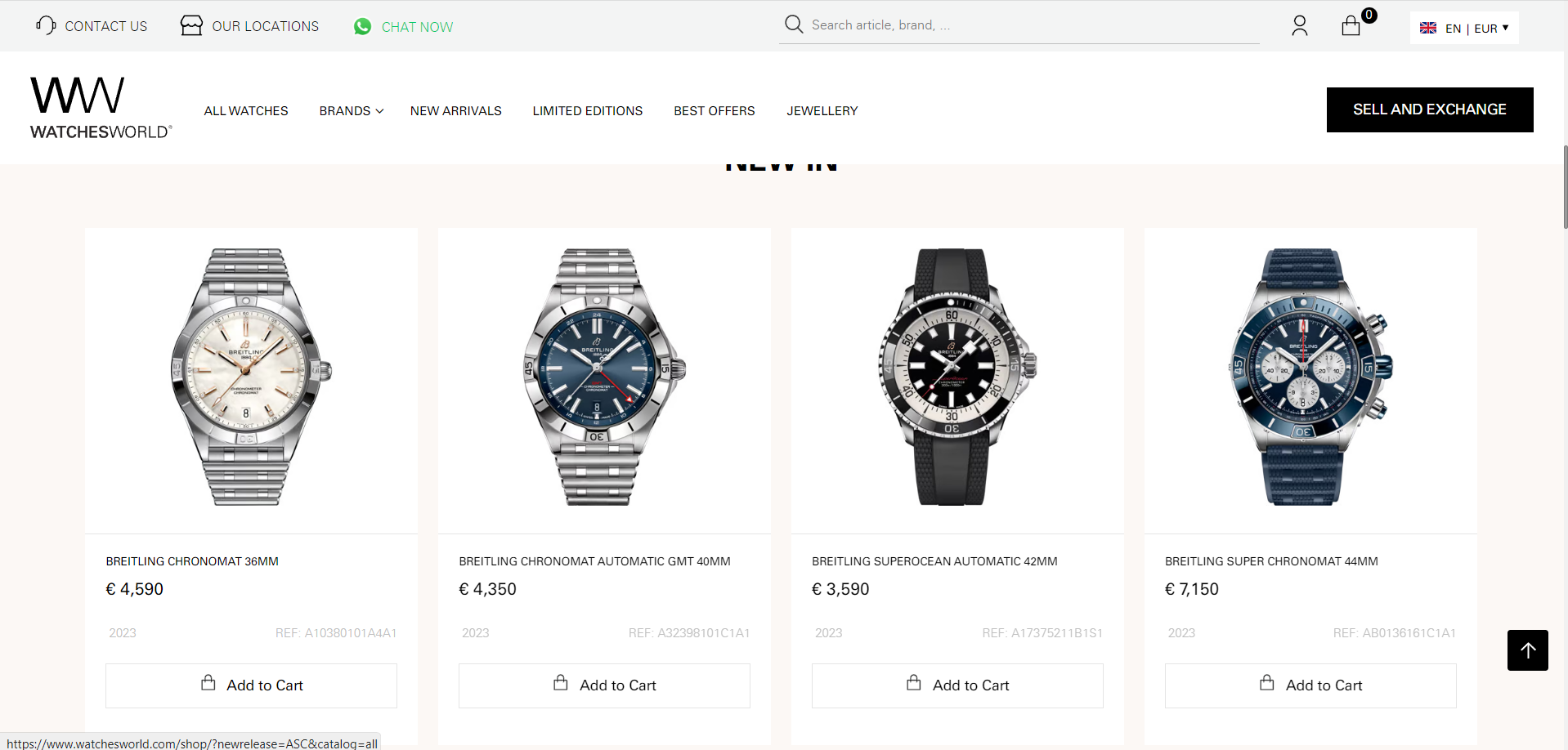


Рисунок 1.1 – Интерфейс «WatchWorld.com»

Преимуществами ресурса являются:

* приятный и стильный дизайн;
* просмотр информации о проданных товарах.

Недостатком данного интернет-ресурса является то, что в приложении не предусмотрена возможность настройки интерфейса в соответствии с индивидуальными предпочтениями пользователей.

* + 1. Интернет-ресурс «TimeWearshop.by»

TimeWearshop.by - это интернет-магазин часов, где можно найти широкий выбор моделей часов различных брендов и стилей. На сайте представлены как мужские, так и женские часы, а также аксессуары для часов. Клиенты могут использовать различные фильтры для поиска нужной модели, например, по бренду, стилю, цвету или цене. Интерфейс интернет-ресурса «Pandora.net» представлен на рисунке 1.2.

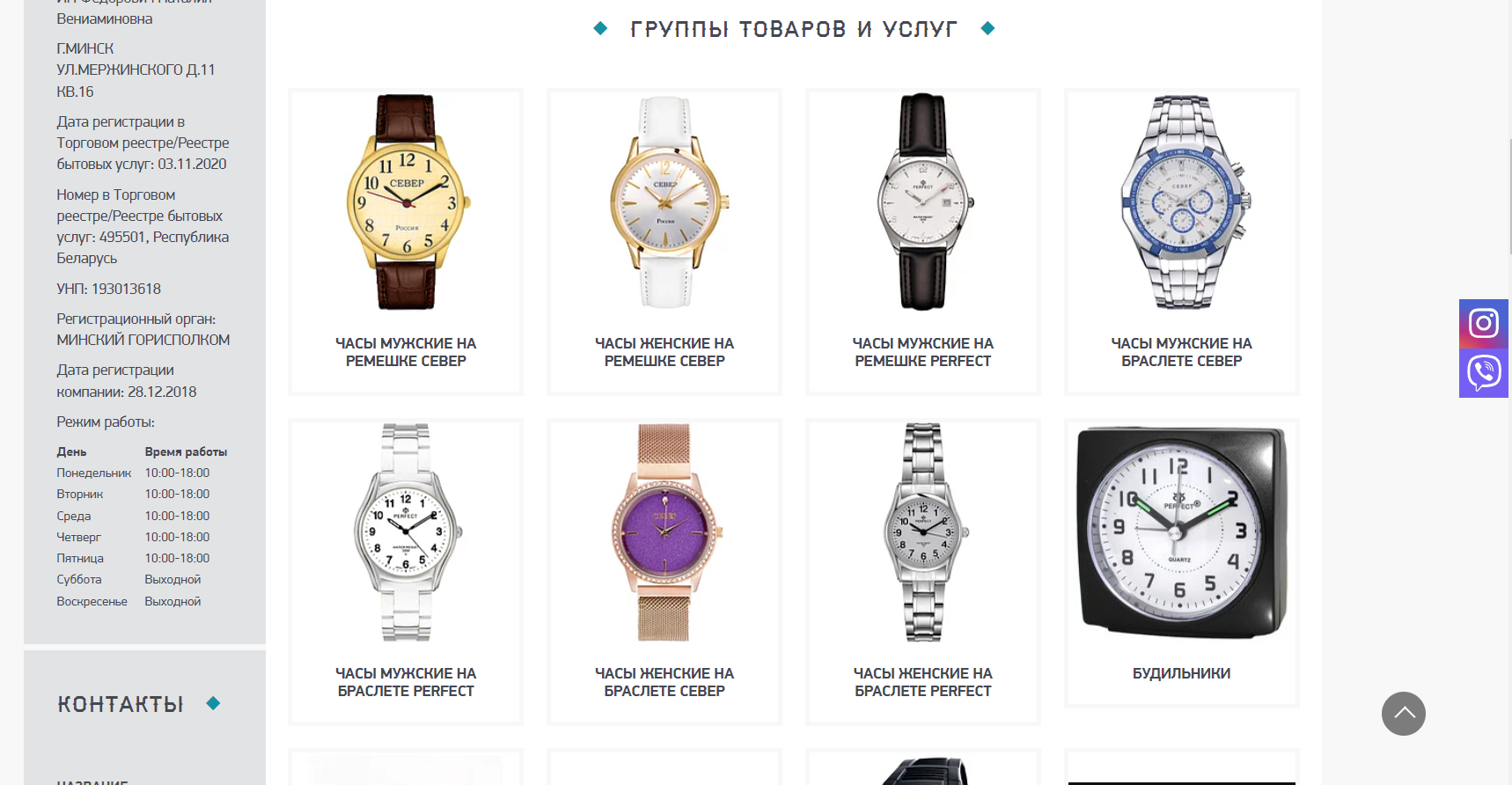


Рисунок 1.2 – Интерфейс «TimeWearshop.by»

Преимуществами ресурса являются:

* + фильтрация и сортировка: Приложение TimeWearshop предоставляет функционал фильтрации и сортировки часов по различным параметрам, таким как бренд, материал корпуса, тип механизма и т.д. Это помогает пользователям быстро находить и выбирать часы в соответствии с их предпочтениями.

