Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Институт информационных технологий

Кафедра «Микропроцессорные системы и сети»

К защите допустить

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
К ДИПЛОМНОМУ ПРОЕКТУ  
НА ТЕМУ:  
«Портфолио криптовалют»**

Выполнила:

КДП

<http://t.me/kalilask4>

МИНСК 2022

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Институт повышения квалификации и переподготовки руководящих работников и специалистов

по информационным технологиям и радиоэлектронике

Кафедрамикропроцессорных систем и сетей

Специальность 1-40 01 73 – Программное обеспечение информационных систем

**УТВЕРЖДАЮ**

**ЗАДАНИЕ**

по дипломному проектированию

1.Тема проекта *Программное средство “Портфолио криптовалют”,*

3. Исходные данные к проекту *Программное средство должно быть построено c использованием*

*сети Internet, а также для работы в ОС Windows. Совместимость с операционной системой не ниже Windows 7, СУБД не ниже Microsoft SQL Server 2019. Основной функционал: создание портфолио*

*криптовалют, добавление монет, анализ состава портфеля.*

1. Перечень подлежащих разработке вопросов, содержащихся в пояснительной записке:

*4.1. Анализ существующих решений по теме проекта и постановка задачи.*

*4.2. Методы и модели, положенные в основу проекта*

*4.3. Разработка проекта программного средства*

*4.4. Разработка и реализация программного кода*

*4.5. Тестирование разработанного программного средства*

5. Перечень графического материала (с точным указанием обязательных чертежей и графиков):

*5.1. Схема ресурсов системы (десктоп), ф. А3, чертеж.*

*5.2. Схема работы системы (десктоп), ф.А3, чертеж.*

*5.3. Схема работы системы (веб), ф.А3, чертеж.*

*5.4. Добавление новой монеты (десктоп). Схема программы, ф.А3*

*5.5. Добавление новой транзакции обмена (десктоп). Схема программы, ф.А3*

Задание принял к исполнению

**Содержание**

[Нормативные ссылки, определения и сокращения 4](#_Toc97948124)

[Введение 5](#_Toc97948125)

[1 Аналитический обзор и постановка задачи 7](#_Toc97948126)

[1.1 Аналитический обзор существующих аналогов 7](#_Toc97948127)

[1.2 Постановка задачи 8](#_Toc97948128)

[2 Методы и модели абстрагирования объектов и функций предметной области 11](#_Toc97948129)

[2.1 Модели представления системы и их описание 11](#_Toc97948130)

[3 Разработка проекта программного обеспечения 13](#_Toc97948131)

[3.1 Функциональный анализ предметной области и разработка спецификаций требований к разрабатываемому программному обеспечению с использованием CASE-средств 13](#_Toc97948132)

[3.2 Разработка структуры объекта и функции составляющих его элементов 14](#_Toc97948133)

[3.2.1 Архитектура проекта 14](#_Toc97948134)

[3.3 Разработка схемы работы системы, отображающей последовательность основных операций и потоки данных в системе 18](#_Toc97948135)

[3.4 Выбор СУБД и разработка проекта базы данных 19](#_Toc97948136)

[3.5 Разработка интерфейсов программных средств 20](#_Toc97948137)

[3.6 Разработка структуры классов 23](#_Toc97948138)

[4 Разработка алгоритмов и их программная реализация 24](#_Toc97948139)

[5 Тестирование созданного программного средства 26](#_Toc97948140)

[5.1. Методы тестирования 26](#_Toc97948141)

[5.2. Разработка тестов и тестирование 27](#_Toc97948142)

[Заключение 32](#_Toc97948143)

[Список использованных источников 33](#_Toc97948144)

[Приложение А (справочное) Нормативно-справочные данные 34](#_Toc97948145)

[Приложение Б (обязательное) Текст программы 35](#_Toc97948146)

Нормативные ссылки, определения и сокращения

Криптовалюта – это любой вид валюты в цифровой или виртуальной форме; для защиты транзакций в криптовалюте используется шифрование (криптография). Не существует центрального органа по выпуску или регулированию криптовалют. Для записи транзакций и выпуска новых единиц используется децентрализованная система.

Биржа криптовалюты – место (площадка, сервис), где можно как обменять криптовалюту на другую, так и торговать ими на постоянной основе для получения прибыли.

Спот [расчётов](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D1%82%D1%91%D0%B6) - сделки, при которых оплата по [сделке](https://ru.wikipedia.org/wiki/%D0%A1%D0%B4%D0%B5%D0%BB%D0%BA%D0%B0) производится немедленно (как правило, в течение двух дней).

Транзакция - операция, состоящая в переводе монет внутри портфеля; сделка купли-продажи.

Транзакция перевода (transfer transaction) – транзакция увеличения объема уже имеющейся монеты.

Транзакция обмена (exchange transaction) – транзакция изменения объема двух уже имеющихся монет.

Торговля на бирже — процесс перехода прав собственности на монеты и (или [ценные бумаги](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D1%83%D0%BC%D0%B0%D0%B3%D0%B8)) на организованных торговых площадках ([биржах](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%80%D0%B6%D0%B0)).

Бро́кер (от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Broker — [маклер](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BA%D0%BB%D0%B5%D1%80), брокер, посредник) — [юридическое лицо](https://ru.wikipedia.org/wiki/%D0%AE%D1%80%D0%B8%D0%B4%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BB%D0%B8%D1%86%D0%BE), выполняющее [посреднические функции](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D1%80%D0%B5%D0%B4%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D1%83%D1%81%D0%BB%D1%83%D0%B3%D0%B8) между [продавцом](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B4%D0%B0%D0%B2%D0%B5%D1%86) и [покупателем](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BA%D1%83%D0%BF%D0%B0%D1%82%D0%B5%D0%BB%D1%8C) (на [фондовой](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D0%BD%D0%B4%D0%BE%D0%B2%D0%B0%D1%8F_%D0%B1%D0%B8%D1%80%D0%B6%D0%B0), [товарной](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D0%B2%D0%B0%D1%80%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D1%80%D0%B6%D0%B0), [валютной](https://ru.wikipedia.org/wiki/%D0%92%D0%B0%D0%BB%D1%8E%D1%82%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D1%80%D0%B6%D0%B0) [биржах](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%80%D0%B6%D0%B0)).

Электронная наличность ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) e-cash, electronic cash) или цифровая наличность ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) digital cash) — термин, который используется в платёжных системах. Данное название связано с возможностью совершать электронные платежи, аналогично оплате обычными [наличными](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D0%BB%D0%B8%D1%87%D0%BD%D1%8B%D0%B5_%D0%B4%D0%B5%D0%BD%D1%8C%D0%B3%D0%B8): без обязательного посредничества третьего лица

Введение

При ведении финансового учета в современном мире высокое значение приобретает качество учета, скорость и точность передачи информации. Традиционные бумажные и электронные носители не позволяют оперативно выполнять требуемые операции без создания “узких” мест. Учет финансов в виде крипто-монет производится в кошельках различных трейдинговых систем, холодных кошельках, что требует дополнительной системы для общего учета и мониторинга, анализа, наглядного представления.

В этом случае пользователю требуется единая информационная система (ИС) позволяющая формировать необходимую документацию доступную в текущем режиме. В рамках ИС происходит формирование отчетов для анализа и прогнозирования, формирование статистической отчетности.

Целью проекта является предоставление сервиса по ведению портфолио криптовалют, анализу состава портфеля, повышению оперативности реагирования при изменениях на рынке.

Задачи, решаемые в рамках реализации ИС:

− снижение человеческого фактора при внесении данных;

− реализация учета;

− реализация необходимых статистических отчетов;

− анализ и наглядное представление данных.

Крипто-портфолио приложения играют очень важную роль как для профессиональных пользователей, так и для новичков в блокчейне, торговле, инвестировании, обеспечивая возможность мониторинга баланса и движения монет в портфеле, анализа.

Учет монет производится в кошельках различных трейдинговых систем, холодных кошельках с возможностью формирования отчетов. Важным моментом является обеспечение мониторинга всех монет в одном месте.

Внесение данных в портфолио, управление данными осуществляется пользователем системы.

Программные средства, инструменты и языки программирования для решения поставленных задач приведены ниже.

**C# 10** [объектно-ориентированный](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) язык программирования для разработки приложений для платформы Microsoft .NET Framework и .NET Core.

Среда разработки программы **Microsoft Visual Studio 2019** с применением платформы пользовательского интерфейса для создания приложений WPF. Применение обусловлено необходимостью разработки приложения с экранными формами, простотой формирования отражения данных.

Среда разработки **JetBrains Rider**. Применение обусловлено необходимостью обеспечения удобной разработки в быстрой среде с инспекций кода, с наличием различных рефакторингов, удобной отладкой.

Система управления базами данных **Microsoft SQL Server 2017**. Применение обусловлено необходимостью использования базы данных совместимой с модулями ИС.

Программное средство для построения схем и диаграмм **MS Visio 2019**.

Среда моделирования **RationalRose 2003**. Применение обусловлено необходимостью создания моделей и представлений предметной области для дальнейшего использования при разработке программы.

Система контроля версий **git** и веб-сервис **github.com**.

Десктопный **git-клиент - Fork 1.68**.

Онлайн-сервис для проектирования и прототипирования **Figma**.

1 Аналитический обзор и постановка задачи

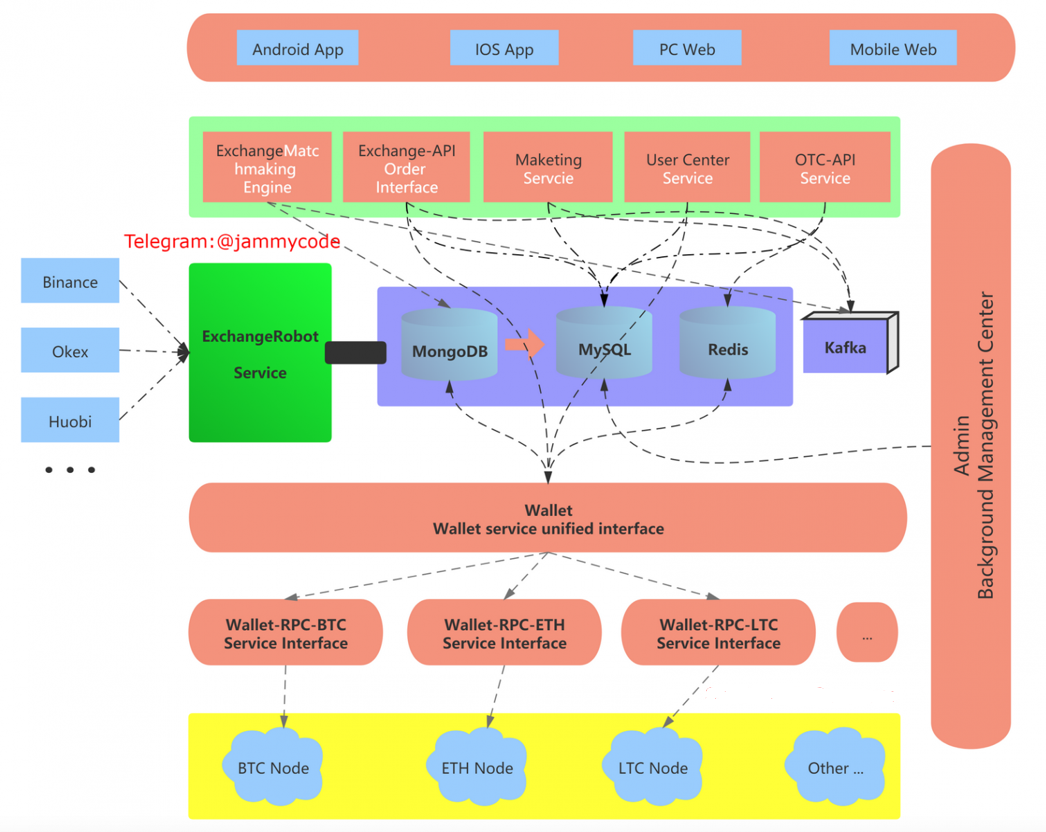
Криптовалю́та — разновидность [цифровой валюты](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%B2%D0%B0%D0%BB%D1%8E%D1%82%D0%B0), учёт внутренних расчётных единиц которой обеспечивает децентрализованная [платёжная система](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D1%82%D1%91%D0%B6%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0)(нет внутреннего или внешнего администратора или какого-либо его аналога), работающая в полностью автоматическом режиме. Сама по себе криптовалюта не имеет какой-либо особой [материальной](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D1%89%D1%8C) или электронной формы— это просто число, обозначающее количество данных расчётных единиц, которое записывается в соответствующей позиции информационного пакета протокола передачи данных и зачастую даже не подвергается шифрованию, как и вся иная информация о [транзакциях](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B0%D0%BA%D1%86%D0%B8%D1%8F_(%D1%84%D0%B8%D0%BD%D0%B0%D0%BD%D1%81%D1%8B)) между адресами системы. В настоящее время на более чем 10 000криптовалют приходится около менее 400 криптобирж, что связано с высокой стоимостью запуска и высокой зарегулированностью. Криптобиржа, конечно, слишком сложный инструмент, однако изучение открытого ПО и документации поможет в проектировании нашего программного средства.

Рисунок 1 - Устройство криптобиржи с открытым кодом

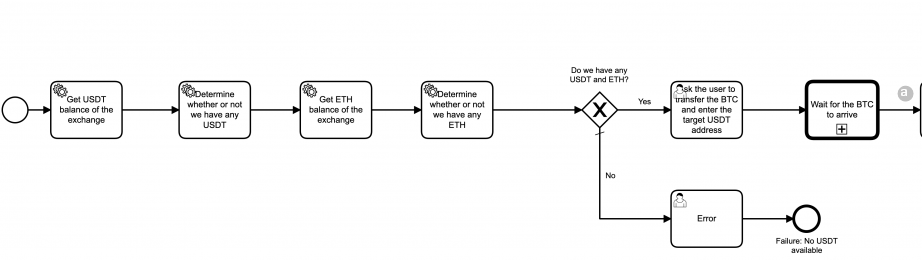
Процесс обмена валют представлен на рисунках 2-3. Здесь представлены служебными задачами (service tasks), перечень задач(task list), подпроцессы, с отдельной схемой.

Рисунок 2 - Процесс обмена валют

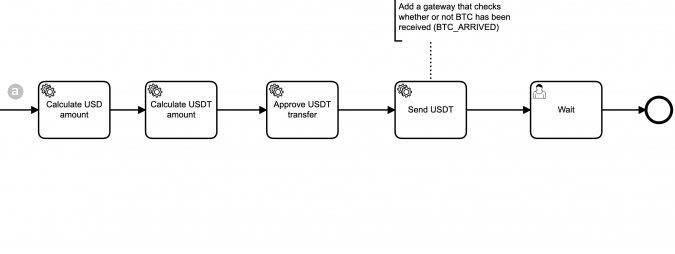


Рисунок 3 - Процесс обмена валют (продолжение)

1.1 Аналитический обзор существующих аналогов

Программное средство, которое подлежит разработке назовем «Портфолио криптовалют». Оно будет разработано для пользователя, интересующегося крипто индустрией, трейдера, инвестора.

При разработке данного программного средства, велось консультирование у практикующих специалистов, с целью сделать программное средство широко применяемой в практической деятельности пользователей. Доступны обзору некоторые аналоги данного проекта. Рассмотрим следующие аналоги программных средств, которые могли бы использоваться ведения портфеля, учета криптовалют: сoinmarketcap,

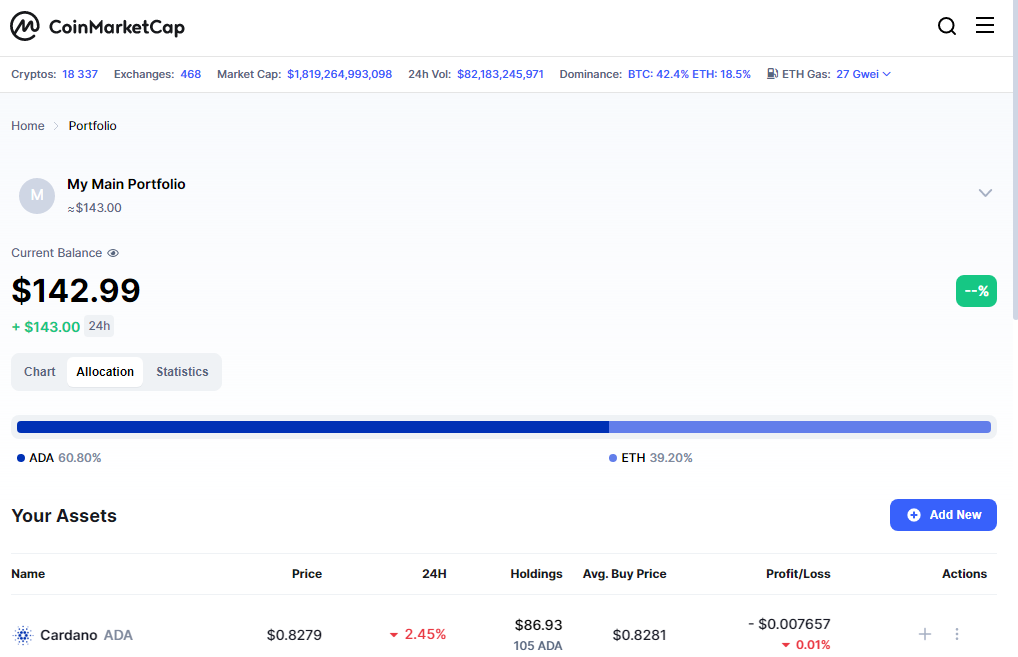


Рисунок 4 – Интерфейс coinmarketcap

Программное средство «сoinmarketcap» (рисунок 1) содержит хороший функционал для учета криптовалют, однако отсутствует гибкость в настройке интерфейса, интерфейс перегружен в результате чего новичку сложно ориентироваться, что может являться проблемой первичного взаимодействия с программным средством. Основная функциональность данного программного средства в предоставлении справочной информации о текущем рынке криптовалют.

Другие аналоги, такие как pionex (рисунок 5), binance, currency (рисунок 6) являются торговыми площадками и не позволяют хранить информацию о монетах на других биржах и кошельках.



Рисунок 5 – Главное окно Pionex

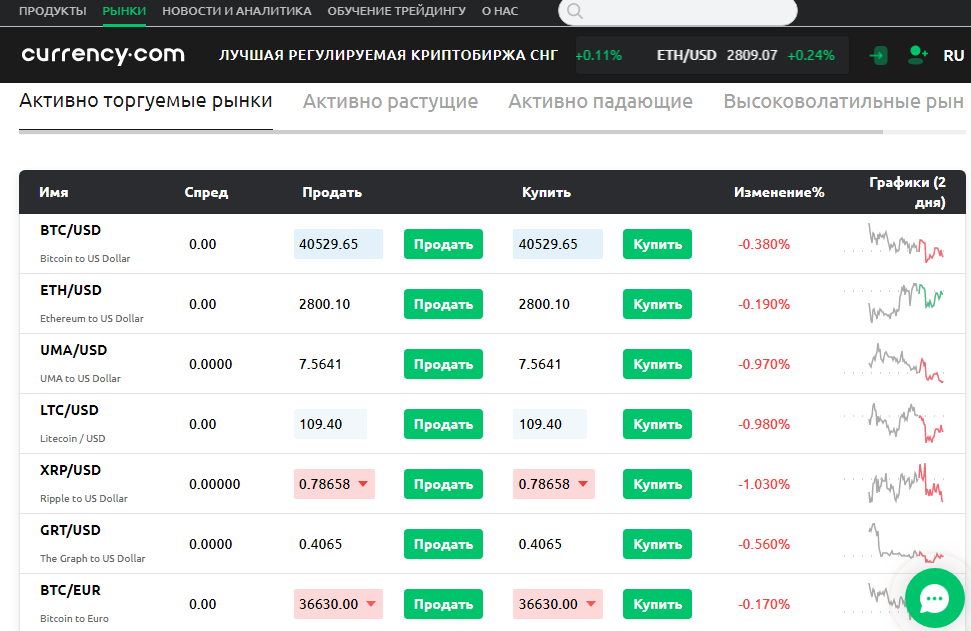


Рисунок 6 – Главное окно Currency

1.2 Постановка задачи

С точки зрения предметной области система должна разделяться на следующие компоненты:

1. учет поступления монет:
   * 1. создание новой монеты в системе;
     2. корректировка монеты;
     3. изменение количества;
2. учет транзакций:
   * 1. создание транзакции на изменение количества монеты;
     2. создание транзакции на обмен монет;
3. статистические отчеты:
   * 1. наличие монет с расчетом прибыли относительно текущей цены;
     2. транзакции за весь период;
     3. расчеты относительно средней цены покупки и текущей цены;
4. представление:
   * 1. диаграмма портфолио с пропорциональным представлением каждой монеты.

Для разработки программных средств используется язык программирования C# – это один из самых мощных, быстроразвивающихся и востребованных языков в ИТ-отрасли [10]. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программ до крупных веб-порталов и веб-серверов, обслуживающих ежедневно миллионы пользователей. Графический интерфейс десктопного приложения реализуется при помощи технологии WPF (Windows Presentation Foundation), которая является частью экосистемы платформы [6] .NET и представляет собой подсистему для построения графических интерфейсов, с использованием паттерна проектирования MVVM, доступ к данным реализован на основе технологии LINQ to Entities. Для разработки и создания базы данных использована система управления реляционными базами данных Microsoft SQL Server, с использованием технологии EntityFramework Code First [11]. В качестве CASE-средств выступает Rational Rose. Работа данного продукта основана на унифицированном языке моделирования UML [6] (Unified Modeling Language). Для разработки веб-сервиса используется ASP.NET.

Работа по реализации приложения разделена на следующие задачи:

1. моделирование предметной области для выявления сущностей, участников, процессов;
2. проектирование базы данных на основании п.1;
3. разработка интерфейса пользователей;
4. разработка десктопного приложения;
5. тестирование работы базы данных, приложения;
6. разработка веб-сервиса;
7. тестирование работы базы данных, приложения, взаимодействия десктопного приложения и веб-сервиса.

Для заданной предметной области, используя подход «Code First» создать базу данных, содержащую две таблицы, связанные отношением «один-ко-многим».

Для объектов одной из таблиц предусмотреть вывод изображения. Изображения должны храниться в папке Images, а в базе данных должно храниться только имя файла (без имени папки). (Для привязки данных реализовать конвертор значений – ValueConverter)

В программе на основе LINQ-to-Entities реализовать для спроектированной БД возможность просмотра и редактирования данных. В программе использовать многослойную архитектуру, выделив в отдельные сборки уровень доступа к базе данных (DAL – Data Access Level) и уровень бизнес-логики (BLL – Business Logic Level).

В главном окне реализовать отображение информации в виде MasterSlave (Главный – Подчиненный), когда при выборе в списке строки одной таблицы автоматически отображается содержимое подчиненной таблицы В главном окне приложения реализовать шаблон MVVM.

Для реакции модели представления на действия пользователя использовать механизм команд. Составить UML диаграмму классов для спроектированной системы.

Основной целью проекта является разработка программного продукта учитывающего вышесказанное и реализующего функционал:

1. обеспечивать работу одного пользователя в текущем режиме;
2. реализовывать внесения данных о монетах и транзакциях;
3. формировать наглядное представление данных в виде таблиц;
4. монеты, внесенные в ИС;
5. транзакции добавления и обмена монет;
6. обеспечивать отказоустойчивость при ошибочных данных введенных пользователем;
7. использовать базу данных доступную другим модулям при расширении ИС.

Приложение должно выводить следующие данные:

1. в виде диаграммы - монеты, внесенные в ИС отображением пропорционально общей оценочной стоимости портфеля (суммарно и в процентах);
2. в виде графика - изменение общего количества монеты в портфеле с течением времени;
3. суммарную начальную стоимость портфеля, стоимость портфеля с учетом текущего курса.

2 Методы и модели абстрагирования объектов и функций предметной области

2.1 Модели представления системы и их описание

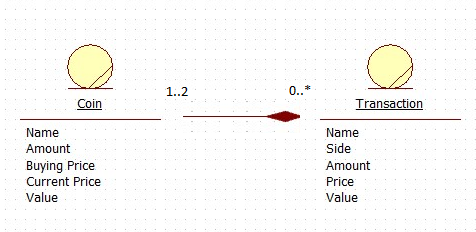
Для разностороннего изучения разрабатываемой системы представим различные модели, разработанные в WhiteStarUML, figma. Диаграмма сущностей предметной области представлена на рисунке 7.

Рисунок 7 - Диаграмма сущностей предметной области

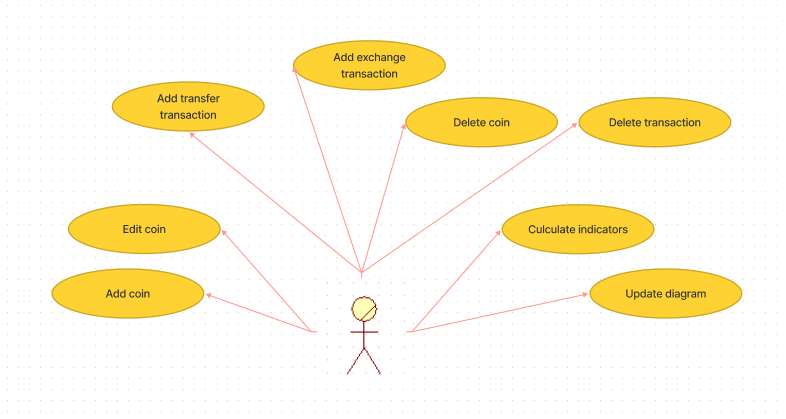
Основные функции, выделенные для разрабатываемой системы представлены на диаграмме вариантов использования (рисунок 8).

Рисунок 8 - Диаграмма вариантов использования

В результате бизнес-анализа выявлены следующие сущности: Монета, Транзакция. Отношения между сущностями и атрибуты отражены выше в диаграмме сущностей предметной области. Диаграмма классов приведена на Рисунке 22. Отношения между классами: к одной монете может относится несколько транзакций, одна транзакция может содержать от 1 до 2 монет. Для веб-сервиса выделены следующие сущности: Пользователь, Монета, Монета портфеля, Портфель.

