**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**МИНСКИЙ КОЛЛЕДЖ ПРЕДПРИНИМАТЕЛЬСТВА**

Отделение «Программное обеспечение информационных технологий»

Специальность «Программное обеспечение информационных технологий»

**ДОПУЩЕН К ЗАЩИТЕ**

Заведующий отделением

/М.В Меньшикова/

**ИНТЕРНЕТ-МАГАЗИН «TEMPUSHIRING»**

Пояснительная записка

МКП ДП 2-40 01 01 33 015 ПЗ

**Учащийся** /С.В. Сипов/

**Руководитель**  /М.В. Кулакович/

**Консультант по экономической части**  /Е.Я. Моложавая/

**Консультант по охране труда** /И.В. Ахрименя/

**Председатель цикловой комиссии**  /Ю.А. Доманова/

**Нормоконтроль** /И.Ю. Дроздова /

**Рецензент** /О.П. Рябычина /

Минск, 2021

**МИНСКИЙ КОЛЛЕДЖ ПРЕДПРИНИМАТЕЛЬСТВА**

Отделение «Программное обеспечение информационных технологий»

Специальность «Программное обеспечение информационных технологий»

Специализация «Обработка экономической и деловой информации» 2-40 01 01 33 гр. П-1822

УТВЕРЖДАЮ

**Зам. директора**

**по учебно-методической работе**

Т.А. Андреева\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ЗАДАНИЕ**

**на дипломный проект**

Учащемуся \_Сипову Сергею Вадимовичу \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема проекта \_Интернет-магазин «TempusHiring» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Закреплено приказом по колледжу \_№168 от **23** ноября 2020 г.\_\_\_\_\_\_\_\_\_\_\_

2. Срок сдачи учащимся законченного проекта \_10.03.2021 г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Исходные данные по проекту: \_действия пользователя, название часов, \_\_\_ имя, логин, пароль, повторный пароль, дата, производитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. СОСТАВ ПРОЕКТА

а) Пояснительная записка (перечень вопросов со специальной частью, подлежащих разработке)

1 Постановка задачи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2 Вычислительная система\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3 Проектирование\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4 Описание программы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5 Отладка и испытание программы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6 Описание применения программы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

7 Охрана труда\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

8 Экономический раздел\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Заключение\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Список использованных источников\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Приложение(я)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

б) Графическая часть проекта (с точным указанием обязательных чертежей и графиков)

Лист 1\_\_Диаграмма «Сущность-связь» – плакат\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Лист 2\_\_Диаграмма «Вариантов использования» – плакат\_\_\_\_\_\_

5. Консультанты (с указанием разделов, которые они консультируют)

консультант по экономическому разделу: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Е.Я.Моложавая/\_

консультант по охране труда: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/И.В.Ахрименя/\_

6. Календарный график работы на весь период проектирования

07.12.20 − 15.12.20 – Постановка задачи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

16.12.20 – 23.12.20 – Вычислительная система\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

24.12.20 – 14.01.21 – Проектирование\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

15.01.21 – 01.02.21 – Кодирование\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

02.02.21 – 08.02.21 – Отладка и тестирование\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

16.02.21 – 22.02.21 – Разработка экономического раздела и раздела охраны\_\_ труда

23.02.21 – 01.03.21 – Оформление пояснительной записки и графической\_\_\_\_ части

**Председатель цикловой комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/**Ю.А. Доманова/

**Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/**М.В. Кулакович/

Дата выдачи задания\_05.12.2020 г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Задание принял для выполнения, дата \_\_\_05.12.2020 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Подпись учащегося**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*y*

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

МКП КП 2-40 01 01 33 015 ПЗ

Разраб.

Сипов

Пров.

*Кулакович*

Реценз.

Н. контр.

Утв.

*Интернет-магазин «TempusHiring»*

Лит.

Листов

95

МКП

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc65159199)

[1 Постановка задачи 6](#_Toc65159200)

[1.1 Формулировка задания 6](#_Toc65159201)

[1.2 Обоснование необходимости автоматизации 7](#_Toc65159202)

[1.3 Определение данных и их представление 7](#_Toc65159203)

[2 Вычислительная система 9](#_Toc65159204)

[2.1 Обоснование выбора языка программирования 9](#_Toc65159205)

[2.2 Обоснование выбора среды разработки 10](#_Toc65159206)

[2.3 Требования к конфигурации программного и аппаратного обеспечения 12](#_Toc65159207)

[3 Проектирование 14](#_Toc65159208)

[3.1 Проектирование интерфейса 14](#_Toc65159209)

[3.2 Инфологическая модель предметной области 14](#_Toc65159210)

[3.3 Логическая модель данных 16](#_Toc65159211)

[3.4 Физическая модель базы данных 20](#_Toc65159212)

[4 Описание программы 23](#_Toc65159213)

[4.1 Физическая структура 23](#_Toc65159214)

[4.2 Логическая структура 23](#_Toc65159215)

[5 Отладка и испытание программы 26](#_Toc65159216)

[5.1 Тестовые примеры 26](#_Toc65159217)

[5.2 Анализ полученных результатов 28](#_Toc65159218)

[6 Описание применения программы 34](#_Toc65159219)

[6.1 Назначение программы и условия применения 34](#_Toc65159220)

[6.2 Описание работы программы 34](#_Toc65159221)

*Рябычина*

*Дроздова*

*Меньшикова*

Изм.

Лист

№ докум.

Подпись

Дата

МКП КП 2-40 01 01 33 015 ПЗ

Лист

*3*

[7 Охрана труда 46](#_Toc65159222)

[7.1 Правовые и организационные вопросы охраны труда 46](#_Toc65159223)

[7.2 Производственная санитария и гигиена труда. 48](#_Toc65159224)

[7.3 Техника безопасности (электробезопасность) 49](#_Toc65159225)

[7.4 Пожарная безопасность 51](#_Toc65159226)

[8 Экономический раздел 55](#_Toc65159227)

[8.1 Определение общей трудоемкости разработки программного  
 обеспечения 55](#_Toc65159228)

[8.2 Определение затрат на оплату труда специалистов 56](#_Toc65159229)

[8.3 Определение полной себестоимости программного обеспечения 58](#_Toc65159230)

[8.4 Определение цены заказанного программного обеспечения 60](#_Toc65159231)

[Заключение 63](#_Toc65159232)

[Список использованных источников. 65](#_Toc65159233)

[Приложение А (обязательное) Листинг программы 66](#_Toc65159234)

**ВВЕДЕНИЕ**

С середины 1990 годов веб-разработка была одной из наиболее динамично развивающихся отраслей в мире. К росту данной отрасли толкает желание компаний продвигать свои товары и услуги для своих клиентов. За это время стоимость разработки сайтов и стоимость хостингов резко упали. Если ранее, подобные разработки стояли несколько десятков тысяч долларов, то сейчас это можно сделать гораздо дешевле, в зависимости от сложности заказов и стоимости содержания разработки.

Целью дипломного проекта является разработка интернет-магазина «TempusHiring». Подобные сайты позволяют ознакомиться с каталогом часов, не выходя из дома, либо в любом другом месте, в котором оказался потенциальный пользователь и при желании приобрести понравившуюся продукцию.

Задачами выполнения дипломного проекта являются:

– систематизация знаний в области программирования с использованием языка С#, SQL;

– углубление и расширение объема профессионально значимых знаний, умений и навыков в области разработки программ в среде Visual Studio 2019 и Microsoft SQL Server Management Studio 18;

– формирование умений и навыков самостоятельной организации исследовательской работы;

– овладение современными методами поиска, обработки и использования информации.

Данная пояснительная записка содержит разделы:

1. Постановка задачи – постановка задачи с указанием всех функций проекта, определение круга задач, которые должны быть автоматизированы или модифицированы в данном дипломном проекте.
2. Вычислительная система – описание выбранного языка программирования и среды разработки для реализации проекта, их достоинств и недостатков. Обоснование выбранного языка и среды.
3. Проектирование – описание внешнего пользовательского интерфейса − системы меню, диалоговых окон и элементов управления, описание назначения всех компонент приложения.
4. Описание программы – описание объектов со всеми функциями, входящие в приложение – назначение и взаимодействие.
5. Отладка и испытание программы – проверка работы программы на различных тестах – в нормальных и экстремальных условиях. Проверка обработки исключений. Проверка реакции программы на действия пользователя.
6. Описание применения программы – сведения о назначении программного средства, области применения, классе решаемых задач, ограничениях, накладываемых на область применения.
7. Охрана труда – описываются организационно-правовые вопросы, требования безопасности труда, санитарно-гигиенические требования, электробезопасность и пожарная безопасность предприятия.
8. Экономический раздел – содержит характеристики разработки, расчет затрат на разработку и расчет экономического эффекта.

Заключение – в этом разделе приводятся краткая формулировка проблемы и пути её решения, использованные методы и средства, полнота раскрытия проблемы.

# ПОСТАНОВКА ЗАДАЧИ

* 1. **Формулировка задания**

Необходимо разработать интернет-магазин «TempusHiring» для приобретения часов.

Реализовать базу данных в которой должна хранится следующая информация о:

* пользователях;
* ролях;
* сторонних сервисах;
* часах;
* товарах в корзине;
* проданных товарах;
* производителях;

Реализовать две роли для работы интернет-магазина: пользователь и администратор.

Для администратора необходимо разработать панель со следующими функциями:

* редактирование информации в базе данных;
* добавление информации в базу данных;
* удаление информации из базы данных;
* просмотра товара;
* просмотра отзывов;
* просмотра заказов;
* просмотра пользователей.

Для пользователей необходимо реализовать следующие функции:

* авторизация;
* просмотра товаров;
* просмотра личной информации;
* оформление заказа;
* просмотра корзины;
* добавление в корзину.
  1. **Обоснование необходимости автоматизации**

В наши дни интернет коммерция имеет огромное значение для успешной реализации любой продукции или услуги. Современный покупатель привык к возможности, не выходя из дома, или находясь в любом другом месте совершать покупки онлайн при помощи различных веб или интернет приложений. Если магазин не представлен в интернете, потенциальный клиент вероятнее всего не станет утруждать себя поездкой в тороговую точку и предпочтет конкурента, у которого имеется ресурс, позволяющий быстро, легко и удобно подобрать и заказать интересующий для себя продукт в онлайн режиме. Данное приложение поспособствует увеличению продаж за счет упрощения процесса подбора и покупки, сэкономит время клиента, и сможет позволить владельцу ликвидировать торговые площади в пользу реализации продукции при помощи сети интернет.

В качестве аналогов интернет-магазину «TempusHiring» можно привести такие сайты как: Hublot (https://www.hublot.com), Patek Philippe (https://www.patek.com), TimeCity (https://timecity.by).

* 1. **Определение данных и их представление**

Входные данные – это величины, которые задаются до начала алгоритма или определяются динамически во время его работы.

Выходные данные – это данные, получаемые в результаты выполнения алгоритма.

В данном проекте входными данными являются:

1. Личные данные пользователей.
2. Информация о товарах.
3. Информация о заказе.
4. Действия пользователя.
5. Товары в корзине.

Выходными данными данного проекта будут:

1. Информация о товарах.
2. Информация о заказе.
3. Результаты оплаты.

Количество товаров в корзине.

1. **Вычислительная система**
   1. **Обоснование выбора языка программирования**

В данном дипломном проекте используется язык программирования C# плюсами которого являются:

1. Быстро развивается.
2. Много синтаксического сахара. Синтаксический сахар – это конструкции, которые созданы для облегчения написания и понимания кода (особенно если это код другого программиста) и не играют роли при компиляции.
3. Средний порог вхождения.
4. Благодаря Xamarin на C# теперь можно писать под Android и iOS.
5. Добавлено функциональное программирование (F#).
6. Большое сообщество программистов.
7. Много вакансий на должность C# программиста в любом регионе.
8. С# использует технологию LINQ. [1, с. 1212]

К минусам C# можно отнести:

1. Ориентированность, в основном, только на .NET (на Windows платформу).
2. Бесплатность только для небольших компании, учащихся и программистов-одиночек. Для больших команд покупка лицензий обойдется недешево.
3. NET использует концепцию JIT-компиляции. Это означает, что программа будет скомпилирована в машинные коды по мере необходимости прямо во время работы приложения.[2]

Для создания реляционной базы данных (БД) в данном проекте был использован язык структурных запросов Structured Query Language (SQL).

SQL – это язык структурированных запросов, применяемый для создания, модификации и управления данными в БД, управляемой соответствующей системой управления базами данных (СУБД).

Плюсы SQL:

1. Наличие стандартов.
2. Скорость.
3. Масштабируемость.
4. Обширный функционал.
5. Независимость от конкретной СУБД.
6. Декларативность.

Минусы SQL:

1. Сложность вхождения.
2. Несоответствие реляционной модели данных.
3. Сложность работы с иерархическими структурами. [3]
   1. **Обоснование выбора среды разработки**

В качестве среды разработки выбрана платформа ASP.NET Core.

ASP.NET Core — свободно-распространяемый кросс-платформенный фреймворк для создания веб-приложений с открытым исходным кодом.

Плюсы платформы .ASP Net Core:

* 1. Межплатформенная поддержка.
  2. Производительность.
  3. Переносимые библиотеки классов.
  4. Переносимое или автономное развертывание.
  5. Полная поддержка командной строки.
  6. Открытый код.
  7. Возможность взаимодействия с полной платформой .Net Framework. [7]

Минусы платформы:

* + 1. Слабо развита экосистема.
    2. Нет русификации. [7]

Плюсы Visual Studio являются:

1. Встроенный Web-сервер.
2. Поддержка множества языков при разработке.
3. Меньше кода для написания.
4. Интуитивный стиль кодирования.
5. Более высокая скорость разработки.
6. Возможности отладки.
7. Возможность рефакторизации кода.
8. Встроенная функция управления исходным кодом.

Минусы Visual Studio являются:

1. Невозможность отладчика отслеживать в коде режима ядра.
2. Очень дорогая полная версия. [6]

## Плюсы SQL Server:

1. СУБД масштабируется, поэтому работать с ней можно на портативных ПК или мощной мультипроцессорной технике. Процессор может одновременно обрабатывать большой объем запросов.
2. Размер страниц – до восьми кб, поэтому данные извлекаются быстро, подробную и сложную информацию хранить удобнее. Система позволяет обрабатывать транзакции в интерактивном режиме, есть динамическая блокировка.
3. Рутинные административные задачи автоматизированы: это управление блокировками, памятью, редактура размеров файлов. У системы продуманы настройки, можно создать профили пользователей.
4. Реализован поиск по фразам, тексту, словам, можно создавать ключевые индексы.
5. В SQL Server есть репликации через интернет, предусмотрена синхронизация. Есть полноценный веб-ассистент для форматирования страниц.
6. В систему интегрирован сервер интерактивного анализа для принятия решений, создания корпоративных отчетов. Есть службы преобразования информации.
7. Запросы можно формулировать на английском языке, без программирования.
8. СУБД поддерживает работу с другими продуктами Microsoft: Access, MS Excel.

## Недостатки SQL Server:

1. Высокая цена программы. [4]
   1. **Требования к конфигурации программного и аппаратного обеспечения**

Windows 10 – новая операционная система от компании Microsoft, релиз которой состоялся 29 июля 2015 года. Разработчики искренне постарались учесть все пожелания пользователей и создать единую платформу для всех устройств. Для настольных компьютеров, планшетов, смартфонов и т.д. Кроме того, ещё одной интересной особенностью выпущенного продукта является то, что он является последним в линейке Windows. На нём нумерация прекращается, а обновления будут устанавливаться только для конкретных программ.

**Плюсы:**

1. Устанавливается совершенно бесплатно у пользователей 7 и 8 версий в течение года после официального релиза.
2. Полноэкранные приложения можно уменьшить до нужных размеров, чего нельзя было сделать в восьмой версии.
3. Поддержка нескольких рабочих столов без зависаний и торможений, характерных для предыдущей системы линейки.
4. Универсальность. Новая Windows может использоваться как на настольных ПК, так и на карманных, переносных устройствах. Все официальные приложения устанавливаются на любой гаджет с поддержкой Windows 10, а их интерфейс подстраивается под его особенности.
5. Также стоит отметить, что системные требования для установки Windows 10 такие же, как для седьмой версии. Она без проблем запустится на устаревших системах с ограниченным объемом оперативной памяти и слабым процессором. Официальная спецификация очень демократична.
6. Можно вернуться к предыдущей Windows 7 или Vista, если имеется резервная копия на жестком диске, а также к восьмой версии.

**Минусы:**

1. На некоторых устройствах устанавливается только со 2-3-й попытки. К счастью, это больше касается переносных ПК – среди пользователей персональных компьютеров о таких проблемах особо не упоминалось.
2. Проблемы с региональными настройками. Многие пользователи отметили, что старались внести настройки своего региона, но ОС автоматически устанавливает США, с чем ничего нельзя поделать.
3. Отсутствие приложения Media Center, а с ним и привычного приложения для воспроизведения DVD-дисков. Это объясняется тем, что такого рода носители в мире современных технологий являются устаревшими (не в странах СНГ, где поддержка технологии все еще нужна).
4. Проблемы с приватностью. Лицензионное соглашение Windows10 дает Microsoft право на сбор вашей персональной информации, статистики – данных о местоположении, истории браузера и прочего в таком духе. [5]

Требования к аппаратным и операционным ресурсам для разработки:

1. Операционная система Windows10.
2. Процессор Intel Core i7-3770 CPU 3,40 GHz.
3. Видеоадаптер Gigabyte Gtx 1050 Ti 4gb.
4. Оперативная память 16GB.
5. SSD Kingston 128Gb.
6. **Проектирование**

## 3.1 Проектирование интерфейса

В данном проекте должны быть страницы: «Home», «Catalog», «ShoppingCart», «Orders», «Notification», «Sign In», «Sign Up», «ForgotPassword».

На странице «Home» выводится общяя информация о предметной области магазина.

На странице «Catalog» выводится общий список часов, к которому авторизованный пользовать имеет возможность применять фильтры.

На странице «ShoppingCart» выводится список часов, которые пользователь добавил в корзину.

На странице «Orders» выводится инфрмация обо всех заказах текущего пользователя.

На странице «Notification» выводится информация общего храктера для уведомления пользователя о результате совершенного действия.

На страницах «Sign In», «Sign Up» выводятся формы для авторизации или регистрации пользователя в приложении при помощи стандартных средств или с использованием сторонних сервисов.

На странице «ForgotPassword» выводится поля для ввода электронной почты, на которую будет отправлена инструкция для дальнейших действий по восстановлению доступа к аккаунту.

## 3.2 Инфологическая модель предметной области

Интернет-магазин является очень удобным способом приобрести необходимую продукцию не выходя из дома. Профильные интернет-магазины имеют возможность дистанционно предоставить удобную и интуитивно понятную инфраструктуру для подбора удовлетворяющих требованию клиента часов. Интернет-магазин должен хранить большое количество информации о производителе, часах, клиентах и заказах.

Эту информацию весьма удобно хранить в базе данных, поскольку это обеспечивает быстрый поиск, надежность хранения и, что главное, упрощает работу как продавца, так и покупателя.

Проанализируем объекты реального мира. Для формирования концептуальной модели необходимо провести идентификацию объектов сущности базы данных.

Необходимо спроектировать базу данных интернет-магазина, информация из которой будет использоваться для продажи часов.

Для контроля за наличием часов в БД необходимо хранить список всех имеющихся часов и их характеристики.

Для контроля информации о часах, необходимо вести учет количества часов и количества активных заказов.

При проектировании БД необходимо учитывать следующее:

* + 1. Предоставлять возможность оформления заказа только на те часы, которые есть в наличии.
    2. Осущестлять изменение количества и наименования часов в корзине.
    3. Осуществлять удаление из корзины.
    4. Осуществлять оформление заказа.
    5. Не допускать добавление информации о пользователе с существующим E-mail.
    6. Не допускать использование аккаунта без подтверждения электронной почты.
    7. Не допускать добавление информации о пользователе с паролем менее шести символов.
    8. Выдавать информацию только имеющихся часов.
    9. В корзине выводить только те товары, на которые не был офрмлен заказ.
    10. Оформлять заказы могут только авторизованные пользователи.
    11. Добавлять товары в корзину могут только авторизованные пользователи.

Таким образом база данных содержит следующие сущности как пользователь, заказы, заказ-часы, корзина, часы, производитель, механизм, тип механизма, материал корпуса.

Связь между сущностями часы и производитель, связывает производителя с часами**.** Данная связь предназначена для получения информации о производителе часов.

Связь между сущностями часы и механизм**.** Данная связь предназначена для связи часов с механизмом, из которого оно сделано.

Сущность часы имеет связь с такими сущностями как заказ-часы, корзина.

## 3.3 Логическая модель данных

В проекте было принято использовать девять таблиц. Первая таблица будет содержать информацию о пользователях (Id, UserName, Email, PasswordHash, EmailConfirmed). Вторая таблица будет содержать информацию о заказах (Id, PaymentMethod, OrderDate, TotalCount, TotalPrice, UserId, IsOrderCompeted). Третья таблица будет содержать информацию о часах в заказе (Id, OrderId, WatchId, Count). Четвертая таблица будет содержать информацию о корзине (Id, UserId, WatchId, Count). Пятая таблица будет содержать информацию о часах (Id, Title, Description, Price, Photo, ManufacturerId, MechanismId). Шестая таблица будет содержать информацию о производителях (Id, Title, Country). Седьмая таблица будет содержать информацию о механизмах (Id, Description, PowerReserveDays, Title, MechanismTypeId). Восьмая таблица будет содержать информацию о типе механизма (Id, Title). Девятая таблица будет содержать информацию о материале корпуса (Id, Title). На плакате 1 приведена ER диаграмма. Информация о клиентах хранится в таблице 3.1.

Таблица 3.1­ ­– Информация о клиентах

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Примечание |
| Id | Счетчик | PK |
| UserName | Текст | Имя пользователя |
| Email | Текст | Электронная почта |
| PasswordHash | Текст | Хэш пароля |
| EmailConfirmed | Логический | Подтверждена ли почта |

Информация о заказах хранится в таблице 3.2.

Таблица 3.2­ ­– Информация о заказах

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Примечание |
| Id | Счетчик | PK |
| PaymentMethod | Текст | Способ оплаты |
| OrderDate | Текст | Дата оформления заказа |
| TotalCount | Число | Общее количество заказанных часов |
| TotalPrice | Число | Общая стоимость заказанных часов |
| UserId | Число | FK (Таблица User поле Id) |
| IsOrderCompleted | Логический | Выполнен ли заказ |

Информация о часах в заказе хранится в таблице 3.3.

Таблица 3.3 ­– Информация о часах в заказе

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Примечание |
| 1 | 2 | 3 |
| Id | Счетчик | PK |
| OrderId | Число | FK (Таблица Order поле Id) |
| Продолжение таблицы 3. 3 ­– Информация о часах в заказе | | |
| 1 | 2 | 3 |
|  |  |  |
| WatchId | Число | FK (Таблица Watch поле Id) |
| Count | Число | Количество |

Информация о корзине хранится в таблице 3.4.

Таблица 3.4­ ­– Информация о корзине

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Примечание |
| Id | Счетчик | PK |
| UserId | Число | FK (Таблица User поле Id) |
| WatchId | Число | FK (Таблица Watch поле Id) |
| Count | Число | Количество |

Информация о часах хранится в таблице 3.5.