Недостатки:

* ограниченная доступность платформ: Приложение TimeWearshop может быть доступно только для определенных операционных систем или устройств, что ограничивает доступность и удобство использования для некоторых пользователей. некоторые страницы загружаются довольно медленно.
* медленная загрузка и ответ: Приложение TimeWearshop может быть подвержено проблемам с загрузкой и медленным откликом, особенно при большом объеме данных или при использовании в условиях низкой скорости интернета. Это может вызывать разочарование у пользователей и снижать общую эффективность использования приложения.
  + 1. Интернет-ресурс «TimeCity.by»

TimeCity.by – это привлекательный интернет-магазин, специализирующийся на продаже часов. На их веб-сайте вы сможете найти широкий выбор моделей различных брендов и стилей, чтобы удовлетворить свои потребности и предпочтения в области часового дизайна. При посещении сайта TimeCity.by вы сразу же заметите его удобный и интуитивно понятный интерфейс, который облегчает процесс поиска и покупки часов. Навигация по каталогу часов структурирована и позволяет легко фильтровать часы по бренду, стилю, функциональности и другим параметрам, чтобы быстро найти нужную модель.

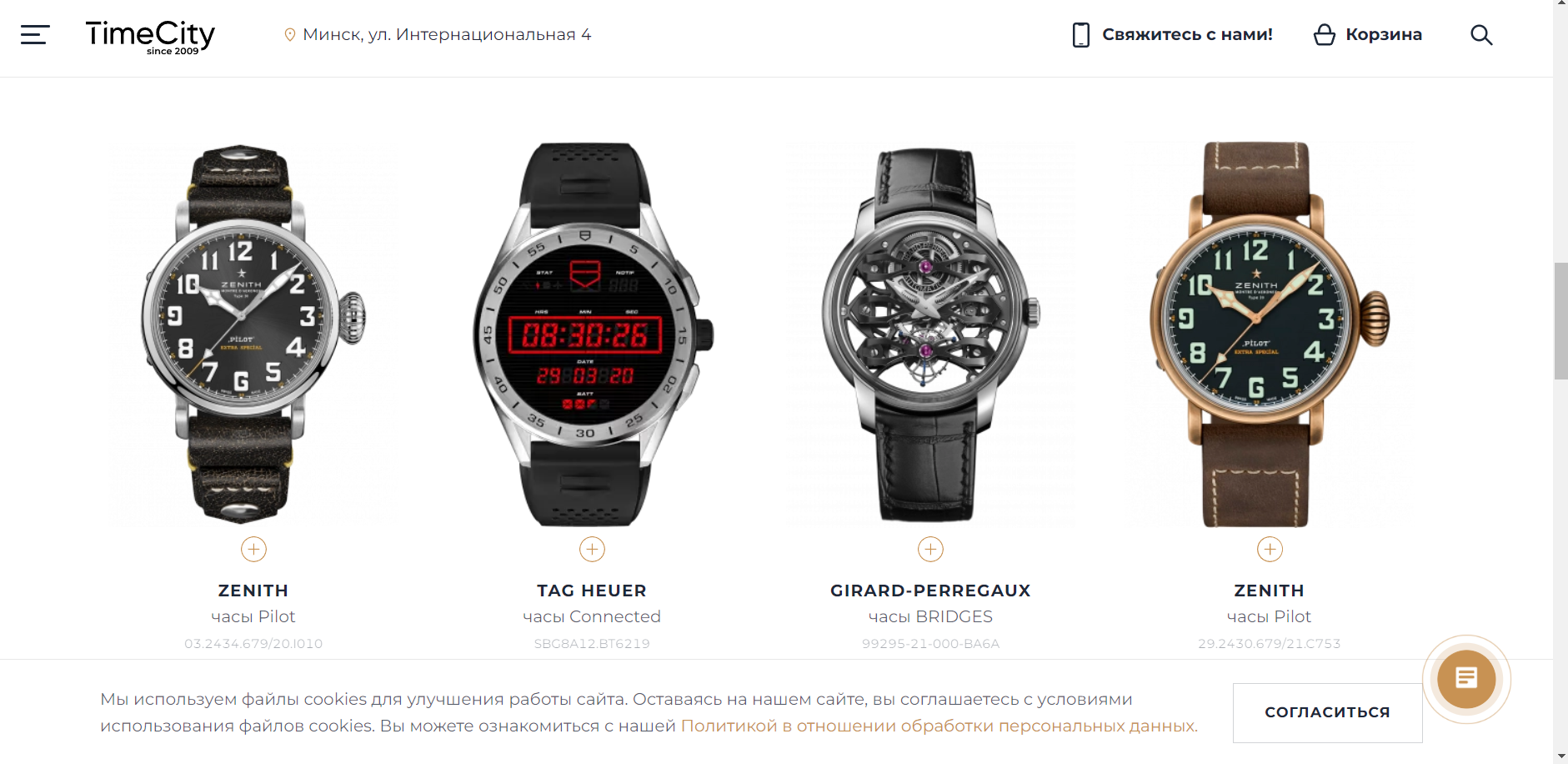


Рисунок 1.3 – Интерфейс интернет-ресурса «TimeCity.by»

Достоинства:

* удобный интерфейс: Веб-сайт TimeCity.by имеет удобный и интуитивно понятный интерфейс, что облегчает навигацию и поиск нужной модели. Клиенты могут легко фильтровать часы по различным параметрам, чтобы быстро найти подходящую модель. ассортимент продукции - сайт Asprey.com предлагает большой выбор продукции, включая ювелирные изделия, часы, драгоценные украшения, кожгалантерею и многое другое.

Недостатки:

* нестабильная работа приложения: Возможны проблемы с неполадками и сбоями при использовании приложения TimeCity.by, что может вызывать неудобство у пользователей и приводить к снижению эффективности покупок;
* ограниченные возможности обратной связи: Приложение TimeCity.by может не предоставлять достаточные возможности для обратной связи с клиентами, такие как возможность задать вопросы или получить дополнительную информацию о товарах. Это может ограничивать возможности клиентов получить необходимую помощь и консультацию.

1.2 Постановка задачи

Проанализировав прототип, были выделены основная задача и функциональные требования проекта.

Целью курсового проекта стало написание десктопного интернет-магазина продажи часов, в котором можно сравнивать различные модели, а также предусмотреть возможность настройки интерфейса в соответствии с индивидуальными предпочтениями пользователей.

Функциональные требования описаны в главе 2.

2. Анализ требований к программному средству и разработка функциональных требований

Для начала этапа разработки необходимо сформулировать функциональные требования к программному средству.

2.1 Описание средств разработки

Задачей курсового проекта является разработка приложения, предназначенного для коммуникации пользователей, а именно отправки текстовых сообщений или изображений. Пользователи могут отправлять жалобы на сообщения, затем администратор имеет возможность скрыть их.

Программный продукт реализован на объектно-ориентированном языке C# с использованием технологий WPF.

Windows Presentation Foundation (WPF) — система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление.

Графической технологией, лежащей в основе WPF, является DirectX, в отличие от WindowsForms, где используется GDI/GDI+. Производительность WPF выше, чем у GDI+ за счёт использования аппаратного ускорения графики через DirectX.

База данных была разработана на SQL Server в приложении DataGrip.

SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире.

SQL Server характеризуется такими особенностями как:

* Производительность. SQL Server работает очень быстро.
* Надежность и безопасность. SQL Server предоставляет шифрование данных.
* Простота. С данной СУБД относительно легко работать и вести администрирование.