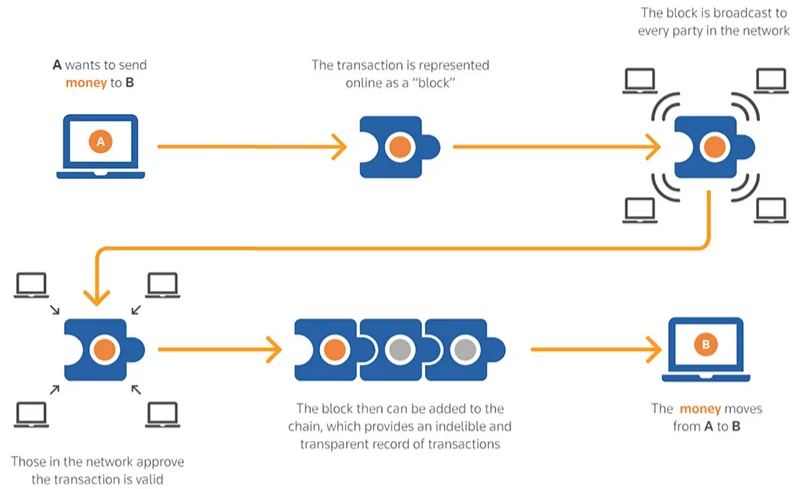
Рассмотрим принцип, по которому возникает и действует криптовалюта, на примере биткоина. Сеть биткоина основана на блокчейне. Блокчейн - это многофункциональная и многоуровневая информационная технология, предназначенная для надежного учета различных активов. Технология надежного распределенного хранения записей обо всех когда-либо совершенных транзакциях. Блокчейн представляет собой цепочку блоков данных, объем которой постоянно растет по мере добавления новых блоков с записями самых последних транзакций. В максимальном упрощении, blockchain работает как некая «распределенная книга записей» в сети с равноправными узлами. У каждого участника обмена есть копия данных этой единой бухгалтерской книги. После того, как стороны в сделке достигают согласия, все дополнения размножаются по сети. Схема работы блокчейна представлена на рисунке 10.

Рисунок 9 - Схема работы блокчейна

Хэш в этой схеме вслучае криптовалют это информация о транзакции, в более сложных системах - это информация об умных контрактах и актуальное состояние программного кода, внесенного в блокчейн.

Рисунок 10 - Общий вид блоков данных

3 Разработка проекта программного обеспечения

3.1 Функциональный анализ предметной области и разработка спецификаций требований к разрабатываемому программному обеспечению с использованием CASE-средств

Актером программного средства «Портфолио криптовалют» является пользователь-трейдер, -криптоинвестор, использующий разные торговые площадки, биржи, кошельки для спотовой торговли, хранения крипто валют. Приложение позволяет хранить данные по всем кошелькам и площадкам в одном месте, получать обновленные данные по текущим ценам, анализировать, получать сообщения при срабатывании триггеров – приближении цены к линии поддержки или сопротивления. Пользователь может инициировать следующие функции:

1. создавать, изменять, удалять монеты;
2. добавлять, изменять, удалять транзакции;
3. обновлять данные по последним ценам;
4. устанавливать триггеры дня нотификации на линии поддержки/сопротивления;
5. получать сообщения при срабатывании триггера.
6. отображать диаграмму по составу портфеля.

Рассмотрим сценарии основных вариантов использования программного средства пользователем. Для этого представим в табличном виде несколько ключевых сценариев, которые необходимо тестировать.

В таблице 1 представлен сценарий использования системы «Добавление монеты».

Таблица 1 – Сценарий функции «Добавление монеты»

|  |  |
| --- | --- |
| **Наименование варианта использования** | Добавление монеты. |
| **Актёры** | Пользователь |
| **Цель исполнения** | Добавление новой монеты. |

Продолжение таблицы 1

|  |  |
| --- | --- |
| **Краткое описание** | Пользователь производит добавление новой монеты путем нажатия кнопки на главном экране «Add coin». Пользователь заполняет все поля данного окна, после чего нажимает кнопку «Ok». Данные о новой монете сохраняются в базе данных, создается транзакция типа «transfer». Окно «Edit coin» закрывается. Новое монета добавляется в список на главной форме программного средства «Портфолио криптовалют» |
| **Предварительные условия** | Отображается главное окно программного средства «Портфолио криптовалют».  Кнопка «Add coin» активна. |
| **Типовой поток событий** | |
| **Действия актёра** | **Отклик системы** |
| 1. Пользователь нажимает кнопку «Add coin» на главном экране.  3. Пользователь заполняет все поля и нажимает на кнопку «Ok». | 2. Открывается окно «Edit coin».  4. Создается новый объект класса «Coin». Данный объект добавляется в базу, и происходит сохранение изменений в базе данных. Создается новый объект класса «Transaction» типа «Transfer». Данный объект добавляется в базу, и происходит сохранение изменений в базе данных. Появляется всплывающее окно с сообщением об успешном создании монеты и транзакции. Обновляются списки монет и транзакций на главном экране, на вкладке Transactions. |
| **Конечные условия** | Обновляется главное окно программного средства «Портфолио», в список которого добавлено новое тестирование. |

Продолжение таблицы 1

|  |  |
| --- | --- |
| **Расширение сценария** | |
| 3а.\_Пользователь не заполнил поле «Short name».  3б.\_Пользователь не заполнил поле «image». | 4а – система создает дефолтную монету и транзакцию. Появляется всплывающее окно с сообщением об успешном создании монеты и транзакции. Обновляются списки монет и транзакций на главном экране, на вкладке Transactions.  4б – система создает монету c именем «default.png». |

В таблице 2 представлен сценарий использования системы «Удаление монеты».

Таблица 2 – Сценарий функции «Удаление монеты»

|  |  |
| --- | --- |
| **Наименование варианта использования** | Удаление монеты. |
| **Актёры** | Пользователь |
| **Цель исполнения** | Удаление существующей монеты |
| **Краткое описание** | Пользователь производит удаление существующей монеты путем выбора его из списка на вкладке главного окна «Портфолио криптовалют» и последующего нажатия кнопки в меню «Delete coin». В появившемся диалоговом окне пользователь подтверждает удаление выбранной монеты, после чего происходит удаление монеты и сохранение изменений в базе данных. Появляется сообщение об успешном удалении. При смене вкладок или правого клика мышкой обновляются индикаторы с учетом удаленной монеты. На вкладке с диаграммой монета не отображается. |
| **Предварительные условия** | Отображается главное окно программного средства «Портфолио криптовалют».  В программном средстве «Портфолио криптовалют» существует хотя бы одна монета. |
| **Типовой поток событий** | |
| **Действия актёра** | **Отклик системы** |
| 1. Пользователь выбирает монету из списка на панели главного экрана программного средства «Портфолио криптовалют».  3. Пользователь нажимает кнопку в меню «Delete coin»  5. Пользователь нажимает кнопку «Yes» | 2. Выделяется строка выбранной монеты.  4. Появляется диалоговое окно с вопросом об удалении монеты.  6. Выбранная монета удаляется из базы данных. Изменения сохраняются в базе данных. Отображается сообщение об успешном удалении монеты. |
| Конечные условия | Выбранное тестирование удалено из главного окна программного средства «Портфолио криптовалют». |

3.2 Разработка структуры объекта и функции составляющих его элементов

3.2.1 Схема работы системы

Схема работы проектируемого программного средства представленная на чертеже ГУИР 003220.002, на чертеже ГУИР 003220.001 представлена схема ресурсов системы, на чертеже ГУИР 003220.003 - схема работы системы веб-приложения.

В начале работы с программным средством на экране отображается главное окно программного средства со вкладками «Coins», «Transactions» «Coins profile». Пользователь может выбрать одну из следующих операций: создание, удаление или редактирование монеты, транзакции, просмотр данных, диаграмм.

Операция создания новой монеты предполагает следующие этапы: открытие для пользователя диалогового окна «Edit coin», в котором отображаются поля для создания монеты, ввод пользователем данных, сохранение записи о новой монете в БД.

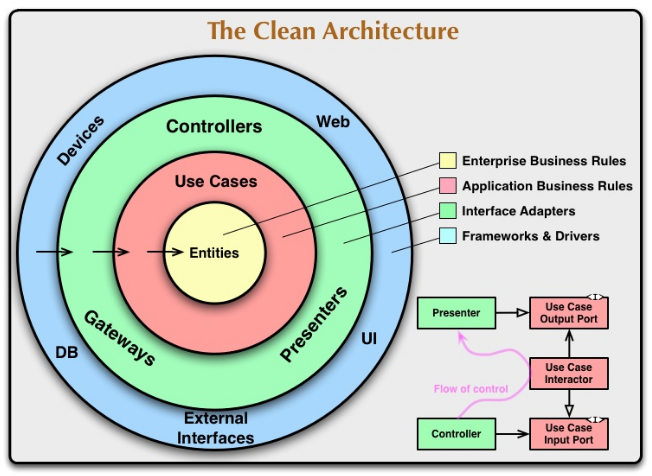
В начале работы с программным веб-средством пользователь проходит авторизацию в системе. Для этого он вводит логин и пароль в поля формы авторизации. Приложение выполняет проверку на наличие в базе данных записи, содержащей введенные логин и пароль. В случае ее отсутствия пользователю предлагается пройти авторизацию снова. Если же авторизация пройдена успешно, система определяет роль пользователя (admin, user) и открывает стартовое окно со списком монет пользователя. В главном окне пользователь может выбрать одну из следующих операций: создание, удаление или редактирование монеты, портфеля, просмотр данных.

Далее из приведенной схемы работы системы и ее описания выделим ресурсы, которые будет использовать данная система. Самые очевидные из них – ядро приложения, реализующее бизнес-логику, и база данных для хранения информации. Для WPF-приложения, будет разумно использовать в качестве СУБД MS SQL SERVER, так как это технологии одного стека, разработанного Micosoft, и они легко между собой взаимодействуют. Взаимодействие с БД предлагается реализовать с использованием подхода Code First к существующей БД, основанного на технологии EntityFramework, что существенно облегчает описание взаимодействия с БД для программиста. Таким образом определяется еще один используемый системой ресурс: библиотека EntityFramework 6.0.

Подход Code First к существующей БД предполагает разработку классов-сущностей вручную. Для упорядочивания внутренней структуры приложения предлагается вынести эти классы в отдельную библиотеку и в ней же описать класс контекста данных. Таким образом, еще один ресурс, который будет использовать приложение, – библиотека с классами-сущностями БД.

3.2.2 Архитектура проекта

Ориентируясь на подход описанный в статье Роберта Мартина «[The Clean Architecture](https://8thlight.com/blog/uncle-bob/2012/08/13/the-clean-architecture.html)»[8], и рекомендации к лабораторным работам [4] разработана архитектура проекта представлена в Приложения А. Оригинальная схема из статьи о Clean Architecture представлена на рисунке 9.

Рисунок 9 - Чистая архитектура

Архитектура проекта представлена в Приложении А. Архитектура проекта с зависимостями представлена на рисунке 10.

Библиотека Domain содержит описание сущностей, а также интерфейсы IRepository (хранилище данных), IUnitOfWork и ICoinClonable (рисунок 11)

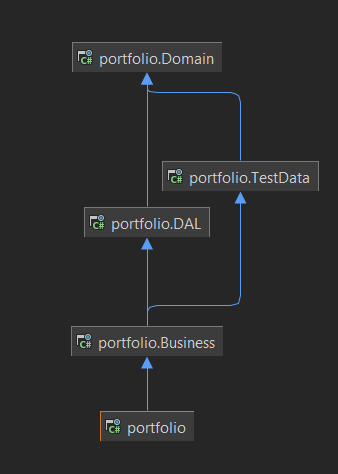


Рисунок 10 - Зависимости проекта.

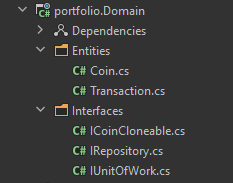


Рисунок 11 - Слой Domain

Библиотека Business содержит классы менеджеров для работы с монетами и транзакциями. Менеджеры осуществляют работу с базой данных через интерфейсы IRepository.

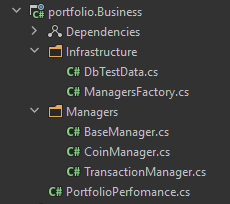


Рисунок 12 - Слой Business

Библиотека DAL – это реализация IRepository и IUnitOfWork с использованием Entity Framework.

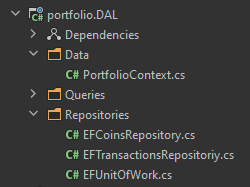
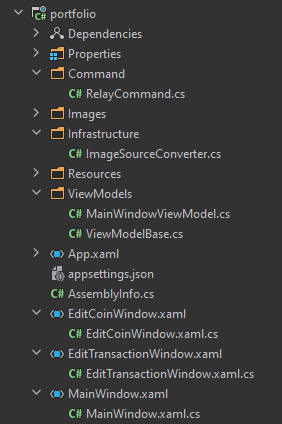


Рисунок 13 - Слой DAL

Рисунок 14 - Структура основного проекта приложения

3.3 Разработка схемы работы системы, отображающей последовательность основных операций и потоки данных в системе

Схема работы проектируемого программного средства представленная на чертеже ГУИР 003220.003

В начале работы с программным средством на экране отображается главное окно программного средства со вкладками «Coins», «Transactions» «Coins profile». Пользователь может выбрать одну из следующих операций: создание, удаление или редактирование монеты, транзакции, просмотр данных, диаграмм.

Операция создания новой монеты предполагает следующие этапы: открытие для пользователя диалогового окна «Edit coin», в котором отображаются поля для создания монеты, ввод пользователем данных, сохранение записи о новой монете в БД.

В начале работы с программным веб-средством пользователь проходит авторизацию в системе. Для этого он вводит логин и пароль в поля формы авторизации. Приложение выполняет проверку на наличие в базе данных записи, содержащей введенные логин и пароль. В случае ее отсутствия пользователю предлагается пройти авторизацию снова. Если же авторизация пройдена успешно, система определяет роль пользователя (admin, user) и открывает стартовое окно со списком монет пользователя. В главном окне пользователь может выбрать одну из следующих операций: создание, удаление или редактирование монеты, портфеля, просмотр данных.

3.4 Выбор СУБД и разработка проекта базы данных

При разработке базы данных необходимо решить такие задачи, как обеспечение хранения в базе данных всей необходимой информации, обеспечение возможности получения данных по всем необходимым запросам, возможности удаления ненужной информации, сокращение избыточности и дублирования данных и обеспечение целостности базы данных [13]. На рисунке 8 представлена схема базы данных, спроектированной для работы с программным средством.

Важным моментом при разработке базы данных является процесс нормализации. Нормализация – это процесс организации данных в базе данных, включающий создание таблиц и установление отношений между ними в соответствии с правилами, которые обеспечивают защиту данных и делают базу данных более гибкой, устраняя избыточность и несогласованные зависимости. В теории реляционных баз данных разработана следующая последовательность нормальных форм:

1. первая нормальная форма;
2. вторая нормальная форма;
3. третья нормальная форма;
4. нормальная форма Бойса-Кодда;
5. четвертая нормальная форма;
6. пятая нормальная форма [13].

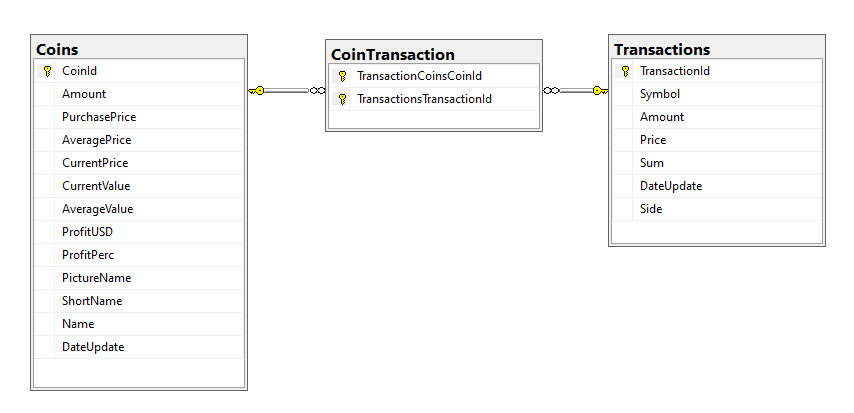


Рисунок 15 - Схема базы данных десктопного приложения

3.5 Разработка интерфейсов программных средств

Графический интерфейс клиентского десктопного приложения разработан по технологии .NET Windows Presentation Foundation. Для разработки приложения был выбран шаблон проектирования архитектуры MVVM (Model-View-ViewModel) для разделения логики приложения и отображения.При реализации приложения принято решение, что экранные формы редактирования монеты и редактирования транзакции должны представлять отдельное окно, отображение таблиц монет и транзакций а также диаграмм и графиков в отдельных вкладках главного окна. Графический интерфейс приложения должен содержать следующие элементы:

1. главное окно приложение, предоставлющее возможность просмотра всех монет в виде таблицы, каждая строка которого представляет собой блок с информацией о конкретной монете, а именно:
2. сокращенное наименование;
3. иллюстрация в виде рисунка;
4. количество монеты;
5. средняя цена покупки;
6. последняя цена по курсу;
7. объем в USD;
8. разница относительно покупки и последней цены по курсу в USD;
9. разница относительно покупки и последней цены по курсу в %;
10. дата добавления.
11. дополнительная панель главного окна предоставлющая возможность просмотра всех транзакций.

Каждая строка главного окна представляет собой блок с информацией о конкретной транзакции, а именно:

1. наименование, содержащее в себе наименование монет участвующих в транзакции;
2. количество монеты для добавления/продажи/покупки;
3. объем покупки;
4. вид транзакции (transfer - для увеличения объема существующей монеты, buy/sell - для обмена монет, участвующих в транзакции),
5. дата добавления транзакции;
6. транзакций, которые относятся к выделенной монете. Отображение информации реализовано в виде MasterSlave, когда при выборе в списке строки одной таблицы автоматически отображается содержимое подчиненной таблицы);
7. окно для ввода информации о новой монете, для добавления его в базу данных;
8. окно для изменения информации о новой монете, для изменения в базе данных;
9. окно для ввода информации о новой транзакции изменения количества выделенной монеты;
10. окно для ввода информации о новой транзакции обмена одной монеты на другую.

Главное окно приложения будет содержать следующие элементы:

1. диаграмма портфолио с пропорциональным представлением каждой монеты;
2. таблицу рассчитанных индикаторов по портфелю;
3. кнопка “Add coin”;
4. кнопка “Edit coin”;
5. кнопка “Recount coin”;
6. кнопка “Add transfer transaction”;
7. кнопка “Add exchange transaction”;
8. кнопка “Delete coin”;
9. кнопка “Delete transaction”.

Кроме того в главном окне предусмотрено отображение изображения монеты. Изображения хранится в папке Images, а в базе данных хранится только имя файла (без имени папки). (Для привязки данных реализован конвертор значений – ValueConverter).

Окно для ввода информации о новой монете будет вызываться по клику на кнопке «Add coin», содержащее текстовые поля, предназначенные для ввода информации о новой монете, а именно:

1. поле “Name”;
2. поле “ShortName”;
3. поле “Amount”;
4. поле “Buying price”;
5. поле “Current price”.

а также кнопку для выбора картинки для отображения в главном окне и кнопки для изменения при необходимости даты и кнопку “Cancel”, по нажатию на которую осуществляется закрытие окна ввода информации. В разрабатываемом приложении возможность просмотра и редактирования данных спроектированной базы данных будет реализована на основе LINQ to Entities. Для разработки приложения используем технологию Windows Presentation Foundation с разработкой многоуровневой архитектуры приложения. Используя подход «Code First» создана база данных, содержащую две таблицы, связанные отношением «многие-ко-многим».

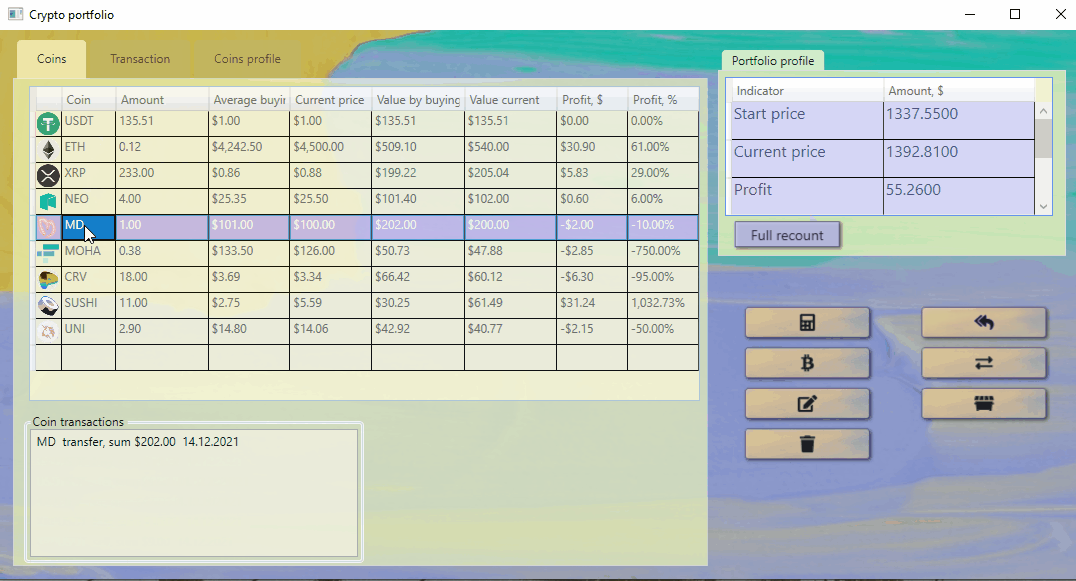
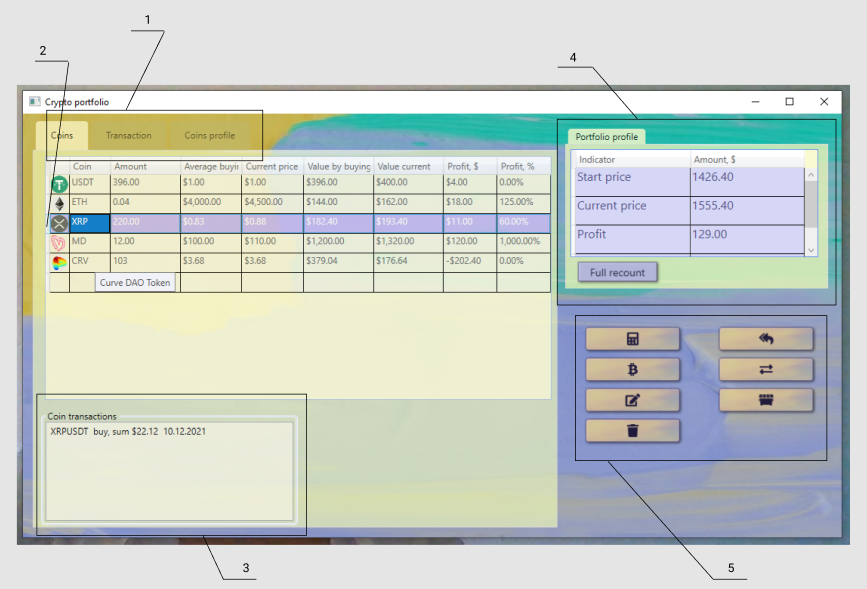
Вид главного окна программного средства приведен на рисунке 9, окно добавления монеты представлено на рисунке 16.

Рисунок 16 – Интерфейс главного окна десктопного приложения

После запуска программы пользователь работает с главным окном (рис. 17). Главное окно приложения предоставляет возможность просмотра всех монет в виде таблицы на вкладке Coins, на вкладке Transaction всех транзакций (рисунок 18), на вкладке Coins profile - диаграмма отражающая отношение монеты к общему объема портфеля (рисунок 16). Отображение информации в панели Coin Transaction (рис.17, п.3) реализовано в виде MasterSlave , когда при выборе в списке строки одной таблицы (рис.17, п.2) автоматически отображается содержимое подчиненной таблицы).

Рисунок 17 - Главное окно приложения: 1 - вкладки переключения на таблицу с монетами, таблицу с транзакциями и диаграмму; 2 - строка, отражающая основные данные о монете; 3 - панель транзакций, относящихся к выбранной монете; 4 - рассчитанные индикаторы и кнопка для пересчета; функциональные кнопки: “Recount coin”, “Add coin”, “Edit coin”, “Delete coin”, “Add transfer-transaction”, “Add exchange transaction”, , “Delete transaction”.