Таблица 3.5­ ­– Информация о часах

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Примечание |
| 1 | 2 | 3 |
| Id | Счетчик | PK |
| Title | Текст | Название |
| Description | Текст | Описание |
| Price | Число | Цена |
| Photo | Текст | Фото |
| ManufacturerId | Число | FK (Таблица Manufacturer поле Id) |
| MechanismId | Число | FK (Таблица Mechanism поле Id) |

Информация о производителе хранится в таблице 3.6.

Таблица 3.6­ ­– Информация о производителе

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Примечание |
| Id | Счетчик | PK |
| Title | Текст | Название |
| Country | Текст | Страна |

Информация о механизмах хранится в таблице 3.7.

Таблица 3.7­ ­– Информация о механизмах

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Примечание |
| Id | Счетчик | PK |
| Title | Число | Название |
| Description | Число | Описание |
| PowerReserveDays | Число | Запас хода |
| MechanismTypeId | Число | FK (Таблица MechanismType поле Id) |

Информация о типе механизма хранится в таблице 3.8.

Таблица 3.8­ ­– Информация о типе механизма

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Примечание |
| Id | Счетчик | PK |
| Title | Текст | Название |

Информация о материале корпуса хранится в таблице 3.9.

Таблица 3.9­ ­– Информация о материале корпуса

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Примечание |
| Id | Счетчик | PK |
| Title | Текст | Название |

## 3.4 Физическая модель базы данных

Для создания базы данных принято решение использовать Microsoft SQL Server Management Studio 18.

USE [TempusHiring]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[AspNetUsers](

[Id] [int] IDENTITY(1,1) NOT NULL,

[UserName] [nvarchar](256) NULL,

[NormalizedUserName] [nvarchar](256) NULL,

[Email] [nvarchar](256) NULL,

[NormalizedEmail] [nvarchar](256) NULL,

[EmailConfirmed] [bit] NOT NULL,

[PasswordHash] [nvarchar](max) NULL,

[SecurityStamp] [nvarchar](max) NULL,

[ConcurrencyStamp] [nvarchar](max) NULL,

[PhoneNumber] [nvarchar](max) NULL,

[PhoneNumberConfirmed] [bit] NOT NULL,

[TwoFactorEnabled] [bit] NOT NULL,

[LockoutEnd] [datetimeoffset](7) NULL,

[LockoutEnabled] [bit] NOT NULL,

[AccessFailedCount] [int] NOT NULL,

CONSTRAINT [PK\_AspNetUsers] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

USE [TempusHiring]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Watches](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Diameter] [float] NOT NULL,

[Description] [nvarchar](max) NULL,

[Price] [decimal](18, 2) NOT NULL,

[Photo] [nvarchar](max) NULL,

[ManufacturerId] [int] NOT NULL,

[GlassMaterialId] [int] NOT NULL,

[MechanismId] [int] NOT NULL,

[BodyMaterialId] [int] NOT NULL,

[StrapId] [int] NOT NULL,

[Title] [nvarchar](max) NULL,

[Gender] [nvarchar](max) NOT NULL,

CONSTRAINT [PK\_Watches] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

USE [TempusHiring]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Orders](

[Id] [int] IDENTITY(1,1) NOT NULL,

[PaymentMethod] [nvarchar](max) NULL,

[OrderDate] [datetime2](7) NOT NULL,

[TotalCount] [int] NOT NULL,

[TotalPrice] [decimal](18, 2) NOT NULL,

[IsOrderCompleted] [bit] NOT NULL,

[UserId] [int] NOT NULL,

CONSTRAINT [PK\_Orders] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

1. **Описание программы**

## 4.1 Физическая структура

В проекте реализовать:

Классы Watch, User, Order, Mechanism, ShoppingCart, TempusHiringDbContext служат для обращения к БД.

Классы AcountService, CatalogService, OrderService, ShoppingCartService содержат бизнес логику для контроллеров.

Класс AccountController предназначен для авторизации и регистрации пользователя.

Класс CatalogController предназначен для вывода данных на страницы «Shop», «Ladies Watches» и «Mens Watches».

Класс OrdersController предназначен для обработки заказов

Класс ShoppingCartContrtoller предназначен для обработки корзины.

Класс HomeController предназначен для вывода информации на стартовую страницу.

## 4.2 Логическая структура

В проекте реализовать следующие методы:

1. LoginAsync(LoginDTO loginModel, bool lockoutOnFailure)– предназначен для входа в свой профиль.
2. LogoutAsync()– предназначен для выхода из своего профиля.
3. RegisterAsync(RegisterDTO registerModel, string claimRole = ClaimRoles.User) – предназначен для регистрации новых пользователей.
4. Index(FilteredWatchViewModel filteredViewModel, int pageNum = 1) – возвращает стартовую страницу.
5. MensWatches(FilteredWatchViewModel filteredViewModel, int pageNum = 1) – возвращает каталог мужских часов.
6. WomensWatches(FilteredWatchViewModel filteredViewModel, int pageNum = 1) – возвращает каталог женских часов.
7. Details(int watchId) – возвращает страницу с конкретной информацией о часах.
8. UpdatePriceRange(int startPrice, int endPrice) – обновляет диапазон цен для фильтра.
9. HashPassword() – возвращает HASH пароль пользователя.
10. GetPriceRange() – возвращает ценовой диапазон.
11. ChangeCount(int watchId, int count) – изменяет количество часов в базе данных.
12. ChangeSelection(int watchId, bool isChecked) – изменяет набор выбранных в корзине товаров.
13. GetWatchCountInStock(int watchId) – возвращает количество указанного экземпляра часов на складе.
14. GetWatchPrice(int watchId) – возвращает цену для указанного экземпляра часов.
15. GetOrderSummary() – возвращает текущее состояние корзины пользователя.
16. Remove(int cartId) – удаляет запись из корзины пользователя.
17. Buy() – оформляет заказ на выбранные пользователем часы.
18. LogoutAsync() – выход пользователя из системы.
19. ConfigureExternalAuthentication(string provider, string redirectUrl) – конфигурирует аутентификацию для стороннего сервиса.
20. ExternalLoginAsync() – авторизация пользователя в системе через сторонний сервис.
21. ExternalRegisterAsync(UserDTO userModel) – регистрация пользователя через сторонний сервис.
22. GeneratePasswordResetTokenAsync(string email) – генерирует токен для сброса пароля пользователя.
23. GenerateEmailConfirmationAsync(string email) – генерирует токен для подтверждения какого-либо действия через электронную почту.
24. ResetPasswordAsync(ResetPasswordDTO resetPasswordModel) – сбрасывает пароль пользователя.
25. SendEmailAsync(string email, string subject, string message) – отправляет письмо на почту пользователя.
26. AddAsync(TModel model) – добалвяет сущность в базу данных.
27. AddRangeAsync(IEnumerable<TModel> models) – добавляет набор сущностей в базу данных.
28. ReadAsync(int id) – возвращает сущность из базы данных по указанному идентификатору.
29. GetPagedResult(int pageNum, int itemsOnPage) – возвращает элементоы, которые представляют собой часть разбитого на страницы набора данных.
30. ReadAll() – считывает все записи из базы данных.
31. UpdateRange(IEnumerable<TModel> items) – обновляет набор указанных данных.
32. Delete(TModel item) – удаляет запись из базы данных.
33. SaveChanges() – сохраняет изменения в базе данных.
34. SaveChangesAsync() – выполняет асинхронное сохранение изменений в базе данных.
35. GetPaged<T>(this IQueryable<T> source, int pageNum, int itemsOnPage) – генерирует страницу для пагинации.
36. **ОТЛАДКА И ИСПЫТАНИЕ ПРОГРАММЫ**

## 5.1 Тестовые примеры

Тестирование программного обеспечения – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом.

Проверка функциональности (тестирование методом «черного ящика») – проверка соответствия программного обеспечения требованиям, заявленным в спецификации. Может проводиться как полное тестирование заявленной функциональность, так и проверка только базовой функциональности.

Системное тестирование ­ высокоуровневая проверка функционала всей программы или системы в целом.

Тестирование производительности ­ тестирование, которое проводится с целью определения, как быстро работает система или её часть под определённой нагрузкой.

Нагрузочное тестирование ­ тестирование предназначено для проверки работоспособности системы при стандартных нагрузках и для определения максимально возможного пика, при котором система работает правильно.

Стресс тестирование ­ тестирование предназначено для проверки работоспособности системы при нестандартных нагрузках и для определения максимально возможного пика, при котором система работает правильно. Так же предназначено для выявления результатов, при которых система переходит в нерабочее состояние.

Тестирование удобства пользования – это метод тестирования, направленный на установление степени удобства использования, обучаемости, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий. Выявлять проблемы, связанные со специфическим механизмом интерфейса определять, существуют ли проблемы с удобностью интерфейса для навигации, использования основного функционала.

Тестирование безопасности – это стратегия тестирования, используемая для проверки безопасности системы, а также для анализа рисков, связанных с обеспечением целостного подхода к защите приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным.

Каждая сложная программная система состоит из отдельных частей – модулей, выполняющих ту или иную функцию в составе системы. Для того, чтобы удостовериться в корректной работе всей системы, необходимо вначале протестировать каждый модуль системы по отдельности. В случае возникновения проблем при тестировании системы в целом это позволяет проще выявить модули, вызвавшие проблему, и устранить соответствующие дефекты в них. Такое тестирование модулей по отдельности получило называние модульного тестирования.

Регрессионное тестирование проводится с целью проверить, не влияют ли новые функции, улучшения и исправленные дефекты на существующую функциональность продукта и не возникают ли старые дефекты.

В результате проверки программного средства на работоспособность, были проведены следующие тестовые испытания:

1. Регистрация с существующим E-mail.
2. Регистрация с несуществующим E-mail.
3. Вход не существующего пользователя.
4. Вход существующего пользователя.
5. Проверка работы интерфейса в Mozilla Firefox.
6. Переход на станицу «ShoppingCart» не авторизованным пользователем.
7. Фильтрация часов по возрастанию по цене.
8. Фильтрация часов по убыванию по цене.
9. Добавление часов в корзину.
10. Добавление часов в заказы.
11. Выход из профиля.

**5.2 Анализ полученных результатов**

Тест 1 показал, что при регистрация с существующим E-mail, выдавалась ошибка «Данный E-mail занят», что изображено на рисунке 5.1.



Рисунок 5.1 ­– Сообщение об ошибке «Данный email занят»

Тест 2 показал, что при регистрация с не занятым E-mail, происходил переход на страницу авторизации, что изображено на рисунке 5.2.

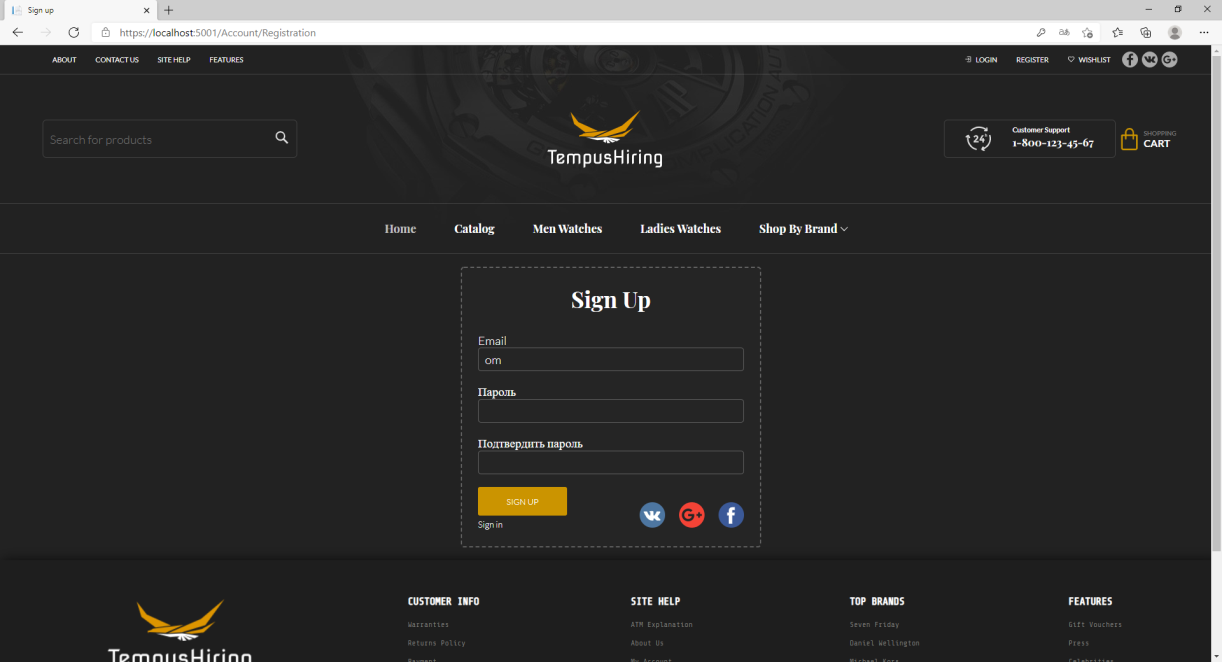


Рисунок 5.2 – Переход на страницу авторизации

Тест 3 показал, что при входе несуществующего пользователя, выдавалась ошибка «Неверный логин и(или) пароль», что изображено на рисунке 5.3.

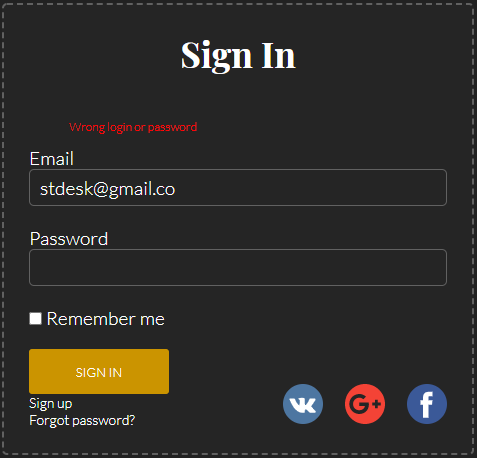


Рисунок 5.3 – Сообщение об ошибке «Неверный логин или пароль»

Тест 4 показал, что при входе существующего пользователя, происходила авторизация и перенаправление на главную страницу, что изображено на рисунке 5.4.

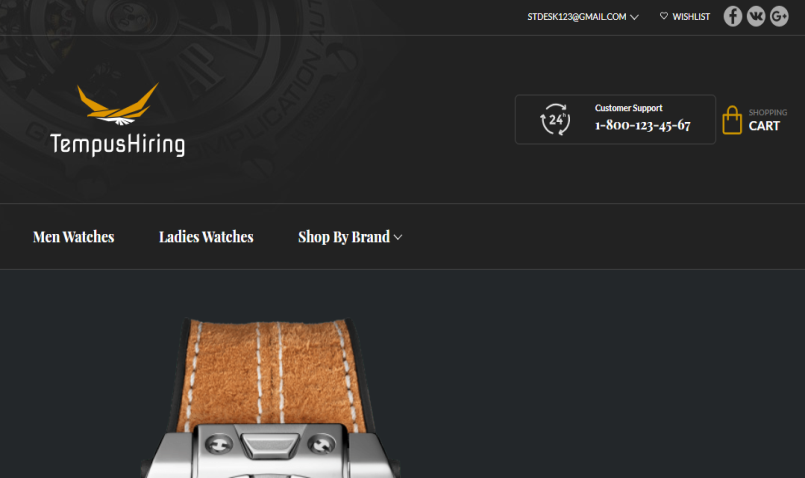
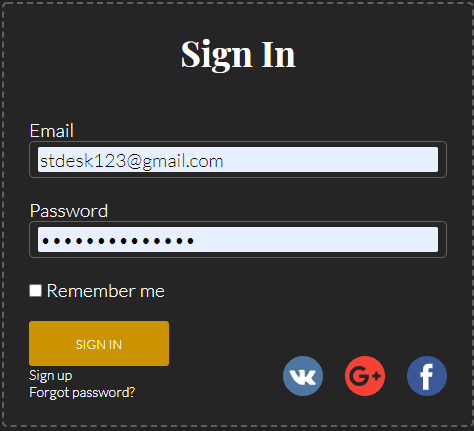


Рисунок 5.4 – Переход на главную страницу

Тест 5 показал, что запуске программы в Mozilla Firefox, интерфейс остался прежним, что изображено на рисунке 5.5.



Рисунок 5.5 – Корректный запуск в браузере Mozilla Firefox

Тест 6 показал, что при переходе на станицу «ShoppingCart» не- авторизованным пользователем происходит перенаправление на станицу авторизации, что изображено на рисунке 5.6.

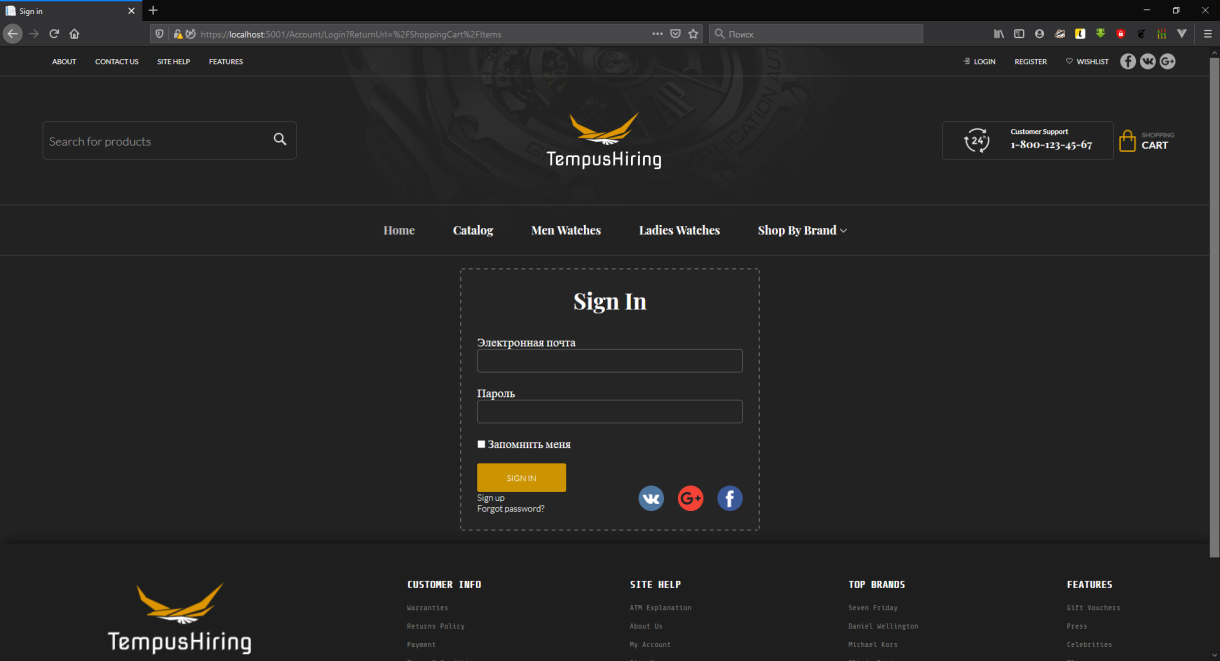
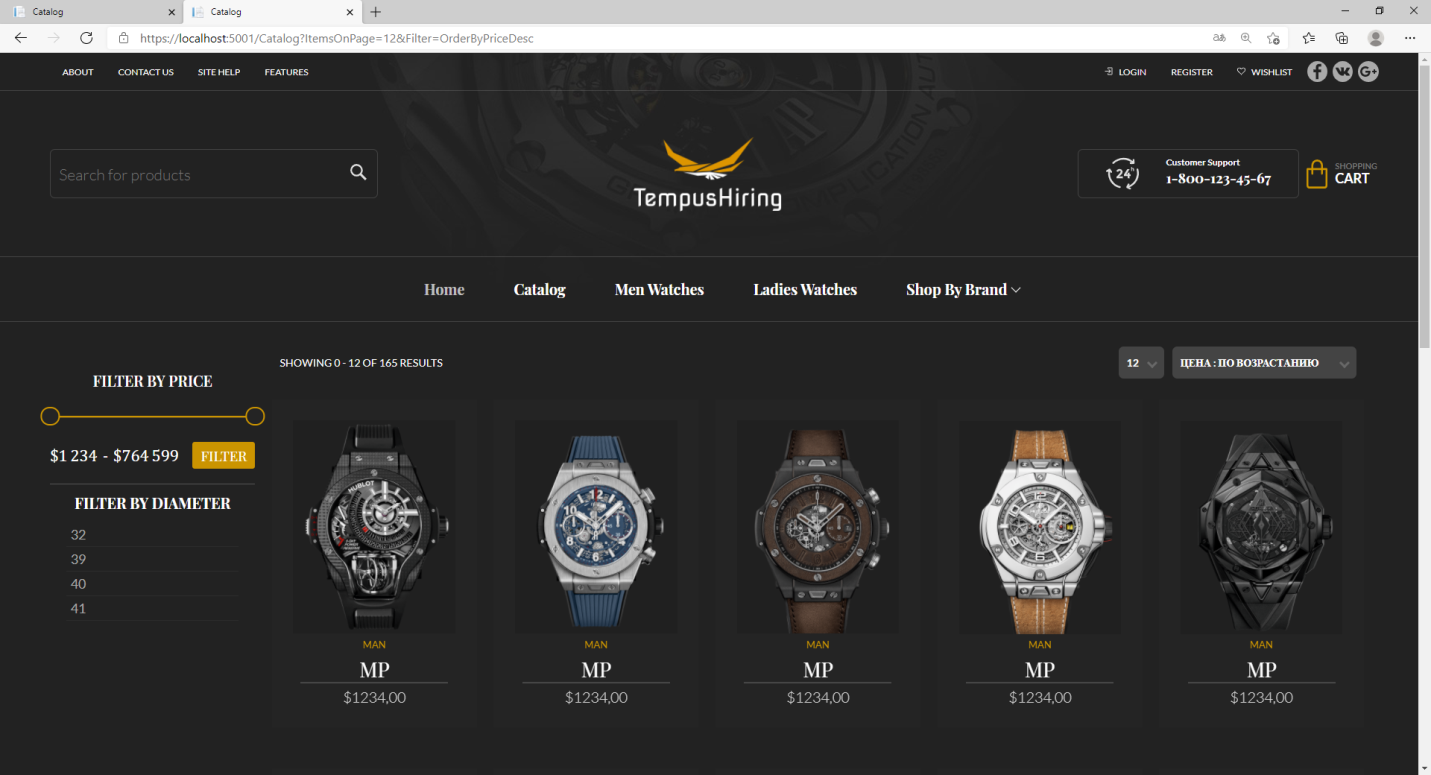
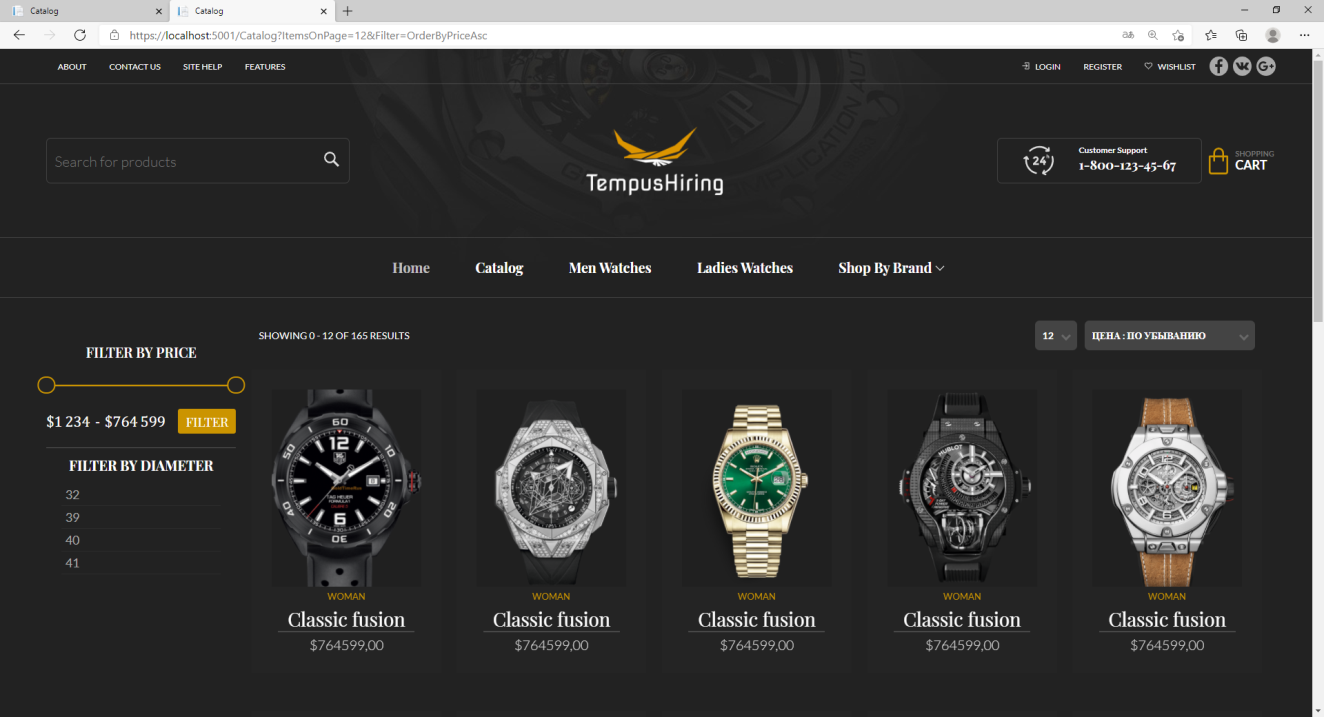


Рисунок 5.6 – Страница авторизации

Тест 7 показал, что при фильтрации часов по цене по возрастанию сначала будут выводиться самые дешевые часы, что изображено на рисунке  5.7.