Для организации баз данных MS SQL Server использует реляционную модель, которая предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта.

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language).

Клиент (например, внешняя программа) отправляет запрос на языке SQL должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.

Основной используемый язык запросов – Transact-SQL – реализован на структурированном языке запросов (SQL) с расширениями.

Для осуществления связи между базой данных и приложением на C# необходим посредник, которым будет являться технология Entity Framework. Она основана на платформе .NET Framework и предназначена 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. Эта технология предоставляет нам набор классов, через которые мы можем отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций.

Entity Framework предполагает три возможных способа взаимодействия с базой данных:

* database first: Entity Framework создает набор классов, которые отражают модель конкретной базы данных;
* model first: сначала разработчик создает модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере;
* code first: разработчик создает класс модели данных, которые будут храниться в базе данных, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы.

В курсовом проектировании для разработки был выбран первый подход.

Приложение необходимо реализовать с использованием шаблона проектирования MVVM и с организацией взаимодействия с базой данных. В качестве клиентской части необходимо реализовать оконное приложение с использованием стандартных и иных библиотек. Приложение должно иметь возможность запускаться без использования интегрированных средств разработки.

2.2 Определение требований к программному средству

Для разработанного продукта необходимо предусмотреть следующие задачи:

* удобная система навигации;
* интуитивно понятный интерфейс;
* высокая скорость работы;

По итогу работы над продуктом, его необходимо протестировать и отладить, проанализировать его возможности и, по необходимости, внести дополнения или улучшения проекта, с возможностью дальнейшего расширения проекта.

Программное средство предоставляет пользователю следующие функциональные возможности:

* регистрация пользователей;
* авторизация пользователей;
* добавлять в корзину;
* добавлять в сравнение;
* поиск и фильтрация часов;
* изменение настроек приложения;

Администратор имеет следующие возможности:

* просмотр имени и почты пользователей;
* добавление, редактирование часов и новостей;

Описание функциональности программного средства представлено на UML-схеме, находящийся в графической части.

3. Проектирование программного средства

3.1. Архитектура системы

В данном приложении используется архитектурный паттерн MVVM (Model-View-ViewModel). Он позволяет отделить логику приложения от визуальной части. MVVM состоит из трех частей:

* Модель описывает используемые в приложении данные.
* Представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением.
* Модель представления связывает модель и представление через механизм привязки данных. ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление.

Программное средство «интернет-магазин часов “Золотое время”» имеет следующую структуру, представленную на рисунке 3.1.

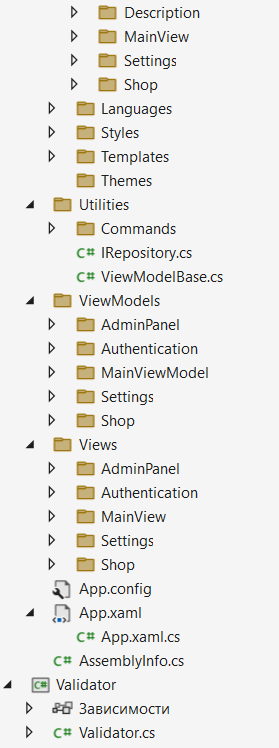
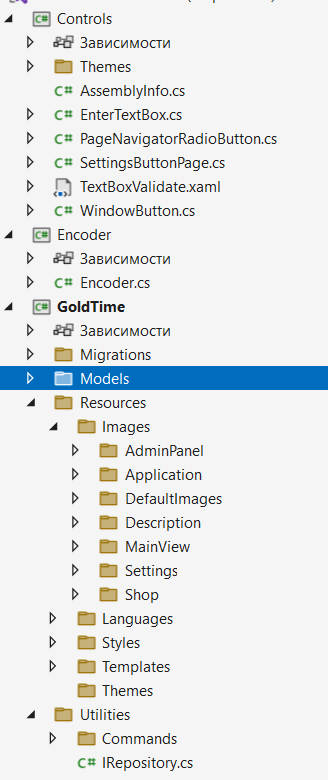


Рисунок 3.1 – Структура проекта

Описание структуры основных папок проекта представлено в таблице 3.1.

Таблица 3.1 – Описание структуры папок и файлов проекта

|  |  |
| --- | --- |
| Имя файла/папки | Содержание |
| Validator | Классы, содержащие функции проверки валидности полей элементов |
| Models | Классы сущностей базы данных, а так же вспомогательные классы |
| Resources | Стили, картинки, языковые ресурсы приложения. |
| Controls | Реализация пользовательских элементов управления. |
| ViewModels | Классы, реализующие слой ViewModel паттерна MVVM. |
| Views | Классы-представления слоя View паттерна MVVM. |
| Encoder | Классы, содержащие функции шифрования и хеширования |

Таким образом, сформированная таблица помогает понять общую структуру проектируемого программного средства.

3.2. Взаимоотношения между классами

При проектировании базы данных курсового проекта использовался Entity Framework. Он 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 позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища.

Отличительной чертой Entity Framework является использование запросов LINQ для выборки данных из БД. С помощью LINQ мы можем получать объекты, связанные различными ассоциативными связями. Диаграмма классов представлена в Приложении Б.

3.3. Модель базы данных

Всего в базе данных 9 таблиц, описание которых приведено ниже. Описания таблиц базы данных представлены в таблицах 3.2 – 3.10:

Таблица 3.2 – описание таблицы Watches

|  |  |
| --- | --- |
| Столбец | Описание |
| Id | Идентификатор |
| NameModel | Название модели |
| NameCompany | Производитель |
| DescriptionModel | Описание часов |
| Price | Цена часов |
| ImagePath | Путь к картинке |

Таблица 3.3 – описание таблицы EnrollmentComparison

|  |  |
| --- | --- |
| Столбец | Описание |
| ComparisonWatchesId | Идентификатор списка сравнения пользователя |
| WatchesId | Идентификатор часов, добавленных в сравнение пользователем |

Таблица 3.4 – описание таблицы EnrollmentBasket

|  |  |
| --- | --- |
| Столбец | Описание |
| BasketWatchesId | Идентификатор корзины сравнения пользователя |
| WatchesId | Идентификатор часов, добавленных в корзину пользователем |

Таблица 3.5 – описание таблицы Users

|  |  |
| --- | --- |
| Столбец | Описание |
| Id | Идентификатор пользователя |
| Name | Имя пользователя |
| Email | Почта пользователя |
| Password | Пароль пользователя |
| Status | Статус пользователя |
| WatchesBasketId | Идентификатор корзины часов |
| ComparisonWatchesId | Идентификатор списка сравнения часов |
| OrdersUserId | Идентификатор списка заказов часов |

Таблица 3.6 – описание таблицы EnrollmentOrders

|  |  |
| --- | --- |
| Столбец | Описание |
| OrdersId | Идентификатор заказа |
| OrdersUserId | Идентификатор списка заказов |

Таблица 3.7 – описание таблицы OrdersUser

|  |  |
| --- | --- |
| Столбец | Описание |
| Id | Идентификатор списка заказов |

Таблица 3.8 – описание таблицы ComparisonWatches

|  |  |
| --- | --- |
| Столбец | Описание |
| Id | Идентификатор списка сравнения |

Таблица 3.9 – описание таблицы BasketWatches

|  |  |
| --- | --- |
| Столбец | Описание |
| Id | Идентификатор списка корзины |

Таблица 3.10 – описание таблицы Orders

|  |  |
| --- | --- |
| Столбец | Описание |
| Id | Идентификатор |
| DateOrder | Дата заказа |
| QuentityWatchOrder | Количество купленных часов |
| PriceOrder | Цена заказа |

3.4. Проектирование последовательности взаимодействия

При проектировании последовательности взаимодействия объектов приложения была использована UML-диаграмма последовательности.

Диаграмма последовательности (Sequence Diagram) – это один из видов UML-диаграмм, который используется для описания последовательности взаимодействия объектов в системе.

Диаграмма последовательности взаимодействия покупателя и администратора с приложением представлена на рисунке 3.2.