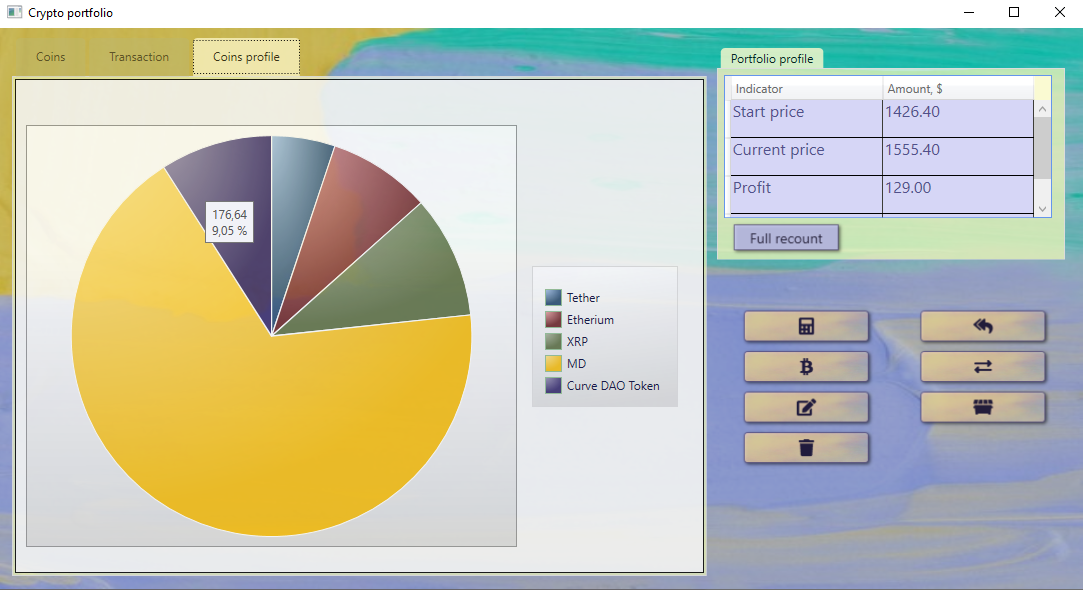
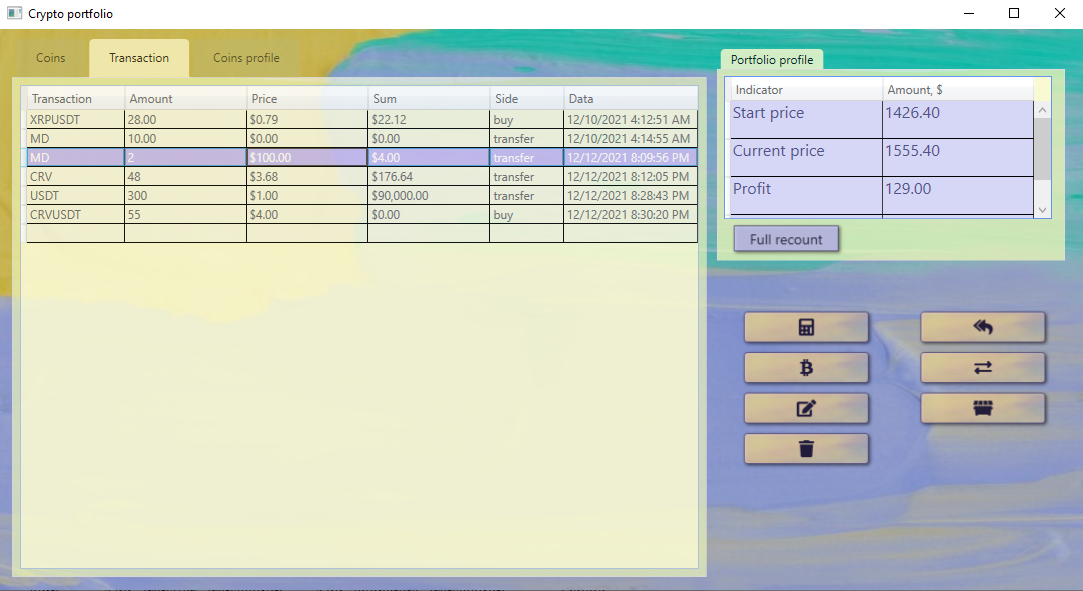
****Рисунок 18 - Главное окно приложения, вкладка Transaction

Рисунок 19 - Главное окно приложения, вкладка Coins profile

После добавления монет или транзакций обновляются автоматически Coins, Transactions и Coins profile. Для пересчета индикаторов портфеля необходимо нажатие кнопки “Full recount”. Вкладка Coins представлена на рис. 20, пересчитанная диаграмма после добавления нескольких монет представлена на рис. 21.

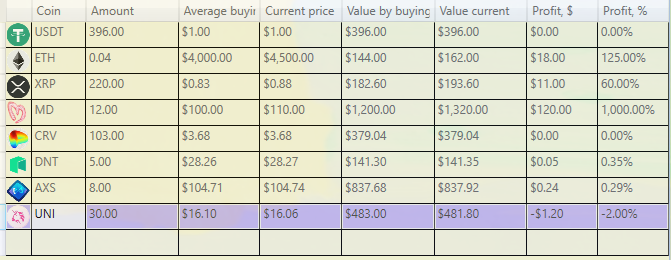
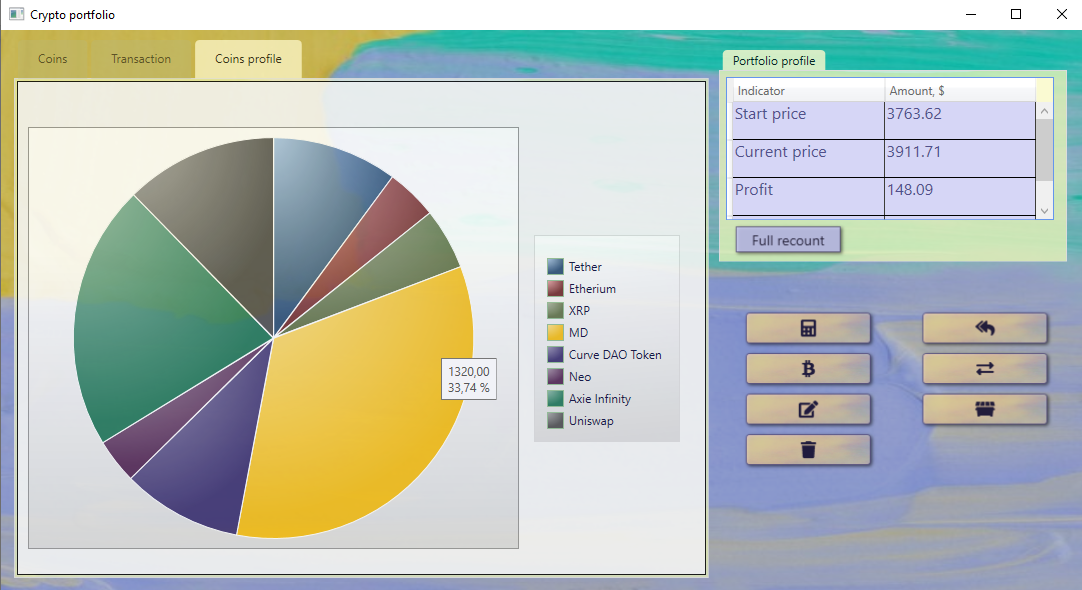


Рисунок 20 - Обновленная таблица, вкладка Coins

Рисунок 21 - Обновленная диаграмма на вкладке Coin profile

3.6 Разработка структуры классов

Для десктопного приложения разработана диаграмма классов, приведенная на рисунке 22. Отношения между классами: к одной монете может относится несколько транзакций, одна транзакция может содержать несколько монет (many-to-many).

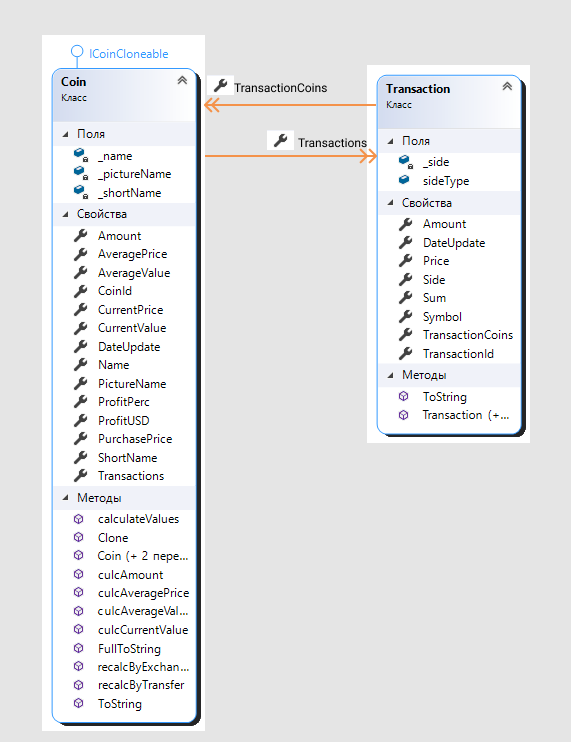


Рисунок 22 - Диаграмма классов

4 Разработка алгоритмов и их программная реализация

4.1 Алгоритм добавления монеты (десктопное приложение)

Основные операции взаимодействия с пользователем реализованы с помощью команд. Командыпредставляют механизм выполнения какой-либо задачи. Использование команд помогает нам сократить объем кода и использовать одну и ту же команду для нескольких элементов управления в различных местах программы. Таким образом, команды позволяют абстрагировать набор действий от конкретных событий конкретных элементов. Команды программного средства «Портфолио криптовалют» являются реализацией общераспространенного паттерна Команда.

Метод **Execute** предназначен для хранения логики команды. Функция CanExecute (таблица 1) возвращает *true*, если команда включена и доступна для использования, и *false*, если команда отключена. Событие CanExecuteChanged вызывается при изменении состояния команды.

|  |
| --- |
| class RelayCommand: ICommand  {  public event EventHandler CanExecuteChanged  {  add { CommandManager.RequerySuggested += value; }  remove { CommandManager.RequerySuggested -= value; }  }  […]  public RelayCommand(Action<object> executeMethod, Func<object, bool> canExecute)  {  this.executeMethod = executeMethod;  this.canExecute = canExecute;  } |

В WPF этот интерфейс реализован встроенным классом System.Windows.Input.RoutedCommand, который является базовым для всех встроенных команд. Поэтому, если нам потребуется создать свой класс команды, мы можем либо реализовать ICommand, либо унаследовать свой класс команды от RoutedCommand.

Процесс добавления новой монеты в базу реализуется через ***NewCoinCommand.***

|  |
| --- |
| #region Add coin  private ICommand \_newCoinCommand;  public ICommand NewCoinCommand =>  \_newCoinCommand ??= new RelayCommand(OnNewCoinExecuted);  private void OnNewCoinExecuted(object id)  {  var dialog = new EditCoinWindow  {  DateUpdate = DateTime.Now  };  if (dialog.ShowDialog() != true) return;  if(dialog.ShortName == null)  {  dialog.ShortName = "DEFAULT";  }  var transaction = new Transaction  {  Symbol = dialog.ShortName.ToUpper(),  Side = "transfer",  Amount = dialog.Amount,  Price = dialog.PurchasePrice,  Sum = dialog.Amount \* dialog.PurchasePrice,  };  var coin = new Coin  {  Name = dialog.Name,  ShortName = dialog.ShortName.ToUpper(),  Amount = dialog.Amount,  PurchasePrice = dialog.PurchasePrice,  CurrentPrice = dialog.CurrentPrice,  AveragePrice = dialog.PurchasePrice,  CurrentValue = dialog.CurrentPrice \* dialog.Amount,  AverageValue = dialog.PurchasePrice \* dialog.Amount,  DateUpdate = dialog.DateUpdate,  Transactions = new List<Transaction>()  {  transaction  }  };  var fileName = Path.GetFileName(dialog.PictureName);  if (fileName != null)  {  coin.PictureName = fileName;  }  else  {  coin.PictureName = "no.png";  }  if (dialog.PictureName != null)  {  try  {  var target = Path.Combine(Directory.GetCurrentDirectory(), "Images", fileName);  File.Copy(dialog.PictureName, target);  }  catch (IOException e)  {  MessageBox.Show("This file already exist. Choose another one. \n (Select coin, click button Edit)");  }  }  transaction.TransactionCoins.Add(coin);  transactionManager.CreateTransaction(transaction);  Coins.Add(coin);  AllTransactions.Add(transaction);  }  #endregion |

При нажатии на кнопку “Add coin” открывается окно “Edit coin”, которое предлагает пользователю заполнить данные по новой монете. Если поля не заполнены монета будет создаваться с дефолтными данными.

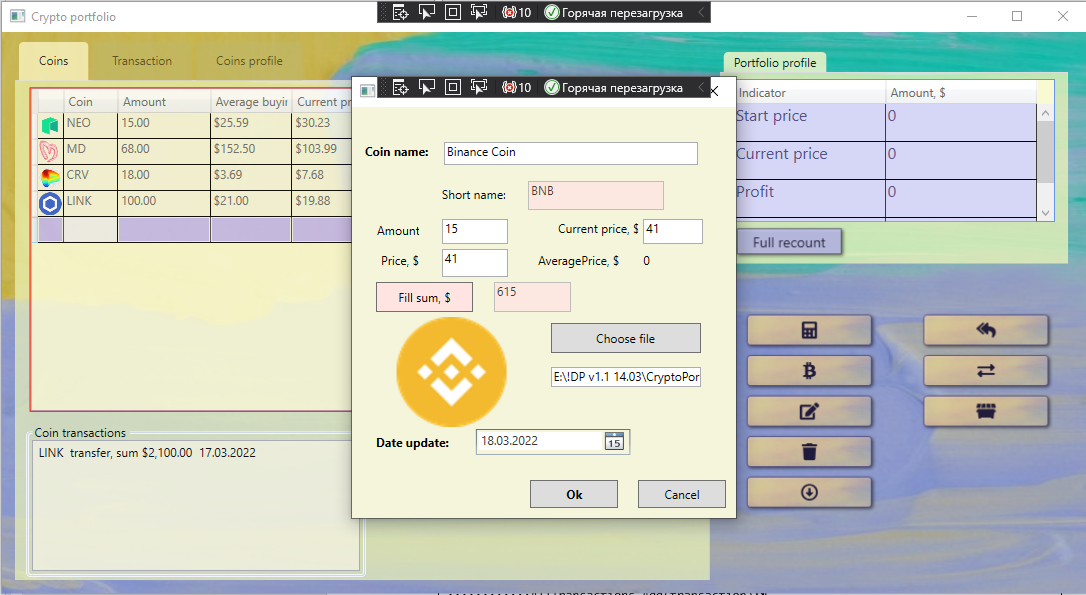


Рисунок 23 – Окно добавления монеты

После нажатия пользователем кнопки «Ok» происходит выполнение проверки корректности заполнения полей (нет ли незаполненных полей; полей с неконвертируемыми в числовые форматы значениями, где это необходимо; комбобоксов без выбранного объекта). Если метод возвращает значение false, то выводится сообщение об ошибке, а в визуальных элементах, данные в которых неверны, устанавливается красный фон. При корректных введенных данных в соответствующие публичные поля диалогового окна сохраняются введенные данные, характеризующие текущий анализ, а также вновь созданные объекты, характеризующие записи в таблицах и окно закрывается с результатом true. Схема добавления монеты в базу данных представлена на чертеже ГУИР 3220.004.

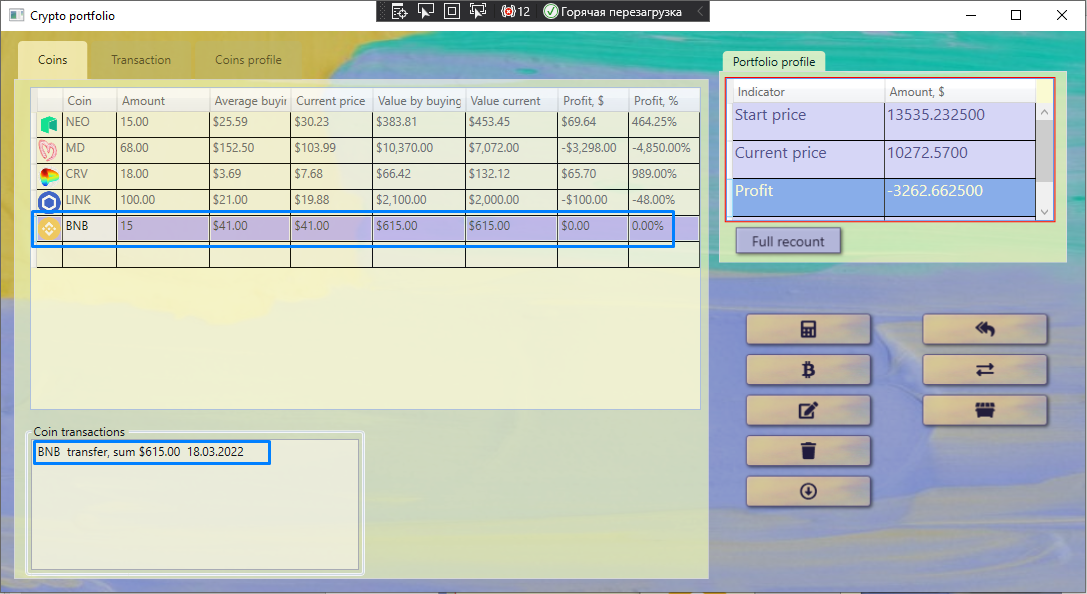


Рисунок 24 – Главное окно после добавления монеты

4.2 Алгоритм удаления монеты

Процесс удаления данных о монете из базы данных реализуется также посредством механизма команд, метод – ***DelCoinCommand.***

|  |
| --- |
| #region Delete coin  private ICommand \_deleteCoinCommand;  public ICommand DeleteCoinCommand =>  \_deleteCoinCommand ??= new RelayCommand(OnDeleteCoinExecuted, DeleteCoinCanExecute);  // check if can del  private bool DeleteCoinCanExecute(object p) =>  \_selectedCoin != null;  private void OnDeleteCoinExecuted(object id)  {  var result = MessageBox.Show("Are you sure?", $"Delete coin {\_selectedCoin.Name}?", MessageBoxButton.YesNo);  if (result == MessageBoxResult.Yes)  {  coinManager.Delete(\_selectedCoin.CoinId); //deleting only coin and relating, not transaction  OnGetTransactionExecuted(\_selectedCoin.CoinId);  Coins.Remove(SelectedCoin);  }  }  #endregion |

Перед удалением монеты пользователь выбирает ее из списка на вкладке Coins главного окна. При нажатии на кнопку “Delete coin” открывается окно, где пользователю отображается выбранная монета и пользователю предлагается подтвердить удаление. Если пользователь нажмет кнопку “Escape” удаление монеты будет отменено, если пользователь нажмет кнопку “Ok” монета будет удалена. Появится всплывающее окно с сообщением о том, что монета удалена. Вкладка Coins главного окна программного средства будет обновлена. Схема удаления монеты представлена на чертеже ГУИР 3220.005.

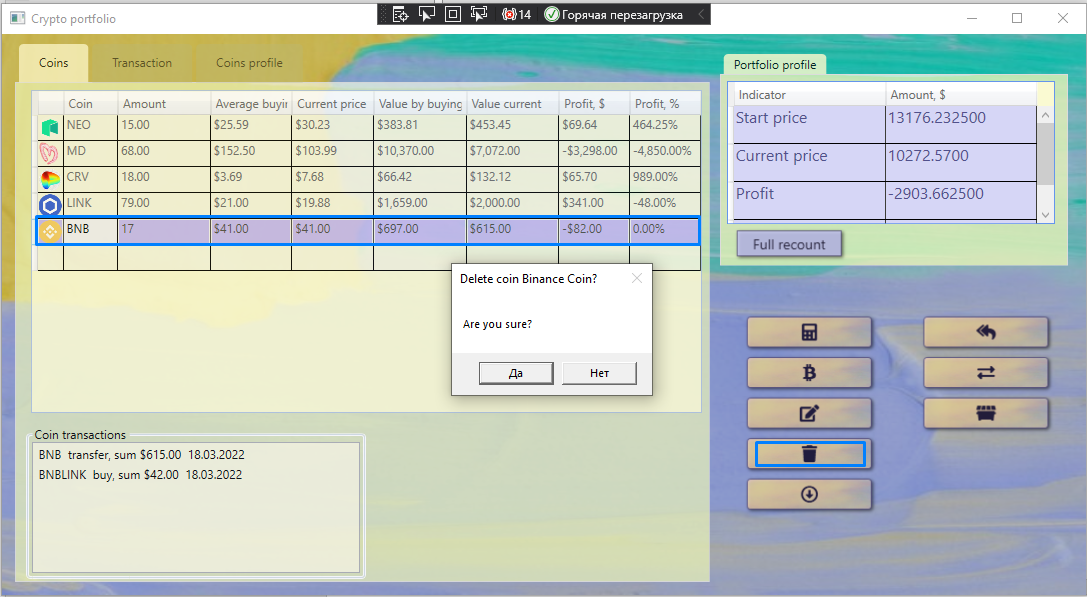


Рисунок 25 – Главное окно. Удаление монеты

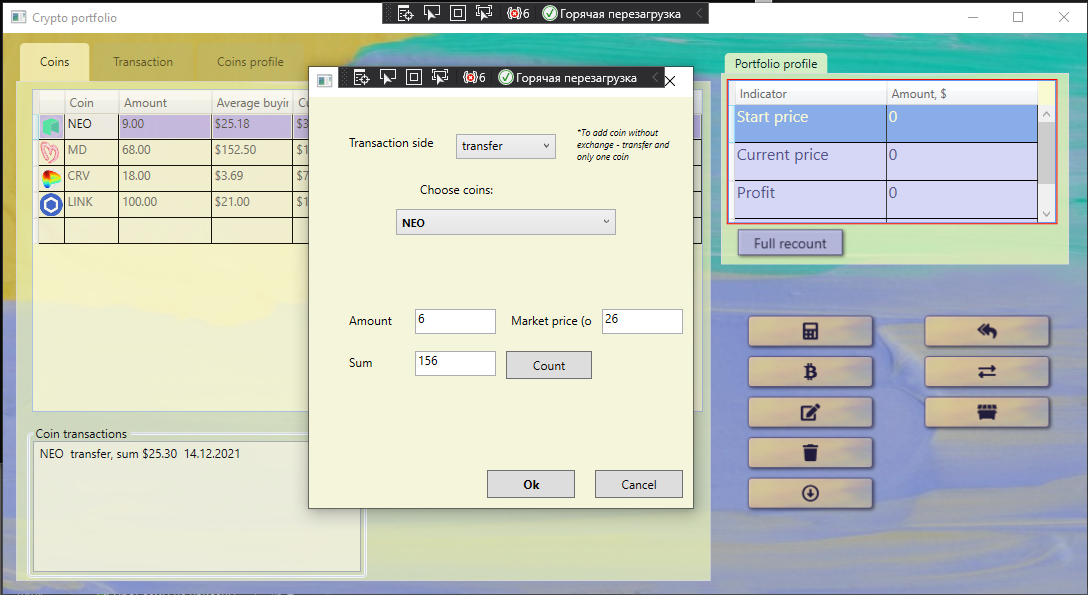
4.3 Алгоритм добавления транзакции перевода

Транзакция перевода (transfer-transaction) – транзакция увеличения объема уже имеющейся монеты. Процесс добавления новой транзакции в базу реализуется через ***NewTransferTransactionCommand*** (также реализовано через команду).

|  |
| --- |
| #region Add Transfer transaction  private ICommand \_newTransferTransactionCommand;  public ICommand NewTransferTransactionCommand =>  \_newTransferTransactionCommand ??= new RelayCommand(OnNewTransferTransactionExecuted);  private void OnNewTransferTransactionExecuted(object id)  {  var dialog = new EditTransactionWindow(id)  {  DebetCoin = \_selectedCoin,  };  if (dialog.ShowDialog() != true) return;  Coin debetCoin = coinManager.GetById(((Coin) dialog.cBoxCoinDebet.SelectedItem).CoinId);  var transaction = new Transaction()  {  DateUpdate = DateTime.Now,  Side = dialog.Side,  Symbol = debetCoin?.ShortName,  Amount = dialog.Amount,  Price = dialog.Price,  Sum = dialog.Amount \* dialog.Price,  TransactionCoins = new List<Coin>()  {  debetCoin,  }  };  debetCoin?.recalcByTransfer(dialog.Amount, dialog.Price);  coinManager.Update(debetCoin);  transactionManager.CreateTransaction(transaction);  coinManager.AddTransactionToCoin(transaction, debetCoin.CoinId);  AllTransactions.Add(transaction);  }  #endregion |

При нажатии на кнопку “Add transfer transaction” открывается окно “Edit transaction”, которое предлагает пользователю заполнить данные по транзакции (рисунок 26):

1. название транзакции в выпадающем меню выбрано автоматически (ShortName монеты);
2. пункт transfer в выпадающем меню выбран автоматически;
3. количество (поле Amount);
4. цена покупки (поле Market price);
5. поле Sum заполнится автоматически при нажатии на кнопку Count.

Рисунок 26 – добавление transfer-транзакции

После нажатия пользователем кнопки «Ok» происходит выполнение проверки корректности заполнения полей (заполнены поля Amount и Marker price). Если метод возвращает значение false, то выводится сообщение об ошибке. При корректных введенных данных в базе данных создается новый объект класса “Transaction”, соответствующий объект класса “Coin” пересчитывается (увеличивается объем и пересчитывается средняя цена – рисунок 20), обновляется главное окно приложения.

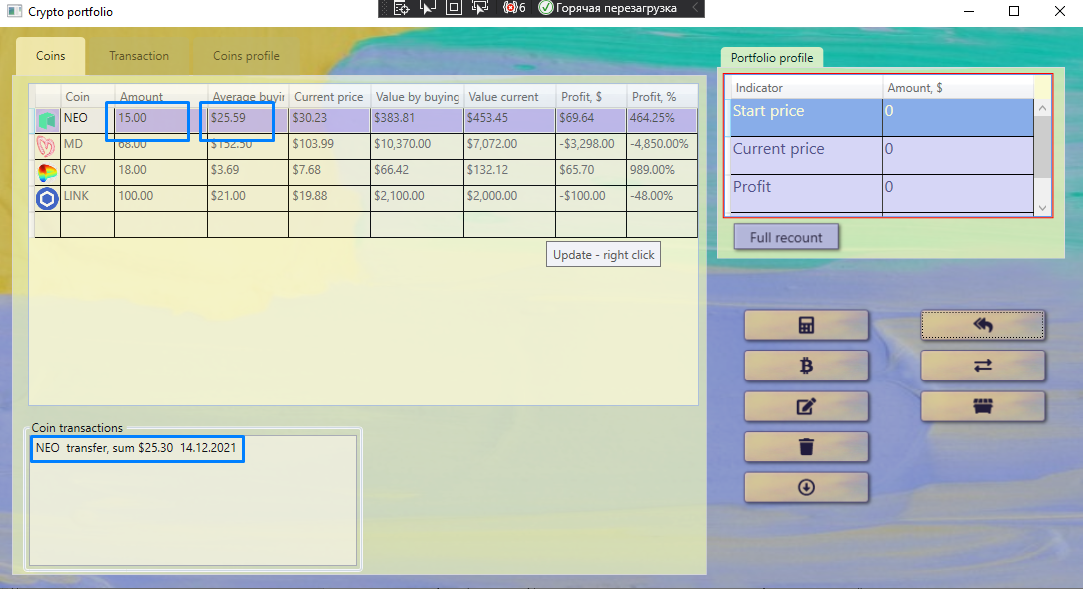
Схема добавления монеты в базу данных представлена на чертеже ГУИР 3220.005.

Рисунок 27 - Главное окно приложения после добавления transfer-транзакции.