 Рисунок 5.7 – Корректная работа сортировки по цене по возрастанию

Тест 8 показал, что при фильтрации часов по цене по убыванию сначала будут выводиться самые дорогие часы, что изображено на рисунке 5.8.

 Рисунок 5.8 – Корректная работа сортировки по цене по убыванию

Тест 9 показал, что при нажатии кнопки «Add to cart» часы будут добавлены в корзину, что изображено на рисунке 5.9.

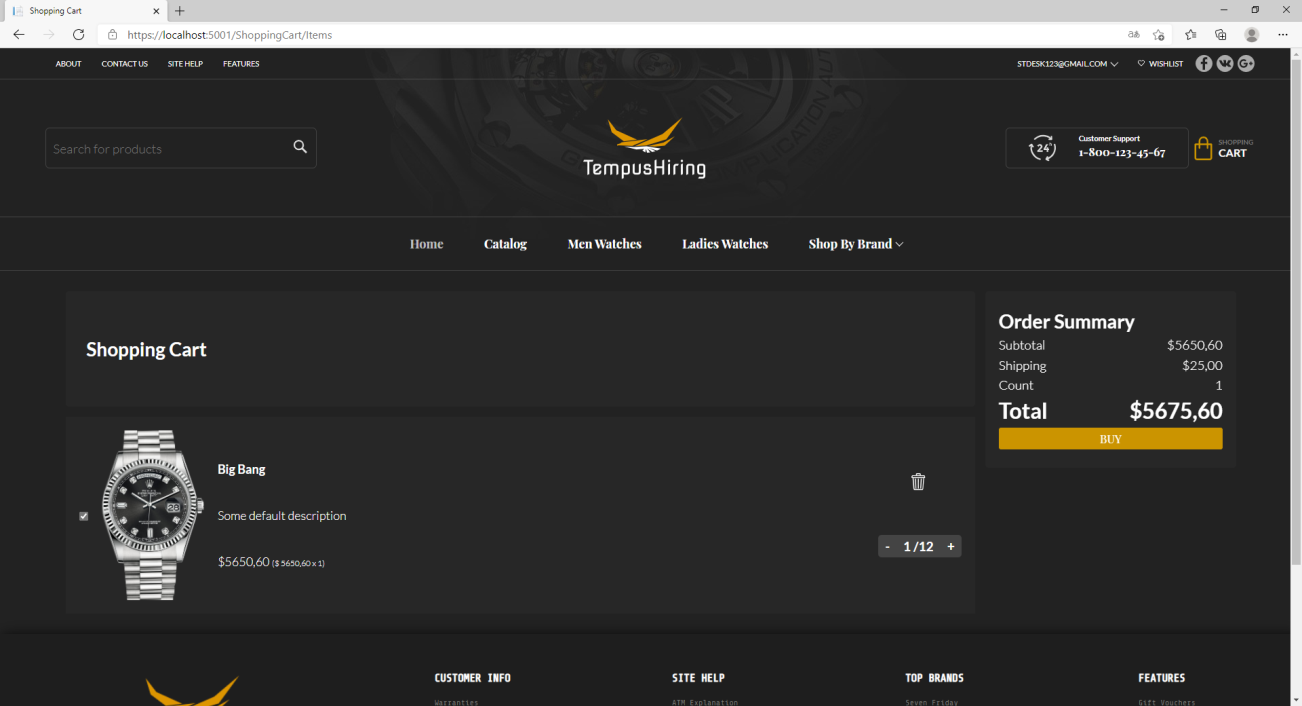


Рисунок 5.9 – Добавление в корзину

Тест 10 показал, что при нажатии кнопки «Buy» в корзине часы будут добавлены в заказ, что изображено на рисунке 5.10.

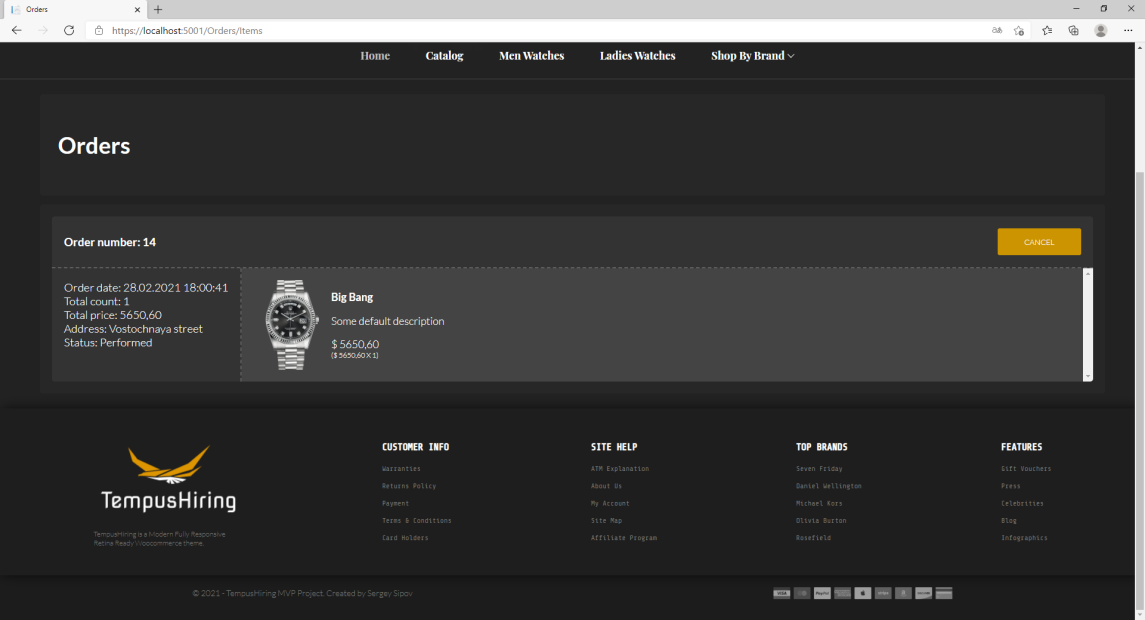


Рисунок 5.10 – Корректное добавление в корзину

Тест 11 показал, что при выходе из профиля, пользователя перенаправило на главную страницу, что изображено на рисунке 5.11.

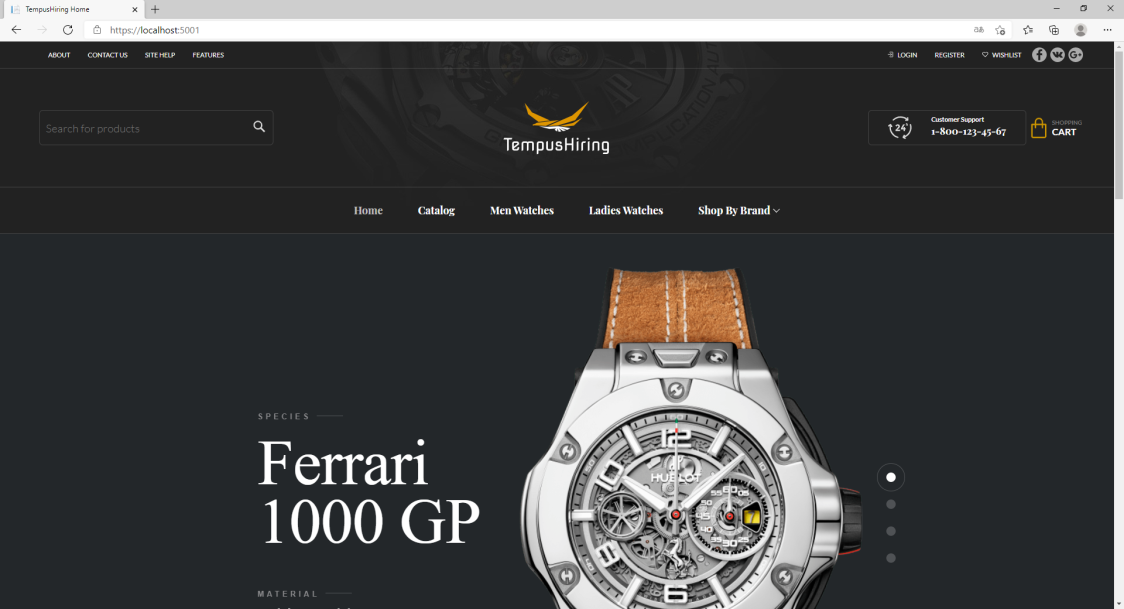


Рисунок 5.11 – Корректный выход из профиля

1. **Описание применениЯ ПРОГРАММЫ**

## 6.1 Назначение программы и условия применения

Данное программное средство предназначено для облегчения работы розничного интернет-магазина по продаже часов.

Минимальные системные требования – это набор условий, необходимых для возможности запуска и работы программного продукта. Однако, наличие минимальных системных требований не отменяет возможность запуска ПО на компьютерах, которые по характеристикам слабее минимальных.

Минимальные системные требования:

1. 1,8 ГГц или более быстрый процессор. Рекомендуется четырехъядерный или лучше.
2. 2 ГБ ОЗУ. Рекомендуется 8 ГБ.
3. Место на жестком диске: от 800 МБ до 5 ГБ свободного места.
4. Скорость жесткого диска: для повышения производительности установите Windows и Visual Studio на твердотельный диск (SSD).
5. Видеокарта, поддерживающая минимальное разрешение экрана 720p.
6. Google Chrome, Яндекс, Microsoft Edge.
   1. **Описание работы программы**

При запуске сайта загружается страница «Home», на которой отображено навигационное меню, кнопка для регистрации и авторизации, кнопка для перехода в корзину, что изображено на рисунке 6.1.



Рисунок 6.1 – Стартовая страница

При нажатии на кнопку «Catalog» происходит переход на страницу с каталогом часов, что изображено на рисунке 6.2.

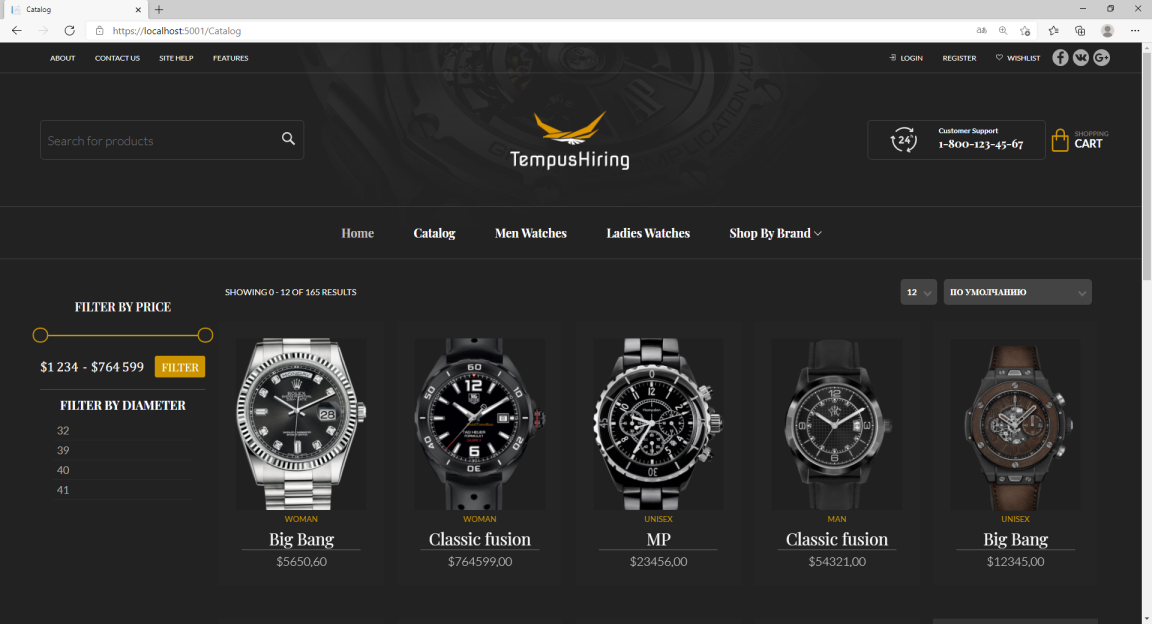


Рисунок 6.2 – Каталог часов

При нажатии на ссылку «Mens Watches» происходит переход на страницу с каталогом мужских часов, что изображено на рисунке 6.3.

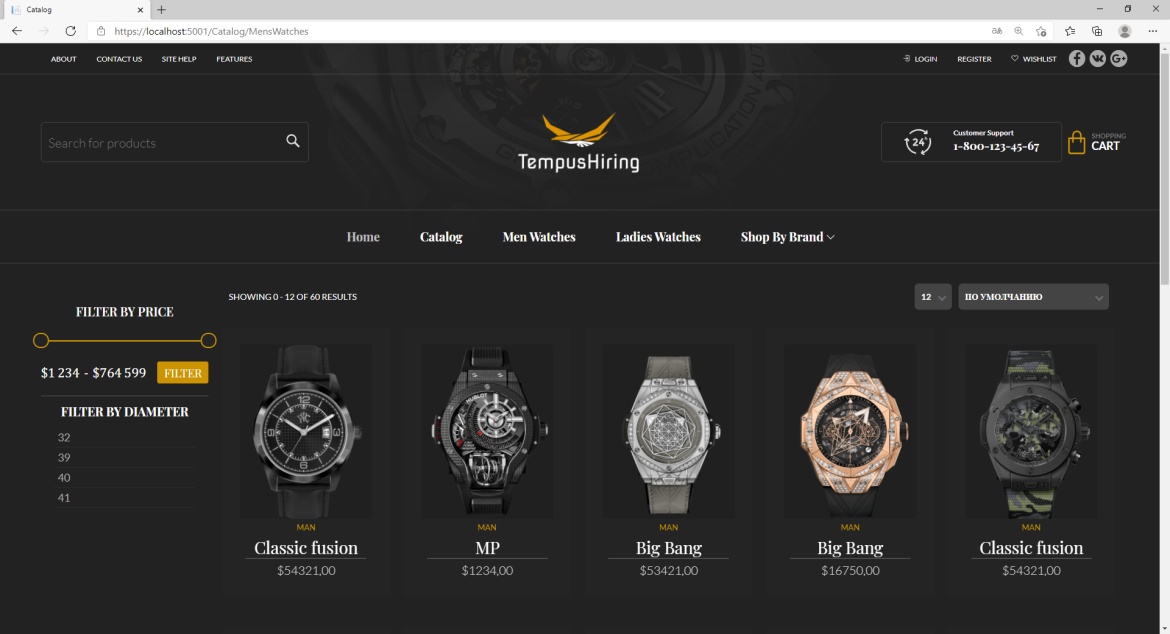


Рисунок 6.3 – Каталог мужских часов

При нажатии на ссылку «Ladies Watches» происходит переход на страницу с каталогом женских часов, что изображено на рисунке 6.4.

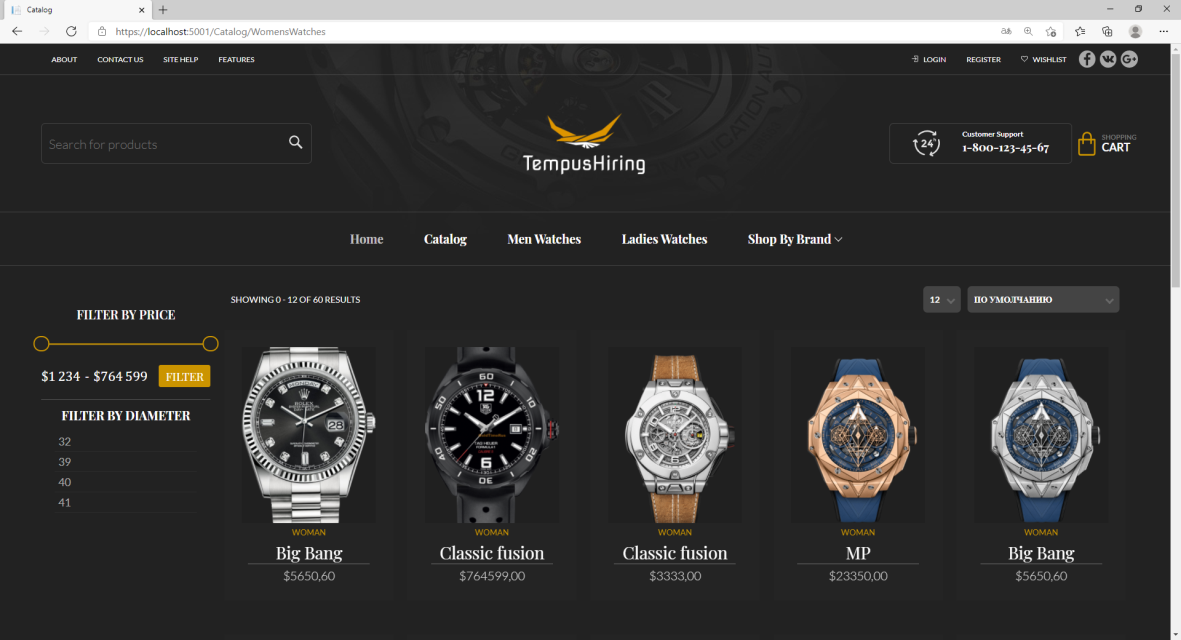


Рисунок 6.4 – Каталог женских часов

При наведении курсора мыши на часы в каталоге, они разворачиваются на 180 градусов, и с обратной стороны будет выведено подробное описание товара и кнопка «Add to cart», при нажатии на которую товар будет добавлен в корзину, что изображено на рисунке 6.5.

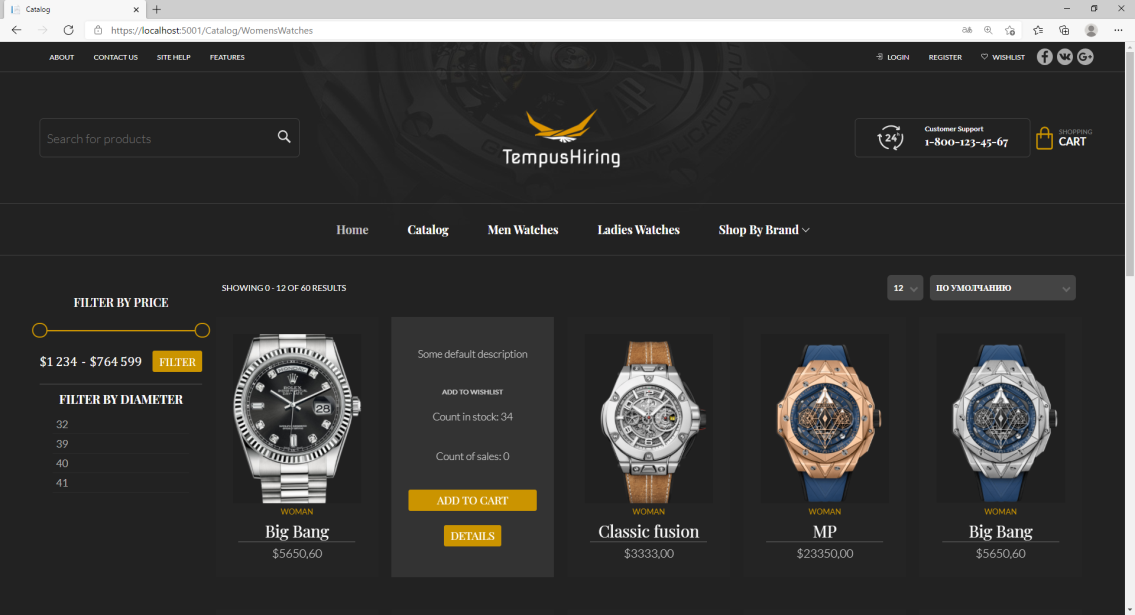


Рисунок 6.5 – Информация о часах

При нажатии на кнопку «По умолчанию» появляется выпадающее меню, в котором можно выбрать параметр сортировки, что изображено на рисунке 6.6.