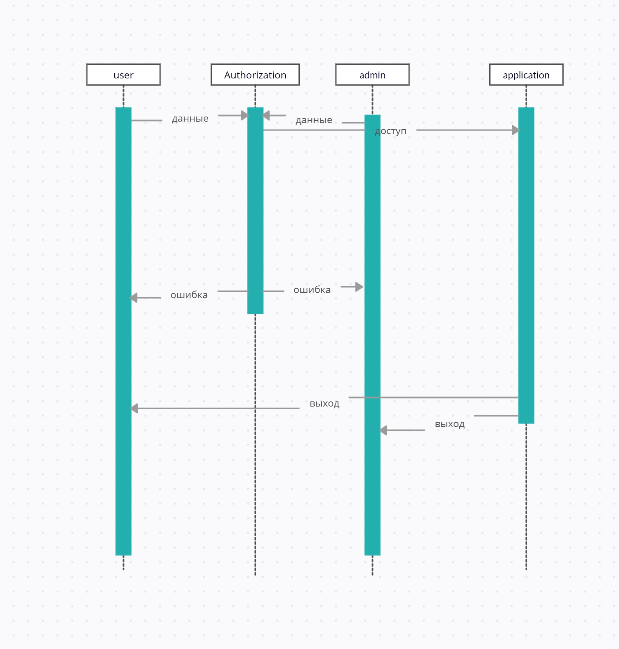


Рисунок 3.2 – Диаграмма последовательности

На диаграмме последовательности объекты представлены в виде вертикальных линий, а сообщения между объектами - в виде стрелок, направленных от одного объекта к другому. Каждое сообщение имеет свойство времени жизни, которое указывает, сколько времени занимает передача сообщения между объектами. Диаграмма последовательности позволяет описать не только последовательность сообщений между объектами, но и условия ветвления и циклов в процессе взаимодействия.

4. Реализация программного средства

4.1 Реализация шаблона MVVM и другие паттерны

При создании приложения использовался паттерн проектирования MVVM. Он заключается в разделении представления от бизнес-логики. Это достигается за счёт ввода новой логической конструкции ViewModel. Она связывает представление и бизнес-логику приложения.

Все бизнес-классы в данном курсовом проекте наследовались от класса PropertyChangedNotification, который в свою очередь является реализацией интерфейсов INotifyPropertyChanged. Данный класс предоставляет методы GetValue и SetValue, которые могут непосредственно вызываться в методах get и set свойств, принадлежащих ViewModel. Они предоставляют модель оповещения об изменении свойства, а также позволяют ViewModel обойтись без огромного количества приватных полей.

Реализация класса PropertyChangedNotification представлена на рисунке 4.1.

|  |
| --- |
| public class PropertyChangedNotification : INotifyPropertyChanged  {  private readonly Dictionary<string, object> \_values = new Dictionary<string, object>();  protected void SetValue<T>(Expression<Func<T>> propertySelector, T value)  {  string propertyName = GetPropertyName(propertySelector);  SetValue<T>(propertyName, value);  }  protected void SetValue<T>(string propertyName, T value)  {  if (string.IsNullOrEmpty(propertyName))  {  throw new ArgumentException("Invalid property name", propertyName);  }  \_values[propertyName] = value;  NotifyPropertyChanged(propertyName);  }  protected T GetValue<T>(Expression<Func<T>> propertySelector)  {  string propertyName = GetPropertyName(propertySelector);  return GetValue<T>(propertyName);  }  protected T GetValue<T>(string propertyName)  {  if (string.IsNullOrEmpty(propertyName))  {  throw new ArgumentException("Invalid property name", propertyName);  }  object value;  if (!\_values.TryGetValue(propertyName, out value))  {  value = default(T);  \_values.Add(propertyName, value);  }  return (T)value;  }  public event PropertyChangedEventHandler PropertyChanged;  protected void NotifyPropertyChanged(string propertyName)  {  this.VerifyPropertyName(propertyName);  PropertyChangedEventHandler handler = this.PropertyChanged;  if (handler != null)  {  var e = new PropertyChangedEventArgs(propertyName);  handler(this, e);  }  }  private string GetPropertyName(LambdaExpression expression)  {  var memberExpression = expression.Body as MemberExpression;  if (memberExpression == null)  {  throw new InvalidOperationException();  }  return memberExpression.Member.Name;  }  } |

Рисунок 4.1 – Реализация класса PropertyChangedNotification

В приложении используется паттерн Command который позволяет инкапсулировать запрос на выполнение определенного действия в виде отдельного объекта. В WPF команды представлены интерфейсом ICommand. Также есть классы которые позволяет делегировать командную логику методам, передаваемым в качестве параметров, пример представлен на рисунке 4.2

|  |
| --- |
| public class DelegateCommand : ICommand  {  #region Constructors  /// <summary>  /// Constructor  /// </summary>  public DelegateCommand(Action executeMethod)  : this(executeMethod, null, false)  {  }  /// <summary>  /// Constructor  /// </summary>  public DelegateCommand(Action executeMethod, Func<bool> canExecuteMethod)  : this(executeMethod, canExecuteMethod, false)  {  }  /// <summary>  /// Constructor  /// </summary>  public DelegateCommand(Action executeMethod, Func<bool> canExecuteMethod, bool isAutomaticRequeryDisabled)  {  if (executeMethod == null)  {  throw new ArgumentNullException("executeMethod");  }  \_executeMethod = executeMethod;  \_canExecuteMethod = canExecuteMethod;  \_isAutomaticRequeryDisabled = isAutomaticRequeryDisabled;  }  #endregion  #region Public Methods  /// <summary>  /// Method to determine if the command can be executed  /// </summary>  public bool CanExecute()  {  if (\_canExecuteMethod != null)  {  return \_canExecuteMethod();  }  return true;  }  /// <summary>  /// Execution of the command  /// </summary>  public void Execute()  {  if (\_executeMethod != null)  {  \_executeMethod();  }  }  void ICommand.Execute(object parameter)  {  Execute();  }  #endregion  #region Data  private readonly Action \_executeMethod = null;  private readonly Func<bool> \_canExecuteMethod = null;  private bool \_isAutomaticRequeryDisabled = false;  private List<WeakReference> \_canExecuteChangedHandlers;  #endregion  } |

Рисунок 4.2 – пример класса DelegateCommand

В программном средстве «Золотое время» также реализованы шаблоны Repository и Unit Of Work.

Для реализации паттерна Repository был написан интерфейс IRepository, структура которого представлена на рисунке 4.3. В этом интерфейсе описаны методы для взаимодействия с контекстом базы данных.

|  |
| --- |
| interface IRepository<T> where T : class  {  List<T> GetAll();  T Get(int id);  void Add(T item);  void Update(T item);  void Delete(int id);  } |

Рисунок 4.3 – Структура интерфейса IRepository

На примере с сущностью Users продемонстрирована реализация интерфейса IRepository (рисунок 4.4).

|  |
| --- |
| public class UsersRepository : IRepository<User>  {  private ApplicationContext db;  public UsersRepository(ApplicationContext context)  {  db = context;  }  public void Create(User item)  {  db.Users.Add(item);  db.SaveChanges();  }  public void Delete(int id)  {  User student = db.Users.Find(id);  if (student != null)  db.Users.Remove(student);  db.SaveChanges();  }  public User Get(int id)  {  List<User> users = db.Users.Include(x => x.WatchesBasket.Watches).Include(x => x.OrdersUser.Orders).Include(x => x.ComparisonWatches.Watches).ToList();  foreach (User user in users)  {  if(user.Id == id)  return user;  }  return new User();  }  public IEnumerable<User> GetAll()  {  return db.Users;  }  public void Update(User item)  {  db.Entry(item).State = EntityState.Modified;  db.SaveChanges();  }  } |

Рисунок 4.4 – Имплементация интерфейса IRepository в классе UserRepository

Далее все репозитории объединяются в общий класс, реализующий шаблон Unit Of Work, структура которого представлена в приложении Г. Unit Of Work предоставляет доступ к репозиториям через свойства, а также некоторые методы, такие как Save для сохранения изменений в базе данных.

4.2 Реализация хеширования паролей

Хранение паролей в базе данных является небезопасным, поскольку злоумышленники могут получить несанкционированный доступ у учетным записям пользователей. Поэтому было принято решение использовать алгоритм sha256 для хеширования паролей.