4.4 Алгоритм добавления транзакции обмена

Транзакция обмена (exchange transaction) – транзакция изменения объема двух уже имеющихся монет. Процесс добавления новой транзакции обмена в базу также реализуется через команду – ***NewTransactionCommand:***

|  |
| --- |
| #region Add transaction  private ICommand \_newTransactionCommand;  public ICommand NewTransactionCommand =>  \_newTransactionCommand ??= new RelayCommand(OnNewTransactionExecuted);  private void OnNewTransactionExecuted(object id)  {  var dialog = new EditTransactionWindow  {  DebetCoin = \_selectedCoin,  };  if (dialog.ShowDialog() != true) return;  Coin debetCoin = (Coin) dialog.cBoxCoinDebet.SelectedItem;  Coin creditCoin = (Coin) dialog.cBoxCoinCredit.SelectedItem;  Coin outDebetCoin;  Coin outCreditCoin;  Coin.recalcByExchangeTransaction(debetCoin, creditCoin, dialog.Amount, dialog.Price, dialog.Side,  out outDebetCoin, out outCreditCoin);  var transaction = new Transaction  {  DateUpdate = DateTime.Now,  Side = dialog.Side,  Symbol = debetCoin != null ? debetCoin.ShortName + creditCoin?.ShortName : "empty DebetCoin",  Amount = dialog.Amount,  Price = dialog.Price,  Sum = dialog.Amount \* dialog.Price,  TransactionCoins = new List<Coin>(2)  {  debetCoin,  creditCoin,  }  };  coinManager.Update(debetCoin);  coinManager.Update(creditCoin);  transactionManager.CreateTransaction(transaction);  coinManager.AddTransactionToCoin(transaction, debetCoin.CoinId, debetCoin.CoinId);  AllTransactions.Add(transaction);  }  #endregion |

При нажатии на кнопку “Add exchange transaction” открывается окно “Edit transaction”, которое предлагает пользователю заполнить данные по транзакции (рисунок 28):

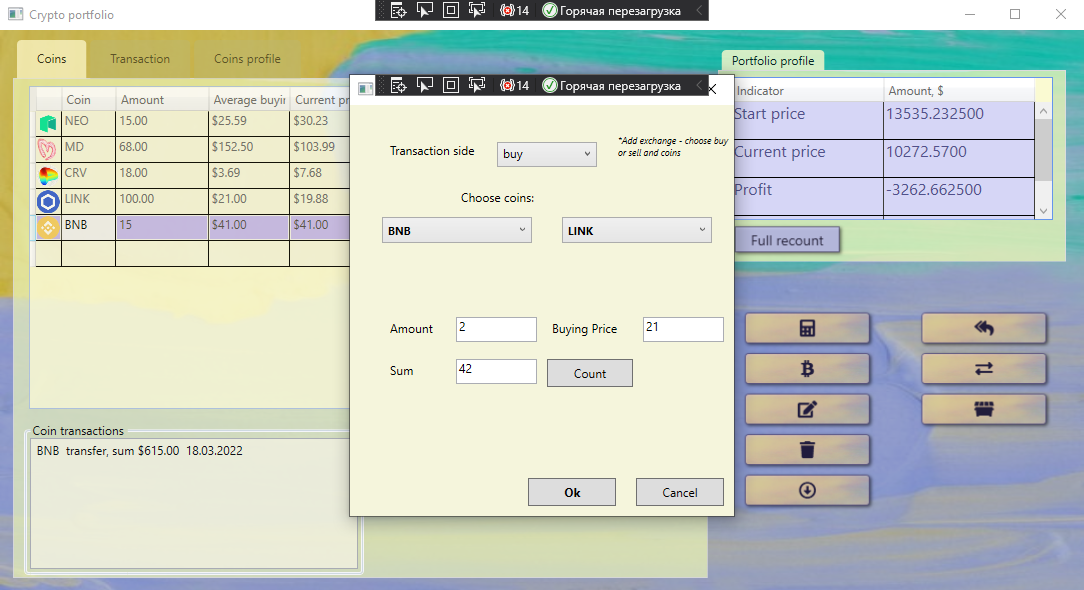
1. названия монет участвующих в транзакции;
2. пункт buy или sell в выпадающем меню;
3. количество (поле Amount);
4. цена покупки (поле Buying price);
5. поле Sum заполнится автоматически при нажатии на кнопку Count.

Рисунок 28 – Добавление exchange-транзакции

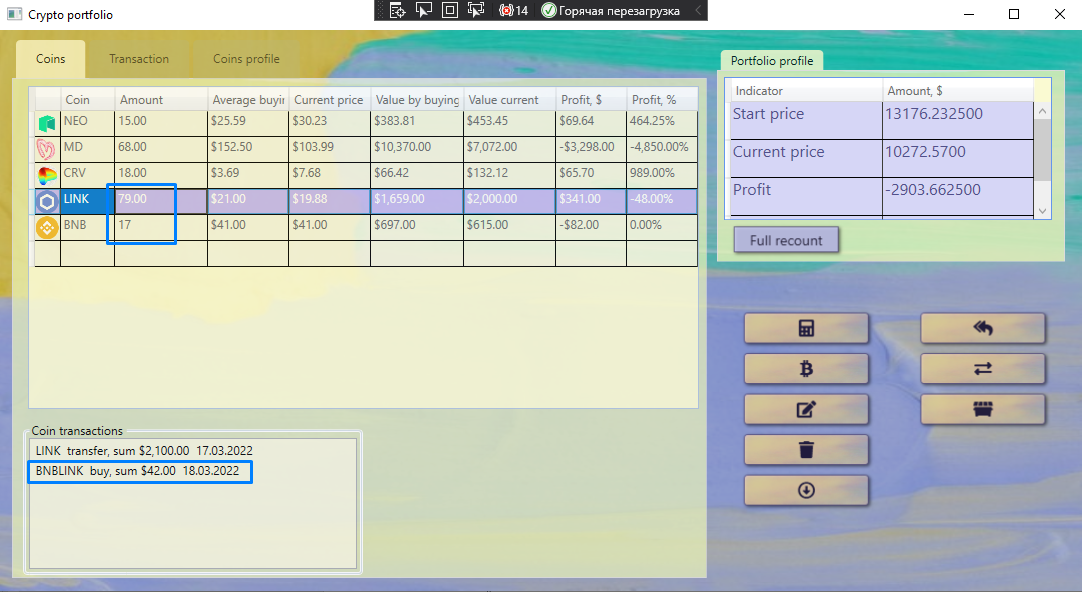


Рисунок 29 – Главное окно после добавления exchange-транзакции

4.5 Алгоритм авторизации пользователя (веб-приложение)

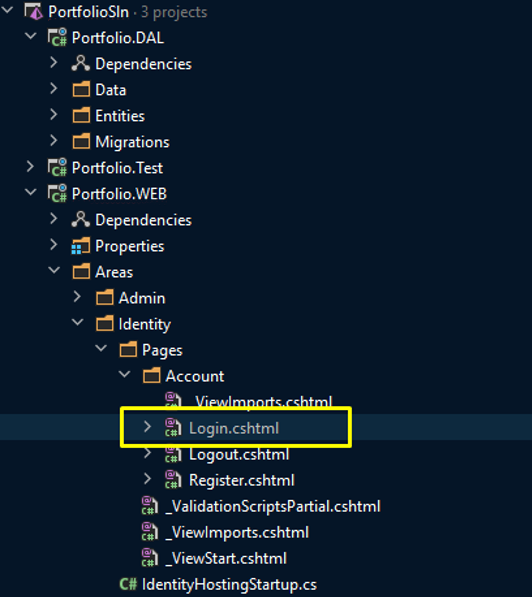
Авторизация - предоставление определённому лицу или группе лиц прав на выполнение определённых действий, а также процесс проверки (подтверждения) данных прав при попытке выполнения этих действий. Для регистрации регистрировать учетные записи пользователей в проекте используется Core Identity. ASP.NET Core Identity– это система членства, позволяющая регистрировать учетные записи пользователей, регистрировать роли и назначать роли пользователям для реализации механизма аутентификации и авторизации.

В ASP.NET Core аутентификация обрабатывается службой IAuthenticationService, которая регистрируется в качестве сервиса используется промежуточным программным обеспечением (Middleware) аутентификации. Компонент middleware аутентификации, сериализует данные пользователя в зашифрованные аутентификационные куки и передает их на сторону клиента.

При получении запроса от клиента, в котором содержатся аутентификационные куки, происходит их валидация, десериализация и инициализация свойства User объекта HttpContext.

Свойство User является объектом класса ClaimsPrincipal. Один из методов ClaimsPrincipal - Identity: возвращает объект ClaimsIdentity, который реализует интерфейс IIdentity и представляет текущего пользователя. Для работы с пользователями и ролями в пространстве имен Microsoft.AspNetCore.Identity описаны классы менеджеров.

Для авторизации используем ***SignInManager<TUser>*** – реализует функции входа в/выхода из системы пользователя и формирует куки аутентификации. UserManager вызывается в LoginPartial (рисунок 23).

Рисунок 30 – Структура Identity веб-сервиса

Алгоритм запускается нажатием кнопки «Войти», доступной любому пользователю. Далее происходит проверка введенных данных со списком зарегистрированных пользователей в таблице Пользователи. Вся информация храниться в таблице Users, которая заполняется в процессе регистрации по таким полям как: email, Пароль.

|  |
| --- |
| public async Task<IActionResult> OnPostAsync(string returnUrl = null)  {  returnUrl = returnUrl ?? Url.Content("~/");  if (ModelState.IsValid)  {  var result = await \_signInManager.PasswordSignInAsync(Input.Email, Input.Password, Input.RememberMe, lockoutOnFailure: false);  if (result.Succeeded)  {  \_logger.LogInformation("User logged in.");  return LocalRedirect(returnUrl);  }  if (result.RequiresTwoFactor)  {  return RedirectToPage("./LoginWith2fa", new { ReturnUrl = returnUrl, RememberMe = Input.RememberMe });  }  if (result.IsLockedOut)  {  \_logger.LogWarning("User account locked out.");  return RedirectToPage("./Lockout");  }  else  {  ModelState.AddModelError(string.Empty, "Invalid login attempt.");  return Page();  }  }  return Page();  } |

Отличие авторизации разрабатываемого проекта состоит в наличии двух типов пользователя: администратор и пользователь. Тип пользователя администратор предполагается использовать при расширении сервиса и обеспечения сохранности данных путем ограничения функциональности в зависимости от ролей.

Схема авторизации пользователя данных представлена на чертеже ГУИР 003220.007.

4.6 Алгоритм обновления текущей цены

Данная функция реализована частично и представляет собой симуляцию подключения к бирже и обновления текущей цены каждой монеты. Реализовано также посредством паттерна Команда (***UpdateCoinsCommand***)

|  |
| --- |
| //simulation  #region UpdateCoins  private ICommand \_updateCoinsCommand;  public ICommand UpdateCoinsCommand =>  \_updateCoinsCommand ??= new RelayCommand(OnUpdateCoinsExecuted);  private void OnUpdateCoinsExecuted(object id)  {  MessageBoxTimeout((System.IntPtr)0, "Check connection", "Message", 0, 0, 1000);  MessageBoxTimeout((System.IntPtr)0, "Updating", "Message", 0, 0, 4000);  Random rand = new Random(-1);  Random randSign = new Random();  foreach (var coin in Coins)  {  var alteration = (coin.CurrentPrice \* (decimal)rand.NextDouble())/10;  var sign = randSign.Next(-1,2);  //Console.WriteLine($"alteration - { alteration}");  //Console.WriteLine($"sign- {sign}");  coin.CurrentPrice = coin.CurrentPrice + alteration\*sign;  //Console.WriteLine($"price before - {coin.CurrentPrice}, price after update - {coin.CurrentPrice}");  }  }  #endregion |

5 Тестирование созданного программного средства

5.1. Методы тестирования

Среди всевозможных методов тестирования наиболее эффективным в нашем случае будет воспользоваться функциональным ручным тестированием десктопного приложения, а для веб-сервиса разработать автоматизорованные тесты.

Планируемые методы тестирования перечислены ниже.

**Функциональное тестирование** (functional testing) — вид тестирования, направленный на проверку корректности работы функциональности приложения (корректность реализации функциональных требований).

**Ручное тестирование** (manual testing) — тестирование, в котором тест-кейсы выполняются человеком вручную без использования средств автоматизации. От тестировщика требуются такие качества, как терпеливость, наблюдательность, креативность, умение ставить нестандартные эксперименты, а также умение видеть и понимать, что происходит «внутри системы», т.е. как внешние воздействия на приложение трансформируются в его внутренние процессы.

**Автоматизированное тестирование** (automated testing, test automation) — набор техник, подходов и инструментальных средств, позволяющий исключить человека из выполнения некоторых задач в процессе тестирования. Тест-кейсы частично или полностью выполняет специальное инструментальное средство, однако разработка тест-кейсов, подготовка данных, оценка результатов выполнения, написания отчётов об обнаруженных дефектах — всё это и многое другое по-прежнему делает человек.

**Модульное (компонентное) тестирование** (unit testing, module testing, component testing) направлено на проверку отдельных небольших частей приложения, которые (как правило) можно исследовать изолированно от других подобных частей. При выполнении данного тестирования могут проверяться отдельные функции или методы классов, сами классы, взаимодействие классов, небольшие библиотеки, отдельные части приложения.

5.2 Разработка тестов и тестирование

Тестирование программного средства – проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом. В более широком смысле, тестирование – это одна из техник контроля качества, включающая в себя активности по планированию работ (Test Management), проектированию тестов (Test Design), выполнению тестирования (Test Execution) и анализу полученных результатов (Test Analysis) [12].

Запуск и тестирование десктопного программного средства программы выполнялось на операционной системе Windows 10 для одного языка 64-бит. Запуск и тестирование веб сервиса в браузерах Chrome, Mozilla, Opera и в программном средстве Postman.

Для тестирования веб-сервиса также разработаны автотесты.

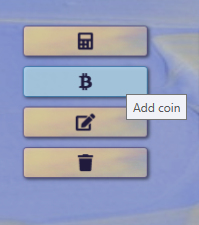
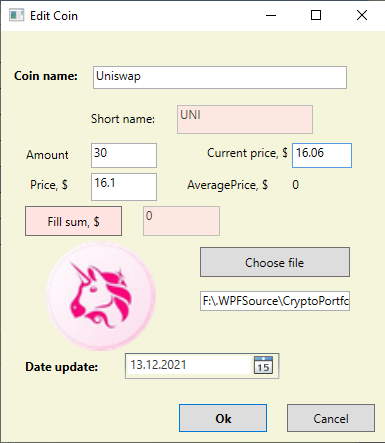
Рассмотрим функционал, предлагаемый пользователю по добавлению новой монеты. Вызов данной функциональности осуществляется по кнопке «Add coin» (рисунок 31).

Рисунок 31 – Кнопка добавления монеты

Вызывается окно добавления монеты, предоставляющее возможность создания новой монеты (рисунок 32).

Рисунок 32 – Окно редактирования монеты

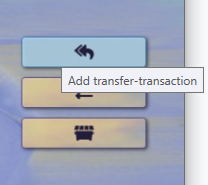
Окна для ввода информации о новых транзакциях вызывается по клику на кнопке «Add transfer-transaction» (рис. 33) или «Add exchange transaction» (рис. 34).

Рисунок 33 – Кнопка вызова окна добавления трансфер-транзакции

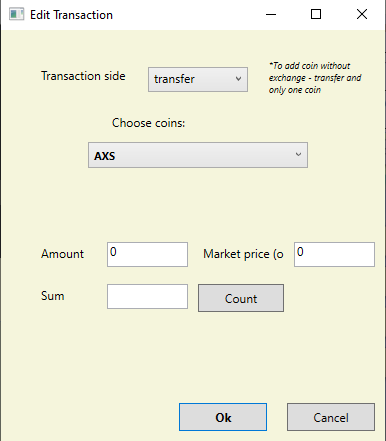


Рисунок 34 - Окно добавления трансфер-транзакции

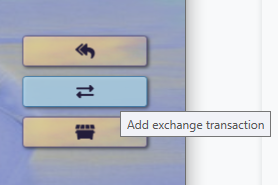
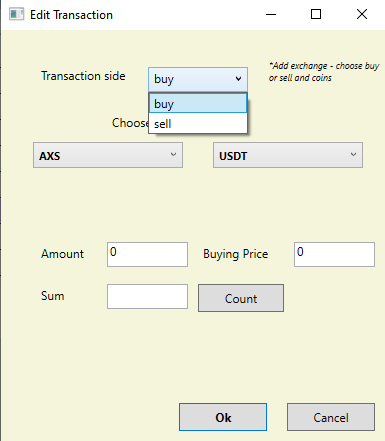
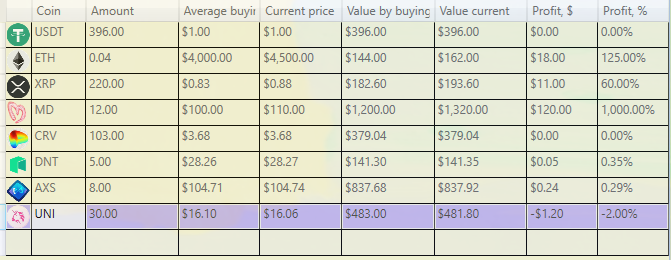


Рисунок 35 – Кнопка вызова окно добавления транзакции обмена монет

Рисунок 36 - Окно добавления транзакции обмена монет

После добавления монет или транзакций обновляются автоматически Coins, Transactions и Coins profile. Для пересчета индикаторов портфеля необходимо нажатие кнопки “Full recount”. Вкладка Coins представлена на рисунке 37, пересчитанная диаграмма после добавления нескольких монет представлена на рисунке 38.



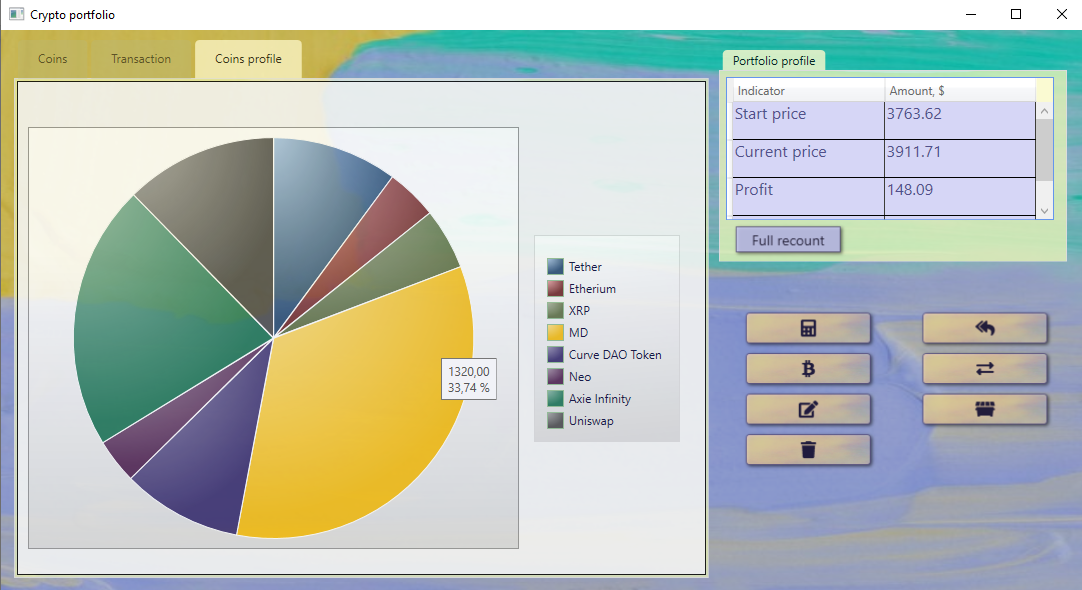
Рисунок 37 - Обновленная таблица, вкладка Coins

Рисунок 38 - Обновленная диаграмма на вкладке Coin profile

**5.3** **Функциональное тестирование программного средства**

Тестовое покрытие – критерий, отображающий добротность тестирования. Характеризует полноту охвата тестами программного кода либо требований к нему. Основной подход к оцениванию – формирование тестового пула. Значение тестового покрытия кода находится в прямой зависимости от количества отобранных вариантов проверки к нему. Многозадачность и универсальность современного софта обусловливает невозможность организации тестового покрытия с показателем в 100%. Так что для максимального охвата тестируемого кода, разработаны особые приёмы и инструменты [12].

В данном дипломном проекте применяется дымовое тестирование.

Понятие дымовое тестирование пошло из инженерной среды: «При вводе в эксплуатацию нового оборудования («железа») считалось, что тестирование прошло удачно, если из установки не пошел дым» [12].

В области же программного обеспечения, дымовое тестирование рассматривается как короткий цикл тестов, выполняемый для подтверждения того, что после сборки кода (нового или исправленного) устанавливаемое приложение, стартует и выполняет основные функции.

Вывод о работоспособности основных функций делается на основании результатов поверхностного тестирования наиболее важных модулей приложения на предмет возможности выполнения требуемых задач и наличия быстро находимых критических и блокирующих дефектов. В случае отсутствия таковых дефектов дымовое тестирование объявляется пройденным, и приложение передается для проведения полного цикла тестирования, в противном случае, дымовое тестирование объявляется проваленным, и приложение уходит на

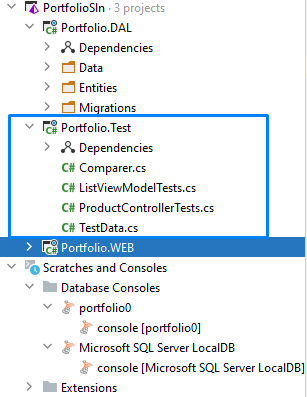
доработку [12]. Для тестирования веб-сервиса также разработаны тесты – проект «Portfolio.Test» - рисунок 39.

Рисунок 39 Тестирование веб-сервиса

Заключение

При работе над проектом была разработана программные средства для ведения учёта криптовалют. Программы поможет облегчить ведение учёта и анализа криптовалют и транзакций по ним, а также даст возможность получения наглядного представления распределения объема монет в портфеле. Программа рассчитана для использования специалистами в криптоиндустрии.

В ходе выполнения работы были применены знания полученные при изучении курса “Средства визуального программирования приложений”, “ Организация и проектирование база данных”, “ Технологии проектирования программного обеспечения информационных систем”, “ Объектно-ориентированное программирование”, “Web-технологии”, “Виртуализация и облачные вычисления” и др.

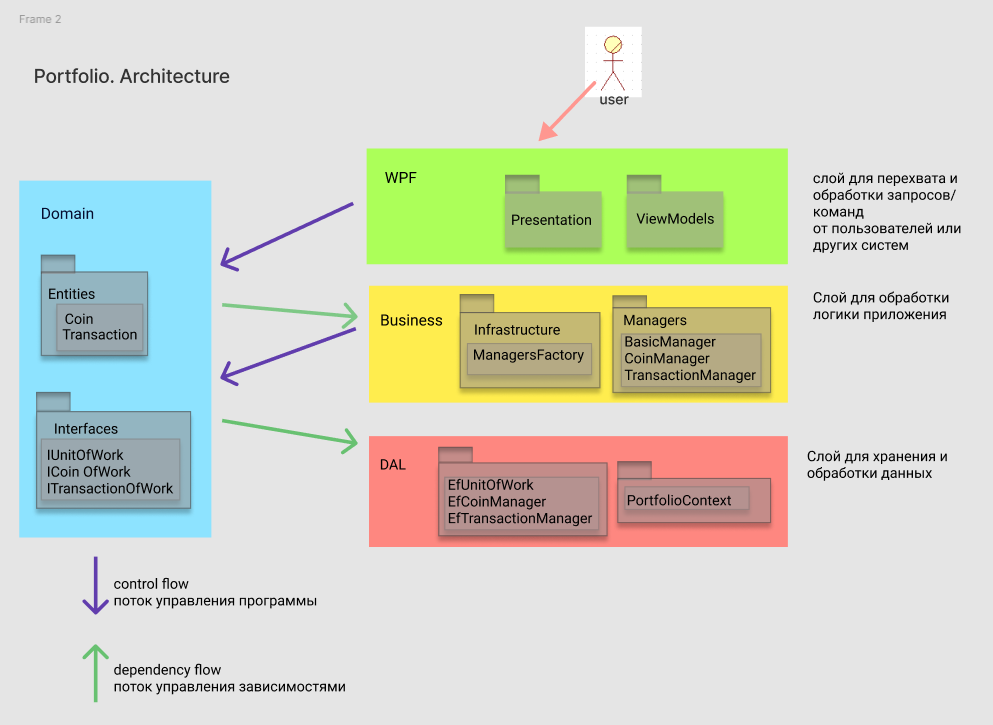
В проекте использованы наиболее актуальные и современные технологии и методологии, применяемые при разработке современных программных продуктов основанных на принципах ООП.

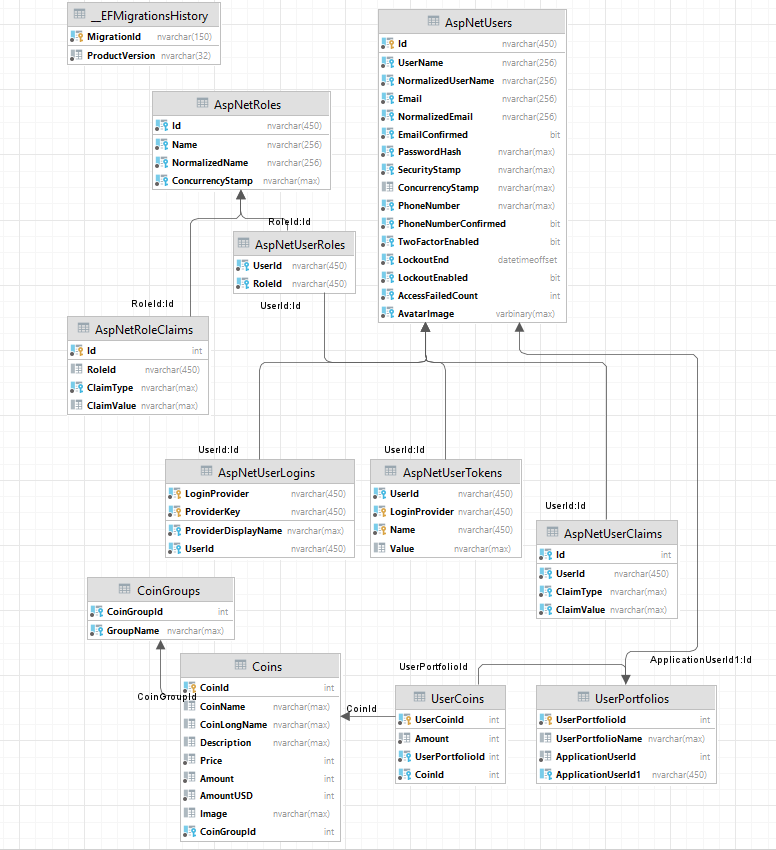
Все поставленные на дипломное проектирование задачи выполнены полностью.