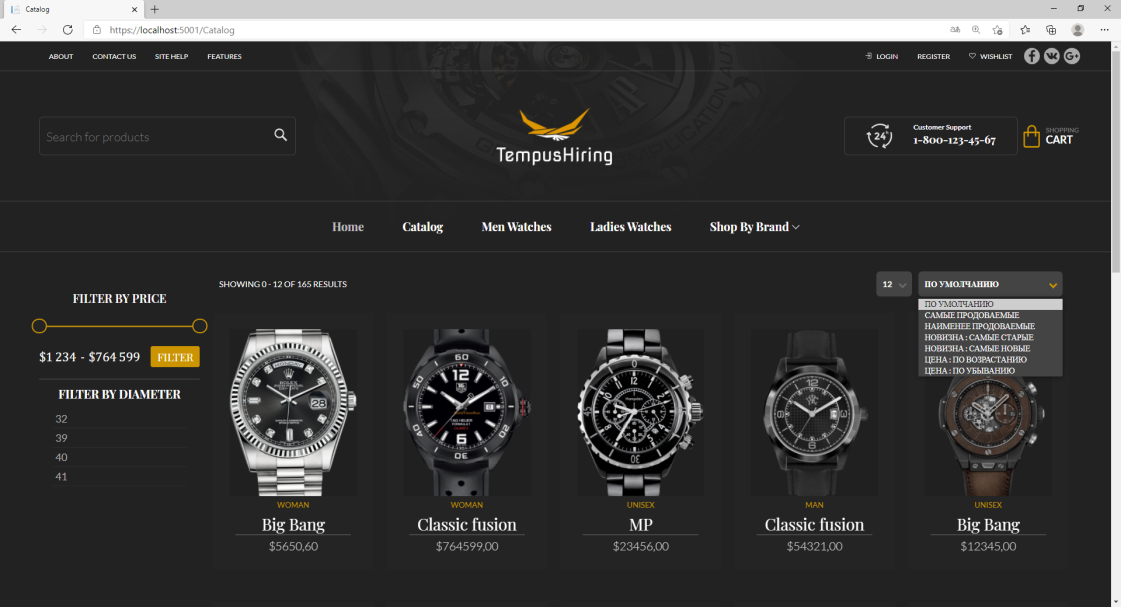


Рисунок 6.6 – Фильтрация в каталоге

При нажатии на кнопку «12» появляется выпадающее меню, в котором можно выбрать количество элементов на странице, что изображено на рисунке  6.7.

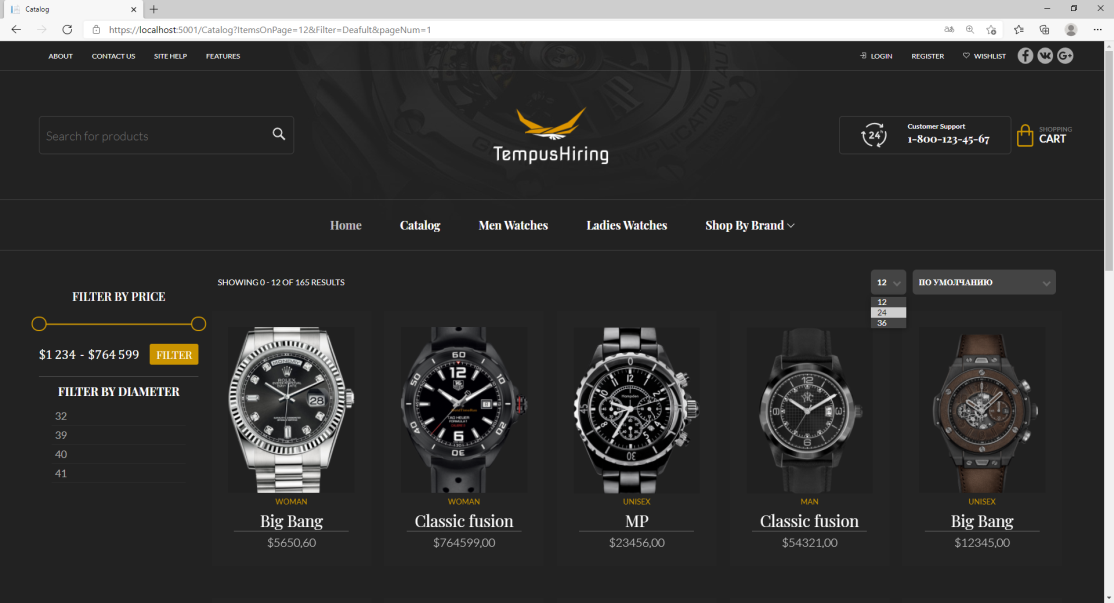


Рисунок 6.7 – Сброс фильтраций

Внизу страницы расположены кнопки для перехода по содержимому, при нажатии на которые будет выгружаться следующий пакет данных, что изображено на рисунке 6.8.

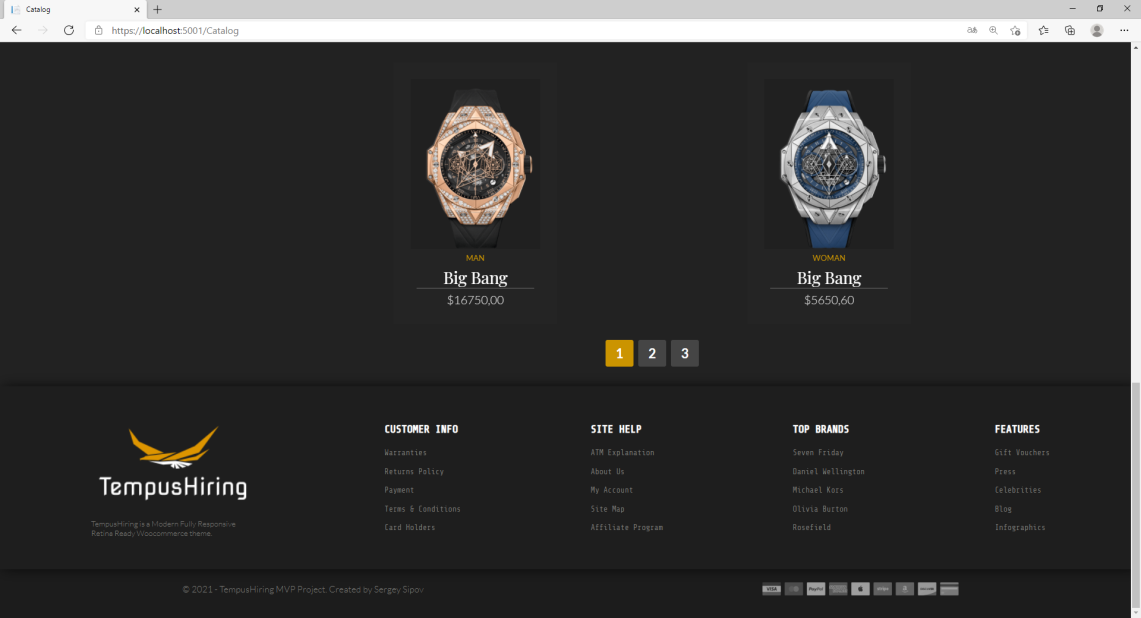


Рисунок 6.8 – Навигация на странице

При нажатии на ссылку «Login» в правом верхнем углу страницы пользователь перейдет на страницу авторизации «Sign In», что изображено на рисунке 6.9.

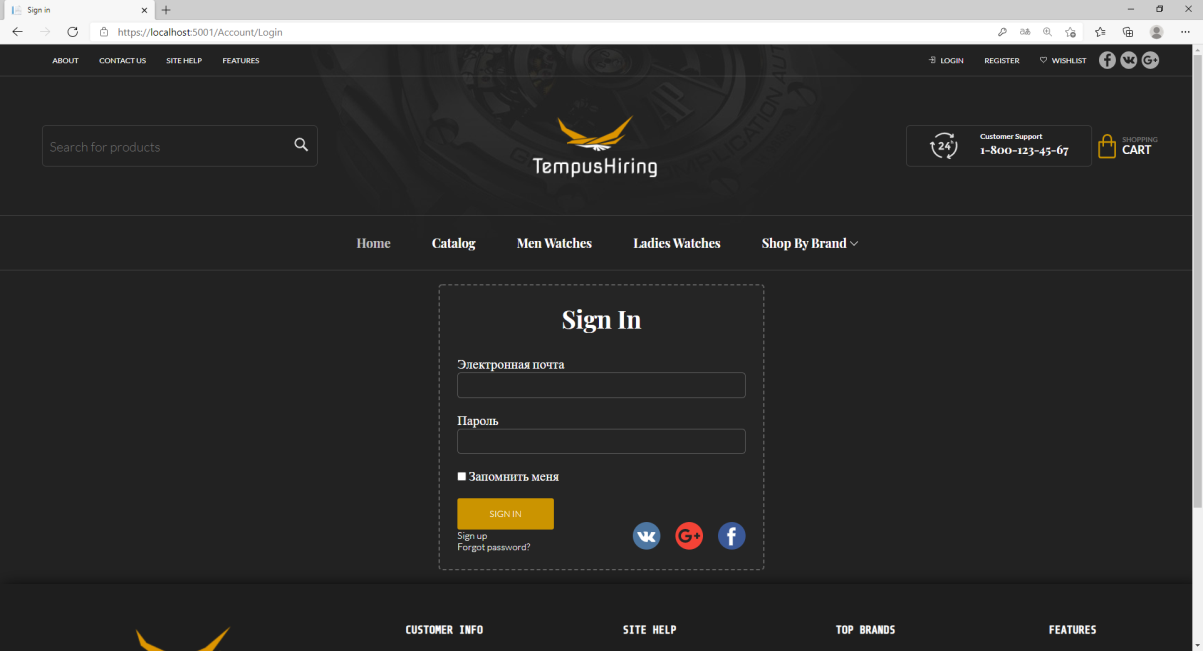


Рисунок 6.9 – Страница авторизации

При нажати на ссылку «Register» пользователь перейдет на страницу реигстрации «Sign Up», что изображено на рисунке 6.10.

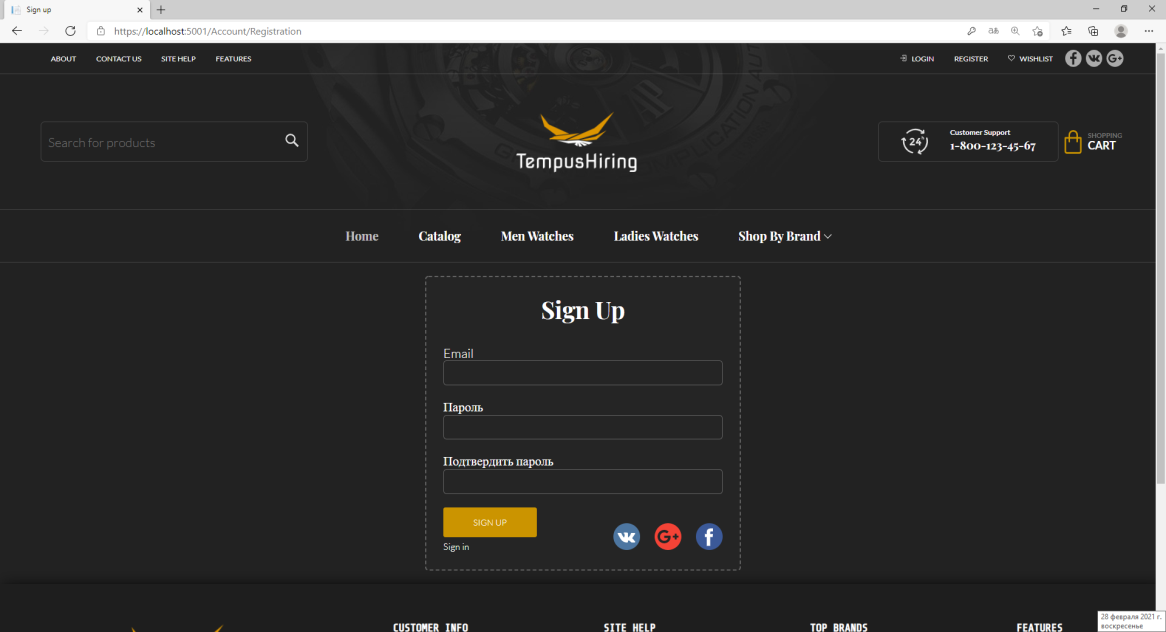


Рисунок 6.10 – Форма для регистрации

В правом нижнем углу формы имеются три иконки, позволяющие войти или зарегистрироваться в системе через сторонние ресурсы, при нажатии на которую произойдет переадресация на соответствующий ресурс, что изображено на рисунке 6.11.

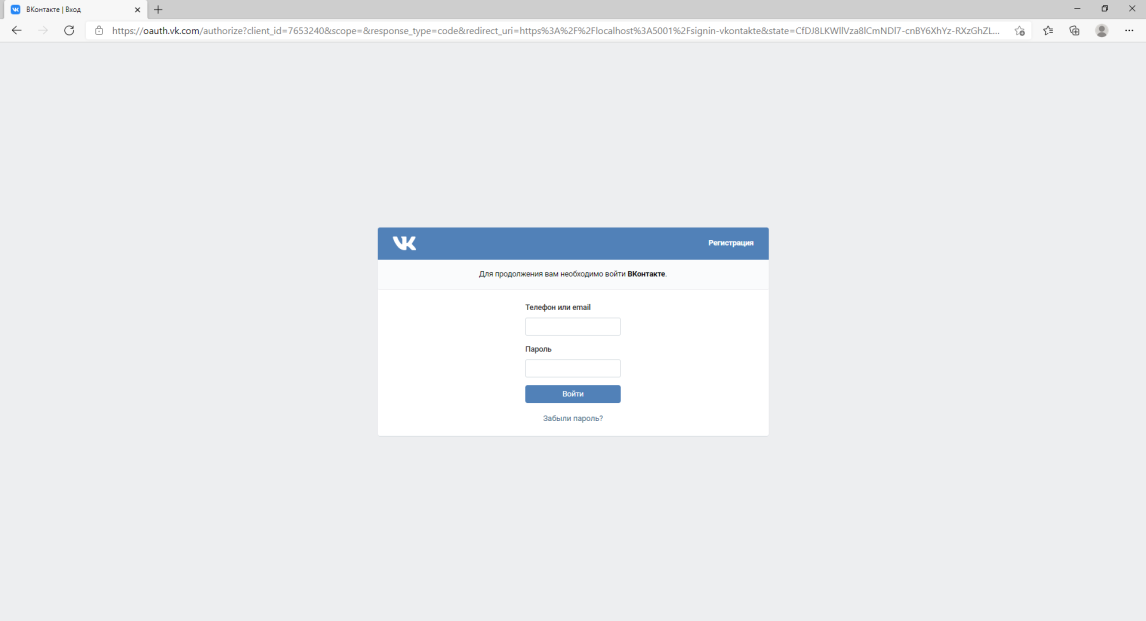


Рисунок 6.11 – Вход через сторонний ресурс

На форме «Sign In» надо заполнить два поля: логин и пароль. Длина пароля должна составлять минимум 8 символов. При нажатии на кнопку «Войти», идет проверка на заполнение полей, если поля не содержат информацию, то возле незаполненного поля будет надпить «Не указан E-mail» или «Не указан пароль», что изображено на рисунке 6.12.

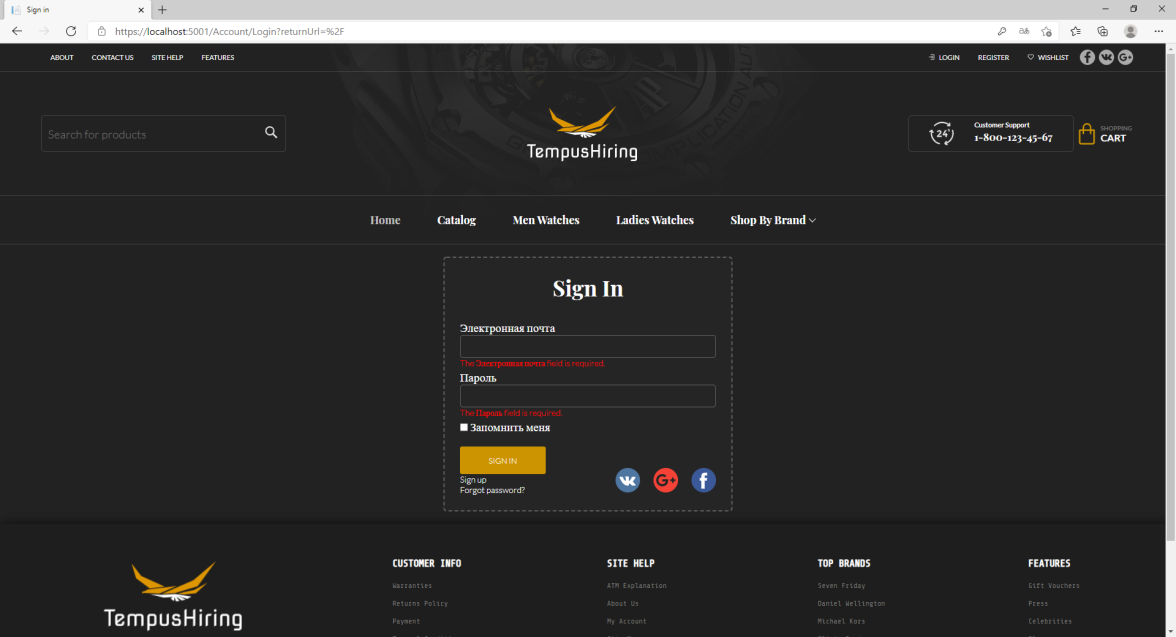


Рисунок 6.12 – Не указаны данные

Если поля корректно заполнены, то проверяется существование пользователя с данным логином и паролем. Если пользователь не существует, по выведется «Некорректный логин и (или) пароль», что изображено на рисунке 6.13.

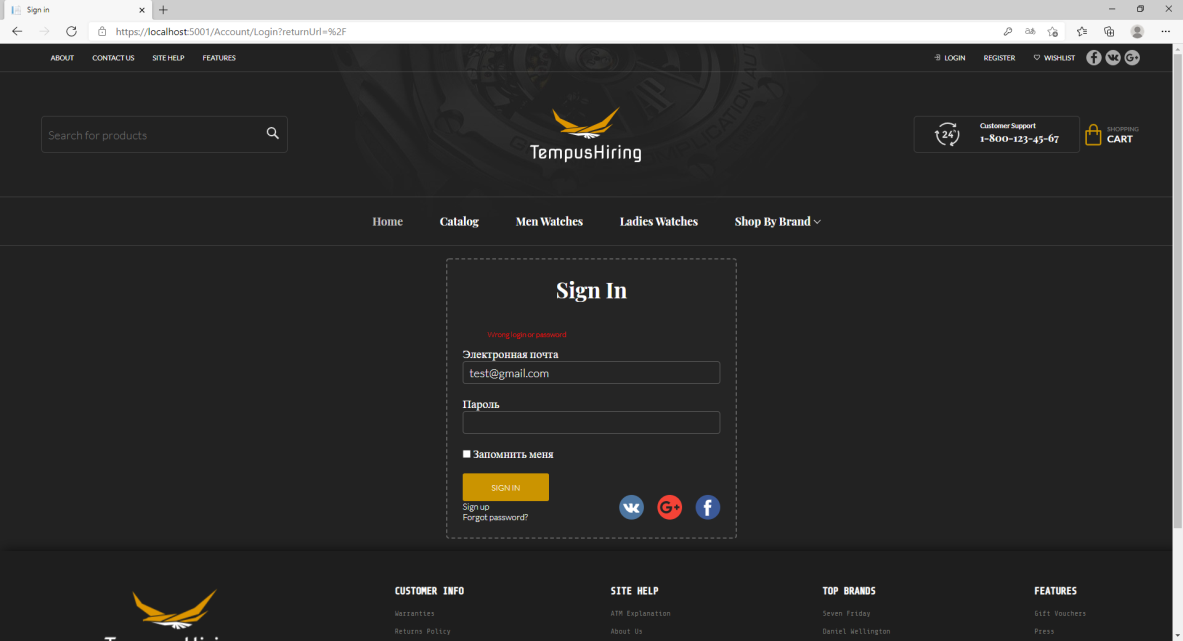


Рисунок 6.13 – Некорректны данные

На форме регистрации «Sign Up» происходит валидация полей, и в случае, если данные введены некорректно будет выведено соответствующее сообщение, что изображено на рисунке 6.14.



Рисунок 6.14 – Некорректные данные

Если пароли не совпадают, будет выведено соответствующее сообщение об ошибке, что изображено на рисунке 6.15.

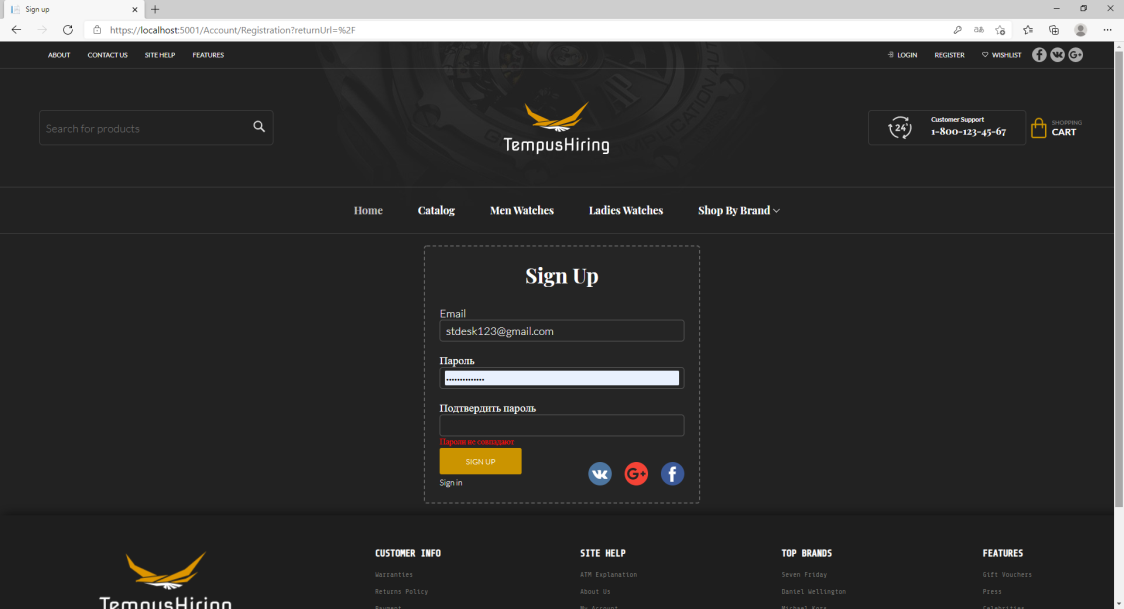


Рисунок 6.15 – Разные пароли

Если Email уже занят, то выведется соответствующее сообщение, что изображено на рисунке 6.16.

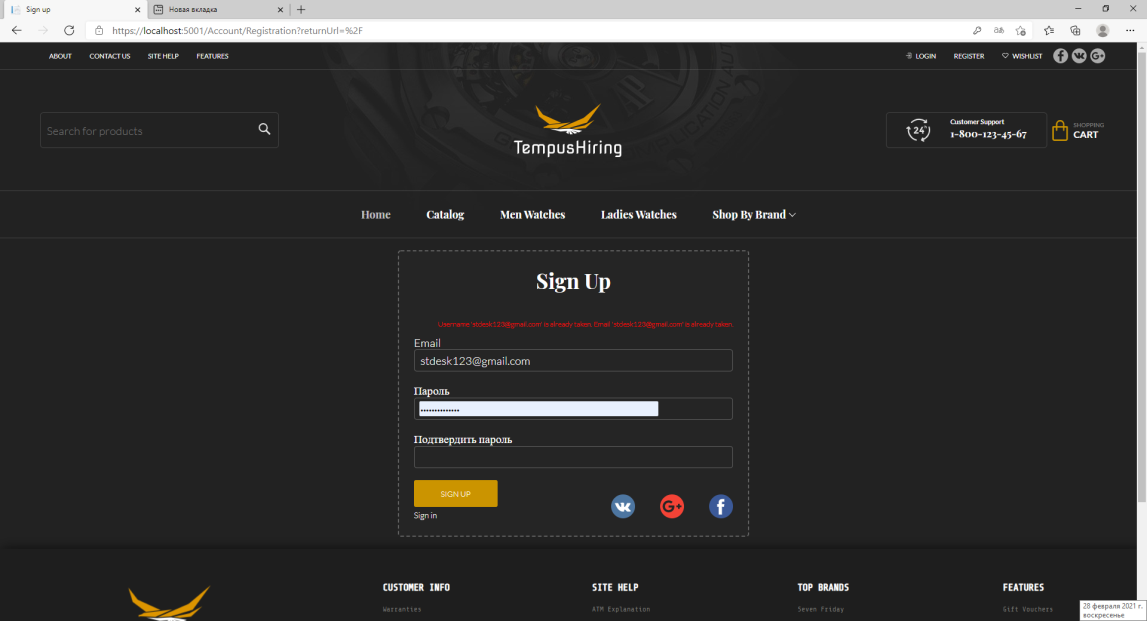


Рисунок 6.16 – Email занят

При успешной регистрации или авторизации пользователь возвращается на страницу, с которой произошел переход, в правом верхнем углу появляется логин пользователя, при наведении на который отображается выпадающий список со ссылками на страницы «ShoppingCart», «Orders» и кнопкой «Exit» для выхода из профиля, что изображено на рисунке 6.17.