Класс EncoderL реализует данный алгоритм. Структура класса представлена на рисунке 4.5.

|  |
| --- |
| public static class EncoderL  {  public static string HashPassword(string password)  {  using (var sha256 = SHA256.Create())  {  byte[] hashedBytes = sha256.ComputeHash(System.Text.Encoding.UTF8.GetBytes(password));  return Convert.ToBase64String(hashedBytes);  }  }  public static bool VerifyPassword(string password, string hashedPassword)  {  string passwordHash = HashPassword(password);  return string.Equals(passwordHash, hashedPassword);  }  } |

Рисунок 4.5 – Структура класса EncoderL

5. Тестирование, проверка работоспособности и анализ полученных результатов

5.1. Тестирование авторизации и регистрации

В момент авторизации и регистрации возможны ситуации, когда пользователь вводит некорректные данные, например, неверный пароль, незарегистрированный логин, некорректный адрес электронной почты и т.д. Такие исключения обрабатываются программным средством, оповещая пользователя о некорректных значениях.

На рисунке 5.1 приведен пример обработки пустых полей в форме регистрации.

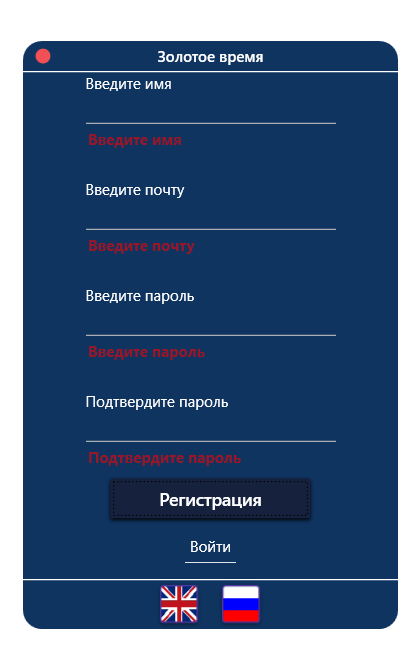
****

Рисунок 5.1 – Обработка пустых полей

На рисунке 5.2 приведен сценарий, при котором пользователь пытается зарегистрировать логин, который уже занят другим пользователем.

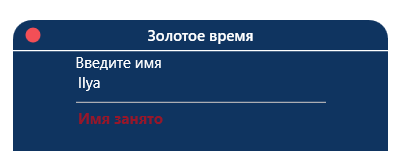
****

Рисунок 5.2 – Обработка регистрации, если логин уже занят

На рисунке 5.3 отображена ситуация, при которой пользователь вводит некорректные данные при входе в приложение. В это случае поля очищаются и пользователю выводится информирующее об ошибке входа сообщение.

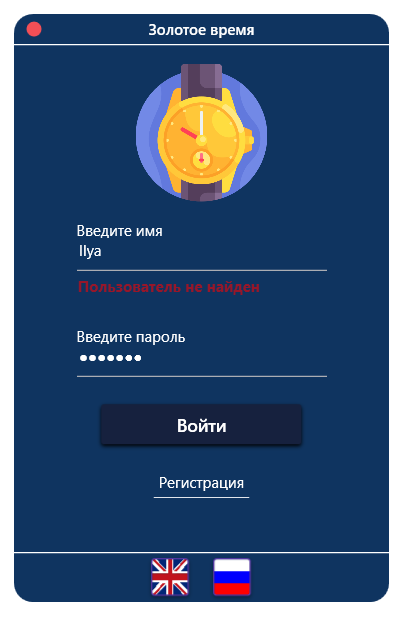
****

Рисунок 5.3 – Обработка некорректных данных при входе в приложение

Таким образом, приложение предотвращает поступление некорректных данных в базу данных и препятствует возникновению пользователей с одинаковыми логинами или адресами электронной почты.

5.3. Тестирование панели администратора

В панели администратора возможно генерирование многих исключительных ситуаций, таких как: отправка пустой формы, попытка записать

Примеры обработки пустых полей формы приведен на рисунке 5.7.

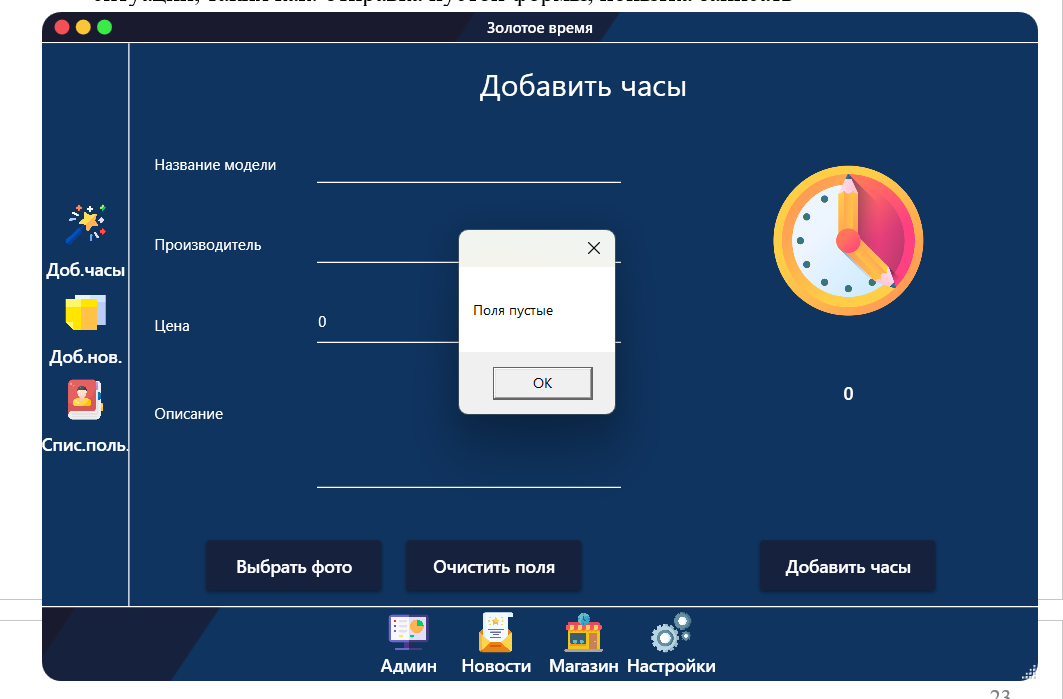
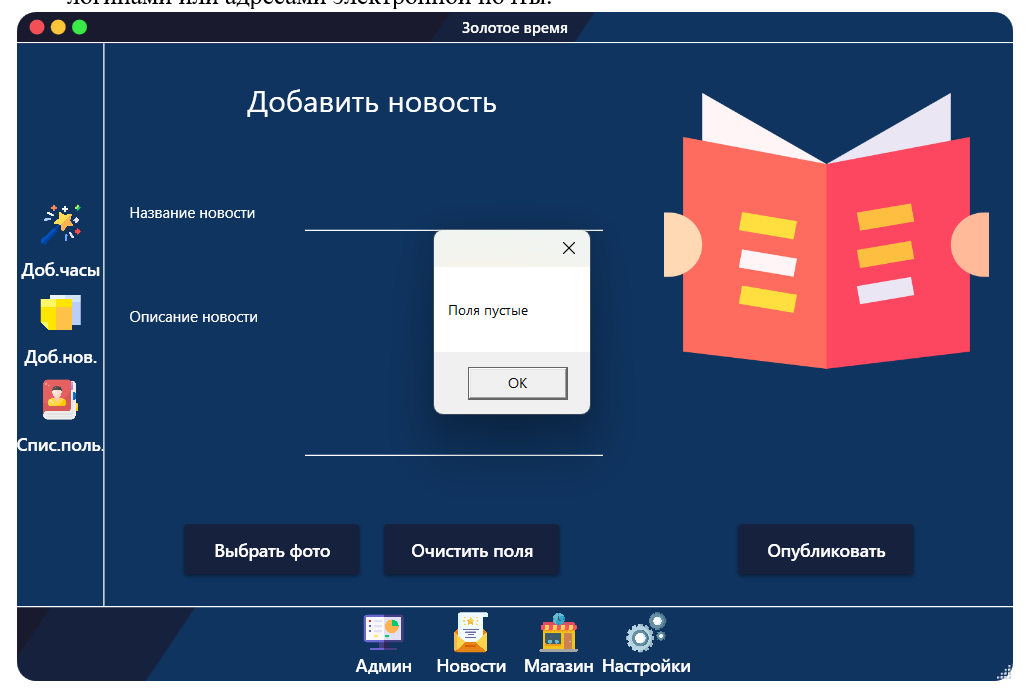
 

Рисунок 5.4 – Отправка незаполненной формы

6. Руководство по установке и использованию

При запуске программного средства «Золотое время» пользователь попадает на страницу авторизации, содержащую формы входа, регистрации и восстановления пароля. Если у пользователя еще нет аккаунта, ему следует нажать на кнопку «Регистрация», которая его перенаправит на форму для регистрации. Интерфейс стартового окна представлен на рисунке 6.1.