Список использованных источников

1. Чистая архитектура. Искусство разработки программного обеспечения. - Роберт С. Мартин. - СПб.: Питер, 2018. — 352.
2. Э.Троелсен, Ф.Джепикс: Язык программирования C# 7 и платформы .NET и .NET Core.
3. CLR via C# Программирование на платформе Microsoft .NET Д. Рихтер
4. Методические указания к лабораторным работам по СВПП. - Гламаздин И.И.
5. Видеокурс Роман Труфанов. Занятие “Слоеная структура решений“ [Электронный ресурс]. Режим доступа <https://www.youtube.com/watch?v=7RaOja0NsPc&ab_channel=RomanTrufanov>, свободный. — Загл. с экрана.
6. Руководство по WPF [Электронный ресурс]. - Режим доступа <https://metanit.com/sharp/wpf/>, свободный. — Загл. с экрана.
7. Windows Presentation Foundation [Электронный ресурс]. - Режим доступа <https://professorweb.ru/my/WPF/base_WPF/level1/info_WPF.php>, свободный. — Загл. с экрана.
8. Чистая архитектура Роберт С. Мартин - [Электронный ресурс]. - Режим доступа w, свободный. — Загл. с экрана.
9. Руководство по ASP.NET 6.0 [Электронный ресурс]. - Режим доступа <https://metanit.com/sharp/wpf/>, свободный. — Загл. с экрана.
10. Введение в С#. Язык C# и платформа .NET [Электронный ресурс] – Режим доступа: https://metanit.com/sharp/tutorial/1.1.php. – Дата доступа: 07.03.2019.
11. Microsoft SQL Server [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Microsoft\_SQL\_Server. – Дата доступа: 07.03.2019.
12. Тестирование. Фундаментальная теория [Электронный ресурс] – Режим доступа: https://habrahabr.ru/post/279535/. – Дата доступа: 08.02.2019.
13. Microsoft SQL Server [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Microsoft\_SQL\_Server. – Дата доступа: 07.03.2019.

Приложение А   
Нормативно-справочные данные





Приложение Б   
Текст программы

Десктопное приложение «Портфолио криптовалют»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Input;

namespace portfolio.Command

{

class RelayCommand: ICommand

{

public event EventHandler CanExecuteChanged

{

add { CommandManager.RequerySuggested += value; }

remove { CommandManager.RequerySuggested -= value; }

}

/// <summary>

/// Признак CanExecute

/// </summary>

private Func<object, bool> canExecute;

/// <summary>

/// делегат метода Execute

/// </summary>

private Action<object> executeMethod;

/// <summary>

/// Конструктор

/// </summary>

/// <param name="executeMethod">делегат метода Execute</param>

/// <param name="canExecute">делегат признака Can Execute</param>

public RelayCommand(Action<object> executeMethod, Func<object, bool> canExecute)

{

this.executeMethod = executeMethod;

this.canExecute = canExecute;

}

/// <summary>

/// Конструктор

/// </summary>

/// <param name="executeMethod">делегат метода Execute</param>

public RelayCommand(Action<object> executeMethod)

{

this.executeMethod = executeMethod;

}

public bool CanExecute(object parameter)

{

return canExecute?.Invoke(parameter) ?? true;

}

public void Execute(object parameter)

{

executeMethod(parameter);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Data;

using System.Globalization;

using System.IO;

using System.Windows.Media.Imaging;

using System.Diagnostics;

namespace portfolio.Infrastructure

{

public class ImageSourceConverter : IValueConverter

{

string root = Directory.GetCurrentDirectory();

string ImageDirectory => Path.Combine(root, "Images");

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

return Path.Combine(ImageDirectory, (string)value);

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

throw new NotImplementedException();

}

}

}

using portfolio.Business.Infrastructure;

using portfolio.Business.Managers;

using portfolio.Command;

using portfolio.Domain.Entities;

using System;

using System.Collections.Generic;

using System.Windows.Controls.DataVisualization.Charting;

using System.Windows.Input;

using portfolio.Business;

namespace portfolio.ViewModels

{

public class MainWindowViewModel : ViewModelBase

{

ManagersFactory factory;

CoinManager coinManager;

TransactionManager transactionManager;

private string titleCoins = "Coin Window";

private string titleTransactions = "Transaction Window";

private string titleCoinPerformance = "Coin Performance Window";

#region Public properties

public static ObservableCollection<Coin> Coins { get; set; }

public static ObservableCollection<Transaction> AllTransactions { get; set; }

public static ObservableCollection<PortfolioIndicator> PortfolioIndicators { get; set; }

public ObservableCollection<Transaction> TransactionsFromCoin { get; set; }

[System.Runtime.InteropServices.DllImport("user32.dll", SetLastError = true)]

static extern int MessageBoxTimeout(IntPtr hwnd, String text, String title,

uint type, Int16 wLanguageId, Int32 milliseconds);

public MainWindowViewModel()

{

factory = new ManagersFactory("DefaultConnection");

coinManager = factory.GetCoinManager();

transactionManager = factory.GetTransactionManager();

//init db

if (coinManager.Coins.Count() == 0)

{

DbTestData.SetupData(coinManager, transactionManager);

}

Coins = new ObservableCollection<Coin>(coinManager.Coins);

Coins.CollectionChanged += CoinsOnCollectionChanged;

AllTransactions = new ObservableCollection<Transaction>(transactionManager.Transactions);

AllTransactions.CollectionChanged += AllTransactionsOnCollectionChanged;

TransactionsFromCoin = new ObservableCollection<Transaction>();

PortfolioIndicators = new ObservableCollection<PortfolioIndicator>();

PortfolioIndicators.CollectionChanged += PortfolioIndicatorsOnCollectionChanged;

var startPrice = new PortfolioIndicator

{

IndicatorName = "Start price"

};

PortfolioIndicators.Add(startPrice);

var currentPrice = new PortfolioIndicator

{

IndicatorName = "Current price"

};

PortfolioIndicators.Add(currentPrice);

var profit = new PortfolioIndicator

{

IndicatorName = "Profit"

};

PortfolioIndicators.Add(profit);

//CollectionChanged

if (Coins.Count > 0)

OnGetTransactionExecuted(Coins[0].CoinId);

}

private void PortfolioIndicatorsOnCollectionChanged(object? sender, NotifyCollectionChangedEventArgs e)

{

switch (e.Action)

{

case NotifyCollectionChangedAction.Replace:

PortfolioIndicator replasedIndicator = e.OldItems[0] as PortfolioIndicator;

MessageBoxTimeout((System.IntPtr) 0, $"{replasedIndicator.IndicatorName} recounted.",

"Indicators", 0, 0, 1000);

break;

}

}

private void AllTransactionsOnCollectionChanged(object? sender, NotifyCollectionChangedEventArgs e)

{

switch (e.Action)

{

case NotifyCollectionChangedAction.Add:

Transaction transaction = e.NewItems[0] as Transaction;

MessageBoxTimeout((System.IntPtr) 0, $"{transaction.Symbol} added.", "Transaction", 0, 0, 2000);

break;

}

}

private static void CoinsOnCollectionChanged(object sender, NotifyCollectionChangedEventArgs e)

{

switch (e.Action)

{

case NotifyCollectionChangedAction.Add:

Coin coin = e.NewItems[0] as Coin;

MessageBoxTimeout((System.IntPtr) 0, $"{coin.Name} - {coin.ShortName} added.", "Coins", 0, 0, 2000);

break;

case NotifyCollectionChangedAction.Remove:

Coin oldCoin = e.OldItems[0] as Coin;

MessageBoxTimeout((System.IntPtr) 0, $"{oldCoin.Name} deleted.", "Coins", 0, 0, 2000);

break;

case NotifyCollectionChangedAction.Replace:

Coin replasedCoin = e.OldItems[0] as Coin;

Coin replasingCoin = e.NewItems[0] as Coin;

MessageBoxTimeout((System.IntPtr) 0, $"{replasedCoin.Name} replased {replasingCoin.Name}.", "Coins",

0, 0, 2000);

break;

}

}

#region selected coin

private Coin \_selectedCoin;

public Coin SelectedCoin

{

get => \_selectedCoin;

set { Set(ref \_selectedCoin, value); }

}

#endregion

#region selected transaction

private Transaction \_selectedTransaction;

public Transaction SelectedTransaction

{

get => \_selectedTransaction;

set { Set(ref \_selectedTransaction, value); }

}

#endregion

#endregion

#region Command

#region Add coin

private ICommand \_newCoinCommand;

public ICommand NewCoinCommand =>

\_newCoinCommand ??= new RelayCommand(OnNewCoinExecuted);

private void OnNewCoinExecuted(object id)

{

var dialog = new EditCoinWindow

{

DateUpdate = DateTime.Now

};

if (dialog.ShowDialog() != true) return;

if(dialog.ShortName == null)

{

dialog.ShortName = "DEFAULT"; //default

}

var transaction = new Transaction

{

Symbol = dialog.ShortName.ToUpper(),

Side = "transfer",

Amount = dialog.Amount,

Price = dialog.PurchasePrice,

Sum = dialog.Amount \* dialog.PurchasePrice,

};

var coin = new Coin

{

Name = dialog.Name,

ShortName = dialog.ShortName.ToUpper(),

Amount = dialog.Amount,

PurchasePrice = dialog.PurchasePrice,

CurrentPrice = dialog.CurrentPrice,

AveragePrice = dialog.PurchasePrice,

CurrentValue = dialog.CurrentPrice \* dialog.Amount,

AverageValue = dialog.PurchasePrice \* dialog.Amount,

DateUpdate = dialog.DateUpdate,

Transactions = new List<Transaction>()

{

transaction

}

};

var fileName = Path.GetFileName(dialog.PictureName);

if (fileName != null)

{

coin.PictureName = fileName;

}

else

{

coin.PictureName = "no.png";

}

if (dialog.PictureName != null)

{

try

{

var target = Path.Combine(Directory.GetCurrentDirectory(), "Images", fileName);

File.Copy(dialog.PictureName, target);

}

catch (IOException e)

{

MessageBox.Show("This file already exist. Choose another one. \n (Select coin, click button Edit)");

}

}

transaction.TransactionCoins.Add(coin);

transactionManager.CreateTransaction(transaction);

Coins.Add(coin);

AllTransactions.Add(transaction);

}

#endregion

#region Choose coin in list

private ICommand \_getTransactionsFromCoinCommand;

public ICommand GetTransactionsFromCoinCommand =>

\_getTransactionsFromCoinCommand

??= new RelayCommand(OnGetTransactionExecuted);

private void OnGetTransactionExecuted(object id)

{

TransactionsFromCoin.Clear();

if (id is not null)

{

var transactions = coinManager.GetTransactionsOfCoin((int) id);

foreach (var transaction in transactions)

TransactionsFromCoin.Add(transaction);

}

}

#endregion

#region Edit coin

private ICommand \_editCoinCommand;

public ICommand EditCoinCommand =>

\_editCoinCommand ??= new RelayCommand(OnEditCoinExecuted, EditCoinCanExecute);

// check if can edit

private bool EditCoinCanExecute(object p) =>

\_selectedCoin != null;

private void OnEditCoinExecuted(object id)

{

var dialog = new EditCoinWindow

{

Name = \_selectedCoin.Name,

ShortName = \_selectedCoin.ShortName,

Amount = \_selectedCoin.Amount,

PurchasePrice = \_selectedCoin.PurchasePrice,

CurrentPrice = \_selectedCoin.CurrentPrice,

AveragePrice = \_selectedCoin.AveragePrice,

DateUpdate = \_selectedCoin.DateUpdate,

PictureName = \_selectedCoin.PictureName

};

if (dialog.ShowDialog() != true) return;

// to Images

var imageFolderPass = Path.Combine(Directory.GetCurrentDirectory(), "Images");

// for new picture

try

{

if (!\_selectedCoin.PictureName.Equals(dialog.PictureName))

{

// delete old picture

//\_selectedCoin.PictureName = null;

File.Delete(Path.Combine(imageFolderPass, \_selectedCoin.PictureName));

// get new picture

var newImage = Path.GetFileName(dialog.PictureName);

// copy file to Images

File.Copy(dialog.PictureName, Path.Combine(imageFolderPass, newImage));

\_selectedCoin.PictureName = newImage;

}

}

catch

{

\_selectedCoin.PictureName = dialog.PictureName;

}

\_selectedCoin.Name = dialog.Name;

\_selectedCoin.ShortName = dialog.ShortName;

\_selectedCoin.Amount = dialog.Amount;

\_selectedCoin.PurchasePrice = dialog.PurchasePrice;

\_selectedCoin.CurrentPrice = dialog.CurrentPrice;

\_selectedCoin.CurrentValue = dialog.Amount \* dialog.CurrentPrice;

\_selectedCoin.AverageValue = dialog.Amount \* dialog.PurchasePrice;

\_selectedCoin.DateUpdate = dialog.DateUpdate;

coinManager.Update(SelectedCoin);

OnGetTransactionExecuted(\_selectedCoin.CoinId);

}

#endregion

#region Delete coin

private ICommand \_deleteCoinCommand;

public ICommand DeleteCoinCommand =>

\_deleteCoinCommand ??= new RelayCommand(OnDeleteCoinExecuted, DeleteCoinCanExecute);

// check if can del

private bool DeleteCoinCanExecute(object p) =>

\_selectedCoin != null;

private void OnDeleteCoinExecuted(object id)

{

var result = MessageBox.Show("Are you sure?", $"Delete coin {\_selectedCoin.Name}?", MessageBoxButton.YesNo);

if (result == MessageBoxResult.Yes)

{

coinManager.Delete(\_selectedCoin.CoinId);

OnGetTransactionExecuted(\_selectedCoin.CoinId);

Coins.Remove(SelectedCoin);

}}

#endregion

#region Calculate coin Values

private ICommand \_countCoinCommand;

public ICommand CountCoinCommand =>

\_countCoinCommand ??= new RelayCommand(OnCulcCoinExecuted, CulcCoinCanExecute);

// check if can del

private bool CulcCoinCanExecute(object p) =>

\_selectedCoin != null;

private void OnCulcCoinExecuted(object id)

{

Coin newCoin = coinManager.Recount(\_selectedCoin.CoinId);

coinManager.Update(newCoin);

MessageBoxTimeout((System.IntPtr) 0, $"{\_selectedCoin.Name} - {\_selectedCoin.ShortName} recounted. ",

"Coins", 0, 0, 1000);

OnGetTransactionExecuted(\_selectedCoin.CoinId);

}

#endreion

#region Full recount

private ICommand \_fullRecountCommand;

public ICommand FullRecountCommand =>

\_fullRecountCommand ??= new RelayCommand(OnFullRecountExecuted);

private void OnFullRecountExecuted(object id)

{

PortfolioIndicator startPriceIndicator = new PortfolioIndicator();

startPriceIndicator.IndicatorName = "Start price";

startPriceIndicator.Value = 0;

decimal startPrice = 0;

foreach (var coin in Coins)

{

startPrice += coin.AverageValue;

}

startPriceIndicator.Value = startPrice;

foreach (var ind in PortfolioIndicators)

{

if (ind.IndicatorName.Equals(startPriceIndicator.IndicatorName))

{

ind.Value = startPrice;

}

}

PortfolioIndicator currentPriceIndicator = new PortfolioIndicator();

currentPriceIndicator.IndicatorName = "Current price";

currentPriceIndicator.Value = 0;

decimal currentPrice = 0;

foreach (var coin in Coins)

{

currentPrice += coin.CurrentValue;

}

currentPriceIndicator.Value = currentPrice;

foreach (var ind in PortfolioIndicators)

{

if (ind.IndicatorName.Equals(currentPriceIndicator.IndicatorName))

{

ind.Value = currentPrice;

}

}

PortfolioIndicator profit = new PortfolioIndicator();

profit.IndicatorName = "Profit";

profit.Value = currentPriceIndicator.Value - startPriceIndicator.Value;

foreach (var ind in PortfolioIndicators)

{

if (ind.IndicatorName.Equals(profit.IndicatorName))

{

ind.Value = profit.Value;

}}}

#endregion

#region Add transaction

private ICommand \_newTransactionCommand;

public ICommand NewTransactionCommand =>

\_newTransactionCommand ??= new RelayCommand(OnNewTransactionExecuted);

private void OnNewTransactionExecuted(object id) {

var dialog = new EditTransactionWindow

{

DebetCoin = \_selectedCoin,

};

if (dialog.ShowDialog() != true) return;

Coin debetCoin = (Coin) dialog.cBoxCoinDebet.SelectedItem;

Coin creditCoin = (Coin) dialog.cBoxCoinCredit.SelectedItem;

Coin outDebetCoin;

Coin outCreditCoin;

Coin.recalcByExchangeTransaction(debetCoin, creditCoin, dialog.Amount, dialog.Price, dialog.Side, out outDebetCoin, out outCreditCoin);

var transaction = new Transaction

{

DateUpdate = DateTime.Now,

Side = dialog.Side,

Symbol = debetCoin != null ? debetCoin.ShortName + creditCoin?.ShortName : "empty DebetCoin",

Amount = dialog.Amount,

Price = dialog.Price,

Sum = dialog.Amount \* dialog.Price,

TransactionCoins = new List<Coin>(2)

{

debetCoin,

creditCoin,

} };

coinManager.Update(debetCoin);

coinManager.Update(creditCoin);

transactionManager.CreateTransaction(transaction);

coinManager.AddTransactionToCoin(transaction, debetCoin.CoinId, debetCoin.CoinId);

AllTransactions.Add(transaction);

}

#endregion

#region Add Transfer transaction

private ICommand \_newTransferTransactionCommand;

public ICommand NewTransferTransactionCommand =>

\_newTransferTransactionCommand ??= new RelayCommand(OnNewTransferTransactionExecuted);

private void OnNewTransferTransactionExecuted(object id)

{

var dialog = new EditTransactionWindow(id)

{

DebetCoin = \_selectedCoin,

};

if (dialog.ShowDialog() != true) return;

Coin debetCoin = coinManager.GetById(((Coin) dialog.cBoxCoinDebet.SelectedItem).CoinId);

var transaction = new Transaction()

{

DateUpdate = DateTime.Now,

Side = dialog.Side,

Symbol = debetCoin?.ShortName,

Amount = dialog.Amount,

Price = dialog.Price,

Sum = dialog.Amount \* dialog.Price,

TransactionCoins = new List<Coin>()

{ debetCoin, }

};

debetCoin?.recalcByTransfer(dialog.Amount, dialog.Price);

coinManager.Update(debetCoin);

transactionManager.CreateTransaction(transaction);

coinManager.AddTransactionToCoin(transaction, debetCoin.CoinId);

AllTransactions.Add(transaction);

}

#endregion

#region Delete transaction

private ICommand \_deleteTransactionCommand;

public ICommand DelTransactionCommand => \_deleteTransactionCommand ??=

new RelayCommand(OnDeleteTransactionExecuted, DeleteTransactionCanExecute);

// check if can delete

private bool DeleteTransactionCanExecute(object p) =>

\_selectedCoin != null;

private void OnDeleteTransactionExecuted(object id)

{

var result = MessageBox.Show("Are you sure?", $"Delete Transaction {\_selectedTransaction.Symbol}?",

MessageBoxButton.YesNo);

if (result == MessageBoxResult.Yes) {

transactionManager.Delete(\_selectedTransaction.TransactionId); //deleting only transaction

OnGetTransactionExecuted(\_selectedTransaction.TransactionId);

AllTransactions.Remove(SelectedTransaction);

}}

#endregion

//simulation

#region UpdateCoins

private ICommand \_updateCoinsCommand;

public ICommand UpdateCoinsCommand =>

\_updateCoinsCommand ??= new RelayCommand(OnUpdateCoinsExecuted);

private void OnUpdateCoinsExecuted(object id)

{

MessageBoxTimeout((System.IntPtr)0, "Check connection", "Message", 0, 0, 1000);

MessageBoxTimeout((System.IntPtr)0, "Updating", "Message", 0, 0, 4000);

Random rand = new Random(-1);

Random randSign = new Random();

foreach (var coin in Coins)

{ var alteration = (coin.CurrentPrice \* (decimal)rand.NextDouble())/10;

var sign = randSign.Next(-1,2);

coin.CurrentPrice = coin.CurrentPrice + alteration\*sign;

} }

#endregion

#endregion}}

using System;

using System.Collections.Generic;

using System.ComponentModel;

namespace portfolio.ViewModels{

//In order to be able to bind the markup and model data two-way

public class ViewModelBase : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged([CallerMemberName] string propertyName=null)

{

PropertyChanged?.Invoke(this, new

PropertyChangedEventArgs(propertyName));

}

/// <summary>

/// Method for changing property (event PropertyChanged)

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="prop">property for changing</param>

/// <param name="value">new value</param>

/// <param name="propName">property name</param>

/// <returns></returns>

protected bool Set<T>(ref T prop, T value, [CallerMemberName] string propName=null)

{

if (Equals(prop, value)) return false;

prop = value;

OnPropertyChanged(propName);

return true;

}

}

}

<Window x:Class="portfolio.EditCoinWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:portfolio"

mc:Ignorable="d"

DataContext="{Binding RelativeSource={RelativeSource Self}}"

Title="Edit Coin" Height="450" Width="400"

WindowStartupLocation="CenterScreen" Loaded="Window\_Loaded">

<DockPanel Background="Beige">

<TextBlock DockPanel.Dock="Top" HorizontalAlignment="Center" Margin="0,7,0,0"

FontWeight="Bold" FontSize="15" />

<UniformGrid DockPanel.Dock="Bottom" Rows="1" HorizontalAlignment="Right" Margin="0,0,0,10">

<Button Content="Ok" Margin="10,0" Padding="35,5" IsDefault="True" FontWeight="Bold"

Command="{Binding OkCommand}" />

<Button Content="Cancel" Margin="10,0" IsCancel="True" />

</UniformGrid>

<Grid Margin="3">

<Grid.Resources>

<Style TargetType="TextBlock">

<Style.Triggers>

<Trigger Property="Grid.Column" Value="0">

<Setter Property="HorizontalAlignment" Value="Right" />

<Setter Property="VerticalAlignment" Value="Center" />

<Setter Property="FontWeight" Value="Bold" />

<Setter Property="Margin" Value="0,3" />

</Trigger>

</Style.Triggers>

</Style>

<Style TargetType="TextBox">

<Setter Property="Margin" Value="3,2,0,2" />

</Style>

</Grid.Resources>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="77\*" />

<ColumnDefinition Width="93\*" />

<ColumnDefinition Width="117\*" />

<ColumnDefinition Width="107\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="30" />

<RowDefinition Height="20" />

<RowDefinition Height="30" />

<RowDefinition Height="30" />

<RowDefinition Height="30" />

<RowDefinition Height="40" />

<RowDefinition Height="40" />

<RowDefinition Height="40" />

<RowDefinition Height="30" />

<RowDefinition Height="30" />

</Grid.RowDefinitions>

<TextBlock x:Name="tbName" Grid.Row="0" Grid.Column="0" Text="Coin name:" HorizontalAlignment="Left"

Margin="10,0,0,0" Height="18" Width="73" />

<TextBox Grid.Row="0" Grid.Column="1" Grid.ColumnSpan="3" x:Name="tboxName"

Text="{Binding Name, FallbackValue='New coin'}" Margin="15,5,35,2"

ToolTip="Fill coin name" ToolTipService.ShowDuration="1000" />

<TextBlock x:Name="tbShortName" Grid.Column="1" Grid.Row="1" Text="Short name:" HorizontalAlignment="Left"

Margin="13,19,0,0" VerticalAlignment="Top" Width="78" Height="16" Grid.RowSpan="2">

<TextBox x:Name="tboxShortName" Grid.ColumnSpan="2" Grid.Column="2" Grid.Row="1" Margin="10,14,69,7"

Text="{Binding ShortName}" Background="MistyRose" Opacity="0.75" Grid.RowSpan="2"

ToolTip="Fill short coin name" ToolTipService.ShowDuration="1000" />

<Button x:Name="btnFill" Grid.Row="5" Grid.Column="0" Margin="21,5,45,5"

ToolTip="Calculate" ToolTipService.ShowDuration="1000"

Click="btnFill\_Click" Background="MistyRose" Opacity="1" Grid.ColumnSpan="2">

Fill sum, $

</Button>

<Label Grid.Row="3" Grid.Column="0" Content="Amount" HorizontalAlignment="Left" Margin="17,0,0,0"

Width="52" />

<TextBox Grid.Column="1" Grid.Row="3" x:Name="textBoxAmount"

ToolTip="Amount buying or transh" ToolTipService.ShowDuration="1000"

Text="{Binding Amount}" Margin="13,2,10,3" />

<Label Grid.Row="2" Grid.Column="2" Content="Current price, $" HorizontalAlignment="Left"

Margin="35,28,0,2" Width="90" Grid.RowSpan="2" Grid.ColumnSpan="2" />

<TextBox Grid.Column="3" Grid.Row="3" x:Name="textBoxCurrentPrice"

ToolTip="Set last brocker price" ToolTipService.ShowDuration="1000"

Text="{Binding CurrentPrice}" Margin="13,2,30,3" />

<Label Grid.Column="0" Grid.Row="4" Content="Price, $" HorizontalAlignment="Left" Margin="21,0,0,0" Width="49" />

<TextBox x:Name="tbPurchasePrice" Grid.Column="1" Grid.Row="4"

Text="{Binding PurchasePrice}" Margin="13,2,10,0"

ToolTip="Purchase price" ToolTipService.ShowDuration="1000" />

<Label Grid.Column="2" Grid.Row="4" Content="AveragePrice, $" HorizontalAlignment="Left" Margin="15,0,0,0" Width="92" />

<TextBlock Grid.Column="3" Grid.Row="4" x:Name="textBoxAveragePrice"

ToolTip="Average price" ToolTipService.ShowDuration="1000"

Text="{Binding Path=AveragePrice}" Margin="13,5,10,0" />

<!--<Label Grid.Column="1" Grid.Row="5" Content="Value, $" HorizontalAlignment="Right"/>-->

<TextBox Grid.Column="1" Grid.Row="5" x:Name="textBoxValueUSD" Margin="65,5,59,5"

Background="MistyRose" Opacity="0.75" Text="{Binding Path=ValueUSD, FallbackValue='0'}"