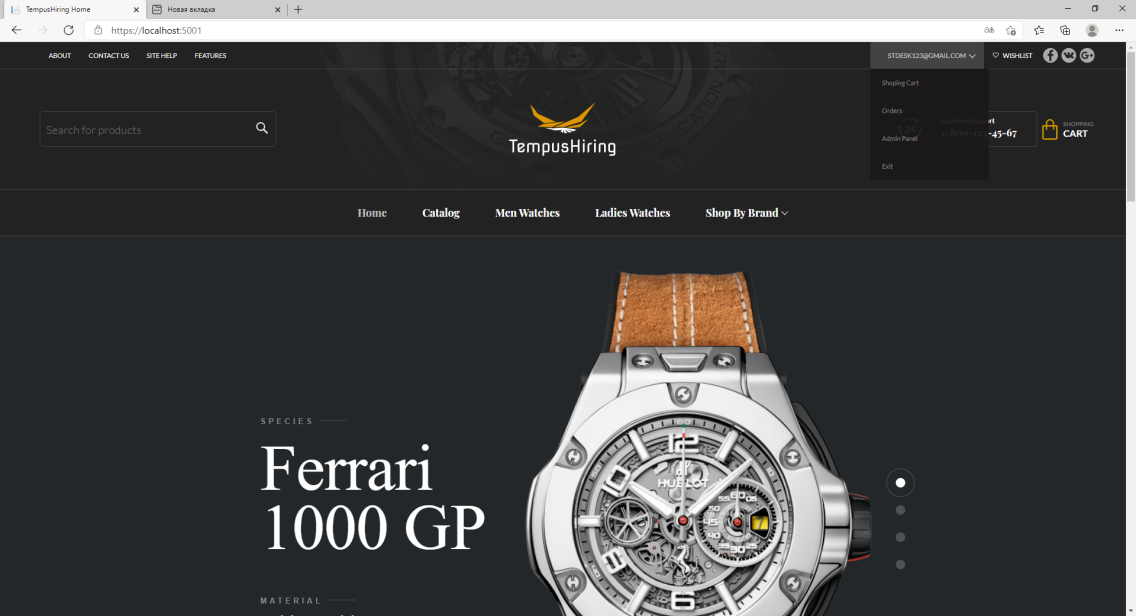


Рисунок 6.17 – Вход авторизованного пользователя

При нажатии на иконку «Корзины» или в выпадающем меню на кнопку «ShoppingCart» для авторизованного пользователя он сможет перейти в корзину для просмотра ее содержимого, что изображено на рисунке 6.18.

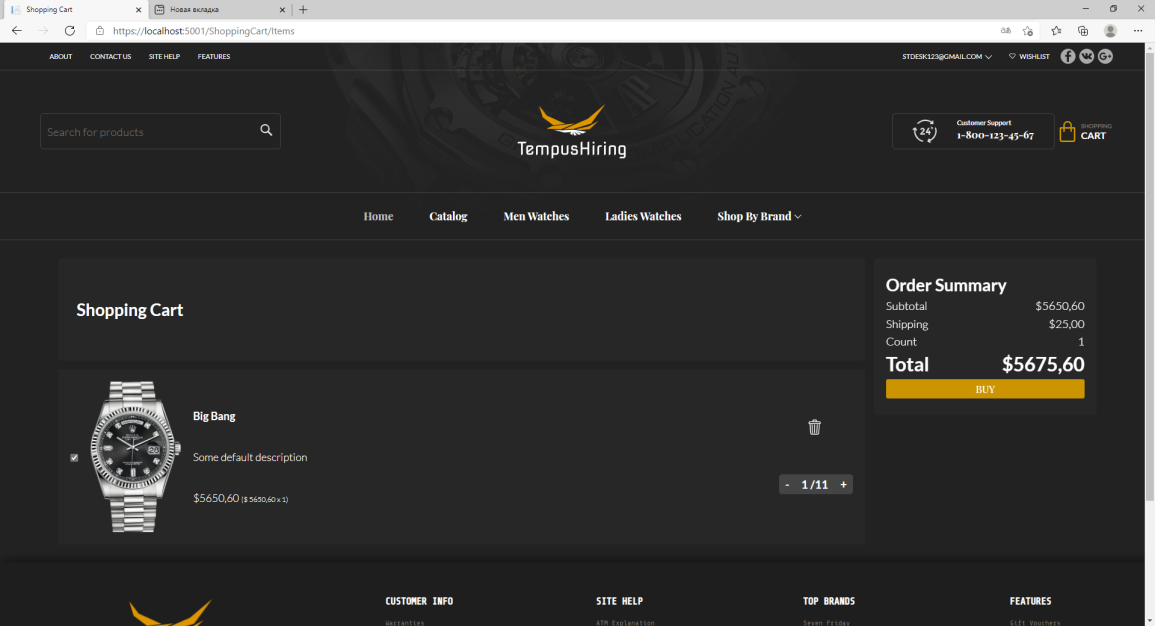


Рисунок 6.18 – Корзина

При выборе товара он должен стать отслеживаемым и его стоимость должна быть выведена в окне «Order Summary», в котором представлена общая стоимость и количество часов, что изображено на рисунке 6.19.

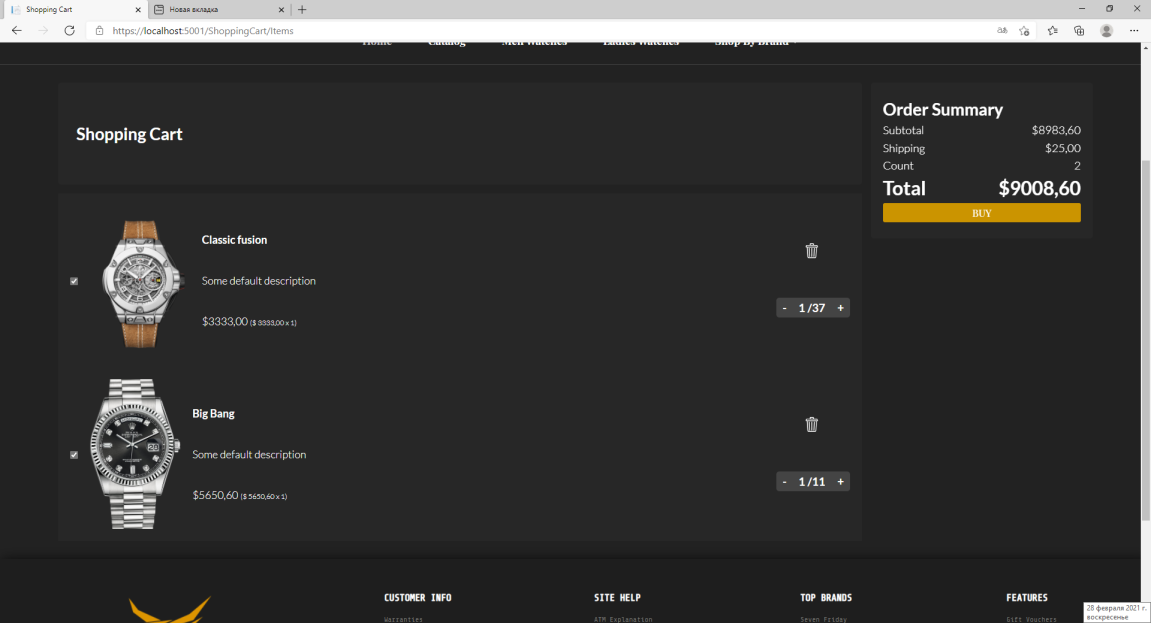


Рисунок 6.19 – Стоимость отслеживаемых товаров в корзине

При нажатии на кнопку «Buy» происходит добавление отслеживаемых товаров в корзину, что изображено на рисунке 6.20.

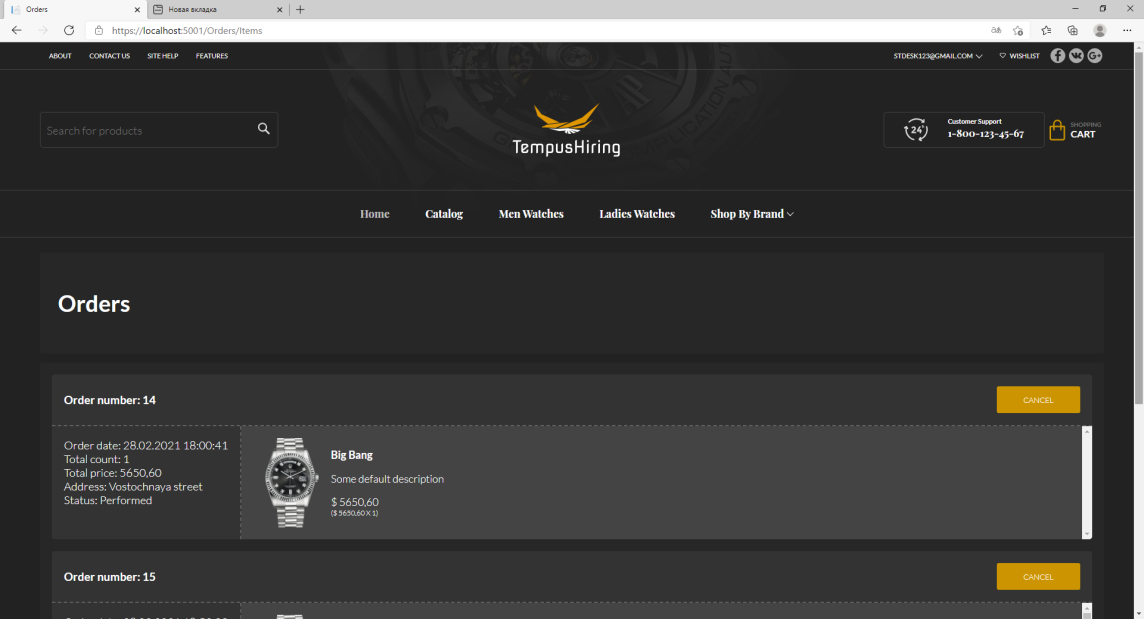


Рисунок 6.20 – Корзина

При нажатии на кнопку «Exit», пользователь выходит из своего профиля, происходит переход на страницу «Home», исчезает логин пользователя с возможностью просмотра выпадающего меню и вместо него появляются ссылки «Login», «Register» что изображено на рисунке 6.21.

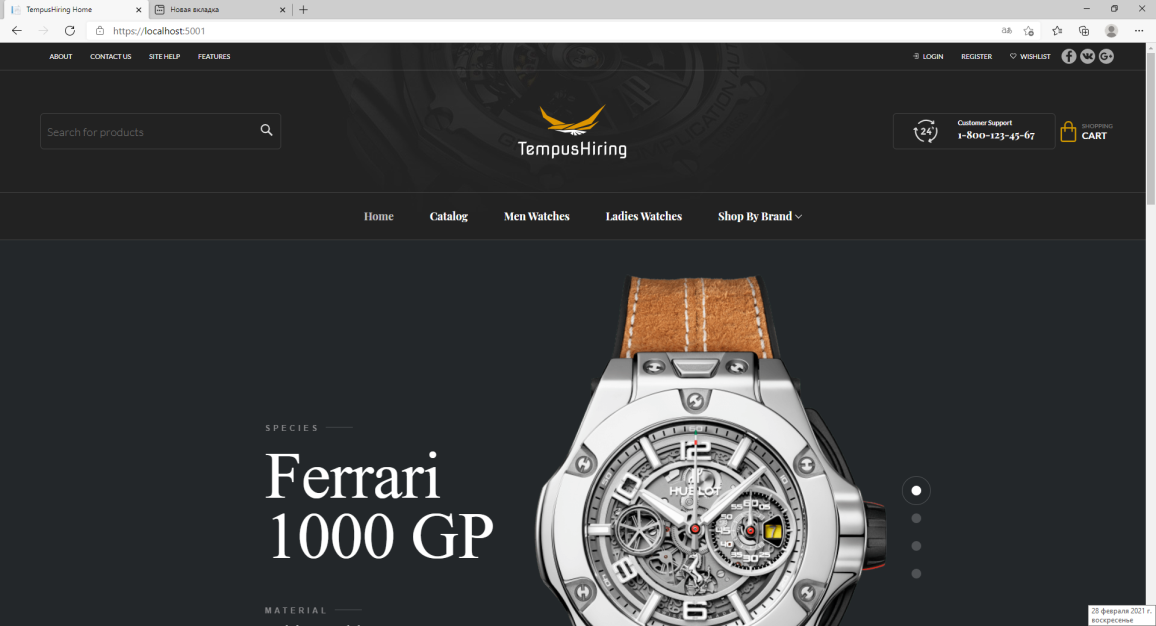


Рисунок 6.21 – Выход из профиля

1. ОХРАНА ТРУДА

## 7.1 Правовые и организационные вопросы охраны труда

Охрана труда – система обеспечения безопасности жизни и здоровья работников в процессе трудовой деятельности, включающая правовые, социально-экономические, организационные, технические, психофизиологические, санитарно-гигиенические, лечебно-профилактические, реабилитационные и иные мероприятия и средства.

Преддипломная практика проходила в научно-производственном республиканском унитарном предприятии «Белорусский государственный институт стандартизации и сертификации» (НПРУП БелГИСС). В данной организации для всех вновь принятых сотрудников или практикантов проводится вводный и первичный инструктажи, выделяется персональное рабочее место с установленными техническими средствами. Общий контроль и руководство по соблюдению правил охраны труда в БелГИСС осуществляет отдельное должностное лицо – инженер по охране труда.

При проведении вводного инструктажа особое внимание уделялось особенностям организации работы в учреждении образования, правилам внутреннего трудового распорядка, требованиям по охране труда.

При проведении первичного инструктажа особое внимание уделялось самостоятельному подключению приборов к электросети. При пользовании средствами вычислительной техники и периферийным оборудованием каждый работник должен внимательно и осторожно обращаться с электропроводкой, приборами, аппаратами и всегда помнить, что пренебрежение правилами безопасности угрожает и здоровью, и жизни человека.

Неоднократно руководителем подчёркивалось, что на своем рабочем месте необходим постоянный контроль за исправным состоянием электропроводки, выключателей, штепсельных розеток, при помощи которых оборудование включается в сеть, и заземления. При обнаружении неисправности нужно немедленно обесточить электрооборудование, оповестить администрацию. Продолжение работы возможно только после устранения неисправности.

В общих правилах обязательных для всех работников БелГИСС запрещается:

– хранить и применять горючие жидкости, взрывчатые вещества, баллоны с газами и др.;

– использовать электронагревательные приборы;

– эксплуатировать провода электроприборов с поврежденной изоляцией;

– пользоваться поврежденными розетками, рубильниками, вилками и прочим электрооборудованием;

– обертывать (накрывать) светильники, бытовые приборы бумагой, тканью и другими горючими материалами;

– применять открытый огонь;

– курить в помещении;

– оставлять без наблюдения включенную в сеть радиоэлектронную аппаратуру, ПЭВМ, оргтехнику, бытовую технику;

– пользоваться неисправной или незаземленной аппаратурой;

– нарушать правила эксплуатации ПЭВМ и оргтехники, а также инструкции по работе на ПЭВМ и средствах оргтехники, действующие в организации;

Среди основных правил, направленных на исключение поражения электрическим током, можно назвать следующие:

– запрещается часто включать и выключать компьютер без необходимости;

– запрещается прикасаться к экрану и к тыльной стороне блоков компьютера;

– не разрешается работать на средствах вычислительной техники и периферийном оборудовании мокрыми руками;

– не разрешается работать на средствах вычислительной техники и периферийном оборудовании, имеющих нарушения целостности корпуса, нарушения изоляции проводов, неисправную индикацию включения питания, с признаками электрического напряжения на корпусе;

– запрещается класть на средства вычислительной техники и периферийные устройства посторонние предметы.

* 1. Производственная санитария и гигиена труда

В НПРУП БелГИСС, на моем рабочем месте соблюдались следующие правила для оборудования:

Помещение, в котором я работал имело естественное и искусственное освещение. Оконные проемы были оборудованы регулируемыми устройствами: занавесями, внешними козырьками. Мой рабочий стол размещался таким образом, что экран компьютера был ориентирован боковой стороной к световым проемам и естественный свет падал преимущественно слева. Освещенность поверхности моего стола в зоне размещения рабочих документов была 300-500 люкс. Освещение не создавало блики на поверхности моего экрана. Светильники на моем рабочем месте имели непросвечивающий отражатель с защитным углом не менее 40 градусов. Были установлены светильники с рассеивателями и экранирующей решеткой. Отсутствовали на моем рабочем месте периферийные устройства. Проводилась ежедневная влажная уборка и систематическое проветривание после часа работы с компьютером.

Оборудование моего рабочего места соответствовало следующим показателям:

1. Площадь рабочего места составляла не менее 4,5 м2.
2. Расстояние между рабочими столами было не менее 2,0 м, а расстояние между боковыми поверхностями видеомониторов – не менее 1,2 м.
3. Высота рабочей поверхности стола 725 мм.
4. Ширина рабочей поверхности была 1200 мм; глубина 1000 мм.

Конструкция рабочего стула:

1. Ширина и глубина поверхности сиденья 400 мм.
2. Поверхность сиденья с закругленным передним краем.
3. Регулировка высоты поверхности сиденья в пределах 400-550 мм и угол наклона вперед до 15 градусов и назад до 5 градусов.
4. Высота опорной поверхности спинки 300, ширина 380 мм и радиус кривизны горизонтальной плоскости – 400 мм.
5. Угол наклона спинки в вертикальной плоскости в пределах 30 градусов.
6. Регулировка расстояния спинки от переднего края сиденья в пределах 260-400 мм.
7. Стационарные подлокотники длиной 250 мм и шириной 60 мм.

Плоскость экрана монитора располагалась ниже уровня глаз на 15° от горизонтали.

Для исключения воздействия повышенных уровней электромагнитных излучений расстояние между экраном монитора и мной составляло не менее 600 мм.

* 1. Техника безопасности (электробезопасность)

Электробезопасность – это комплекс мероприятий, направленных на обеспечение защиты людей от опасного воздействия электрического тока, который, проходя через весь организм человека, вызывает биологические и физико-химические процессы, которые опасны для человека. Эти процессы, которые чаще всего присущи неживой природе, приводят к необратимым биологическим воздействиям на организм человека.

При пользовании средствами вычислительной техники и периферийным оборудованием каждый работник внимательно и осторожно обращаться с электропроводкой, приборами и аппаратами и всегда помня, что пренебрежение правилами безопасности угрожает его здоровью и жизни.

Во избежание поражения электрическим током в организации применяться современное оборудование, предназначенное для безопасного пользования работником.

Вся оргтехника была оборудована бесперебойными источниками питания. При отключении основного электропитания источники бесперебойного питания позволяют избежать потери информации.

Еженедельно проводился комплексный осмотр оборудования на внешние повреждения, а именно следующие технические мероприятия:

– производство отключений;

– проверка отсутствия напряжения;

– наложение заземлений.

Для избежания повреждения изоляции проводов и возникновения коротких замыканий не разрешалось:

– вешать что-либо на провода;

– закрашивать и белить шнуры и провода;

– закладывать провода и шнуры за газовые и водопроводные трубы, за батареи отопительной системы;

– выдергивать штепсельную вилку из розетки за шнур, усилие должно быть приложено к корпусу вилки.

Для исключения поражения электрическим током запрещалось:

– часто включать и выключать компьютер без необходимости;

– прикасаться к экрану и к тыльной стороне блоков компьютера;

– работать на средствах вычислительной техники и периферийном оборудовании мокрыми руками;

– работать на средствах вычислительной техники и периферийном оборудовании, имеющих нарушения целостности корпуса, нарушения изоляции проводов, неисправную индикацию включения питания, с признаками электрического напряжения на корпусе;

– класть на средства вычислительной техники и периферийные устройства посторонние предметы.

Запрещалось под напряжением очищать от пыли и загрязнения электрооборудование.

Запрещалось проверять работоспособность электрооборудования в неприспособленных для эксплуатации помещениях с токопроводящими полами, сырых, не позволяющих заземлить доступные металлические части. Ремонт электроаппаратуры производится только специалистами-техниками с соблюдением необходимых технических требований.

Для избежания поражения электрическим током, при пользовании электроприборами нельзя касаться одновременно каких-либо трубопроводов, батарей отопления, металлических конструкций, соединенных с землей.

При пользовании электроэнергией в сырых помещениях требовалось соблюдать особую осторожность. При обнаружении оборвавшегося провода немедленно сообщить об этом администрации, принять меры по исключению контакта с ним людей.

* 1. **Пожарная безопасность**

Среди возможных причин возникновения пожара при эксплуатации ПК (возникновение аварийных ситуаций) можно назвать следующие:

– короткие замыкания;

– перегрузки;

– повышение переходных сопротивлений в электрических контактах;

– перенапряжение;

– возникновение токов утечки.

В НПРУП БелГИСС принимаются меры, соблюдение которых помогает исключить с большой вероятностью возможность возникновения пожара:

1. Для понижения воспламеняемости и способности распространять пламя кабели были покрыты огнезащитным покрытием.
2. При ремонтно-профилактических работах строго соблюдались правила пожарной безопасности.
3. Каждое из помещений, где производилась эксплуатация устройств ПК, было оборудовано первичными средствами пожаротушения и обеспечено инструкциями по их применению. В качестве средств пожаротушения были представлены углекислотные огнетушители типа ОУ-2, ОУ-5, а также порошковый тип. Применение пенных огнетушителей не допускалось, так как жидкость пропускает ток.
4. Устройства ПК устанавили вдали отопительных и нагревательных приборов (расстояние не менее 1 м и в местах, где не затруднена их вентиляция и нет прямых солнечных лучей).

Средства тушения пожара, предназначенных для локализации небольших загораний в помещениях с ПК:

– пожарные стволы;

– огнетушители;

– сухой песок;

– асбестовые одеяла.

Мой рабочий кабинет был оборудован специальными противопожарными извещателями. Для профилактики действий при пожаре был разработан специальный план эвакуации из помещения.

Применение автоматических средств обнаружения пожаров является одним из основных условий обеспечения пожарной безопасности на производстве, так как позволяет своевременно известить о пожаре и принять меры к быстрой его ликвидации. Наиболее надёжной системой извещения о пожаре является электрическая пожарная сигнализация, которая бывает автоматической и ручной.