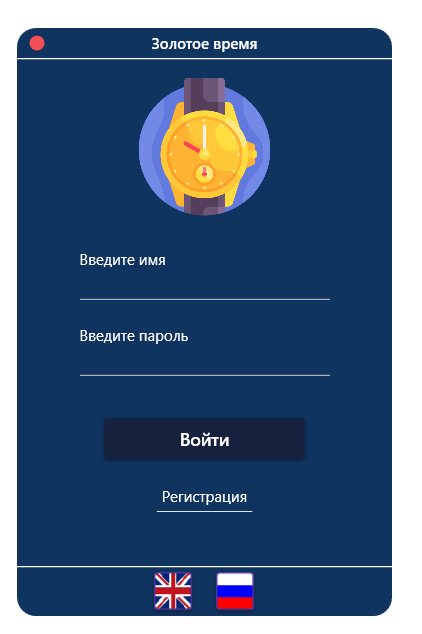


Рисунок 6.1 – Стартовое окно

Если была нажата кнопка «Регистрация», в открывшейся форме следует ввести все данные о регистрируемом пользователе. В случае, если кнопка была нажата ошибочно, форму можно закрыть. Форма регистрации представлена на рисунке 6.2.

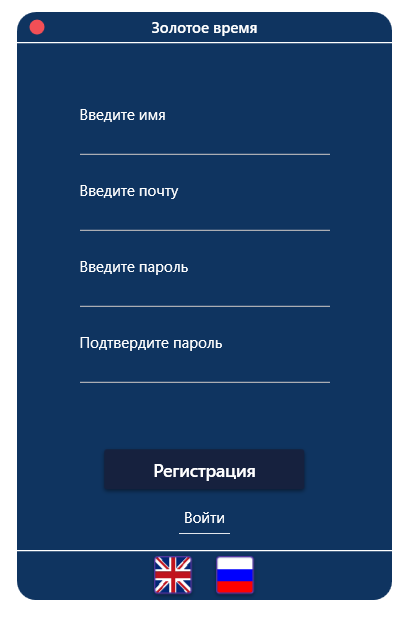


Рисунок 6.2 – Форма регистрации

После того, как пользователь введет необходимые данные он попадает на главную страницу. Она представлена на рисунке 6.3



Рисунок 6.3 – Главная страница

На главной странице пользователь может переходить на страницы корзины(рис.6.5), сравнения(рис.6.4), заказов(6.6), настроек(6.7).