Grid.ColumnSpan="2" RenderTransformOrigin="0.526,0.467" />

<TextBox Grid.ColumnSpan="2" Grid.Column="2" Grid.Row="7" Text="{Binding Path=PictureName}"

Margin="0,10,0,10" Width="150" />

<Button Grid.Column="2" Grid.ColumnSpan="2" Grid.Row="6" Content="Choose file"

Width="150" HorizontalAlignment="Center"

Margin="0,6,0,4" Padding="5"

Command="{Binding SelectPictureNameCommand}" />

<Image Grid.Row="6" Grid.RowSpan="3" Width="150" Grid.Column="0" Grid.ColumnSpan="2" Margin="30,0,0,0"

HorizontalAlignment="Center" Source="{Binding PictureName}" />

<TextBlock Grid.Column="0" Grid.Row="9" Text="Date update:" HorizontalAlignment="Left" Margin="21,0,0,0" Grid.ColumnSpan="2" Height="16" Width="73" />

<DatePicker Grid.Column="1" Grid.Row="9" Margin="47,2,0,2"

SelectedDate="{Binding DateUpdate}" Grid.ColumnSpan="2" />

</Grid>

</DockPanel>

</Window>

using Microsoft.Win32;

using portfolio.Command;

using portfolio.Domain.Entities;

using System;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace portfolio

{

/// <summary>

/// Логика взаимодействия для EditCoinWindow.xaml

/// </summary>

public partial class EditCoinWindow : Window

{

#region Properties

public string Name

{

get { return (string)GetValue(NameProperty); }

set { SetValue(NameProperty, value); }

}

// Using a DependencyProperty as the backing store for

//Name.This enables animation, styling, binding, etc.

public static readonly DependencyProperty NameProperty = DependencyProperty

.Register("Name", typeof(string),

typeof(EditCoinWindow),

new PropertyMetadata(default(string)));

public string ShortName

{

get { return (string)GetValue(ShortNameProperty); }

set { SetValue(ShortNameProperty, value); }

}

public static readonly DependencyProperty ShortNameProperty = DependencyProperty

.Register("ShortName", typeof(string),

typeof(EditCoinWindow),

new PropertyMetadata(default(string)));

public decimal Amount

{

get { return (decimal)GetValue(AmountProperty); }

set { SetValue(AmountProperty, value); }

}

public static readonly DependencyProperty AmountProperty = DependencyProperty

.Register("Amount", typeof(decimal),

typeof(EditCoinWindow),

new PropertyMetadata(default(decimal)));

public decimal CurrentPrice

{

get { return (decimal)GetValue(CurrentPriceProperty); }

set { SetValue(CurrentPriceProperty, value); }

}

public static readonly DependencyProperty CurrentPriceProperty = DependencyProperty

.Register("CurrentPrice", typeof(decimal),

typeof(EditCoinWindow),

new PropertyMetadata(default(decimal)));

public decimal AveragePrice

{

get { return (decimal)GetValue(AveragePriceProperty); }

set { SetValue(AveragePriceProperty, value); }

}

public static readonly DependencyProperty AveragePriceProperty = DependencyProperty

.Register("AveragePrice", typeof(decimal),

typeof(EditCoinWindow),

new PropertyMetadata(default(decimal)));

public decimal PurchasePrice

{

get { return (decimal)GetValue(PurchasePriceProperty); }

set { SetValue(PurchasePriceProperty, value); }

}

public static readonly DependencyProperty PurchasePriceProperty = DependencyProperty

.Register("PurchasePrice", typeof(decimal),

typeof(EditCoinWindow),

new PropertyMetadata(default(decimal)));

public string PictureName

{

get { return (string)GetValue(PictureNameProperty); }

set { SetValue(PictureNameProperty, value); }

}

// Using a DependencyProperty as the backing store for

// ImagePass.This enables animation, styling, binding, etc.

public static readonly DependencyProperty PictureNameProperty = DependencyProperty

.Register("PictureName", typeof(string),

typeof(EditCoinWindow),

new PropertyMetadata(default(string)));

public DateTime DateUpdate

{

get { return (DateTime)GetValue(DateUpdateProperty); }

set { SetValue(DateUpdateProperty, value); }

}

// Using a DependencyProperty as the backing store for

//DateUpdate.This enables animation, styling, binding, etc...

public static readonly DependencyProperty DateUpdateProperty = DependencyProperty

.Register("DateUpdate", typeof(DateTime),

typeof(EditCoinWindow),

new PropertyMetadata(default(DateTime)));

public EditCoinWindow()

{

InitializeComponent();

}

private ICommand \_selectPictureNameCommand;

public ICommand SelectPictureNameCommand =>

\_selectPictureNameCommand

?? new RelayCommand(OnSelectPictureNameExecuted);

public void OnSelectPictureNameExecuted(object param)

{

var dialog = new OpenFileDialog();

if (dialog.ShowDialog() == true)

{

PictureName = dialog.FileName; } }

private ICommand \_okCommand;

public ICommand OkCommand =>

\_okCommand

?? new RelayCommand(OnOkExecuted);

public void OnOkExecuted(object param)

{

this.DialogResult = true;

this.Close();

}

#endregion

private void btnFill\_Click(object sender, RoutedEventArgs e)

{

try

{ //tboxShortName.Text = tboxName.Text.Substring(0, 3).ToUpper();

textBoxValueUSD.Text = (Convert.ToDecimal(textBoxAmount.Text) \* Convert.ToDecimal(tbPurchasePrice.Text)).ToString();

}

catch

{ [System.Runtime.InteropServices.DllImport("user32.dll", SetLastError = true)]

static extern int MessageBoxTimeout(IntPtr hwnd, String text, String title,

uint type, Int16 wLanguageId, Int32 milliseconds);

MessageBoxTimeout((System.IntPtr)0, "Fill Coin name.", "Message", 0, 0, 1000);

//MessageBox.Show("Fill Coin name."); } }

private void Window\_Loaded(object sender, RoutedEventArgs e) { } }}

<Window x:Class="portfolio.EditTransactionWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:portfolio"

xmlns:vm="clr-namespace:portfolio.ViewModels"

mc:Ignorable="d"

DataContext="{Binding RelativeSource={RelativeSource Self}}"

Title="Edit Transaction" Height="450" Width="400"

WindowStartupLocation="CenterScreen" Loaded="Window\_Loaded">

<DockPanel Background="Beige">

<TextBlock DockPanel.Dock="Top" HorizontalAlignment="Center" Margin="0,7,0,0"

FontWeight="Bold" FontSize="15"/>

<UniformGrid DockPanel.Dock="Bottom" Rows="1" HorizontalAlignment="Right" Margin="0,0,0,10">

<Button Content="Ok" Margin="10,0" Padding="35,5" IsDefault="True" FontWeight="Bold"

Command="{Binding OkCommand}"/>

<Button Content="Cancel" Margin="10,0" IsCancel="True"/>

</UniformGrid>

<Grid>

<Grid.Resources>

<Style TargetType="TextBlock">

<Style.Triggers>

<Trigger Property="Grid.Column" Value="0">

<Setter Property="HorizontalAlignment" Value="Right"/>

<Setter Property="VerticalAlignment" Value="Center"/>

<Setter Property="FontWeight" Value="Bold"/>

<Setter Property="Margin" Value="0,3"/>

</Trigger>

</Style.Triggers>

</Style>

</Grid.Resources><Grid.RowDefinitions><RowDefinition/><RowDefinition Height="0.78\*"/> <RowDefinition Height="auto"/><RowDefinition/><RowDefinition/><RowDefinition/> <RowDefinition/> <RowDefinition/> </Grid.RowDefinitions><Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*"></ColumnDefinition><ColumnDefinition/> <ColumnDefinition/> <ColumnDefinition/> </Grid.ColumnDefinitions><Label Margin="35,5,58,5" Grid.Row="0" Grid.ColumnSpan="2" Content="Transaction side"/>

<TextBlock Name="tbNote" TextWrapping="Wrap" FontSize="9" FontStyle="Italic"

Margin="76,2,5,0" Grid.Row="0" Grid.Column="2" Grid.ColumnSpan="2" Text="">

</TextBlock>

<ComboBox x:Name="cBoxSide" Margin="51,10,41,0" Grid.Row="0" Grid.Column="1"

Height="25" VerticalAlignment="Top" Text="{Binding Side}"

Grid.ColumnSpan="2" SelectedIndex="0"/>

<Label Content="Choose coins:" HorizontalAlignment="Left" Grid.Column="1" Margin="10,10,0,42" Grid.RowSpan="2" Grid.Row="1"/>

<StackPanel Grid.Column="0" Grid.Row="2" Grid.ColumnSpan="4" HorizontalAlignment="Center" Orientation="Horizontal">

<ComboBox x:Name="cBoxCoinDebet" Margin="20,10,10,25"

ItemsSource="{Binding Coins}"

SelectedItem="{Binding SelectedCoin}"

SelectedIndex="0" SelectedValuePath="CoinId" HorizontalAlignment="Center"

FontSize="11" Width="150"/>

<ComboBox x:Name="cBoxCoinCredit" Margin="20,10,10,25"

ItemsSource="{Binding Coins}"

SelectedItem="{Binding SelectedCoin}"

SelectedIndex="0" SelectedValuePath="CoinId"

FontSize="11" HorizontalAlignment="Center" Width="150"/>

</StackPanel>

<Label Grid.Row="4" Grid.Column="0" Margin="35 5 0 0" Content="Amount" />

<TextBox x:Name="textBoxAmount" Grid.Column="1" Grid.Row="4" Margin="10,7,5,10"

Text="{Binding Path=Amount}"/>

<Label Name="lPrice" Grid.Column="2" Grid.Row="4" Margin="5 5 0 0 " Content="Buying Price"/>

<TextBox Grid.Column="3" Grid.Row="4" Margin="5,7,10,10" x:Name="textBoxPrice"

Text="{Binding Path=Price}"/>

<Label Grid.Column="0" Grid.Row="5" Margin="35 5 0 0" Content="Sum"/>

<TextBox Grid.Column="1" Grid.Row="5" Margin="10,7,5,10" x:Name="textBoxSum"/>

<Button x:Name="btnCulculateSum" Grid.Row="5" Grid.Column="2" Margin="5,0,5,0" Height="28"

VerticalAlignment="Center" Content="Count" Click="btnCulculateSum\_Click"/>

</Grid>

</DockPanel>

</Window>

using portfolio.Business.Managers;

using portfolio.Command;

using portfolio.Domain.Entities;

using portfolio.ViewModels;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace portfolio

{

public partial class EditTransactionWindow : Window

{

#region Properties

public string Side

{

get { return (string) GetValue(SideProperty); }

set { SetValue(SideProperty, value); }

}

public static readonly DependencyProperty SideProperty = DependencyProperty

.Register("Side", typeof(string),

typeof(EditCoinWindow),

new PropertyMetadata(default(string)));

public ObservableCollection<Coin> Coins

{

get { return (ObservableCollection<Coin>) GetValue(CoinsProperty); }

set { SetValue(CoinsProperty, value); }

}

public static readonly DependencyProperty CoinsProperty = DependencyProperty

.Register("Coins", typeof(string),

typeof(EditCoinWindow),

new PropertyMetadata(default(string)));

public Coin DebetCoin

{

get { return (Coin) GetValue(DebetCoinProperty); }

set { SetValue(DebetCoinProperty, value); }

}

public static readonly DependencyProperty DebetCoinProperty = DependencyProperty

.Register("DebetCoin", typeof(Coin),

typeof(EditCoinWindow),

new PropertyMetadata(default(Coin)));

public decimal Amount

{

get { return (decimal) GetValue(AmountProperty); }

set { SetValue(AmountProperty, value); }

}

public static readonly DependencyProperty AmountProperty = DependencyProperty

.Register("Amount", typeof(decimal),

typeof(EditTransactionWindow),

new PropertyMetadata(default(decimal)));

public decimal Price

{ get { return (decimal) GetValue(PriceProperty); }

set { SetValue(PriceProperty, value); } }

public static readonly DependencyProperty PriceProperty = DependencyProperty

.Register("Price", typeof(decimal),

typeof(EditCoinWindow),

new PropertyMetadata(default(decimal)));

#endregion

public EditTransactionWindow()

{

InitializeComponent();

cBoxSide.ItemsSource = Transaction.sideType.GetRange(1, 2);

cBoxCoinDebet.ItemsSource = MainWindowViewModel.Coins;

cBoxCoinCredit.ItemsSource = MainWindowViewModel.Coins;

tbNote.Text = "\*Add exchange - choose buy or sell and coins";

}

public EditTransactionWindow(object id)

{

InitializeComponent();

cBoxSide.ItemsSource = Transaction.sideType.GetRange(0, 1);

cBoxCoinDebet.ItemsSource = MainWindowViewModel.Coins;

cBoxCoinCredit.Visibility = Visibility.Collapsed;

tbNote.Text = "\*To add coin without exchange - transfer and only one coin";

lPrice.Content = "Market price (opt)";

cBoxCoinDebet.Width = 220;

}

private ICommand \_okCommand;

public ICommand OkCommand =>

\_okCommand

?? new RelayCommand(OnOkExecuted);

public void OnOkExecuted(object param)

{ this.DialogResult = true;

this.Close();

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

cBoxCoinDebet.SelectedItem = DebetCoin;

}

private void btnCulculateSum\_Click(object sender, RoutedEventArgs e)

{

textBoxSum.Text = (Convert.ToDecimal(textBoxAmount.Text) \* Convert.ToDecimal(textBoxPrice.Text)).ToString(); } }}

<Window x:Class="portfolio.MainWindow"

xmlns:i="http://schemas.microsoft.com/xaml/behaviors"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:vm="clr-namespace:portfolio.ViewModels"

xmlns:inf="clr-namespace:portfolio.Infrastructure"

xmlns:fas="http://schemas.fontawesome.com/icons/"

xmlns:chartingToolkit="clr-namespace:System.Windows.Controls.DataVisualization.Charting;assembly=DotNetProjects.DataVisualization.Toolkit"

mc:Ignorable="d"

Title="Crypto portfolio" Height="600" Width="1100" MinHeight="550" MinWidth="1100"

WindowStartupLocation="CenterScreen"

xmlns:datavis="clr-namespace:System.Windows.Controls.DataVisualization;assembly=DotNetProjects.DataVisualization.Toolkit"

xmlns:Custom="{15D9F35E-7F75-4EE3-B7CA-E2986495A3FF}">

<Window.DataContext>

<vm:MainWindowViewModel />

</Window.DataContext>

<Window.Resources>

<inf:ImageSourceConverter x:Key="ImageSourceConverter" />

<Style TargetType="DataGrid">

<Setter Property="RowBackground" Value="Beige" />

<Setter Property="Background" Value="LightGoldenrodYellow" />

<Setter Property="Foreground" Value="DimGray" />

<Setter Property="BorderBrush" Value="LightSteelBlue" />

<Setter Property="BorderThickness" Value="1" />

</Style>

<Style TargetType="DataGridCell">

<Style.Triggers>

<Trigger Property="IsSelected" Value="True">

<Setter Property="Background" Value="#B09FF5" />

<Setter Property="Opacity" Value="0.7" />

</Trigger>

</Style.Triggers>

</Style>

<Style TargetType="Button">

<Setter Property="Margin" Value="5" />

<Setter Property="Padding" Value="5" />

<Setter Property="FontFamily" Value="Brevia" />

<Setter Property="Background">

<Setter.Value>

<ImageBrush ImageSource="pack://application:,,,/Resources/btn5.png" Opacity="1" />

</Setter.Value>

</Setter>

<Setter Property="Foreground" Value="#170F35" />

<Setter Property="BorderBrush" Value="#5A5180" />

<Setter Property="Opacity" Value="0.78" />

<Setter Property="FontSize" Value="12" />

<Setter Property="Button.Effect">

<Setter.Value> <DropShadowEffect Color="Black" Direction="320" ShadowDepth="2" BlurRadius="5" Opacity="0.7" /> </Setter.Value> </Setter>

</Style>

<Style TargetType="Label">

<Setter Property="Margin" Value="5" />

<Setter Property="Padding" Value="5" />

<Setter Property="FontFamily" Value="Brevia Light" />

<Setter Property="FontSize" Value="12" />

</Style>

<Style TargetType="TextBox">

<Setter Property="Margin" Value="5,0,5,7" />

<Setter Property="Padding" Value="5" />

<Setter Property="FontFamily" Value="Brevia" />

<Setter Property="Background" Value="LightBlue" />

<Setter Property="FontSize" Value="13" />

</Style>

<Style TargetType="TextBlock">

<Setter Property="Margin" Value="5,5,5,5" />

<Setter Property="Padding" Value="5" />

<Setter Property="FontFamily" Value="Brevia" />

<Setter Property="FontSize" Value="16" />

<Setter Property="Foreground" Value="DimGray" />

</Style>

<Style x:Key="PieSeriesLegendItemStyle1" TargetType="{x:Type chartingToolkit:LegendItem}">

<Setter Property="IsTabStop" Value="False" />

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type chartingToolkit:LegendItem}">

<Border Background="{TemplateBinding Background}"

BorderThickness="{TemplateBinding BorderThickness}"

BorderBrush="{TemplateBinding BorderBrush}">

<StackPanel Orientation="Horizontal">

<Rectangle Fill="{Binding Background}" Height="17"

Margin="0,5,5,0" StrokeThickness="1"

Stroke="DarkSeaGreen" Width="17" />

<datavis:Title Content="{TemplateBinding Content}" Foreground="#000033"

Margin="0,5,5,0" />

</StackPanel>

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Window.Resources>

<Window.Background>

<ImageBrush ImageSource="pack://application:,,,/Resources/pastelAbstraction.jpg" Opacity="0.8" />

</Window.Background>

<Grid x:Name="gridGeneral" Margin="8. 8. 8. 8."><Grid.ColumnDefinitions> <ColumnDefinition Width="1.9\*" /> <ColumnDefinition Width="1\*"/> </Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="1\*"/>

<RowDefinition Height="1.1\*" />

</Grid.RowDefinitions>

<Grid x:Name="gridData" Grid.RowSpan="2" Opacity="0.9">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="172\*" />

<ColumnDefinition Width="305\*" />

</Grid.ColumnDefinitions>

<TabControl Margin="5,0,0,5" BorderBrush="Gainsboro" Grid.ColumnSpan="2">

<TabControl.Background>

<ImageBrush ImageSource="pack://application:,,,/Resources/light-yellow.jpg"Opacity="0.75" />

</TabControl.Background>

<TabControl.Resources>

<Style TargetType="TabItem">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="TabItem">

<Border Name="Border" BorderThickness="0,0,0,0" orderBrush="LemonChiffon" CornerRadius="4,4,0,0" Margin="2,0">

<ContentPresenter x:Name="ContentSite" VerticalAlignment="Center"

HorizontalAlignment="Center" ontentSource="Header" Margin="10,2" /> </Border>

<ControlTemplate.Triggers>

<Trigger Property="IsSelected" Value="True">

<Setter TargetName="Border" Property="Background" Value="LemonChiffon" />

<Setter TargetName="Border" Property="Opacity" Value="0.8" />

</Trigger>

<Trigger Property="IsSelected" Value="False">

<Setter TargetName="Border" Property="Background" alue="DarkKhaki" /> <Setter TargetName="Border" Property="Opacity" Value="0.6" /> </Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</TabControl.Resources>

<TabItem ContextMenuOpening="rightClickUpdateTabCoin"

ToolTip="Update - right click" ToolTipService.ShowDuration="1000">

<TabItem.Header>

<Label Content="Coins" MouseLeftButtonDown="Tab\_OnMouseLeftButtonDown" />

</TabItem.Header>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="5\*" />

<ColumnDefinition Width="225\*" />

<ColumnDefinition Width="225\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="2.2\*" />

<RowDefinition Height="1\*" />

</Grid.RowDefinitions>

<Border Grid.Column="1" Grid.Row="0" Grid.ColumnSpan="2">

<DockPanel>

<DataGrid

Margin="5,5,5,12" x:Name="grCoinsData" Grid.Row="0" Grid.Column="1"

ItemsSource="{Binding Coins}"

AutoGenerateColumns="False"

SelectedValue="{Binding SelectedCoin}" SelectedIndex="0">

<i:Interaction.Triggers>

<i:EventTrigger EventName="SelectionChanged">

<i:InvokeCommandAction

Command="{Binding GetTransactionsFromCoinCommand}"

CommandParameter="{Binding SelectedCoin.CoinId}" />

</i:EventTrigger>

</i:Interaction.Triggers>

<DataGrid.Columns>

<DataGridTemplateColumn>

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Image

Stretch="Fill"

Width="23" Height="23"

Source="{Binding PictureName,

Converter={StaticResource ImageSourceConverter},

Mode=TwoWay}" />

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTextColumn Width="1\*" Header="Coin" Binding="{Binding Path=ShortName}">

<DataGridTextColumn.CellStyle>

<Style TargetType="DataGridCell">

<Setter Property="ToolTip"

Value="{Binding Name}" />

</Style>

</DataGridTextColumn.CellStyle>

</DataGridTextColumn>

<DataGridTextColumn Width="1.7\*" Header="Amount" Binding="{Binding Path=Amount}">

</DataGridTextColumn>

<DataGridTextColumn Width="1.5\*" Header="Average buying price"

Binding="{Binding Path=AveragePrice, StringFormat={}{0:C}}" />

<DataGridTextColumn Width="1.5\*" Header="Current price" Binding="{Binding Path=CurrentPrice, StringFormat={}{0:C}}" /> <DataGridTextColumn Width="1.7\*" Header="Value by buying"

Binding="{Binding Path=AverageValue, StringFormat={}{0:C}}" />

<DataGridTextColumn Width="1.7\*" Header="Value current"

Binding="{Binding Path=CurrentValue, StringFormat={}{0:C}}" />

<DataGridTextColumn Width="1.3\*" x:Name="tcProfitUSD" Header="Profit, $" Binding="{Binding Path=ProfitUSD, tringFormat={}{0:C2}}" /> <DataGridTextColumn Width="1.3\*" x:Name="tcProfitPerc" Header="Profit, %" Binding="{Binding Path=ProfitPerс, StringFormat={}{0:P2}}"/>

</DataGrid.Columns>

</DataGrid> </DockPanel>

</Border><DockPanel Grid.Column="1" Grid.Row="1" Margin="0,0,0,0" Grid.RowSpan="2">

<GroupBox Header="Coin transactions" Margin="0" Padding="0">

<ListView x:Name="lvTransactions" ItemsSource="{Binding TransactionsFromCoin}"

Background="Beige"> <ListView.ItemTemplate>

<DataTemplate> <StackPanel Orientation="Horizontal">

<TextBlock Text="{Binding Symbol}" /> <TextBlock Text=" " />

<TextBlock Text="{Binding Side}" /> <!-- <TextBlock Text=" amount " /> -->

<!-- <TextBlock Text="{Binding Amount}" /> -->

<!-- <TextBlock Text=", price " /> -->

<!-- <TextBlock Text="{Binding Priсe, StringFormat={}{0:C}}" /> -->

<TextBlock Text=", sum " />

<TextBlock Text="{Binding Sum, StringFormat={}{0:C}}" />

<TextBlock Text=" " />

<TextBlock Text="{Binding DateUpdate, StringFormat={}{0:dd.MM.yyyy}}" />

</StackPanel>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</GroupBox>

</DockPanel>

<DockPanel Grid.Column="2" Grid.Row="1">

<!-- coin profile using api binance or coinmarcetcup -->

</DockPanel>

</Grid>

</TabItem>

<TabItem x:Name="tiTransactionTab">

<TabItem.Header>

<Label Content="Transaction" Name="lTransactionsItems"

MouseLeftButtonDown="Tab\_OnMouseLeftButtonDown" />

</TabItem.Header>

<Border>

<DockPanel Grid.Row="0">

<DataGrid

Margin="5,5,5,5" x:Name="grTransactionData"

ItemsSource="{Binding AllTransactions}"

AutoGenerateColumns="False"

SelectedItem="{Binding SelectedTransaction}"

SelectedIndex="0">

<DataGrid.Columns>

<DataGridTextColumn Header="Transaction" Width="0.8\*" Binding="{Binding Path=Symbol}" />

<DataGridTextColumn Header="Amount" Width="1\*" Binding="{Binding Path=Amount}" />

<DataGridTextColumn Header="Price" Width="1\*"

Binding="{Binding Path=Price, StringFormat={}{0:C}}" />

<DataGridTextColumn Width="1\*" Header="Sum"

Binding="{Binding Path=Sum, StringFormat={}{0:C}}" />

<DataGridTextColumn Header="Side" Width="0.6\*" Binding="{Binding Path=Side}" />

<DataGridTextColumn Header="Data" Width="1.1\*" Binding="{Binding Path=DateUpdate}" />

</DataGrid.Columns>

</DataGrid>

</DockPanel>

</Border>

</TabItem>

<TabItem x:Name="CoinsProfileTab" ContextMenuOpening="rightClickUpdateTabTransaction">

<TabItem.Header>

<Label Content="Coins profile" MouseLeftButtonDown="Tab\_OnMouseLeftButtonDown" />

</TabItem.Header>

<Grid Name="GrCoinsProfile" Background="WhiteSmoke" Opacity="0.95">

<chartingToolkit:Chart Name="mcChart" Width="auto" Height="auto">

<chartingToolkit:Chart.Series>

<chartingToolkit:PieSeries

LegendItemStyle="{DynamicResource PieSeriesLegendItemStyle1}"

Title="Coins" ItemsSource="{Binding Coins}"

IndependentValueBinding="{Binding Path=Name}"

DependentValueBinding="{Binding Path=CurrentValue}"

Margin="0,0,0,0" />

</chartingToolkit:Chart.Series>

</chartingToolkit:Chart> </Grid> </TabItem> </TabControl> </Grid>

<Grid x:Name="gridStatistic" Grid.Column="1" Grid.Row="0">

<TabControl Margin="10,10,10,36" BorderBrush="Gainsboro">

<TabControl.Background>

<ImageBrush ImageSource="pack://application:,,,/Resources/light-yellow.jpg" Opacity="0.75" />

</TabControl.Background>

<TabControl.Resources>

<Style TargetType="TabItem">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="TabItem">

<Border Name="Border" BorderThickness="0,0,0,0" BorderBrush="LemonChiffon"

CornerRadius="4,4,0,0" Margin="2,0">

<ContentPresenter x:Name="ContentSite"