В НПРУП БелГИСС применяются автоматические средства обнаружения пожаров. В качестве такого средства выступает электрическая пожарная сигнализация. Такая система включает: извещатели, линии связи, приемную станцию (коммутатор), источник питания, звуковые и звуковые средства сигнализации.

Для помещения с ПЭВМ были выбраны дымовые пожарные извещатели в количестве 1 штуки. Количество извещателей выбирается исходя из площади помещения, высоты потолков и требований по установке извещателей, для высоты потолка до 3.5м

Помещение содержалось в чистоте. Коридоры, лестничные клетки, двери эвакуационных выходов, подходы к средствам тушения всегда были свободны и ничем не загромождены.

Мебель в помещении не препятствовало быстрой эвакуации людей. Расположение электрических кабелей и различных проводов исключало их повреждение, поражение работников электрическим током, а также они не мешали передвижению по помещению.

В НПРУП БелГИСС в помещениях с наличием ПЭВМ запрещалось:

– хранить и применять горючие жидкости, взрывчатые вещества, баллоны с газами и др.;

– использовать электронагревательные приборы;

– эксплуатировать провода электроприборов с поврежденной изоляцией;

– пользоваться поврежденными розетками, рубильниками, вилками и прочим электрооборудованием;

– обертывать (накрывать) светильники, бытовые приборы бумагой, тканью и другими горючими материалами;

– применять открытый огонь;

– курить в помещении;

– оставлять без наблюдения включенную в сеть радиоэлектронную аппаратуру, ПЭВМ, оргтехнику, бытовую технику;

– пользоваться неисправной или незаземленной аппаратурой;

– нарушать правила эксплуатации ПЭВМ и оргтехники, а также инструкции по работе на ПЭВМ и средствах оргтехники, действующие в организации;

– включать в сетевые фильтры, блоки бесперебойного питания и специализированные розетки, расположенные в коробах бытовую технику и другое, не относящееся к ПЭВМ оборудование.

По завершению работы требовалось обесточить все электроприборы и осмотреть помещение на наличие признаков возгорания.

# ЭКОНОМИЧЕСКИЙ РАЗДЕЛ

## Определение общей трудоемкости разработки программного обеспечения

Данный раздел содержит характеристики разработки, расчет затрат на разработку и расчет экономического эффекта.

Целью данного раздела является определение экономического эффекта от создания нового программного средства. Разработка программного средства предусматривает проведение всех стадий проектирования.

На основании информации о функциях разрабатываемого программного обеспечения по каталогу функций определяется общий объем программного обеспечения. В зависимости от организационных и технологических условий, в которых разрабатывается программное обеспечение (ПО), исполнители по согласованию с руководством организации могут уточнять (корректировать) объем на основе экспертных оценок.

На основании принятого к расчету объема и категории сложности ПО определяется нормативная трудоемкость ПО (TЕр) выполняемых работ.

Нормативная трудоемкость ПО (TЕр) выполняемых работ разработки может корректироваться, при необходимости, с учетом коэффициентов: повышения сложности ПО (Кс), учитывающих новизну ПО (Кн), учитывающих степень использования стандартных модулей (Кт), средства разработки ПО (Кур). Рабочая трудоемкость определяется по формуле (8.1):

, (8.1)

где ТЕн – нормативная трудоемкость разработки программного продукта, определяемая относительно общего количества строк исходного кода, чел.-дней;

Кслож – коэффициент сложности;

Кнов – коэффициент, учитывающий степень новизны;

Кт – коэффициент, учитывающий использование стандартных модулей;

Кур – коэффициент, учитывающий средства разработки.

(дн.)

## Определение затрат на оплату труда специалистов

Основной статьей расходов на создание программного обеспечения является заработная плата разработчиков (исполнителей) проекта, в число которых принято включать инженеров-программистов, участвующих в написании кода, руководителей проекта, системных архитекторов, дизайнеров, разрабатывающих пользовательский интерфейс, отдел тестирующих специалистов и других специалистов, необходимых для решения специальных задач в команде. Заработная плата руководителей организации и работников вспомогательных служб (инфраструктуры) учитывается в накладных расходах. Общая трудоемкость, плановая численность работников и плановые сроки разработки ПО являются базой для расчета основной заработной платы разработчиков проекта. Трудоёмкость разработки проекта в человеко-часах (ТЕр) в расчёте средней продолжительности рабочего дня в 8 часов составит 243,52 часов. Основная заработная плата исполнителей на конкретное программное обеспечение рассчитывается по формуле (8.2):

, (8.2)

где ТСч – часовая тарифная ставка соответствующего разряда, руб.;

ТЕр – трудоемкость разработки, часов;

Кпр – коэффициент премирования.

(руб.)

Дополнительная заработная плата за конкретное ПО (ЗПдоп) включает выплаты, предусмотренные законодательством о труде (оплата отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат, не связанных с основной деятельностью исполнителей), и определяется по нормативу в процентах к основной заработной плате, согласно формуле (8.3):

, (8.3)

где ЗПосн – основная заработная плата исполнителя, руб.;

%ЗПдоп – процент дополнительной заработной платы, %.

(руб.)

Обязательным условием является определение отчислений в фонд социальной защиты населения согласно формуле (8.4):

, (8.4)

где ЗПосн – основная заработная плата исполнителей, руб.;

ЗПдоп – дополнительная заработная плата исполнителей, руб.;

%ФСЗН – установленный законодательством процент отчисления в фонд социальной защиты населения, %.

## Определение полной себестоимости программного обеспечения

Расходы на материалы и комплектующие определяются на основании сметы затрат, разрабатываемой на программное обеспечение с учетом действующих нормативов. В данные затраты включают расходы на магнитные носители, бумагу, красящие ленты и другие материалы, необходимые для разработки программного обеспечения. Нормы расхода материалов в суммарном выражении определяются в расчете на 100 строк исходного кода или по нормативу в процентах основной заработной платы разработчиков, который устанавливается организацией.

Сумма затрат на расходные материалы рассчитывается по формуле (8.5):

, (8.5)

где ЗПосн – основная заработная плата исполнителей, руб.;

Нм – норматив расхода на материалы, %.

(руб.)

Расходы по статье «Машинное время» включают оплату машинного времени, необходимого для разработки и отладки ПО, которое определяется по нормативам (в машино-часах) на 100 строк исходного кода машинного времени в зависимости от характера решаемых задач и типа ПК. Расчет производится по формуле (8.6):

, (8.6)

где Цм– цена одного машино-часа, руб.;

Vстрок– общий объем ПО (количество строк исходного кода);

Нмв– норматив расхода машинного времени на отладку 100 строк исходного кода (машино-часов).

(руб.)

Общехозяйственные расходы связаны с необходимостью содержания аппарата управления, вспомогательных хозяйств и опытных (экспериментальных) производств, а также с расходами на общехозяйственные нужды, и относятся на конкретное программное обеспечение по нормативу в процентном отношении к основной заработной плате исполнителей. Норматив устанавливается в целом по организации (30-80 %) и рассчитывается по формуле (8.7):

, (8.7)

где % ОХР – норматив общехозяйственных расходов, %.

(руб.)

Прочие прямые затраты – это затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы для упрощения работы. Они определяются по нормативу, разрабатываемому в целом по организации, в процентах к основной заработной плате и рассчитывается по формуле (8.8):

, (8.8)

где Нп.з. – норматив прочих прямых затрат, %.

(руб.)

Результаты расчета информации о затратах внесем в таблицу 8.1.

Таблица 8.1 – Расчет полной себестоимости программного обеспечения.

|  |  |
| --- | --- |
| Наименование статей затрат | Сумма, руб. |
| Материальные затраты |  |
| Основная заработная плата |  |
| Дополнительная заработная плата |  |
| Отчисления в ФСЗН |  |
| Машинное время |  |
| Общехозяйственные расходы |  |
| Прочие прямые затраты |  |
| Полная себестоимость | 2102,18 |

* 1. **Определение цены заказанного программного обеспечения**

Для определения цены ПО необходимо рассчитать плановую прибыль.

Рентабельность и прибыль создаваемого ПП определяется исходя из результатов анализа рыночных условий, переговоров с заказчиком и согласования с ним отпускной цены.

Прибыль рассчитывается по формуле (8.10):

, (8.10)

где С/Сполн– полная себестоимость ПО, руб;

Ппл – плановая прибыль от реализации ПО, руб.;

Р – уровень рентабельности ПО, %.

(руб.)

Отпускная цена без НДС определяется по формуле (8.11):

, (8.11)

(руб.)

Отпускная цена ПО с НДСрассчитывается по формуле (6.12):

, (8.12)

где НДС – сумма налога на добавленную стоимость, руб.

(руб.)

Сумма НДС определяется по формуле (8.13):

, (8.13)

где % НДС – ставка НДС, %.

(руб.)

Полученные результаты внесем в таблицу 8.2.

Таблица 8.2 – Расчет отпускной цены программного обеспечения

|  |  |
| --- | --- |
| Показатели | Сумма, руб. |
| Полная себестоимость | 2102,18 |
| Прибыль |  |
| Цена без НДС |  |
| НДС |  |
| Отпускная цена с НДС |  |

Экономическим эффектом от создания программного обеспечения для разработчика является чистая прибыль. Она определяется после уплаты налога на прибыль, формула (8.14):

, (8.14)

где %Нпр – ставка налога на прибыль, %.

(руб.)

Затраты на разработку интернет-магазина «TempusHiring» составляют 2102,18 руб. Отпускная прогнозируемая цена составляет руб. Прибыль разработчика составит 344 руб. Рассчитанная цена разработанного программного обеспечения позволит полностью покрыть все затраты.

**Заключение**

Одной из основных проблем, с которыми сталкиваются магазины розничной торговли (и магазины по продаже часов в частности) является нежелание или невозможность человека потратить необходимое количество сил, времени и энергии для того, чтобы отправиться в магазин и подобрать для себя часы, которые соответствуют желаемым параметрам, имея при этом вероятность значительно выше нуля вовсе не найти желаемое и потратить впустую вышеуказанные невосполнимые ресурсы, и вдобавок к этому получить еще и испорченное настроение от неудавшейся покупки. Оставлять негативные эмоции у своих покупателей – это крайне негативный фактор для повышения продаж.

Для решения данной проблемы был разработан интернет-магазин «TempusHiring», основной задачей которого является предоставление максимально полной и точной информации потенциальному клиенту в кратчайшие сроки, где бы он не находился.

При выполнении дипломного проекта:

1. Были систематизированы знания в области программирования с использованием языка С# и SQL.
2. Изучены паттерны MVC, Repository, UnitOfWork.
3. Углублены знания, умения и навыки в области разработки программ в среде Visual Studio 2019 и Microsoft SQL Server Management Studio 18.
4. Сформированы умения и навыки самостоятельной организации исследовательской работы.
5. Изучены современные методы поиска, обработки и использования информации.
6. Углублены и систематизированы знания в области проектирования архитектуры базы данных.
7. Изучена и проанализирована соответствующая литература.
8. Был проведён сравнительный анализ языков этого класса и показаны преимущества и недостатки языка C#.
9. Углублены знания и умения в использовании технологии построения клиентских приложений ASP NET Core.

Программа была протестирована с использованием 11 тестов. Все тесты показали, что программа работает корректно, как в нормальных, так и в экстремальных условиях.

В разделе «Охрана труда» перечислены и описаны правовые и организационные вопросы охраны труда, а также перечислены санитарно-гигиенические требования к безопасности труда в учреждении образования.

После разработки был осуществлен экономический расчет, в котором была подсчитана прогнозируемая отпускная цена с НДС руб. Затраты на разработку составили 2102,18 руб., а прибыль разработчика – 344 руб.

**Список использованных источников**

1. Джепикс, Ф. Язык программирования С# 7 и платформы .Net и .Net Core 8-издание / Ф. Джепикс, Э. Троелсен. – Москва-Санкт-Петербург, 2019. – 1328 с.
2. CODE BLOG – Программирование C# [Электронный ресурс] / Преимущества и недостатки С#. Режим доступа: [https://shwanoff.ru/plus-minus-c-sharp](https://shwanoff.ru/plus-minus-c-sharp/)/. – Дата доступа: 12.11.2020.
3. Depix – комплексные услуги интернет-маркетинга в Москве [Электронный ресурс] / Система управления базами данных MsSQL. Режим доступа: [https://depix.ru/articles/sistema\_upravleniya\_bazami\_dannyh\_mysql](https://depix.ru/articles/sistema_upravleniya_bazami_dannyh_mysql/)/. –Дата доступа: 12.11.2020.
4. Все новости Москвы [Электронный ресурс] / Что такое SQL Server: плюсы и минусы использования. Режим доступа: <http://muzeon.ru/medicina/2912-chto-takoe-sql-server-plyusy-i-minusy-ispolzovaniya.html>/. – Дата доступа: 18.11.2020.
5. Мой первый компьютер [Электронный ресурс] / Windows 10. Минусы и плюсы операционной системы. Режим доступа: <https://myfirstcomp.ru/operation_systems/windows-10-minusy-i-plyusy-operacionnoj-sistemy/>. – Дата доступа: 12.11.2020.
6. Студенческая библиотека онлайн [Электронный ресурс] / Описание среды разработки Microsoft Visual Studio. Режим доступа: <https://studbooks.net/2258619/informatika/opisanie_sredy_razrabotki_microsoft_visual_studio>/. – Дата доступа: 14.11.2020
7. Вадим Строганов, Ruby on Rails разработчик [Электронный ресурс] / Язык программирования С#. Режим доступа: <https://learn-code.ru/yazyki-programmirovaniya/c-sharp>/. – Дата доступа: 12.11.2020

# Приложение А

(обязательное)

Листинг программы

public class Startup

{

public IConfiguration Configuration { get; }

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public void ConfigureServices(IServiceCollection services)

{

var connectionString = Configuration.GetConnectionString("DefaultConnection");

services.AddDbContext<DbContext, WineDBContext>(options =>

options.UseSqlServer(connectionString)

.UseQueryTrackingBehavior(QueryTrackingBehavior.NoTracking));

services.AddAutoMapper(typeof(MyProfile));

services.AddBLL();

services.AddUnitOfWokr();

services.AddControllers()

.AddNewtonsoftJson(options => options.SerializerSettings.PreserveReferencesHandling = PreserveReferencesHandling.Objects);

services.AddCors(options => options.AddDefaultPolicy(builder => builder.WithOrigins("https://localhost:44397")));

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Home/Error");

app.UseHsts();

}

app.UseCors();

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseEndpoints(endpoints => endpoints.MapControllers());

}

}

public static class BLLSetviceProviderExtensions

{

public static void AddBLL(this IServiceCollection services)

{

services.AddScoped<IClientDTOServices, ClientServices>();

services.AddScoped<IManufacturerDTOServices, ManufacturerServices>();

services.AddScoped<ITastingDTOServices, TastingServices>();

services.AddScoped<IWineDTOServices, WineServices>();

services.AddScoped<IAccount, AccountServices>();

services.AddScoped<IWineRatingDTOServices, WineRatingDTOServices>();

services.AddScoped<ITastingClientDTOServices, TastingClientDTOServices>();

services.AddScoped<IGrapeSortDTOService, GrapeSortServices>();

services.AddScoped<IWineTastingServices, WineTastingServices>();

services.AddScoped<IWineGrapeSortServices, WineGrapeSortServices>();

}

}

public static class UnitOfWokrServiceProviderExtensions

{

public static void AddUnitOfWokr(this IServiceCollection services)

{

services.AddScoped<IUnitOfWork, EFUnitOfWork>();

}

}

[ApiController]

[Route("[controller]/[action]")]

public class MyApiController : ControllerBase

{

}

public class AccountApiController : MyApiController

{

private readonly IClientDTOServices clientServices;

private readonly IAccount account;

private readonly IMapper mapper;

public AccountApiController(IClientDTOServices services, IAccount \_account, IMapper \_mapper)

{

clientServices = services;

mapper = \_mapper;

account = \_account;

}

[HttpPost]

public IActionResult Login([FromBody] LoginModel model)

{

var clientDTO = clientServices.GetAll().FirstOrDefault(x => x.EMail == model.Email && x.PasswordCl == account.HashPassword(model.Password));

if (clientDTO != null)

{

var clientView = mapper.Map<ClientDTO, ClientView>(clientDTO);

return Ok(clientView);

}

return BadRequest();

}

[HttpPost]

public IActionResult Register([FromBody] RegisterModel model)

{

var client = clientServices.GetAll().FirstOrDefault(x => x.EMail == model.EMail);

if (client == null)

{

model.PasswordCl = account.HashPassword(model.PasswordCl);

var newClient = mapper.Map<RegisterModel, ClientDTO>(model);

var id = clientServices.Add(newClient);

if (id != 0)

{

return Ok(id);

}

return NotFound();

}

return BadRequest();

}

}

public class ClientApiController : MyApiController

{

private readonly IClientDTOServices clientServices;

private readonly ITastingDTOServices tastingServices;

private readonly ITastingClientDTOServices tastingClientDTOServices;

private readonly IMapper mapper;

public ClientApiController(IClientDTOServices services, IMapper \_mapper, ITastingDTOServices \_tastingServices, ITastingClientDTOServices \_tastingClientDTOServices)

{

clientServices = services;

tastingServices = \_tastingServices;

tastingClientDTOServices = \_tastingClientDTOServices;

mapper = \_mapper;

}

[HttpGet]

public IEnumerable<ClientView> Item()

{

var clientDTO = clientServices.GetAll();

var clientView = mapper.Map<IEnumerable<ClientDTO>, IEnumerable<ClientView>>(clientDTO);

return clientView;

}

[HttpGet("{id}")]

public IActionResult GetClient(int id)

{

var clientDTO = clientServices.Get(id);

if (clientDTO is null)

{

return NotFound();

}

var clientView = mapper.Map<ClientDTO, ClientView>(clientDTO);

return Ok(clientView);

}

[HttpPost]

public IActionResult AddTasting([FromBody] TastingClientView tastingClient)

{

var tasting = tastingServices.Get(tastingClient.IdTasting);

if (tasting != null && tastingClient.IdClient != null && tasting.CountClient > tastingClientDTOServices.CountClient(tastingClient.IdTasting))

{

var tc = new TastingClientDTO { IdClient = tastingClient.IdClient, IdTasting = tastingClient.IdTasting};

var id = tastingClientDTOServices.Add(tc);

if (id != 0)

{

return Ok(id);

}

return NotFound();

}

return BadRequest();

}

[HttpPost]

public IActionResult RemoveTating([FromBody] TastingClientView tastingClient)

{

if (tastingClient.IdTasting == null && tastingClient.IdClient == null)

{

return BadRequest();

}

var cl = clientServices.Get(tastingClient.IdClient);

var tc = cl.TastingClient.FirstOrDefault(x => x.IdTasting == tastingClient.IdTasting);

if (tc != null)

{

tastingClientDTOServices.Remove(tc);

return NoContent();

}

return BadRequest();

}

[HttpDelete("{id}")]

public IActionResult Remove(int id)

{

var cl = clientServices.Get(id);

if (cl != null)

{

clientServices.Remove(cl);

return NoContent();

}

return BadRequest();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return BadRequest(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

public class GrapeSortApiController : MyApiController

{

private readonly IGrapeSortDTOService grapeSortDTOService;

private readonly IMapper mapper;

public GrapeSortApiController(IGrapeSortDTOService \_grapeSortDTOService, IMapper \_mapper)

{

grapeSortDTOService = \_grapeSortDTOService;

mapper = \_mapper; }

[HttpGet]

public IEnumerable<GrapeSortView> Item()

{

var grapeSortDTO = grapeSortDTOService.GetAll();

var grapeSortView = mapper.Map<IEnumerable<GrapeSortDTO>, IEnumerable<GrapeSortView>>(grapeSortDTO);

return grapeSortView;

}

[HttpGet("{id}")]

public IActionResult About(int id)

{

var grapeSortDTO = grapeSortDTOService.Get(id);

if (grapeSortDTO is null)

{

return NotFound();

}

var grapeSortView = mapper.Map<GrapeSortDTO, GrapeSortView>(grapeSortDTO);

return Ok(grapeSortView);

}

[HttpPost]

public IActionResult Add([FromBody] GrapeSortView value)

{

var grapeSortDTO = mapper.Map<GrapeSortView, GrapeSortDTO>(value);

var id = grapeSortDTOService.Add(grapeSortDTO);

if (id != 0)

{

return Ok(id);

}

return NotFound();

}

[HttpPut]

public IActionResult Update([FromBody] GrapeSortView value)

{

var grapeSortDTO = mapper.Map<GrapeSortView, GrapeSortDTO>(value);

grapeSortDTOService.Update(grapeSortDTO);

return NoContent();

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

var grapeSortDTO = grapeSortDTOService.Get(id);

grapeSortDTOService.Remove(grapeSortDTO);

return NoContent();

}}

public class ManufacturerApiController : MyApiController

{

private readonly IManufacturerDTOServices manufacturerDTOServices;

private readonly IMapper mapper;

public ManufacturerApiController(IManufacturerDTOServices \_manufacturerDTOServices, IMapper \_mapper)

{

manufacturerDTOServices = \_manufacturerDTOServices;

mapper = \_mapper;

}

[HttpGet]

public IEnumerable<ManufacturerView> Item()

{

var manufacturerDTO = manufacturerDTOServices.GetAll();

var manufacturerView = mapper.Map<IEnumerable<ManufacturerDTO>,IEnumerable<ManufacturerView>>(manufacturerDTO);

return manufacturerView;

}

[HttpGet("{id}")]

public IActionResult About(int id)

{

var manufacturerDTO = manufacturerDTOServices.Get(id);

if (manufacturerDTO is null)

{

return NotFound();

}

var manufacturerView = mapper.Map<ManufacturerDTO, ManufacturerView>(manufacturerDTO);

return Ok(manufacturerView);

}

[HttpPost]

public IActionResult Add([FromBody] ManufacturerView value)

{

var manufacturerDTO = mapper.Map<ManufacturerView, ManufacturerDTO>(value);

var id = manufacturerDTOServices.Add(manufacturerDTO);

if (id != 0)

{

return Ok(id);

}

return NotFound();

}

[HttpPut]

public IActionResult Update([FromBody] ManufacturerView value)

{

var manufacturerDTO = mapper.Map<ManufacturerView, ManufacturerDTO>(value);

manufacturerDTOServices.Update(manufacturerDTO);

return NoContent();

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

var manufacturerDTO = manufacturerDTOServices.Get(id);

manufacturerDTOServices.Remove(manufacturerDTO);

return NoContent();

}}

public class TastingApiController : MyApiController

{

private readonly IMapper mapper;

private readonly ITastingDTOServices tastingServices;

public TastingApiController(IMapper mapper, ITastingDTOServices tastingServices)

{

this.mapper = mapper;

this.tastingServices = tastingServices;

}

[HttpGet]

public TastingViewFilter Item()

{

var tastingDTO = tastingServices.GetAll();

var tastingView = mapper.Map<IEnumerable<TastingDTO>, IEnumerable<TastingView>>(tastingDTO);

var ViewModal = new TastingViewFilter { tastingView = tastingView.OrderByDescending(x => x.CountClient - x.TastingClient.Count()), pagination = new Pagination(tastingView.Count(),1) };

return ViewModal;

}

[HttpGet]

public IEnumerable<TastingView> Index()