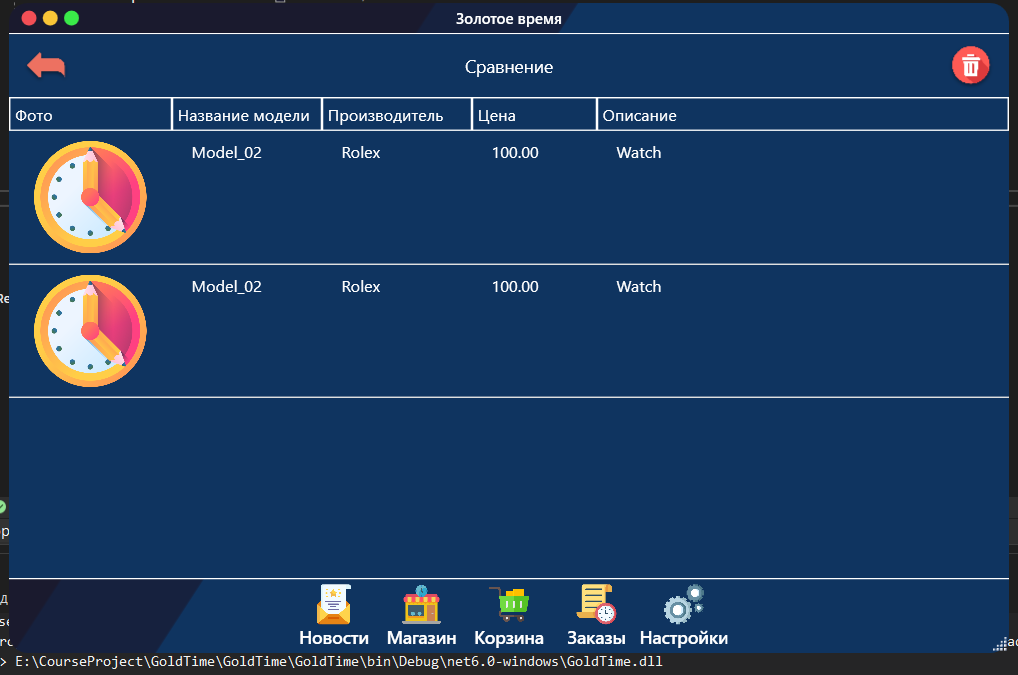


Рисунок 6.4 – Страница сравнения товаров

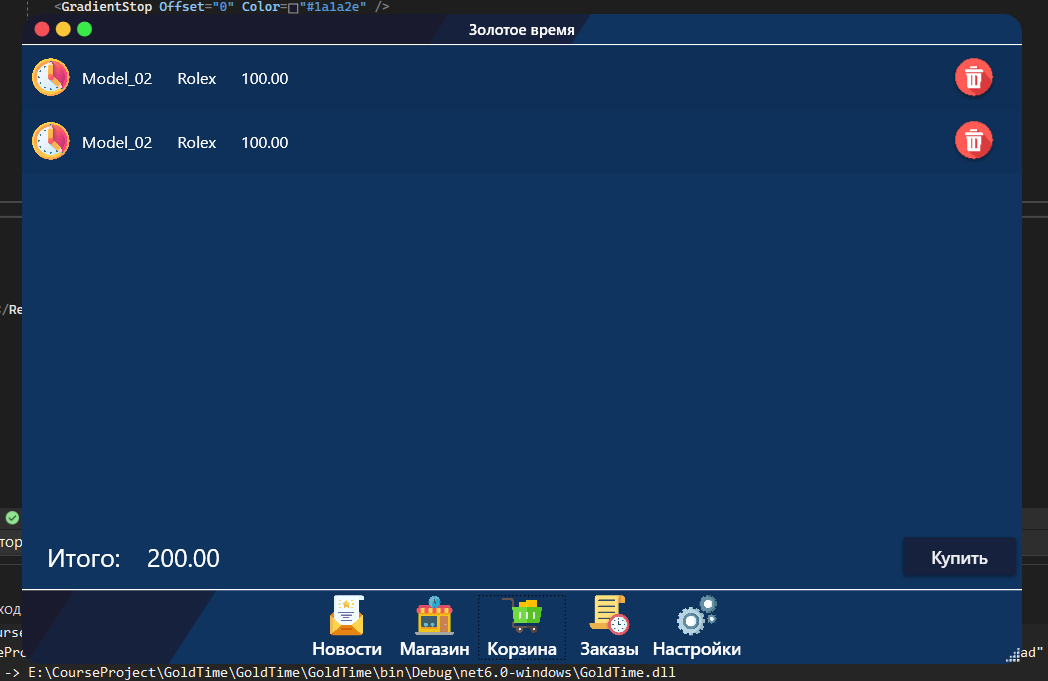


Рисунок 6.5 – Корзина товаров



Рисунок 6.6 – Страница заказов

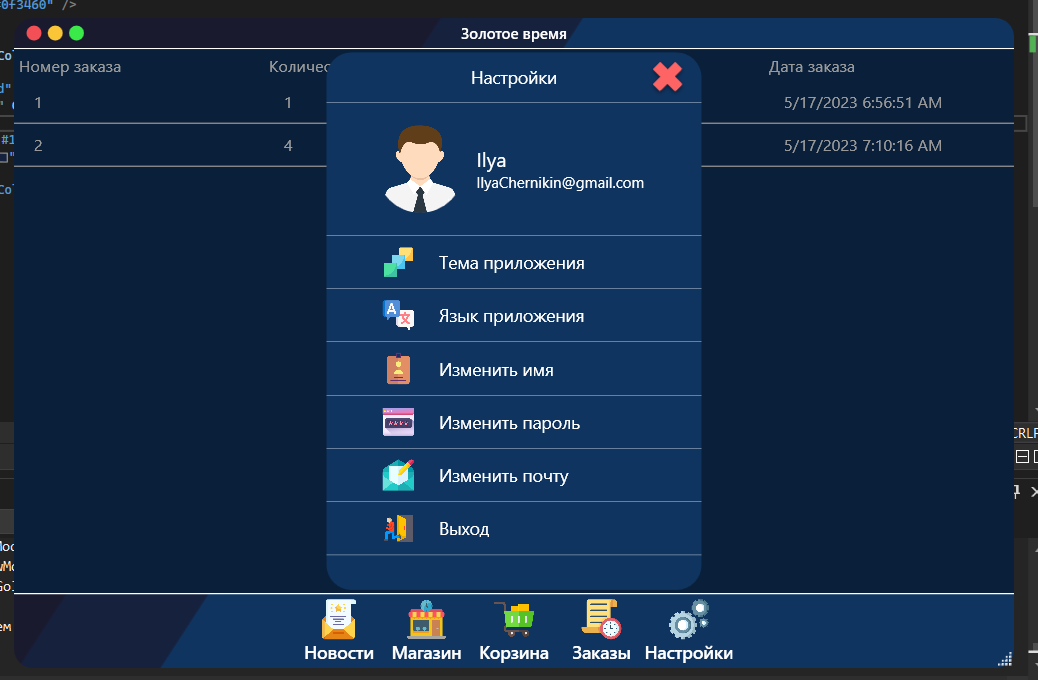


Рисунок 6.7 – Страница настроек



Рисунок 6.8 – Страница новостей

Также он имеет возможность просматривать подробную информацию о новости(рис. 6.9) и часов(рис. 6.10) . Так же на странице подробной информации о часах он может добавить в корзину и в сравнение.

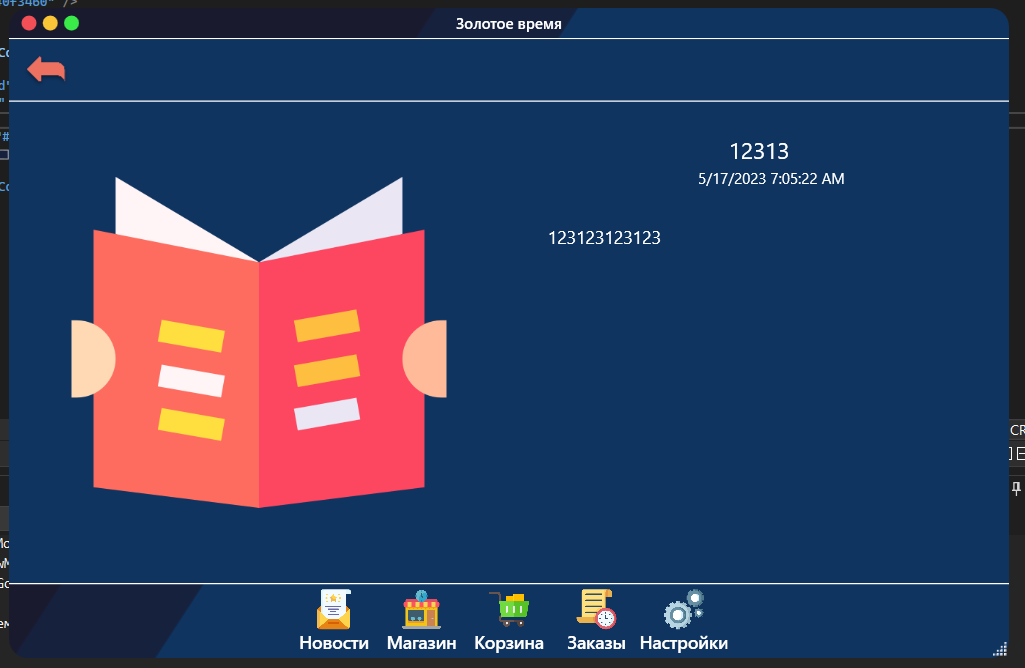


Рисунок 6.9 – Страница подробной информации о новости



Рисунок 6.10 – Страница подробной информации о часах

Заключение

В ходе выполнения курсового проекта было разработано программное средство «интернет-магазин часов ”Золотое время”» на языке C# с использованием технологий Entity Framework Core, WPF.

При разработке программного средства были выполнены все пункты из указанного списка основного функционала приложения, а именно:

* Добавление, изменение и товаров;
* Удаление товаров из каталога;
* Возможность выполнения регистрации и авторизации;
* Поиск товаров;
* Добавление в корзину товаров и в сравнение;
* Возможность изменения настроек приложения;
* Возможность просмотра и редактирования данных об аккаунте.

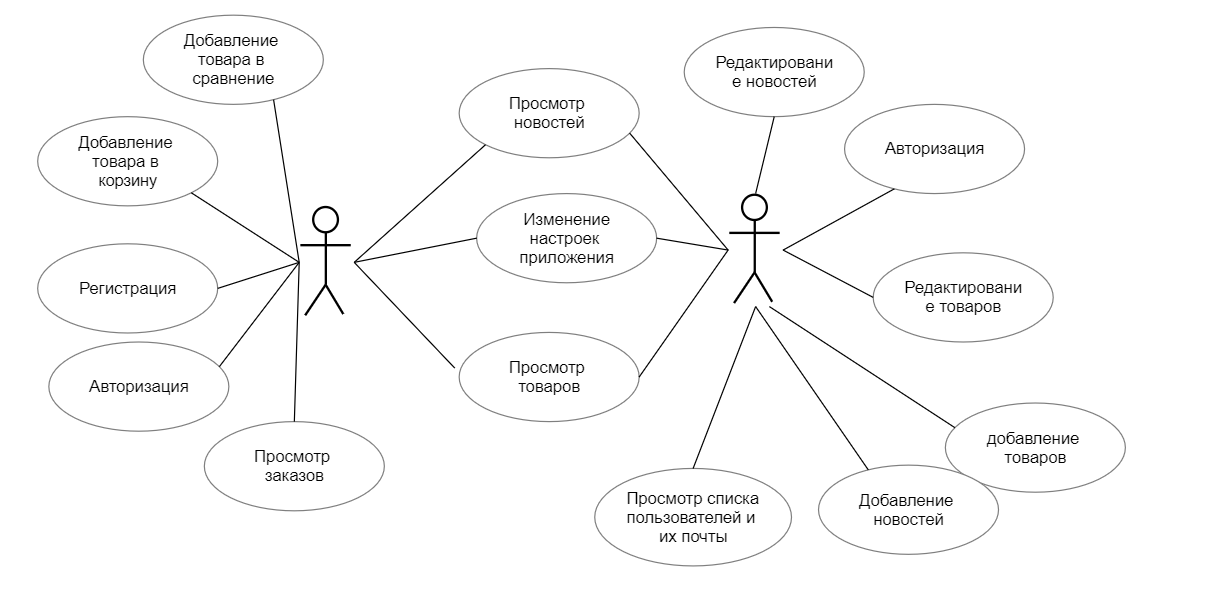
Тестирование программного средства показало, что оно работает корректно и выполняет все свои функции.

Также в процессе выполнения данного курсового проекта были закреплены навыки в программировании на языке C#, создании приложений на WPF, использование Entity Framework Core, работа с современным паттерном MVVM, проектирование базы данных в СУБД MS SQL Server 2019.