VerticalAlignment="Center"

HorizontalAlignment="Center"

ContentSource="Header"

Margin="10,2" />

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsSelected" Value="True">

<Setter TargetName="Border" Property="Background" Value="LemonChiffon" />

<Setter TargetName="Border" Property="Opacity" Value="0.8" />

</Trigger>

<Trigger Property="IsSelected" Value="False">

<Setter TargetName="Border" Property="Background" Value="DarkKhaki" />

<Setter TargetName="Border" Property="Opacity" Value="0.6" />

</Trigger>

</ControlTemplate.Triggers> </ControlTemplate> </Setter.Value> </Setter> </Style> </TabControl.Resources>

<TabItem Header="Portfolio profile">

<Grid Margin="2">

<Grid.RowDefinitions>

<RowDefinition Height="6\*" />

<RowDefinition Height="1.4\*" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="6\*" />

<ColumnDefinition Width="0.1\*" />

</Grid.ColumnDefinitions>

<DataGrid BorderBrush="CornflowerBlue"

Margin="2,2,2,2" x:Name="dgPortfolioIndicators"

ItemsSource="{Binding PortfolioIndicators}"

AutoGenerateColumns="False"

SelectedValue="{Binding SelectedCoin}" SelectedIndex="0">

<DataGrid.CellStyle>

<Style TargetType="DataGridCell">

<Setter Property="BorderThickness" Value="0,0,0,0" />

<Setter Property="BorderBrush" Value="LightGoldenrodYellow" />

<Setter Property="Background" Value="#CCCCFF" />

<Setter Property="Foreground" Value="MidnightBlue" />

<Setter Property="Opacity" Value="0.75" />

<Setter Property="Height" Value="38" />

<Setter Property="FontSize" Value="16" />

<Style.Triggers>

<Trigger Property="IsSelected" Value="True">

<Setter Property="Background" Value="CornflowerBlue" />

<Setter Property="Foreground" Value="LightGoldenrodYellow" />

<Setter Property="BorderBrush" Value="LightGoldenrodYellow" />

</Trigger>

</Style.Triggers>

</Style>

</DataGrid.CellStyle>

<DataGrid.Columns>

<DataGridTextColumn Header="Indicator" Width="1\*" Binding="{Binding Path=IndicatorName}" />

<DataGridTextColumn Width="1\*" Header="Amount, $" Binding="{Binding Path=Value}" />

</DataGrid.Columns>

</DataGrid>

<Button x:Name="btnFullRecount" ToolTip="Recount start portfolio price, current and profit"

ToolTipService.ShowDuration="1000"

Grid.Column="0" Grid.Row="1" Margin="12, 0, 100, 0" HorizontalAlignment="Left"

Height="26" Width="105" FontSize="14"

Background="#CCCCFF"

Command="{Binding FullRecountCommand}">

Full recount </Button> </Grid> </TabItem> <TabItem Header="Settings" Visibility="Hidden">

<Grid> <Grid.RowDefinitions>

<RowDefinition Height="2\*" />

<RowDefinition Height="1.5\*" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*" />

<ColumnDefinition Width="1.8\*" />

</Grid.ColumnDefinitions>

<TextBlock>content</TextBlock>

<StackPanel Grid.Row="1" Grid.Column="1" Background="Coral" Opacity="0.85">

<TextBlock HorizontalAlignment="Center" Foreground="Red">Danger Zone</TextBlock>

<Button Margin="0" HorizontalAlignment="Center" Width="125" Height="35" FontSize="13"

BorderBrush="#DC4E4E" Foreground="DarkRed">

Delete all the data

</Button>

</StackPanel>

</Grid>

</TabItem>

<TabItem Visibility="Hidden" Header="Custom calculator">

<TextBlock>content</TextBlock>

</TabItem>

</TabControl>

</Grid>

<Grid x:Name="gridPanel" Grid.Row="1" Grid.Column="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*" />

<ColumnDefinition Width="1\*" />

</Grid.ColumnDefinitions>

<StackPanel x:Name="st1" Grid.Column="1" Margin="5,10,5,71">

<Button x:Name="AddTransferTransaction"

Command="{Binding NewTransferTransactionCommand}" ToolTip="Add transfer-transaction"

CommandParameter="transfer"

ToolTipService.ShowDuration="2000" Width="125">

<fas:FontAwesome Icon="Solid\_replyall" FontSize="17" Width="21" />

<Button.Resources>

<Style TargetType="Border">

<Setter Property="CornerRadius" Value="3" />

</Style>

</Button.Resources>

</Button>

<Button x:Name="AddTransaction"

Command="{Binding NewTransactionCommand}" ToolTip="Add exchange transaction"

ToolTipService.ShowDuration="2000" Width="125">

<fas:FontAwesome Icon="Solid\_ExchangeAlt" FontSize="17" Width="21" />

<Button.Resources>

<Style TargetType="Border">

<Setter Property="CornerRadius" Value="3" />

</Style>

</Button.Resources>

</Button>

<Button x:Name="DeleteTransaction" ToolTip="Delete transaction" ToolTipService.ShowDuration="2000"

Command="{Binding DelTransactionCommand}" Width="125">

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="3" />

</Style>

</Button.Resources>

<fas:FontAwesome Icon="Solid\_Dumpster" FontSize="17" Width="21" /> </Button>

<Button Visibility="Hidden" Height="26" Margin="0,1,15,1" Background="LightGreen" x:Name="btnInsert"

Content="сurrent" />

<Button Visibility="Hidden" Height="26" Margin="0,1,15,1" Background="Moccasin" x:Name="btnFind"

Content="by bought price" />

<Button Visibility="Hidden" Height="26" Margin="0,1,15,1" Background="LightSalmon" x:Name="btnUpd" Content="ratio" />

</StackPanel>

<StackPanel x:Name="st2" Grid.Column="0" Margin="25,10,10,71">

<Button Width="125" x:Name="Count" ToolTip="Count" ToolTipService.ShowDuration="2000"

Command="{Binding CountCoinCommand}">

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="3" /> </Style>

</Button.Resources>

<fas:FontAwesome Icon="Solid\_calculator" FontSize="17" Width="21" />

</Button>

<Button x:Name="AddCoin" Command="{Binding NewCoinCommand}" ToolTip="Add coin"

ToolTipService.ShowDuration="2000"

Width="125">

<fas:FontAwesome Icon="Brands\_Btc" FontSize="17" Width="21" />

<Button.Resources>

<Style TargetType="Border">

<Setter Property="CornerRadius" Value="3" /> </Style>

</Button.Resources> </Button>

<Button x:Name="Edit" Command="{Binding EditCoinCommand}" ToolTip="Edit coin"

ToolTipService.ShowDuration="2000" Width="125">

<fas:FontAwesome Icon="Solid\_Edit" FontSize="17" Width="21" />

<Button.Resources>

<Style TargetType="Border">

<Setter Property="CornerRadius" Value="3" />

</Style>

</Button.Resources> </Button>

<Button x:Name="Delete" ToolTip="Delete coin" ToolTipService.ShowDuration="2000"

Command="{Binding DeleteCoinCommand}" Width="125">

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="3" /> </Style>

</Button.Resources> <fas:FontAwesome Icon="Solid\_Trash" FontSize="17" Width="21" /> </Button>

<Button x:Name="Update" ToolTip="Update all coins (online - binace.com)" ToolTipService.ShowDuration="2000" Width="125" Command="{Binding UpdateCoinsCommand}" >

<!--Click="Update\_Click">-->

<!--Command="{Binding OnUpdateCoinsCommand}" >-->

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="3" />

</Style>

</Button.Resources>

<fas:FontAwesome Icon="Regular\_ArrowAltCircleDown" FontSize="17" Width="21" />

</Button>

<Button Visibility="Hidden" x:Name="Find" Content="Find" />

</StackPanel>

</Grid>

</Grid>

</Window>

using portfolio.Business.Infrastructure;

using portfolio.Business.Managers;

using portfolio.Domain.Entities;

using portfolio.ViewModels;

using System;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace portfolio

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

[System.Runtime.InteropServices.DllImport("user32.dll", SetLastError = true)]

static extern int MessageBoxTimeout(IntPtr hwnd, String text, String title,

uint type, Int16 wLanguageId, Int32 milliseconds);

public MainWindow()

{

InitializeComponent();

}

private void \_refreshDataShowed()

{

// MessageBoxTimeout((System.IntPtr)0, "Updated", "Message", 0, 0, 1000);

grCoinsData.Items.Refresh();

dgPortfolioIndicators.Items.Refresh();

grTransactionData.Items.Refresh();

lvTransactions.Items.Refresh();

}

private void rightClickUpdateTabCoin(object sender, ContextMenuEventArgs e)

{

\_refreshDataShowed();

MessageBoxTimeout((System.IntPtr)0, "Updated", "Message", 0, 0, 1000);

}

private void rightClickUpdateTabTransaction(object sender, ContextMenuEventArgs e)

{

\_refreshDataShowed();

MessageBoxTimeout((System.IntPtr)0, "Updated", "Message", 0, 0, 1000);

}

private void Tab\_OnMouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

\_refreshDataShowed();

} }}

using Microsoft.Extensions.Configuration;

using portfolio.Business.Managers;

using portfolio.DAL.Repositories;

using portfolio.Domain.Interfaces;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using portfolio.TestData;

namespace portfolio.Business.Infrastructure

{

public class ManagersFactory

{

private readonly IUnitOfWork \_unitOfWork;

private readonly CoinManager \_coinManager;

private readonly TransactionManager \_transactionManager;

private readonly IConfiguration \_configuration;

public ManagersFactory()

{

\_unitOfWork = new TestUnitOfWork();

}

public ManagersFactory(string connStringName)

{

\_configuration = new ConfigurationBuilder()

.SetBasePath(Directory.GetCurrentDirectory())

.AddJsonFile("appsettings.json")

.Build();

var connString = \_configuration

.GetConnectionString(connStringName);

\_unitOfWork = new EFUnitOfWork(connString);

}

public CoinManager GetCoinManager()

{

return \_coinManager

?? new CoinManager(\_unitOfWork);

}

public TransactionManager GetTransactionManager()

{

return \_transactionManager

?? new TransactionManager(\_unitOfWork); } }}

using portfolio.Domain.Entities;

using portfolio.Domain.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace portfolio.Business.Managers

{

public class BaseManager

{

protected readonly IUnitOfWork \_unitOfWork;

protected readonly IRepository<Coin> \_coinRepository;

protected readonly IRepository<Transaction> \_transactionRepository;

public BaseManager(IUnitOfWork unitOfWork)

{

\_unitOfWork = unitOfWork;

\_coinRepository = unitOfWork.CoinRepository;

\_transactionRepository = unitOfWork.TransactioRepository;

}

}

}

using portfolio.Domain.Entities;

using portfolio.Domain.Interfaces;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.IO;

using System.Linq;

using System.Linq.Expressions;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

namespace portfolio.Business.Managers

{

public class CoinManager : BaseManager

{

public CoinManager(IUnitOfWork unitOfWork) : base(unitOfWork)

{

}

public IEnumerable<Coin> Coins

{

get => \_coinRepository.GetAll();

}

#region basic CRUD operations

public Coin Create()

{

Coin coin = new Coin();

\_unitOfWork.SaveChanges();

return coin;

}

public bool Delete(int id)

{

var result = \_coinRepository.Delete(id);

if (!result) return false;

\_unitOfWork.SaveChanges();

return true;

}

public Coin GetById(int id) => \_coinRepository?.Get(id);

public void Update(Coin coin)

{

\_coinRepository.Update(coin);

\_unitOfWork.SaveChanges();

}

#endregion

public void AddRange(List<Coin> coin)

{

coin.ForEach(c => \_coinRepository.Create(c));

\_unitOfWork.SaveChanges();

}

public IEnumerable<Coin> Find(Expression<Func<Coin, bool>> predicate) =>

\_coinRepository.Find(predicate);

public void AddTransactionToCoin(Transaction transaction, int debetCoinId, int creditCoinId)

{

var debetCoin = \_coinRepository.Get(debetCoinId);

var creditCoin = \_coinRepository.Get(creditCoinId);

debetCoin.Transactions.Add(transaction);

creditCoin.Transactions.Add(transaction);

if (debetCoin.CoinId <= 0)

\_coinRepository.Create(debetCoin);

else \_coinRepository.Update(debetCoin);

if (creditCoin.CoinId <= 0)

\_coinRepository.Create(creditCoin);

else \_coinRepository.Update(creditCoin);

transaction.TransactionCoins.Add(debetCoin);

transaction.TransactionCoins.Add(creditCoin);

\_unitOfWork.SaveChanges();

}

public void AddTransactionToCoin(Transaction transaction, int debetCoinId)

{

var debetCoin = \_coinRepository.Get(debetCoinId);

debetCoin.Transactions.Add(transaction);

if (debetCoin.CoinId <= 0)

\_coinRepository.Create(debetCoin);

else \_coinRepository.Update(debetCoin);

transaction.TransactionCoins.Add(debetCoin);

\_unitOfWork.SaveChanges();

}

public void RemoveTransactionFromCoin(Transaction transaction, int coinId)

{

var coin = \_coinRepository.Get(coinId);

coin.Transactions.Remove(transaction);

\_coinRepository.Update(coin);

transaction.TransactionCoins.Remove(coin);

\_transactionRepository.Update(transaction);

\_unitOfWork.SaveChanges();

}

public Coin Recount(int coinId)

{

var newCoin = \_coinRepository.Get(coinId);

newCoin.AverageValue = newCoin.Amount \* newCoin.AveragePrice;

newCoin.CurrentValue = newCoin.Amount \* newCoin.CurrentPrice;

newCoin.ProfitUSD = newCoin.Amount \* newCoin.CurrentPrice - newCoin.Amount \* newCoin.AveragePrice;

newCoin.ProfitPerс = newCoin.ProfitUSD / newCoin.AverageValue \* 10;

return newCoin; }

public static void Test() { }

public ICollection<Transaction> GetTransactionsOfCoin(int coinId) => \_transactionRepository

.Find(transaction => transaction.TransactionCoins.Contains(GetById(coinId)))

.ToList(); }}

using portfolio.Domain.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using portfolio.Domain.Entities;

namespace portfolio.Business.Managers

{ public class TransactionManager : BaseManager

{ public TransactionManager(IUnitOfWork unitOfWork) : base(unitOfWork) { }

public IEnumerable<Transaction> Transactions

{ get => \_transactionRepository.GetAll(); }

#region bacic CRUD operations

public Transaction CreateTransaction(Transaction transaction)

{

\_transactionRepository.Create(transaction);

\_unitOfWork.SaveChanges();

return transaction; }

public Transaction GetById(int id)

=> \_transactionRepository.Get(id);

public void UpdateTransaction(Transaction transaction)

{

\_transactionRepository.Update(transaction);

\_unitOfWork.SaveChanges(); }

public bool Delete(int id)

{

var result = \_transactionRepository.Delete(id);

if (!result) return false;

\_unitOfWork.SaveChanges();

return true;

}

#endregion

public void AddRange(List<Transaction> transactions)

{

transactions.ForEach(transaction => \_transactionRepository.Create(transaction));

\_unitOfWork.SaveChanges();

}

//only for test - transaction should not change like that after creation

public void AddCoinToTransaction(Coin coin, int transactionId)

{

var transaction = \_transactionRepository.Get(transactionId);

coin.Transactions.Add(transaction);

if (coin.CoinId <= 0)

\_coinRepository.Create(coin);

else \_coinRepository.Update(coin);

\_unitOfWork.SaveChanges();

}

public void AddCoinToTransaction(Coin coin)

{

var transaction = new Transaction(coin);

coin.Transactions.Add(transaction);

if (coin.CoinId <= 0)

\_coinRepository.Create(coin);

else \_coinRepository.Update(coin);

\_unitOfWork.SaveChanges();

}

public void RemoveCoinFromTransaction(Coin coin, int transactionId)

{

var transaction = \_transactionRepository.Get(transactionId);

transaction.TransactionCoins.Remove(coin);

coin.Transactions.Remove(transaction);

\_transactionRepository.Update(transaction);

\_coinRepository.Update(coin);

\_unitOfWork.SaveChanges();

}

public ICollection<Coin> GetCoinsOfTransaction(int transactionId) =>

\_coinRepository

.Find(coin => coin.Transactions.Contains(GetById(transactionId)))

.ToList(); }}

namespace portfolio.Business

{ public class PortfolioIndicator

{ public PortfolioIndicator()

{

IndicatorName = "Indicator default name";

Value = 0;

}

public string IndicatorName { get; set; }

public decimal Value { get; set; }

public override string ToString()

{

return $"Name {this.IndicatorName} value {this.Value}"; } }}

using Microsoft.EntityFrameworkCore;

using portfolio.Domain.Entities;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace portfolio.DAL.Data

{

public class PortfolioContext : DbContext

{

public PortfolioContext(DbContextOptions<PortfolioContext> options) : base(options)

{ }

public DbSet<Coin> Coins { get; set; }

public DbSet<Transaction> Transactions { get; set; } }}

using Microsoft.EntityFrameworkCore;

using portfolio.DAL.Data;

using portfolio.Domain.Entities;

using portfolio.Domain.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Linq.Expressions;

using System.Text;

using System.Threading.Tasks;

namespace portfolio.DAL.Repositories{

public class EFCoinsRepository : IRepository<Coin> {

private readonly PortfolioContext \_context;

private readonly DbSet<Coin> \_coins;

public EFCoinsRepository(PortfolioContext context) {

this.\_context = context;

\_coins = context.Coins;

}

public void Create(Coin entity)

{

\_context.AddAsync(entity);

}

public bool Delete(int id)

{

var coin = \_coins.Find(id);

if (coin == null) return false;

\_coins.Remove(coin);

return true;

}

public IQueryable<Coin> Find(Expression<Func<Coin, bool>> predicate)

{

return \_coins.Where(predicate);

}

public Coin Get(int id, params string[] includes)

{

IQueryable<Coin> query = \_coins;

foreach (var include in includes)

query = query.Include(include);

try

{

return query.First(c => c.CoinId == id);

}

catch {

return new Coin(); }

}

public IQueryable<Coin> GetAll()

{

return \_coins.AsQueryable();

}

public void Update(Coin coin)

{

\_coins.Update(coin);

}

}

}

using Microsoft.EntityFrameworkCore;

using portfolio.DAL.Data;

using portfolio.Domain.Entities;

namespace portfolio.DAL.Repositories

{

public class EFTransactionsRepositoriy : IRepository<Transaction>

{

private readonly PortfolioContext \_context;

private readonly DbSet<Transaction> \_transactions;

public EFTransactionsRepositoriy(PortfolioContext context)

{

this.\_context = context;

\_transactions = context.Transactions;

}

public void Create(Transaction entity)

{

\_context.AddAsync(entity);

}

public bool Delete(int id)

{

var transaction = \_transactions.Find(id);

if (transaction == null)

return false;

if(transaction.TransactionCoins.Count > 0)

{

foreach (var coin in transaction.TransactionCoins)

{

\_context.Coins

.Find(transaction.TransactionCoins[coin.CoinId])

.Transactions

.Remove(transaction);

}

}

\_transactions.Remove(transaction);

return true;

}

public IQueryable<Transaction> Find(Expression<Func<Transaction, bool>> predicate)

{

return \_transactions.Where(predicate);

}

public Transaction Get(int id, params string[] includes)

{

IQueryable<Transaction> query = \_transactions;

foreach (var include in includes)

query = query.Include(include);

try

{

return query.First(t => t.TransactionId == id);

} catch {

return new Transaction(); }

}

public IQueryable<Transaction> GetAll()

{

return \_transactions.AsQueryable();

}

public void Update(Transaction transaction)

{ \_transactions.Update(transaction); } }}

using Microsoft.EntityFrameworkCore;

using portfolio.DAL.Data;

using portfolio.Domain.Entities;

using portfolio.Domain.Interfaces;

using System.Threading.Tasks;

namespace portfolio.DAL.Repositories

{ public class EFUnitOfWork: IUnitOfWork

{

private readonly PortfolioContext \_context;

private IRepository<Coin> \_coinRepository;

private IRepository<Transaction> \_transactionRepository;

public EFUnitOfWork(string connectionString)

{

var options = new DbContextOptionsBuilder<PortfolioContext>()

.UseSqlServer(connectionString)

.Options;

\_context = new PortfolioContext(options);

\_context.Database.EnsureCreated();

}

public IRepository<Coin> CoinRepository =>

\_coinRepository ?? new EFCoinsRepository(\_context);

public IRepository<Transaction> TransactioRepository =>

\_transactionRepository ?? new EFTransactionsRepositoriy(\_context);

public void SaveChanges()

{

\_context.SaveChanges(); } }}

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Diagnostics;

using System.Windows;

using portfolio.Domain.Interfaces;

namespace portfolio.Domain.Entities

{

public class Coin : ICoinCloneable

{

[Key] public int CoinId { get; set; }

private string \_name = "No name coin";

private string \_shortName = "NONAME";

[Required] public decimal Amount { get; set; }

public decimal PurchasePrice { get; set; } //last purchase price

public decimal AveragePrice { get; set; } //Average purchase price

public decimal CurrentPrice { get; set; }

public decimal CurrentValue { get; set; }

public decimal AverageValue { get; set; }

public decimal ProfitUSD { get; set; }

public decimal ProfitPerс { get; set; }

private string \_pictureName = "no.png";

public string PictureName

{

get { return \_pictureName; }

set { \_pictureName = value; }

}

public string ShortName

{

get { return \_shortName; }

set {

if (value != null) {

\_shortName = value;

} } }

public string Name {

get { return \_name; }

set {if (value != null) { \_name = value; } } }

public DateTime DateUpdate { get; set; }

// навигационное свойство

public List<Transaction> Transactions { get; set; }

public Coin()

{

//Trace.WriteLine($"constr 1");

AveragePrice = PurchasePrice;

Transactions = new List<Transaction>();

}

//for SetupData for test

public Coin(string name, Transaction transaction)

{

//Trace.WriteLine($"constr 2");

Name = name + "Tww";

ShortName = "DNT";

Amount = 0;

CurrentValue = 0;

AverageValue = 0;

PictureName = ShortName + ".png";

DateUpdate = DateTime.Now;

Transactions = new List<Transaction>();

Transactions.Add(transaction);

}

public Coin(string name, decimal amount, decimal purchasePrice, decimal currentPrice)

{

Name = name;

ShortName = name.Substring(0, 3).ToUpper();

Amount = culcAmount(amount);

PurchasePrice = purchasePrice;

CurrentPrice = currentPrice;

AveragePrice = culcAveragePrice(purchasePrice);

CurrentValue = culcCurrentValue(amount, currentPrice);

AverageValue = culcAverageValue(amount, purchasePrice);

ProfitUSD = CurrentValue - AverageValue;

ProfitPerс = (CurrentValue - AverageValue) / CurrentPrice;

PictureName = ShortName + "png";

DateUpdate = DateTime.Today;

Transactions = new List<Transaction>();

}

public void recalcByTransfer(decimal amountTransfer, decimal priceTransfer)

{

AveragePrice = (AveragePrice + priceTransfer) / 2;

Amount += amountTransfer;

PurchasePrice = priceTransfer;

AverageValue = Amount \* AveragePrice;

CurrentValue = Amount \* CurrentPrice;

ProfitUSD = CurrentValue - AverageValue;

ProfitPerс = (CurrentValue - AverageValue) / Amount;

}

public static void recalcByExchangeTransaction(

Coin debetCoin, Coin creditCoin,

decimal dialogAmount, decimal dialogPrice, string dialogSide,

out Coin outDebetCoin, out Coin outCreditCoin)

{

if (dialogSide == "buy")

{

debetCoin.Amount += dialogAmount;

creditCoin.Amount -= dialogPrice;

}

else if (dialogSide == "sell")

{

debetCoin.Amount -= dialogAmount;

creditCoin.Amount += dialogPrice;

}

debetCoin.AverageValue = debetCoin.Amount \* debetCoin.AveragePrice;

creditCoin.AverageValue = creditCoin.Amount \* creditCoin.AveragePrice;

debetCoin.ProfitUSD = debetCoin.CurrentValue - debetCoin.AverageValue;

creditCoin.ProfitUSD = creditCoin.CurrentValue - creditCoin.AverageValue;

outDebetCoin = debetCoin;

outCreditCoin = creditCoin; }

public void calculateValues()

{

CurrentValue = Amount \* CurrentPrice;

AverageValue = Amount \* AveragePrice;

ProfitUSD = CurrentValue - AverageValue;

ProfitPerс = (CurrentValue - AverageValue) / Amount;

}

public decimal culcAmount(decimal amount)

{ return this.Amount + amount;

}

public decimal culcAveragePrice(decimal purchasePrice) {

if (this.AveragePrice != 0)

{

return (purchasePrice + this.AveragePrice) / (decimal)2.0;

}

return purchasePrice; }

public decimal culcCurrentValue(decimal amount, decimal currentPrice) {

return this.Amount\*currentPrice;

}

public decimal culcAverageValue(decimal amount, decimal purchasePrice)

{

if (this.AveragePrice != 0)

{

return this.culcAveragePrice(purchasePrice) \* amount;

}

return (this.Amount + amount) \* purchasePrice;

}

public override string ToString()

{

return $"{ShortName}" ;

}

public string FullToString()

{

return $"{ShortName} {Name} amount {Amount} average price {AveragePrice} profit {ProfitUSD}" ;

}

public Coin Clone()

{

return new Coin

{

CoinId = CoinId,

Name = Name,

ShortName = ShortName,

Amount = Amount,

PurchasePrice = PurchasePrice,

CurrentPrice = CurrentPrice,

AveragePrice = AveragePrice,

CurrentValue = CurrentValue,

AverageValue = AverageValue,

ProfitUSD = ProfitUSD,

ProfitPerс = ProfitPerс,

\_pictureName = \_pictureName, }; } }}

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Reflection;

using System.Runtime.CompilerServices;

namespace portfolio.Domain.Entities

{

public class Transaction

{

[Key]

public int TransactionId { get; set; }

public string Symbol { get; set; } = "def";

public decimal Amount { get; set; } = 0;

public decimal Price { get; set; } = 0;

public decimal Sum { get; set; } = 0;

public DateTime DateUpdate { get; set; } = DateTime.Now;