{

var tastingDTO = tastingServices.GetAll();

var tastingView = mapper.Map<IEnumerable<TastingDTO>, IEnumerable<TastingView>>(tastingDTO);

tastingView = tastingView.OrderBy(x => x.DateTast)

.Where(x => x.DateTast >= DateTime.Now)

.Take(3);

return tastingView;

}

[HttpGet("{id}")]

public IActionResult About (int Id)

{

var tastingDTO = tastingServices.Get(Id);

if (tastingDTO is null)

{

return NotFound();

}

var tastingView = mapper.Map<TastingDTO, TastingView>(tastingDTO);

return Ok(tastingView);

}

[HttpPost]

public IActionResult Filter([FromBody] TastingViewFilter viewFilter)

{

var tastingDTO = viewFilter.Filter switch

{

FilterTasting.OrderByPriceAsc => tastingServices.ReadOrderedByPriceAsc(),

FilterTasting.OrderByPriceDesc => tastingServices.ReadOrderedByPriceDesc(),

FilterTasting.SortByDateAsc => tastingServices.SortByDateAsc(),

FilterTasting.SortByDateDesc => tastingServices.SortByDateDesc(),

\_ => tastingServices.GetAll().OrderByDescending(x => x.CountClient - x.TastingClient.Count())

};

if (!viewFilter.wine.IsNullOrEmpty())

{

tastingDTO = tastingServices.GetByWine(tastingDTO, viewFilter.wine);

}

if(viewFilter.date.ToShortDateString()!="01.01.0001")

{

tastingDTO = tastingServices.GetByDate(tastingDTO, viewFilter.date);

}

var tastingView = mapper.Map<IEnumerable<TastingDTO>, IEnumerable<TastingView>>(tastingDTO);

viewFilter.tastingView = tastingView.Skip((viewFilter.pagination.PageNumber - 1) \* 5).Take(5);

viewFilter.pagination = new Pagination(tastingView.Count(), viewFilter.pagination.PageNumber);

return Ok(viewFilter);

}

[HttpPost]

public IActionResult Add([FromBody] TastingView tastingView)

{

var tastingDTO = mapper.Map<TastingView, TastingDTO>(tastingView);

var id = tastingServices.Add(tastingDTO);

if (id != 0)

{

return Ok(id);

}

return NotFound();

}

[HttpPut]

public IActionResult Update([FromBody] TastingView value)

{

var tastingDTO = mapper.Map<TastingView, TastingDTO>(value);

tastingServices.Update(tastingDTO);

return NoContent();

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

var manufacturerDTO = tastingServices.Get(id);

tastingServices.Remove(manufacturerDTO);

return NoContent();

}

}

public class WineApiController : MyApiController

{

private readonly IWineDTOServices wineServices;

private readonly IWineRatingDTOServices wineRatingServices;

private readonly IManufacturerDTOServices manufacturerDTOServices;

private readonly IWebHostEnvironment appEnvironment;

private readonly IMapper mapper;

public WineApiController(IWebHostEnvironment \_appEnvironment,

IWineDTOServices services,

IWineRatingDTOServices \_wineRatingServices,

IManufacturerDTOServices \_manufacturerDTOServices,

IMapper \_mapper)

{

appEnvironment = \_appEnvironment;

wineServices = services;

wineRatingServices = \_wineRatingServices;

manufacturerDTOServices = \_manufacturerDTOServices;

mapper = \_mapper;

}

[HttpGet]

public WineViewFilter Item()

{

var wineDTO = wineServices.GetAll();

var wineView = mapper.Map<IEnumerable<WineDTO>, IEnumerable<WineView>>(wineDTO);

var ViewModal = new WineViewFilter

{

wineView = wineView,

ListIdManufacturer = CreateSelectListManufacturerDTO(),

pagination = new Pagination(wineView.Count(), 1)

};

return ViewModal;

}

[HttpGet("{id}")]

public IActionResult About (int Id)

{

var GetWineDTO = wineServices.Get(Id);

if (GetWineDTO is null)

{

return NotFound();

}

var GetWineView = mapper.Map<WineDTO, WineView>(GetWineDTO);

return Ok(GetWineView);

}

[HttpPost]

public IActionResult AddWineRating([FromBody] WineRatingView wineRating)

{

var wineRatingDTO = wineRatingServices.GetAll().FirstOrDefault(x => x.IdClient == wineRating.IdClient&& x.IdWine == wineRating.IdWine);

if (wineRatingDTO is null)

{

wineRatingDTO = new WineRatingDTO { IdWine = wineRating.IdWine, IdClient= wineRating.IdClient, Mark = wineRating.Mark };

var id = wineRatingServices.Add(wineRatingDTO);

if (id != 0)

{

return Ok(id);

}

return NotFound();

}

else

{

wineRatingDTO.Mark = wineRating.Mark;

wineRatingServices.Update(wineRatingDTO);

}

return NoContent();

}

[HttpPost]

public IActionResult Filter([FromBody] WineViewFilter viewFilter)

{

var wine = viewFilter.Filter switch

{

FilterWine.OrderByPriceAsc => wineServices.ReadOrderedByPriceAsc(),

FilterWine.OrderByPriceDesc => wineServices.ReadOrderedByPriceDesc(),

FilterWine.SortByRatingAsc => wineServices.SortByRatingAsc(),

FilterWine.SortByRatingDesc => wineServices.SortByRatingDesc(),

\_ => wineServices.GetAll()

};

if (!viewFilter.nameWine.IsNullOrEmpty())

{

wine = wineServices.GetAllByName(wine, viewFilter.nameWine.Trim());

}

if (viewFilter.IdManufacturer != 0)

{

wine = wineServices.GetAllByManufacturer(wine, viewFilter.IdManufacturer);

}

if (viewFilter.mark != 0)

{

wine = wineServices.GetAllByRating(wine, viewFilter.mark);

}

viewFilter.wineView = mapper.Map<IEnumerable<WineDTO>, IEnumerable<WineView>>(wine).Skip((viewFilter.pagination.PageNumber - 1) \* 5).Take(5);

viewFilter.ListIdManufacturer = CreateSelectListManufacturerDTO();

viewFilter.pagination = new Pagination(wine.Count(), viewFilter.pagination.PageNumber);

return Ok(viewFilter);

}

public async Task<IActionResult> AddFile(IFormFile uploadedFile)

{

string path = "";

if (uploadedFile != null)

{

path = "/img/Wine/" + uploadedFile.FileName;

using (var fileStream = new FileStream(appEnvironment.WebRootPath + path, FileMode.Create))

{

await uploadedFile.CopyToAsync(fileStream);

}

return Ok("https://localhost:44300" + path);

}

return BadRequest();

}

[HttpPost]

public IActionResult Add([FromBody] WineView wine)

{

var wineDTO = mapper.Map<WineView, WineDTO>(wine);

var id = wineServices.Add(wineDTO);

if (id != 0)

{

return Ok(id);

}

return BadRequest();

}

[HttpPut]

public IActionResult Update([FromBody] WineView value)

{

var wineDTO = mapper.Map<WineView, WineDTO>(value);

wineServices.Update(wineDTO);

return NoContent();

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

if (id != 0)

{

var wineDTO = wineServices.Get(id);

wineServices.Remove(wineDTO);

return NoContent();

}

return BadRequest();

}

[NonAction]

private List<SelectListItem> CreateSelectListManufacturerDTO()

{

List<SelectListItem> items = new SelectList(manufacturerDTOServices.GetAll(), "Id", "NameMnf").ToList();

items.Insert(0, (new SelectListItem { Text = "Все", Value = "0" }));

return items;

}

}

public class WineGrapeSortApiController : MyApiController

{

private readonly IWineGrapeSortServices wineGrapeSortServices;

private readonly IMapper mapper;

public WineGrapeSortApiController(IWineGrapeSortServices \_wineGrapeSortServices, IMapper \_mapper)

{

wineGrapeSortServices = \_wineGrapeSortServices;

mapper = \_mapper;

}

[HttpGet]

public IEnumerable<WineGrapeSortView> Item()

{

var wineGrapeSortDTO = wineGrapeSortServices.GetAll();

var wineGrapeSortView = mapper.Map<IEnumerable<WineGrapeSortDTO>, IEnumerable<WineGrapeSortView>>(wineGrapeSortDTO);

return wineGrapeSortView;

}

[HttpGet("{id}")]

public IActionResult About(int id)

{

var wineGrapeSortDTO = wineGrapeSortServices.Get(id);

if (wineGrapeSortDTO is null)

{

return NotFound();

}

var wineGrapeSortView = mapper.Map<WineGrapeSortDTO, WineGrapeSortView>(wineGrapeSortDTO);

return Ok(wineGrapeSortView);

}

[HttpPost]

public IActionResult Add([FromBody] WineGrapeSortView value)

{

var wineGrapeSortDTO = mapper.Map<WineGrapeSortView, WineGrapeSortDTO>(value);

var id = wineGrapeSortServices.Add(wineGrapeSortDTO);

if (id != 0)

{

return Ok(id);

}

return NotFound();

}

[HttpPut]

public IActionResult Update([FromBody] WineGrapeSortView value)

{

var wineGrapeSortDTO = mapper.Map<WineGrapeSortView, WineGrapeSortDTO>(value);

wineGrapeSortServices.Update(wineGrapeSortDTO);

return NoContent();

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

var wineGrapeSortDTO = wineGrapeSortServices.Get(id);

wineGrapeSortServices.Remove(wineGrapeSortDTO);

return NoContent();

}

}

public class WineTastingApiController : MyApiController

{

private readonly IWineTastingServices wineTastingServices;

private readonly IMapper mapper;

public WineTastingApiController(IWineTastingServices \_wineTastingServices, IMapper \_mapper)

{

wineTastingServices = \_wineTastingServices;

mapper = \_mapper;

}

[HttpGet]

public IEnumerable<WineTastingView> Item()

{

var wineTastingDTO = wineTastingServices.GetAll();

var wineTastingView = mapper.Map<IEnumerable<WineTastingDTO>, IEnumerable<WineTastingView>>(wineTastingDTO);

return wineTastingView;

}

[HttpGet("{id}")]

public IActionResult About(int id)

{

var wineTastingDTO = wineTastingServices.Get(id);

if (wineTastingDTO is null)

{

return NotFound();

}

var wineTastingView = mapper.Map<WineTastingDTO, WineTastingView>(wineTastingDTO);

return Ok(wineTastingView);

}

[HttpPost]

public IActionResult Add([FromBody] WineTastingView value)

{

var wineTastingDTO = mapper.Map<WineTastingView, WineTastingDTO>(value);

var id = wineTastingServices.Add(wineTastingDTO);

if (id != 0)

{

return Ok(id);

}

return NotFound();

}

[HttpPut]

public IActionResult Update([FromBody] WineTastingView value)

{

var wineTastingDTO = mapper.Map<WineTastingView, WineTastingDTO>(value);

wineTastingServices.Update(wineTastingDTO);

return NoContent();

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

var wineTastingDTO = wineTastingServices.Get(id);

wineTastingServices.Remove(wineTastingDTO);

return NoContent();

}

}

public class MyProfile : Profile

{

public MyProfile()

{

CreateMap<Client, ClientDTO>().ReverseMap();//+ 1

CreateMap<ClientDTO, ClientView>().ReverseMap();

CreateMap<Manufacturer, ManufacturerDTO>().ReverseMap();//+ 2

CreateMap<ManufacturerDTO, ManufacturerView>().ReverseMap();

CreateMap<Tasting, TastingDTO>().ReverseMap();//+ 3

CreateMap<TastingDTO, TastingView>().ReverseMap();

CreateMap<Wine, WineDTO>().ReverseMap();//+ 4

CreateMap<WineDTO, WineView>().ReverseMap();

CreateMap<TastingClient, TastingClientDTO>().ReverseMap();//+ 5

CreateMap<TastingClientDTO, TastingClientView>().ReverseMap();

CreateMap<WineTasting, WineTastingDTO>().ReverseMap();//+ 6

CreateMap<WineTastingDTO, WineTastingView>().ReverseMap();

CreateMap<WineGrapeSort, WineGrapeSortDTO>().ReverseMap();//+ 7

CreateMap<WineGrapeSortDTO, WineGrapeSortView>().ReverseMap();

CreateMap<WineRating, WineRatingDTO>().ReverseMap();//+ 8

CreateMap<WineRatingDTO, WineRatingView>().ReverseMap();

CreateMap<GrapeSort, GrapeSortDTO>().ReverseMap();//+ 9

CreateMap<GrapeSortDTO, GrapeSortView>().ReverseMap();

CreateMap<RegisterModel, ClientDTO>().ReverseMap();//+ 10

}

public class LoginModel

{

[Required(ErrorMessage = "Не указан E-mail")]

[EmailAddress]

[Display(Name = "Введите E-mail")]

public string Email { get; set; }

[Required(ErrorMessage = "Не указан пароль")]

[Display(Name = "Введите пароль")]

[StringLength(100,MinimumLength =8,ErrorMessage ="Пароль от 8 символов")]

public string Password { get; set; }

}

public class RegisterModel

{

[Required(ErrorMessage = "Не указана фамилия")]

[Display(Name = "Введите фамилию")]

public string Surname { get; set; }

[Required(ErrorMessage = "Не указано имя")]

[Display(Name = "Введите имя")]

public string NameCl { get; set; }

[Required(ErrorMessage = "Не указано отчество")]

[Display(Name = "Введите отчество")]

public string Patronymic { get; set; }

[Required(ErrorMessage = "Не указан E-mail")]

[EmailAddress]

[Display(Name = "Введите E-mail")]

public string EMail { get; set; }

[Required(ErrorMessage = "Не указан пароль")]

[Display(Name = "Введите пароль")]

[StringLength(100, MinimumLength = 8, ErrorMessage = "Пароль должен быть больше 8 символов")]

public string PasswordCl { get; set; }

[Display(Name = "Введите повторно пароль")]

[Compare("PasswordCl", ErrorMessage = "Пароль введен неверно")]

public string RepeatPassword{ get; set; }

[Required(ErrorMessage = "Телефон не указан")]

[Phone(ErrorMessage = "Телефон должен содержать от 8 до 13 цифр")]

[Display(Name = "Введите телефон")]

[StringLength(13, MinimumLength = 8, ErrorMessage = "Телефон должен содержать от 8 до 13 цифр")]

public string Phone { get; set; }

}

public class ClientView

{

public int Id { get; set; }//

[Display(Name = "Фамилия :")]

public string Surname { get; set; }

[Display(Name = "Имя :")]

public string NameCl { get; set; }

[Display(Name = "Отчество :")]

public string Patronymic { get; set; }

[Display(Name = "Email :")]

public string EMail { get; set; }

public string PasswordCl { get; set; }//

[Display(Name = "Телефон :")]

public string Phone { get; set; }

[Display(Name = "Дегустации :")]

public virtual ICollection<TastingClientView> TastingClient { get; set; }

[Display(Name = "Средняя оценка выставляемая вину :")]

public virtual ICollection<WineRatingView> WineRating { get; set; }

public ClientView()

{

TastingClient = new List<TastingClientView>();

WineRating = new List<WineRatingView>();

}

}

public class GrapeSortView

{

public int Id { get; set; }

public string NameSort { get; set; }

public virtual ICollection<WineGrapeSortView> WineGrapeSort { get; set; }

public GrapeSortView()

{

WineGrapeSort = new List<WineGrapeSortView>();

}

}

public class ManufacturerView

{

public int Id { get; set; }

public string Country { get; set; }

public string NameMnf { get; set; }

public string EMail { get; set; }

public string AddressMnf { get; set; }

public virtual ICollection<WineView> Wine { get; set; }

public ManufacturerView()

{

Wine = new List<WineView>();

}

}

public class Pagination

{

public int PageNumber { get; set; }

public int TotalPages { get; set; }

public Pagination(int count, int pageNumber)

{

PageNumber = pageNumber;

TotalPages = (int)Math.Ceiling(count / (double)5);

}

public bool HasPreviousPage

{

get

{

return (PageNumber > 1);

}

}

public bool HasNextPage

{

get

{

return (PageNumber < TotalPages);

}

}

}

public class TastingClientView

{

public int Id { get; set; }

public int? IdTasting { get; set; }

public int? IdClient { get; set; }

public virtual ClientView IdClientNavigation { get; set; }

public virtual TastingView IdTastingNavigation { get; set; }

}

public class TastingView

{

public int Id { get; set; }

public DateTime? DateTast { get; set; }

public string LocationTast { get; set; }

public string NameTast { get; set; }

public int? CountClient { get; set; }

public double? Price { get; set; }

public virtual ICollection<TastingClientView> TastingClient { get; set; }

public virtual ICollection<WineTastingView> WineTasting { get; set; }

public TastingView()

{

TastingClient = new List<TastingClientView>();

WineTasting = new List<WineTastingView>();

}

}

public class TastingViewFilter

{

public IEnumerable<TastingView> tastingView { get; set; }

public DateTime date { get; set; }

public string wine { get; set; }

public FilterTasting Filter { get; set; }

public Pagination pagination { get; set; }

}

public class WineGrapeSortView

{

public int Id { get; set; }

public int? IdWine { get; set; }

public int? IdGrapeSort { get; set; }

public virtual GrapeSortView IdGrapeSortNavigation { get; set; }

public virtual WineView IdWineNavigation { get; set; }

}

public class WineRatingView

{

public int Id { get; set; }

public int? IdWine { get; set; }

public int? IdClient { get; set; }

public int? Mark { get; set; }

public virtual ClientView IdClientNavigation { get; set; }

public virtual WineView IdWineNavigation { get; set; }

}

public class WineTastingView

{

public int Id { get; set; }

public int? IdWine { get; set; }

public int? IdTasting { get; set; }

public virtual TastingView IdTastingNavigation { get; set; }

public virtual WineView IdWineNavigation { get; set; }

}

public class WineView

{

public int Id { get; set; }

public string Fluidity { get; set; }

public string Color { get; set; }

public string Taste { get; set; }

public string Smack { get; set; }

public double? Fortress { get; set; }

public int? IdManufacturer { get; set; }

public double? Price { get; set; }

public string Photo { get; set; }

public string NameW { get; set; }

public virtual ManufacturerView IdManufacturerNavigation { get; set; }

public virtual ICollection<WineGrapeSortView> WineGrapeSort { get; set; }

public virtual ICollection<WineRatingView> WineRating { get; set; }

public virtual ICollection<WineTastingView> WineTasting { get; set; }

public WineView()

{

WineGrapeSort = new List<WineGrapeSortView>();

WineRating = new List<WineRatingView>();

WineTasting = new List<WineTastingView>();

}

}

public class WineViewFilter

{

public List<SelectListItem> ListIdManufacturer { get; set; }

public IEnumerable<WineView> wineView { get; set; }

public string nameWine { get; set; }

public int IdManufacturer { get; set; }

public double mark { get; set; }

public FilterWine Filter { get; set; }

public Pagination pagination { get; set; }

}

Enum:

public enum FilterTasting

{

[Display(Name = "По умолчанию")]

Deafult,

[Display(Name = "Дата : старые")]

SortByDateAsc,

[Display(Name = "Дата : новые")]

SortByDateDesc,

[Display(Name = "Цена : по возрастанию")]

OrderByPriceAsc,

[Display(Name = "Цена : по убыванию")]

OrderByPriceDesc

}

public enum FilterWine

{

[Display(Name = "По умолчанию")]

Deafult,

[Display(Name = "Рейтинг : по возрастанию")]

SortByRatingAsc,

[Display(Name = "Рейтинг : по убыванию")]

SortByRatingDesc,

[Display(Name = "Цена : по возрастанию")]

OrderByPriceAsc,

[Display(Name = "Цена : по убыванию")]

OrderByPriceDesc

}

public class AccountServices : IAccount

{

public string HashPassword(string password)

{

using (SHA1 sha1Hash = SHA1.Create())

{

byte[] sourceBytes = Encoding.UTF8.GetBytes(password);

byte[] hashBytes = sha1Hash.ComputeHash(sourceBytes);

return BitConverter.ToString(hashBytes).Replace("-", String.Empty);

}

}

}

public class ClientServices : IClientDTOServices

{

private readonly IUnitOfWork unitOfWork;

private readonly IMapper mapper;

public ClientServices(IUnitOfWork \_unitOfWork, IMapper \_mapper)

{

unitOfWork = \_unitOfWork;

mapper = \_mapper;

}

public int Add(ClientDTO source)

{

var client = mapper.Map<ClientDTO, Client>(source);

unitOfWork.ClientRepository.Create(client);

unitOfWork.Save();

return client.Id;

}

public void Remove(ClientDTO item)

{

unitOfWork.ClientRepository.Delete(mapper.Map<ClientDTO, Client>(item));

unitOfWork.Save();

}

public ClientDTO Get(int? id)

{

var client = unitOfWork.ClientRepository.ReadAll()

.Include(c => c.TastingClient)

.ThenInclude(tc => tc.IdTastingNavigation)

.Include(c=>c.WineRating)

.ThenInclude(cw=>cw.IdWineNavigation)

.FirstOrDefault(x=>x.Id==id);

return mapper.Map<Client, ClientDTO>(client);

}

public IEnumerable<ClientDTO> GetAll()

{

var clientDTO = unitOfWork.ClientRepository.ReadAll()

.Include(c => c.TastingClient)

.ThenInclude(tc => tc.IdTastingNavigation)

.Include(c => c.WineRating)

.ThenInclude(cw => cw.IdWineNavigation);

return mapper.Map<IEnumerable<Client>, IEnumerable<ClientDTO>>(clientDTO);

}

public void Update(ClientDTO item)

{

unitOfWork.ClientRepository.Update(mapper.Map<ClientDTO, Client>(item));

unitOfWork.Save();

}

}

public class GrapeSortServices : IGrapeSortDTOService

{

private readonly IUnitOfWork unitOfWork;

private readonly IMapper mapper;

public GrapeSortServices(IUnitOfWork \_unitOfWork, IMapper \_mapper)

{

unitOfWork = \_unitOfWork;

mapper = \_mapper;

}

public int Add(GrapeSortDTO source)

{

var grapeSort = mapper.Map<GrapeSortDTO, GrapeSort>(source);

unitOfWork.GrapeSortRepository.Create(grapeSort);

unitOfWork.Save();

return grapeSort.Id;

}

public void Remove(GrapeSortDTO item)

{

unitOfWork.GrapeSortRepository.Delete(mapper.Map<GrapeSortDTO, GrapeSort>(item));

unitOfWork.Save();

}

public GrapeSortDTO Get(int? id)

{

var grapeSort = unitOfWork.GrapeSortRepository.ReadAll()

.Include(g => g.WineGrapeSort)

.ThenInclude(wg=>wg.IdWineNavigation)

.ThenInclude(x=>x.IdManufacturerNavigation)

.Include(g => g.WineGrapeSort)

.ThenInclude(wg => wg.IdWineNavigation)

.ThenInclude(x => x.WineRating)

.ThenInclude(x=>x.IdClientNavigation)

.Include(g => g.WineGrapeSort)

.ThenInclude(wg => wg.IdWineNavigation)

.ThenInclude(x => x.WineTasting)

.ThenInclude(x => x.IdTastingNavigation)

.FirstOrDefault(g => g.Id == id);

return mapper.Map<GrapeSort, GrapeSortDTO>(grapeSort);

}

public IEnumerable<GrapeSortDTO> GetAll()

{

var grapeSort = unitOfWork.GrapeSortRepository.ReadAll()

.Include(g => g.WineGrapeSort)