Список литературы

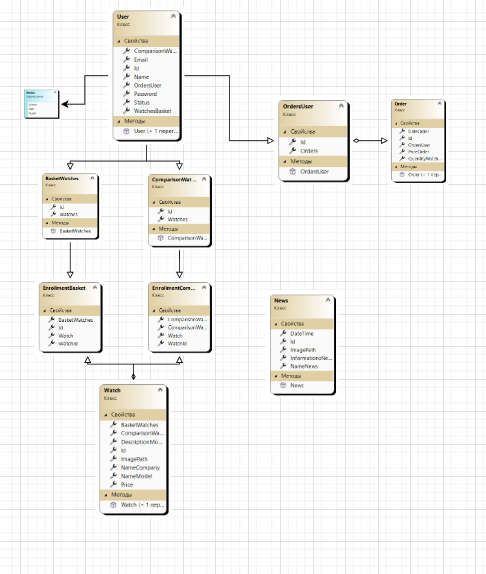
1. Microsoft Visual Studio [Электронный ресурс] – https://ru.wikipedia.org/wiki/Microsoft\_Visual\_Studio – Дата доступа 23.02.2023
2. Полное руководство по языку программирования C# 7.0 и платформе .NET 4.7. Режим доступа: https://metanit.com/sharp/tutorial/ – Дата доступа: 14.03.2023
3. Пацей, Н.В. Курс лекций по языку программирования C# / Н. В. Пацей. – Минск: БГТУ, 2018. – 175 с.
4. Руководство по WPF // [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/wpf/ – Дата доступа: 28.03.2023
5. Работа с Entity Framework Core [Электронный ресурс] – https://professorweb.ru/my/entity-framework/6/level1/ – Дата доступа 03.05.2023
6. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: https://professorweb.ru – Дата доступа: 10.04.2023.

Приложение А

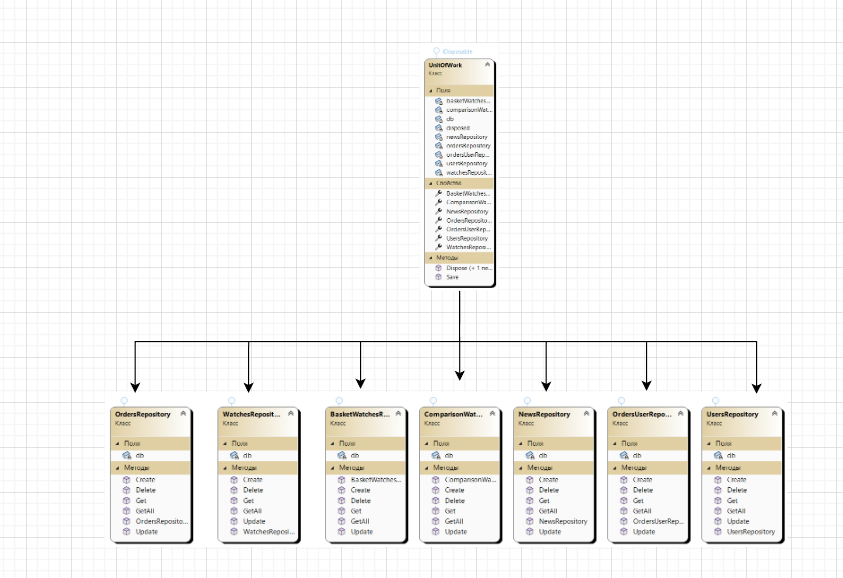


Приложение Б

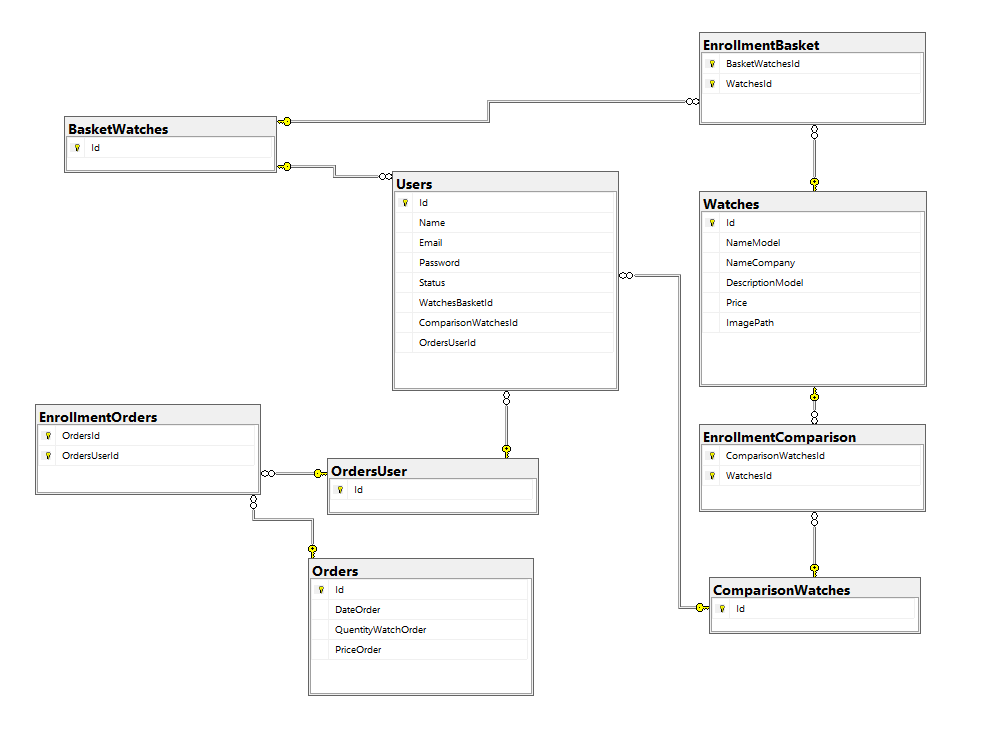
Классы-модели



Классы-репозитории



Приложение В

****

Приложение Г

Класс UnitOfWork

|  |
| --- |
| public class UnitOfWork : IDisposable  {  ApplicationContext db = new ApplicationContext();  #region fields  private BasketWatchesRepository basketWatchesRepository;  private ComparisonWatchesRepository comparisonWatchesRepository;  private NewsRepository newsRepository;  private OrdersUserRepository ordersUserRepository;  private UsersRepository usersRepository;  private WatchesRepository watchesRepository;  private OrdersRepository ordersRepository;  #endregion  #region propertys  public OrdersRepository OrdersRepository  {  get  {  if (ordersRepository == null)  ordersRepository = new OrdersRepository(db);  return ordersRepository;  }  }  public BasketWatchesRepository BasketWatchesRepository  {  get  {  if (basketWatchesRepository == null)  basketWatchesRepository = new BasketWatchesRepository(db);  return basketWatchesRepository;  }  }  public ComparisonWatchesRepository ComparisonWatchesRepository  {  get  {  if (comparisonWatchesRepository == null)  comparisonWatchesRepository = new ComparisonWatchesRepository(db);  return comparisonWatchesRepository;  }  }  public NewsRepository NewsRepository  {  get  {  if (newsRepository == null)  newsRepository = new NewsRepository(db);  return newsRepository;  }  }  public OrdersUserRepository OrdersUserRepository  {  get  {  if (ordersUserRepository == null)  ordersUserRepository = new OrdersUserRepository(db);  return ordersUserRepository;  }  }  public UsersRepository UsersRepository  {  get  {  if (usersRepository == null)  usersRepository = new UsersRepository(db);  return usersRepository;  }  }  public WatchesRepository WatchesRepository  {  get  {  if (watchesRepository == null)  watchesRepository = new WatchesRepository(db);  return watchesRepository;  }  }  #endregion  #region methods  public void Save()  {  db.SaveChanges();  }  private bool disposed = false;  public virtual void Dispose(bool disposing)  {  if (!this.disposed)  {  if (disposing)  {  db.Dispose();  }  this.disposed = true;  }  }  public void Dispose()  {  Dispose(true);  GC.SuppressFinalize(this);  }  #endregion  } |