.ThenInclude(wg => wg.IdWineNavigation)

.ThenInclude(x => x.IdManufacturerNavigation)

.Include(g => g.WineGrapeSort)

.ThenInclude(wg => wg.IdWineNavigation)

.ThenInclude(x => x.WineRating)

.ThenInclude(x => x.IdClientNavigation)

.Include(g => g.WineGrapeSort)

.ThenInclude(wg => wg.IdWineNavigation)

.ThenInclude(x => x.WineTasting)

.ThenInclude(x => x.IdTastingNavigation);

return mapper.Map<IEnumerable<GrapeSort>, IEnumerable<GrapeSortDTO>>(grapeSort);

}

public void Update(GrapeSortDTO item)

{

unitOfWork.GrapeSortRepository.Update(mapper.Map<GrapeSortDTO, GrapeSort>(item));

unitOfWork.Save();

}

}

public class ManufacturerServices : IManufacturerDTOServices

{

private readonly IUnitOfWork unitOfWork;

private readonly IMapper mapper;

public ManufacturerServices(IUnitOfWork \_unitOfWork, IMapper \_mapper)

{

unitOfWork = \_unitOfWork;

mapper = \_mapper;

}

public int Add(ManufacturerDTO source)

{

var manufacturer = mapper.Map<ManufacturerDTO, Manufacturer>(source);

unitOfWork.ManufacturerRepository.Create(manufacturer);

unitOfWork.Save();

return manufacturer.Id;

}

public void Remove(ManufacturerDTO item)

{

unitOfWork.ManufacturerRepository.Delete(mapper.Map<ManufacturerDTO, Manufacturer>(item));

unitOfWork.Save();

}

public ManufacturerDTO Get(int id)

{

var manfDTO = unitOfWork.ManufacturerRepository.ReadAll()

.Include(x => x.Wine).ThenInclude(x => x.WineGrapeSort)

.ThenInclude(x => x.IdGrapeSortNavigation)

.Include(x => x.Wine).ThenInclude(x => x.WineRating)

.ThenInclude(x => x.IdClientNavigation)

.ThenInclude(x => x.TastingClient)

.Include(x => x.Wine).ThenInclude(x => x.WineTasting)

.ThenInclude(x => x.IdTastingNavigation)

.ThenInclude(x => x.TastingClient)

.ThenInclude(x => x.IdClientNavigation)

.FirstOrDefault(m => m.Id == id);

return mapper.Map<Manufacturer, ManufacturerDTO>(manfDTO);

}

public IEnumerable<ManufacturerDTO> GetAll()

{

var manDTO = unitOfWork.ManufacturerRepository.ReadAll()

.Include(x => x.Wine).ThenInclude(x=>x.WineGrapeSort)

.ThenInclude(x=>x.IdGrapeSortNavigation)

.Include(x => x.Wine).ThenInclude(x=>x.WineRating)

.ThenInclude(x=>x.IdClientNavigation)

.ThenInclude(x=>x.TastingClient)

.Include(x => x.Wine).ThenInclude(x => x.WineTasting)

.ThenInclude(x => x.IdTastingNavigation)

.ThenInclude(x=>x.TastingClient)

.ThenInclude(x=>x.IdClientNavigation);

return mapper.Map<IEnumerable<Manufacturer>, IEnumerable<ManufacturerDTO>>(manDTO);

}

public void Update(ManufacturerDTO item)

{

unitOfWork.ManufacturerRepository.Update(mapper.Map<ManufacturerDTO, Manufacturer>(item));

unitOfWork.Save();

}

}

public class TastingClientDTOServices : ITastingClientDTOServices

{

private readonly IUnitOfWork unitOfWork;

private readonly IMapper mapper;

public TastingClientDTOServices(IUnitOfWork \_unitOfWork, IMapper \_mapper)

{

unitOfWork = \_unitOfWork;

mapper = \_mapper;

}

public int Add(TastingClientDTO item)

{

var res = mapper.Map<TastingClientDTO, TastingClient>(item);

unitOfWork.TastingClientRepository.Create(res);

unitOfWork.Save();

return res.Id;

}

public int CountClient(int? idTasting)

{

return unitOfWork.TastingClientRepository.ReadAll()

.Include(tc => tc.IdClientNavigation)

.Include(tc => tc.IdTastingNavigation)

.Where(x => x.IdTasting == idTasting)

.Count();};

public void Remove(TastingClientDTO item)

{

var res = mapper.Map<TastingClientDTO, TastingClient>(item);

unitOfWork.TastingClientRepository.Delete(res);

unitOfWork.Save();

}

}

public class TastingServices : ITastingDTOServices

{

private readonly IUnitOfWork unitOfWork;

private readonly IMapper mapper;

public TastingServices(IUnitOfWork \_unitOfWork, IMapper \_mapper)

{

unitOfWork = \_unitOfWork;

mapper = \_mapper;

}

public int Add(TastingDTO source)

{

var tasting = mapper.Map<TastingDTO, Tasting>(source);

unitOfWork.TastingRepository.Create(tasting);

unitOfWork.Save();

return tasting.Id;

}

public void Remove(TastingDTO item)

{

unitOfWork.TastingRepository.Delete(mapper.Map<TastingDTO, Tasting>(item));

unitOfWork.Save();

}

public TastingDTO Get(int? id)

{

var tastingDTO = unitOfWork.TastingRepository.ReadAll()

.Include(x => x.TastingClient)

.ThenInclude(x => x.IdClientNavigation)

.Include(x => x.WineTasting)

.ThenInclude(x => x.IdWineNavigation)

.FirstOrDefault(t => t.Id == id);

return mapper.Map<Tasting, TastingDTO>(tastingDTO);

}

public IEnumerable<TastingDTO> GetAll()

{

var tastingDTO = unitOfWork.TastingRepository.ReadAll()

.Include(x => x.TastingClient)

.ThenInclude(x => x.IdClientNavigation)

.Include(x => x.WineTasting)

.ThenInclude(x => x.IdWineNavigation);

return mapper.Map<IEnumerable<Tasting>, IEnumerable<TastingDTO>>(tastingDTO);

}

public void Update(TastingDTO item)

{

unitOfWork.TastingRepository.Update(mapper.Map<TastingDTO, Tasting>(item));

unitOfWork.Save();

}

public IEnumerable<TastingDTO> GetByDate(IEnumerable<TastingDTO> item, DateTime date)

{

return item.Where(x => x.DateTast.Value.ToShortDateString() == date.ToShortDateString());

}

public IEnumerable<TastingDTO> GetByWine(IEnumerable<TastingDTO> item, string wine)

{

return item.Select(x => x.WineTasting.Where(x => x.IdWineNavigation.NameW.ToUpper().Contains(wine.ToUpper())))

.Where(x=>!x.IsNullOrEmpty())

.Select(x=>x.Select(x=>x.IdTastingNavigation).FirstOrDefault());

}

public IEnumerable<TastingDTO> ReadOrderedByPriceAsc() => GetAll().OrderBy(x => x.Price);

public IEnumerable<TastingDTO> ReadOrderedByPriceDesc() => GetAll().OrderByDescending(x => x.Price);

public IEnumerable<TastingDTO> SortByDateAsc() => GetAll().OrderBy(x => x.DateTast);

public IEnumerable<TastingDTO> SortByDateDesc() => GetAll().OrderByDescending(x => x.DateTast);

}

public class WineGrapeSortServices : IWineGrapeSortServices

{

private readonly IUnitOfWork unitOfWork;

private readonly IMapper mapper;

public WineGrapeSortServices(IUnitOfWork \_unitOfWork, IMapper \_mapper)

{

unitOfWork = \_unitOfWork;

mapper = \_mapper;

}

public int Add(WineGrapeSortDTO source)

{

var wineGrapeSort = mapper.Map<WineGrapeSortDTO, WineGrapeSort>(source);

unitOfWork.WineGrapeSortRepository.Create(wineGrapeSort);

unitOfWork.Save();

return wineGrapeSort.Id;

}

public WineGrapeSortDTO Get(int? id)

{

var wineGrapeSort = unitOfWork.WineGrapeSortRepository.ReadAll()

.Include(x=>x.IdGrapeSortNavigation)

.Include(x=>x.IdWineNavigation)

.FirstOrDefault(g => g.Id == id);

return mapper.Map<WineGrapeSort, WineGrapeSortDTO>(wineGrapeSort);

}

public IEnumerable<WineGrapeSortDTO> GetAll()

{

var wineGrapeSort = unitOfWork.WineGrapeSortRepository.ReadAll()

.Include(x => x.IdGrapeSortNavigation)

.Include(x => x.IdWineNavigation);

return mapper.Map<IEnumerable<WineGrapeSort>, IEnumerable<WineGrapeSortDTO>>(wineGrapeSort);

}

public void Remove(WineGrapeSortDTO item)

{

unitOfWork.WineGrapeSortRepository.Delete(mapper.Map<WineGrapeSortDTO, WineGrapeSort>(item));

unitOfWork.Save();

}

public void Update(WineGrapeSortDTO item)

{

unitOfWork.WineGrapeSortRepository.Update(mapper.Map<WineGrapeSortDTO, WineGrapeSort>(item));

unitOfWork.Save();

}

}

public class WineRatingDTOServices : IWineRatingDTOServices

{

private readonly IUnitOfWork unitOfWork;

private readonly IMapper mapper;

public WineRatingDTOServices(IUnitOfWork \_unitOfWork, IMapper \_mapper)

{

unitOfWork = \_unitOfWork;

mapper = \_mapper;

}

public int Add(WineRatingDTO wineRatingDTO)

{

var wineRating = mapper.Map<WineRatingDTO, WineRating>(wineRatingDTO);

unitOfWork.WineRatingRepository.Create(wineRating);

unitOfWork.Save();

return wineRating.Id;

}

public IEnumerable<WineRatingDTO> GetAll()

{

var wineRating = unitOfWork.WineRatingRepository.ReadAll()

.Include(wr=>wr.IdClientNavigation)

.Include(wr=>wr.IdWineNavigation);

return mapper.Map<IEnumerable<WineRating>, IEnumerable<WineRatingDTO>>(wineRating);

}

public void Update(WineRatingDTO item)

{

unitOfWork.WineRatingRepository.Update(mapper.Map<WineRatingDTO, WineRating>(item));

unitOfWork.Save();

}

public void Remove(WineRatingDTO item)

{

unitOfWork.WineRatingRepository.Delete(mapper.Map<WineRatingDTO, WineRating>(item));

unitOfWork.Save();

}

}

public class WineServices : IWineDTOServices

{

private readonly IUnitOfWork unitOfWork;

private readonly IMapper mapper;

public WineServices(IUnitOfWork \_unitOfWork, IMapper \_mapper)

{

unitOfWork = \_unitOfWork;

mapper = \_mapper;

}

public int Add(WineDTO source)

{

var wine = mapper.Map<WineDTO, Wine>(source);

unitOfWork.WineRepository.Create(wine);

unitOfWork.Save();

return wine.Id;

}

public void Remove(WineDTO item)

{

unitOfWork.WineRepository.Delete(mapper.Map<WineDTO, Wine>(item));

unitOfWork.Save();

}

public WineDTO Get(int id)

{

var wine = unitOfWork.WineRepository.ReadAll()

.Include(w => w.WineRating)

.ThenInclude(wr=>wr.IdClientNavigation)

.Include(w=>w.IdManufacturerNavigation)

.Include(w=>w.WineTasting)

.ThenInclude(wt=>wt.IdTastingNavigation)

.Include(w=>w.WineGrapeSort)

.ThenInclude(wg=>wg.IdGrapeSortNavigation)

.FirstOrDefault(w => w.Id == id);

return mapper.Map<Wine, WineDTO>(wine);

}

public IEnumerable<WineDTO> GetAllByName(IEnumerable<WineDTO> wineDTO,string nameWine)

{

return wineDTO.Where(w => w.NameW.ToUpper().Contains(nameWine.ToUpper()));

}

public IEnumerable<WineDTO> GetAllByManufacturer(IEnumerable<WineDTO> wineDTO, int IdManufacturer)

{

return wineDTO.Where(w => w.IdManufacturer == IdManufacturer);

}

public IEnumerable<WineDTO> GetAllByRating(IEnumerable<WineDTO> wineDTO, double mark)

{

return wineDTO.Select(w => w.WineRating.GroupBy(x => x.IdWine)

.Where(x => x.Average(x => x.Mark) >= mark)

.Select(x => x.Select(x => x.IdWineNavigation).FirstOrDefault())

.FirstOrDefault()).Where(x => x != null);

}

public IEnumerable<WineDTO> GetAll()

{

var wine = unitOfWork.WineRepository.ReadAll()

.Include(w => w.WineRating)

.ThenInclude(wr => wr.IdClientNavigation)

.Include(w => w.IdManufacturerNavigation)

.Include(w => w.WineTasting)

.ThenInclude(wt => wt.IdTastingNavigation)

.Include(w => w.WineGrapeSort)

.ThenInclude(wg => wg.IdGrapeSortNavigation);

return mapper.Map<IEnumerable<Wine>, IEnumerable<WineDTO>>(wine);

}

public void Update(WineDTO item)

{

unitOfWork.WineRepository.Update(mapper.Map<WineDTO, Wine>(item));

unitOfWork.Save();

}

public IEnumerable<WineDTO> ReadOrderedByPriceAsc() => GetAll().OrderBy(x => x.Price);

public IEnumerable<WineDTO> ReadOrderedByPriceDesc() => GetAll().OrderByDescending(x => x.Price);

public IEnumerable<WineDTO> SortByRatingAsc() => GetAll().OrderBy(x => x.WineRating.Average(x => x.Mark));

public IEnumerable<WineDTO> SortByRatingDesc() => GetAll().OrderByDescending(x => x.WineRating.Average(x => x.Mark));

}

public class WineTastingServices : IWineTastingServices

{

private readonly IUnitOfWork unitOfWork;

private readonly IMapper mapper;

public WineTastingServices(IUnitOfWork \_unitOfWork, IMapper \_mapper)

{

unitOfWork = \_unitOfWork;

mapper = \_mapper;

}

public int Add(WineTastingDTO source)

{

var wineTasting = mapper.Map<WineTastingDTO, WineTasting>(source);

unitOfWork.WineTastingRepository.Create(wineTasting);

unitOfWork.Save();

return wineTasting.Id;

}

public WineTastingDTO Get(int? id)

{

var wineTasting = unitOfWork.WineTastingRepository.ReadAll()

.Include(x => x.IdTastingNavigation)

.Include(x => x.IdWineNavigation)

.FirstOrDefault(x=>x.Id==id);

return mapper.Map<WineTasting, WineTastingDTO>(wineTasting);

}

public IEnumerable<WineTastingDTO> GetAll()

{

var wineTasting = unitOfWork.WineTastingRepository.ReadAll()

.Include(x => x.IdTastingNavigation)

.Include(x => x.IdWineNavigation);

return mapper.Map<IEnumerable<WineTasting>, IEnumerable<WineTastingDTO>>(wineTasting);

}

public void Remove(WineTastingDTO item)

{

unitOfWork.WineTastingRepository.Delete(mapper.Map<WineTastingDTO, WineTasting>(item));

unitOfWork.Save();

}

public void Update(WineTastingDTO item)

{

unitOfWork.WineTastingRepository.Update(mapper.Map<WineTastingDTO, WineTasting>(item));

unitOfWork.Save();

}

}

public class WineDBContext : DbContext

{

public virtual DbSet<Client> Client { get; set; }

public virtual DbSet<GrapeSort> GrapeSort { get; set; }

public virtual DbSet<Manufacturer> Manufacturer { get; set; }

public virtual DbSet<Tasting> Tasting { get; set; }

public virtual DbSet<TastingClient> TastingClient { get; set; }

public virtual DbSet<Wine> Wine { get; set; }

public virtual DbSet<WineGrapeSort> WineGrapeSort { get; set; }

public virtual DbSet<WineRating> WineRating { get; set; }

public virtual DbSet<WineTasting> WineTasting { get; set; }

public WineDBContext(DbContextOptions options) : base(options)

{

Database.EnsureCreated();

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Client>(entity =>

{

entity.Property(e => e.EMail)

.HasColumnName("E\_mail")

.HasMaxLength(100)

.IsUnicode(false);

entity.Property(e => e.NameCl)

.HasMaxLength(100)

.IsUnicode(false);

entity.Property(e => e.PasswordCl)

.HasMaxLength(100)

.IsUnicode(false);

entity.Property(e => e.Patronymic)

.HasMaxLength(100)

.IsUnicode(false);

entity.Property(e => e.Phone)

.HasMaxLength(15)

.IsUnicode(false);

entity.Property(e => e.Surname)

.HasMaxLength(100)

.IsUnicode(false);

});

modelBuilder.Entity<GrapeSort>(entity =>

{

entity.Property(e => e.NameSort)

.HasMaxLength(100)

.IsUnicode(false);

});

modelBuilder.Entity<Manufacturer>(entity =>

{

entity.Property(e => e.AddressMnf)

.HasMaxLength(100)

.IsUnicode(false);

entity.Property(e => e.Country)

.HasMaxLength(100)

.IsUnicode(false);

entity.Property(e => e.EMail)

.HasColumnName("E\_mail")

.HasMaxLength(100)

.IsUnicode(false);

entity.Property(e => e.NameMnf)

.HasMaxLength(100)

.IsUnicode(false);

});

modelBuilder.Entity<Tasting>(entity =>

{

entity.Property(e => e.DateTast).HasColumnType("date");

entity.Property(e => e.LocationTast)

.HasMaxLength(100)

.IsUnicode(false);

});

modelBuilder.Entity<TastingClient>(entity =>

{

entity.ToTable("Tasting\_Client");

entity.HasOne(d => d.IdClientNavigation)

.WithMany(p => p.TastingClient)

.HasForeignKey(d => d.IdClient)

.OnDelete(DeleteBehavior.Cascade)

.HasConstraintName("FK\_\_Tasting\_C\_\_IdCli\_\_32E0915F");

entity.HasOne(d => d.IdTastingNavigation)

.WithMany(p => p.TastingClient)

.HasForeignKey(d => d.IdTasting)

.OnDelete(DeleteBehavior.Cascade)

.HasConstraintName("FK\_\_Tasting\_C\_\_IdTas\_\_31EC6D26");

});

modelBuilder.Entity<Wine>(entity =>

{

entity.Property(e => e.Color)

.HasMaxLength(100)

.IsUnicode(false);

entity.Property(e => e.Fluidity)

.HasMaxLength(100)

.IsUnicode(false);

entity.Property(e => e.Fortress)

.HasMaxLength(100)

.IsUnicode(false);

entity.Property(e => e.Photo)

.HasMaxLength(100)

.IsUnicode(false);

entity.Property(e => e.Smack)

.HasMaxLength(100)

.IsUnicode(false);

entity.Property(e => e.Taste)

.HasMaxLength(100)

.IsUnicode(false);

entity.HasOne(d => d.IdManufacturerNavigation)

.WithMany(p => p.Wine)

.HasForeignKey(d => d.IdManufacturer)

.OnDelete(DeleteBehavior.Cascade)

.HasConstraintName("FK\_\_Wine\_\_IdManufact\_\_2C3393D0");});

modelBuilder.Entity<WineGrapeSort>(entity =>

{

entity.ToTable("Wine\_GrapeSort");

entity.HasOne(d => d.IdGrapeSortNavigation)

.WithMany(p => p.WineGrapeSort)

.HasForeignKey(d => d.IdGrapeSort)

.OnDelete(DeleteBehavior.Cascade)

.HasConstraintName("FK\_\_Wine\_Grap\_\_IdGra\_\_36B12243");

entity.HasOne(d => d.IdWineNavigation)

.WithMany(p => p.WineGrapeSort)

.HasForeignKey(d => d.IdWine)

.OnDelete(DeleteBehavior.Cascade)

.HasConstraintName("FK\_\_Wine\_Grap\_\_IdWin\_\_35BCFE0A");

});

modelBuilder.Entity<WineRating>(entity =>

{

entity.Property(e => e.IdWine).HasColumnName("IdWIne");

entity.HasOne(d => d.IdWineNavigation)

.WithMany(p => p.WineRating)

.HasForeignKey(d => d.IdWine)

.OnDelete(DeleteBehavior.Cascade)

.HasConstraintName("FK\_\_WineRatin\_\_IdWIn\_\_2F10007B");

entity.HasOne(d => d.IdClientNavigation)

.WithMany(p => p.WineRating)

.HasForeignKey(d => d.IdClient)

.OnDelete(DeleteBehavior.Cascade)

.HasConstraintName("FK\_\_WineRatin\_\_IdWIn\_\_2F10228B");

});

modelBuilder.Entity<WineTasting>(entity =>

{

entity.ToTable("Wine\_Tasting");

entity.HasOne(d => d.IdTastingNavigation)

.WithMany(p => p.WineTasting)

.HasForeignKey(d => d.IdTasting)

.OnDelete(DeleteBehavior.Cascade)

.HasConstraintName("FK\_\_Wine\_Tast\_\_IdTas\_\_3A81B327");

entity.HasOne(d => d.IdWineNavigation)

.WithMany(p => p.WineTasting)

.HasForeignKey(d => d.IdWine)

.OnDelete(DeleteBehavior.Cascade)

.HasConstraintName("FK\_\_Wine\_Tast\_\_IdWin\_\_398D8EEE");

});

}

}

public class Repository<TSource> : IRepository<TSource> where TSource : class

{

private readonly DbContext context;

private readonly DbSet<TSource> set;

public Repository(DbContext \_context) => (context, set) = (\_context, \_context.Set<TSource>());

public void Create(TSource source)

{

set.Add(source);}

public void Delete(TSource item)

{

context.Entry(item).State = EntityState.Deleted;

}

public IQueryable<TSource> ReadAll()

{

return set.AsQueryable().AsNoTracking();

}

public void Update(TSource item)

{

context.Entry(item).State = EntityState.Modified;

}

}

public class EFUnitOfWork : IUnitOfWork

{

private readonly DbContext db;

private IRepository<Client> client;

private IRepository<Manufacturer> manufacturer;

private IRepository<Tasting> tasting;

private IRepository<Wine> wine;

private IRepository<TastingClient> tastingClient;

private IRepository<WineRating> wineRatingClient;

private IRepository<GrapeSort> grapeSort;

private IRepository<WineGrapeSort> wineGrapeSort;

private IRepository<WineTasting> wineTasting;

public IRepository<Client> ClientRepository => client ??= new Repository<Client>(db);

public IRepository<Manufacturer> ManufacturerRepository => manufacturer ??= new Repository<Manufacturer>(db);

public IRepository<Tasting> TastingRepository => tasting ??= new Repository<Tasting>(db);

public IRepository<Wine> WineRepository => wine ??= new Repository<Wine>(db);

public IRepository<TastingClient> TastingClientRepository => tastingClient ??= new Repository<TastingClient>(db);

public IRepository<GrapeSort> GrapeSortRepository => grapeSort ??= new Repository<GrapeSort>(db);

public IRepository<WineGrapeSort> WineGrapeSortRepository => wineGrapeSort ??= new Repository<WineGrapeSort>(db);

public IRepository<WineRating> WineRatingRepository => wineRatingClient ??= new Repository<WineRating>(db);

public IRepository<WineTasting> WineTastingRepository => wineTasting ??= new Repository<WineTasting>(db);

public EFUnitOfWork(DbContext dBContext)

{

db = dBContext;

}

public void Save()

{

db.SaveChanges();

}

}