// навигационное свойство

public List<Coin> TransactionCoins { get; set; }

[Required]

private string \_side = "transfer";

public static readonly List<string> sideType = new() { "transfer", "buy", "sell"};

public string Side

{

get { return \_side; }

set

{

if (sideType.Contains(value))

\_side = value; } }

public Transaction(Coin coin)

{

Symbol = coin.ShortName;

Side = Side;

Amount = 0;

Price = 0;

DateUpdate = DateTime.Now;

TransactionCoins = new List<Coin>(2);

TransactionCoins.Add(coin);

}

public Transaction()

{

Symbol = "DefaultSymbol";

Side = "buy";

Amount = 0;

Price = 0;

DateUpdate = DateTime.Now;

TransactionCoins = new List<Coin>(2);

}

public override string ToString()

{

return $"{Symbol} {Side} {Amount} {Price} {Sum} ";

}}}

using portfolio.Domain.Entities;

namespace portfolio.Domain.Interfaces

{

public interface ICoinCloneable

{

Coin Clone();

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Linq.Expressions;

using System.Text;

using System.Threading.Tasks;

namespace portfolio.Domain.Interfaces

{

public interface IRepository<TEntity> where TEntity : class

{

IQueryable<TEntity> GetAll();

TEntity Get(int id, params string[] includes);

IQueryable<TEntity> Find(Expression<Func<TEntity, bool>> predicate);

void Create(TEntity entity);

void Update(TEntity entity);

bool Delete(int id);

using portfolio.Domain.Entities;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace portfolio.Domain.Interfaces

{

public interface IUnitOfWork

{

IRepository<Coin> CoinRepository { get; }

IRepository<Transaction> TransactioRepository { get; }

void SaveChanges();

}

}

using portfolio.Domain.Entities;

using portfolio.Domain.Interfaces;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Linq.Expressions;

using System.Text;

using System.Threading.Tasks;

namespace portfolio.TestData

{

/// <summary>

/// Test repository for Coin\_DEL

/// Find implementation is sufficient

/// </summary>

public class CoinTestRepository : IRepository<Coin>

{

private readonly List<Coin> \_coins;

public CoinTestRepository(List<Coin> coins)

{

\_coins = coins;

SetupData(); //generate test data

}

private static void SetupData()

{

var r = new Random();

var coins = new List<Coin>();

var transaction = new Transaction();

for (var i = 0; i < 5; i++)

{

var coin = new Coin($"C{i}", transaction);

coins.Add(coin);

Trace.WriteLine($"Test setup coin in \"for\": {coin}");

}

Trace.WriteLine("Test setup coin " + coins[0].Transactions[0].ToString());

Trace.WriteLine("Test setup coin " + coins[0].Transactions[0].ToString());

}

public void Create(Coin entity)

{

throw new NotImplementedException();

}

public bool Delete(int id)

{

throw new NotImplementedException();

}

public IQueryable<Coin> Find(Expression<Func<Coin, bool>> predicate)

{

Func<Coin, bool> filter = predicate.Compile();

return \_coins.Where(filter).AsQueryable();

}

public Coin Get(int id, params string[] includes)

{

throw new NotImplementedException();

}

public IQueryable<Coin> GetAll()

{

return \_coins.AsQueryable();

}

public void Update(Coin entity)

{

throw new NotImplementedException();}}}

using portfolio.Domain.Entities;

using portfolio.Domain.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace portfolio.TestData

{

public class TestUnitOfWork : IUnitOfWork

{

private IRepository<Coin> coinRepository;

private IRepository<Transaction> transactioRepository;

private List<Coin> coins;

private List<Transaction> transactions;

public TestUnitOfWork()

{

coins = new List<Coin>();

transactions = new List<Transaction>();

transactioRepository = new TransactionTestRepository(transactions);

foreach (var transaction in transactions)

coins.AddRange(transaction.TransactionCoins);

coinRepository = new CoinTestRepository(coins);

}

public IRepository<Coin> CoinRepository => throw new NotImplementedException();

public IRepository<Transaction> TransactioRepository => throw new NotImplementedException();

public void SaveChanges()

{

throw new NotImplementedException();}}}

using portfolio.Domain.Entities;

using portfolio.Domain.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Linq.Expressions;

using System.Text;

using System.Threading.Tasks;

namespace portfolio.TestData

{

public class TransactionTestRepository : IRepository<Transaction>

{

private readonly List<Transaction> \_transactions;

public TransactionTestRepository(List<Transaction> transactions)

{

this.\_transactions = transactions;

SetupData(); //generate test data

}

private void SetupData()

{

Random r = new Random();

var coins = new List<Coin>();

for (var i = 0; i < 5; i++)

{

coins.Add(new Coin

{

Name = $"Coin{i}"

}); }

for (var i = 1; i <= 5; i++)

{

var transaction = new Transaction

{

TransactionCoins = new List<Coin>

{

coins[r.Next()],

coins[r.Next()],

},

Amount = r.Next(1, 10),

Price = r.Next(1, 100000),

DateUpdate = DateTime.Now + TimeSpan.FromDays(r.Next(10, 20)),

};

\_transactions.Add(transaction);}}

public void Create(Transaction entity)

{

throw new NotImplementedException();

}

public bool Delete(int id)

{

throw new NotImplementedException();

}

public IQueryable<Transaction> Find(Expression<Func<Transaction, bool>> predicate)

{

Func<Transaction, bool> filter = predicate.Compile();

return \_transactions.Where(filter).AsQueryable();

}

public Transaction Get(int id, params string[] includes)

{

return \_transactions.FirstOrDefault(t => t.TransactionId == id);

}

public IQueryable<Transaction> GetAll()

{

return \_transactions.AsQueryable();

}

public void Update(Transaction entity)

{

throw new NotImplementedException();

}

}

}

Веб-сервис Портфолио криптовалют

using Microsoft.EntityFrameworkCore;  
using System;  
using System.Collections.Generic;  
using System.Text;  
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;  
using Portfolio.DAL.Entities;  
namespace Portfolio.DAL.Data  
{  
 public class ApplicationDbContext : IdentityDbContext<ApplicationUser>  
 {  
 public DbSet<Coin> Coins { get; set; }  
 public DbSet<CoinGroup> CoinGroups { get; set; }  
 public DbSet<UserCoin> UserCoins { get; set; }  
 public DbSet<UserPortfolio> UserPortfolios { get; set; }  
  
 public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options) { } }}

using Microsoft.AspNetCore.Identity;  
using System;  
using System.Collections.Generic;  
using System.Text;  
  
namespace Portfolio.DAL.Entities  
{  
 public class ApplicationUser : IdentityUser  
 {  
 public byte[] AvatarImage { get; set; }  
 }  
}

namespace Portfolio.DAL.Entities  
{  
 public class Coin  
 {  
 public int CoinId { get; set; } *// id* public string CoinName { get; set; }  
 public string CoinLongName { get; set; }  
 public string Description { get; set; }  
 public int Price { get; set; } *// last price,$* public int Amount { get; set; } *// amount coins* public int AmountUSD { get; set; } = 10; *// amount, $ (Price\*Amount)* public string Image { get; set; } *// Image name  
 // Navigational properties  
 /// <summary>  
 /// coin group (exs.: platdorm, Defi, oraqle et)  
 /// </summary>* public int CoinGroupId { get; set; }  
  
 public CoinGroup Group { get; set; } }}

using System.Collections.Generic;  
namespace Portfolio.DAL.Entities  
{  
 public class CoinGroup  
 {  
 public int CoinGroupId { get; set; }  
 public string GroupName { get; set; }  
 */// <summary>  
 /// Navigational properties one-to-many  
 /// </summary>* public List<Coin> Coins { get; set; } }}

using System.Collections.Generic;  
namespace Portfolio.DAL.Entities  
{  
 public class UserCoin  
 {  
 public int UserCoinId { get; set; }   
 public int Amount { get; set; }  
 *//public decimal AveragePurchasePrice { get; set; }* public int UserPortfolioId { get; set; }  
 public UserPortfolio UserPortfolio { get; set; }  
   
 public int CoinId { get; set; }  
 public Coin Coin { get; set; } }}

using System.Collections.Generic;  
namespace Portfolio.DAL.Entities  
{  
 public class UserPortfolio  
 {  
 public int UserPortfolioId { get; set; }  
 public string UserPortfolioName { get; set; }  
   
 public List<UserCoin> UserCoins { get; set; }  
   
 public int ApplicationUserId { get; set; }  
 public ApplicationUser ApplicationUser { get; set; } }}

using System;  
using System.Collections.Generic;  
using System.Text;  
namespace Portfolio.Test{  
 class Comparer<T> : IEqualityComparer<T>  
 {  
 public static Comparer<T> GetComparer(Func<T, T, bool> func)  
 {  
 return new Comparer<T>(func);  
 }  
 Func<T, T, bool> comparerFunction;  
 public Comparer(Func<T, T, bool> func)  
 {  
 comparerFunction = func;  
 }  
 public bool Equals(T x, T y)  
 {  
 return comparerFunction(x, y);  
 }  
 public int GetHashCode(T obj)  
 {  
 throw new NotImplementedException();}}}

using Portfolio.DAL.Entities;  
using Portfolio.WEB.Models;  
using Xunit;  
namespace Portfolio.Test  
{  
 public class ListViewModelTests  
 {  
 [Fact]  
 public void ListViewModelCountsPages()  
 {  
 *// Act* var model = ListViewModel<Coin>  
 .*GetModel*(TestData.*GetCoinsList*(), 1, 3);  
 *// Assert* Assert.*Equal*(2, model.TotalPages);  
 }  
 [Theory]  
 [MemberData(memberName: nameof(TestData.*Params*), MemberType = typeof(TestData))]  
 public void ListViewModelSelectsCorrectQty(int page, int qty, int id)  
 {  
 *// Act* var model = ListViewModel<Coin>  
 .*GetModel*(TestData.*GetCoinsList*(), page, 3);  
 *// Assert* Assert.*Equal*(qty, model.Count);  
 }  
 [Theory]  
 [MemberData(memberName: nameof(TestData.*Params*), MemberType = typeof(TestData))]  
 public void ListViewModelHasCorrectData(int page, int qty, int id)  
 {  
 *// Act* var model = ListViewModel<Coin>  
 .*GetModel*(TestData.*GetCoinsList*(), page, 3);  
 *// Assert* Assert.*Equal*(id, model[0].CoinId);  
 }  
 }  
}

using System;  
using System.Collections.Generic;  
using System.Text;  
using Portfolio.DAL.Data;  
using Portfolio.DAL.Entities;  
  
namespace Portfolio.Test  
{ public class TestData  
 { public static List<Coin> *GetCoinsList*()  
 {  
 return new List<Coin>  
 {  
 new Coin{ CoinId=1, CoinGroupId=1},  
 new Coin{ CoinId=2, CoinGroupId=1},  
 new Coin{ CoinId=3, CoinGroupId=2},  
 new Coin{ CoinId=4, CoinGroupId=2},  
 new Coin{ CoinId=5, CoinGroupId=3} }; }  
  
 public static void FillContext(ApplicationDbContext context)  
 {  
 context.CoinGroups.Add(new CoinGroup  
 { GroupName = "fake group" });  
 context.AddRange(new List<Coin>  
 {  
 new Coin{ CoinId=1, CoinGroupId=1},  
 new Coin{ CoinId=2, CoinGroupId=1},  
 new Coin{ CoinId=3, CoinGroupId=2},  
 new Coin{ CoinId=4, CoinGroupId=2},  
 new Coin{ CoinId=5, CoinGroupId=3} });  
 context.SaveChanges(); }  
 public static IEnumerable<object[]> *Params*()  
 {  
 *// 1-я страница, кол. объектов 3, id первого объекта 1* yield return new object[] { 1, 3, 1 };  
 *// 2-я страница, кол. объектов 2, id первого объекта 4* yield return new object[] { 2, 2, 4 };}}}  
 @page  
@model Portfolio.WEB.Areas.Admin.Pages.CreateModel  
  
@{  
 ViewData["Title"] = "Create";  
 Layout = "~/Views/Shared/\_Layout.cshtml";}  
<h1>Create</h1>  
<h4>Coin</h4>  
<hr />  
<div class="row">  
 <div class="col-md-4">  
 <**form method**="post" enctype="multipart/form-data">  
 <**div asp-validation-summary**="*ModelOnly*" class="text-danger"></**div**>  
 <div class="form-group">  
 <**label asp-for**="Coin.CoinName" class="control-label"></**label**>  
 <**input asp-for**="Coin.CoinName" class="form-control" />  
 <**span asp-validation-for**="Coin.CoinName" class="text-danger"></**span**>  
 </div>  
 <div class="form-group">  
 <**label asp-for**="Coin.Description" class="control-label"></**label**>  
 <**input asp-for**="Coin.Description" class="form-control" />  
 <**span asp-validation-for**="Coin.Description" class="text-danger"></**span**>  
 </div>  
 <div class="form-group">  
 <**label asp-for**="Coin.AmountUSD" class="control-label"></**label**>  
 <**input asp-for**="Coin.AmountUSD" class="form-control" />  
 <**span asp-validation-for**="Coin.AmountUSD" class="text-danger"></**span**>  
 </div>  
 <div class="form-group">  
 <**label asp-for**="Coin.Image" class="control-label"></**label**>  
 <div class="custom-file mt-3">  
 <**input type**="file" **asp-for**="Image"  
 class="custom-file-input" id="customFile">  
 <label class="custom-file-label"  
 for="customFile">Выберите файл</label> </div> </div>  
 <div class="form-group">  
 <**label asp-for**="Coin.CoinGroupId" class="control-label"></**label**>  
 <**select asp-for**="Coin.CoinGroupId" class="form-control" **asp-items**="ViewBag.CoinGroupId"></**select**> </div>  
 <div class="form-group">  
 <input type="submit" value="Create" class="btn btn-primary" />  
 </div>  
 </**form**>  
 </div>  
</div>  
  
<div>  
 <**a asp-page**="Index">Back to List</**a**>  
</div>

@page "{id:int}"  
@model Portfolio.WEB.Areas.Admin.Pages.DeleteModel  
@{  
 ViewData["Title"] = "Delete";  
 Layout = "~/Views/Shared/\_Layout.cshtml";  
}  
<h1>Delete</h1>  
  
<h3>Are you sure you want to delete this?</h3>  
<div>  
 <h4>Coin</h4>  
 <hr />  
 <dl class="row">  
 <dt class="col-sm-2">  
 @Html.DisplayNameFor(model => model.Coin.CoinName)  
 </dt>  
 <dd class="col-sm-10">  
 @Html.DisplayFor(model => model.Coin.CoinName)  
 </dd>  
 <dt class="col-sm-2">  
 @Html.DisplayNameFor(model => model.Coin.Description)  
 </dt>  
 <dd class="col-sm-10">  
 @Html.DisplayFor(model => model.Coin.Description)  
 </dd>  
 <dt class="col-sm-2">  
 @Html.DisplayNameFor(model => model.Coin.AmountUSD)  
 </dt>  
 <dd class="col-sm-10">  
 @Html.DisplayFor(model => model.Coin.AmountUSD)  
 </dd>  
 <dt class="col-sm-2">  
 @Html.DisplayNameFor(model => model.Coin.Image)  
 </dt>  
 <dd class="col-sm-10">  
 @Html.DisplayFor(model => model.Coin.Image)  
 </dd>  
 <dt class="col-sm-2">  
 @Html.DisplayNameFor(model => model.Coin.Group)  
 </dt>  
 <dd class="col-sm-10">  
 @Html.DisplayFor(model => model.Coin.Group.CoinGroupId)  
 </dd>  
 </dl>  
 <**form method**="post">  
 <**input type**="hidden" **asp-for**="Coin.CoinId" />  
 <input type="submit" value="Delete" class="btn btn-danger" /> |  
 <**a asp-page**="./Index">Back to List</**a**>  
 </**form**>  
</div>  
@section Scripts{  
 <script src="~/js/bs-custom-file-input.js"></script>  
  
 <script>  
 bsCustomFileInput.init();  
 </script>  
}

@page  
@model Portfolio.WEB.Areas.Admin.Pages.DetailsModel  
@{  
 ViewData["Title"] = "Details";  
 Layout = "~/Views/Shared/\_Layout.cshtml";  
}  
<h1>Details</h1>  
<div>  
 <h4>Coin</h4>  
 <hr />  
 <dl class="row">  
 <dt class="col-sm-2">  
 @Html.DisplayNameFor(model => model.Coin.CoinName)  
 </dt>  
 <dd class="col-sm-10">  
 @Html.DisplayFor(model => model.Coin.CoinName)  
 </dd>  
 <dt class="col-sm-2">  
 @Html.DisplayNameFor(model => model.Coin.Description)  
 </dt>  
 <dd class="col-sm-10">  
 @Html.DisplayFor(model => model.Coin.Description)  
 </dd>  
 <dt class="col-sm-2">  
 @Html.DisplayNameFor(model => model.Coin.AmountUSD)  
 </dt>  
 <dd class="col-sm-10">  
 @Html.DisplayFor(model => model.Coin.AmountUSD)  
 </dd>  
 <dt class="col-sm-2">  
 @Html.DisplayNameFor(model => model.Coin.Image)  
 </dt>  
 <dd class="col-sm-10">  
 @Html.DisplayFor(model => model.Coin.Image)  
 </dd>  
 <dt class="col-sm-2">  
 @Html.DisplayNameFor(model => model.Coin.Group)  
 </dt>  
 <dd class="col-sm-10">  
 @Html.DisplayFor(model => model.Coin.Group.CoinGroupId)  
 </dd>  
 </dl>  
</div>  
<div>  
 <**a asp-page**="./Edit" asp-route-id="@Model.Coin.CoinId">Edit</**a**> |  
 <**a asp-page**="./Index">Back to List</**a**>  
</div>

@page  
@model Portfolio.WEB.Areas.Admin.Pages.IndexModel  
@{  
 ViewData["Title"] = "Index";  
 Layout = "~/Views/Shared/\_Layout.cshtml";  
}  
<h1>Index</h1>  
<p class="mt-2">  
 <**a asp-page**="Create" class="btn btn-outline-info">  
 <span class="fa fa-plus-circle">Добавить</span>  
 </**a**>  
</p>  
<p>  
 <**a asp-page**="Create">Create New</**a**>  
</p>  
<table class="table">  
 <thead>  
 <tr>  
 <th></th>  
 <th>  
 @Html.DisplayNameFor(model => model.Coin[0].CoinName)  
 </th>  
 <th>  
 @Html.DisplayNameFor(model => model.Coin[0].Description)  
 </th>  
 <th>  
 @Html.DisplayNameFor(model => model.Coin[0].AmountUSD)  
 </th>  
 <th>  
 @Html.DisplayNameFor(model => model.Coin[0].Image)  
 </th>  
 <th>  
 @Html.DisplayNameFor(model => model.Coin[0].Group)  
 </th>  
 <th></th>  
 </tr>  
 </thead>  
 <tbody>  
 @foreach (var item in Model.Coin)  
 { <tr>  
 <td> <im src="~/Images/@item.Image" alt="NoImage" width="50" />  
 </td> <td>  
 @Html.DisplayFor(modelItem => item.CoinName)  
 </td> <td>  
 @Html.DisplayFor(modelItem => item.Description)  
 </td> <td> @Html.DisplayFor(modelItem => item.AmountUSD)  
 </td> <td>  
 @Html.DisplayFor(modelItem => item.Image) </td> <td>  
 @Html.DisplayFor(modelItem => item.Group.CoinGroupId)  
 </td> <td>  
 <**a asp-page**="./Edit" asp-route-id="@item.CoinId">Edit</**a**> |  
 <**a asp-page**="./Details" asp-route-id="@item.CoinId">Details</**a**> |  
 <**a asp-page**="./Delete" asp-route-id="@item.CoinId">Delete</**a**>  
 </td>  
 <td>  
 <**a asp-page**="./Edit" asp-route-id="@item.CoinId"  
 class="btn btn-outline-info">  
 <span class="fas fa-edit"></span>  
 </**a**>  
 <**a asp-page**="./Delete" asp-route-id="@item.CoinId"  
 class="btn btn-outline-danger">  
 <span class="fas fa-trash-alt"></span>  
 </**a**> </td> </tr> } </tbody></table>

using Microsoft.AspNetCore.Mvc;  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Threading.Tasks;  
using Portfolio.WEB.Models;  
namespace Portfolio.WEB.Components  
{  
 public class MenuViewComponent : ViewComponent  
 {  
 *// Инициализация списка элементов меню* private List<MenuItem> \_menuItems = new List<MenuItem>  
 {  
 *//new MenuItem{ Controller="Home", Action="Index", Text="Home"},* new MenuItem{ Controller="Product", Action="Index", Text="Coins"},  
 *// new MenuItem{ Controller="Product", Action="Index", Text="Portfolio"},* new MenuItem{ IsPage=true, Area="Identity", Page="/Index", Text="Portfolio"},  
 new MenuItem{ IsPage=true, Area="Admin", Page="/Index", Text="Administration"}  
 };  
 public IViewComponentResult Invoke()  
 {  
 *//Получение значений сегментов маршрута* var controller = ViewContext.RouteData.Values["controller"];  
 var page = ViewContext.RouteData.Values["page"];  
 var area = ViewContext.RouteData.Values["area"];  
 foreach (var item in \_menuItems)  
 {  
 *// Название контроллера совпадает?* var \_matchController = controller?.Equals(item.Controller)  
 ?? false;  
 *// Название зоны совпадает?* var \_matchArea = area?.Equals(item.Area) ?? false;  
 *// Если есть совпадение, то сделать элемент меню активным  
 // (применить соответствующий класс CSS)* if (\_matchController || \_matchArea)  
 {  
 item.Active = "active";  
 }  
 }  
 return View(\_menuItems);}}}  
using System.Collections.Generic;  
using System.Linq;  
using System.Threading.Tasks;  
using Microsoft.AspNetCore.Mvc;  
using Microsoft.EntityFrameworkCore;  
using Portfolio.DAL.Data;  
using Portfolio.DAL.Entities;  
namespace Portfolio.WEB.Controllers  
{  
 public class CoinController : Controller  
 { [Route("api/**[controller]**")]  
 [ApiController]  
 public class CoinsController : ControllerBase  
 {  
 private readonly ApplicationDbContext \_context;  
 public CoinsController(ApplicationDbContext context)  
 {  
 \_context = context;  
 }  
 *// GET: api/Coins* [HttpGet]  
 public async Task<ActionResult<IEnumerable<Coin>>> GetCoins(int? group)  
 { return await \_context  
 .Coins  
 .Where(d => !group.HasValue  
 || d.CoinGroupId.Equals(group.Value))  
 ?.ToListAsync(); }  
 *// GET: api/Coins/5 /api/Coins?group=1* [HttpGet("{id}")]  
 public async Task<ActionResult<Coin>> GetCoin(int id)  
 {  
 var coin = await \_context.Coins.FindAsync(id);  
  
 if (coin == null)  
 {  
 return NotFound();  
 }  
 return coin; }  
 *// PUT: api/Coins/5  
 // To protect from overposting attacks, enable the specific properties you want to bind to, for  
 // more details, see https://go.microsoft.com/fwlink/?linkid=2123754.* [HttpPut("{id}")]  
 public async Task<IActionResult> PutCoin(int id, Coin coin)  
 {  
 if (id != coin.CoinId)  
 {  
 return BadRequest(); }  
  
 \_context.Entry(coin).State = EntityState.*Modified*;  
  
 try {  
 await \_context.SaveChangesAsync();  
 }  
 catch (DbUpdateConcurrencyException)  
 {  
 if (!CoinExists(id))  
 {  
 return NotFound();  
 }  
 else  
 {  
 throw; } }  
 return NoContent(); }  
 *// POST: api/Coins  
 // To protect from overposting attacks, enable the specific properties you want to bind to, for  
 // more details, see https://go.microsoft.com/fwlink/?linkid=2123754.* [HttpPost]  
 public async Task<ActionResult<Coin>> PostCoin(Coin coin)  
 {  
 \_context.Coins.Add(coin);  
 await \_context.SaveChangesAsync();  
  
 return CreatedAtAction("GetCoin", new { id = coin.CoinId }, coin); }  
  
 *// DELETE: api/Coins/5* [HttpDelete("{id}")]  
 public async Task<ActionResult<Coin>> DeleteCoin(int id)  
 {  
 var coin = await \_context.Coins.FindAsync(id);  
 if (coin == null)  
 {  
 return NotFound();  
 }  
  
 \_context.Coins.Remove(coin);  
 await \_context.SaveChangesAsync();  
  
 return coin;  
 }  
  
 private bool CoinExists(int id)  
 {  
 return \_context.Coins.Any(e => e.CoinId == id);}}}}

using System.Diagnostics;  
using Microsoft.AspNetCore.Mvc;  
using Microsoft.Extensions.Logging;  
using Portfolio.WEB.Models;  
namespace portfolio.WEB.Controllers  
{  
 public class HomeController : Controller  
 {  
 private readonly ILogger<HomeController> \_logger;  
  
 public HomeController(ILogger<HomeController> logger)  
 {  
 \_logger = logger; }  
 public IActionResult Index()  
 {  
 return View(); }  
 public IActionResult Privacy()  
 {  
 return View(); }  
 [ResponseCache(Duration = 0, Location = ResponseCacheLocation.*None*, NoStore = true)]  
 public IActionResult Error() {  
 return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });}}}  
 using Microsoft.AspNetCore.Hosting;  
using Microsoft.AspNetCore.Identity;  
using Microsoft.AspNetCore.Mvc;  
using System.Threading.Tasks;  
using Portfolio.DAL.Entities;  
namespace Portfolio.WEB.Controllers  
{  
 public class ImageController : Controller  
 {  
 UserManager<ApplicationUser> \_userManager;  
 IWebHostEnvironment \_env;  
 public ImageController(UserManager<ApplicationUser> userManager, IWebHostEnvironment env)  
 {  
 \_userManager = userManager;  
 \_env = env; }  
 public async Task<FileResult> GetAvatar()  
 {  
 var user = await \_userManager.GetUserAsync(User);  
 if (user.AvatarImage != null)  
 return File(user.AvatarImage, "image/...");  
 else  
 {  
 var avatarPath = "/Images/anonymous.png";  
  
 return File(\_env.WebRootFileProvider  
 .GetFileInfo(avatarPath).CreateReadStream(), "image/...");}}}}  
using Portfolio.WEB.Extensions;  
using Microsoft.AspNetCore.Mvc;  
using Microsoft.Extensions.Logging;  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Threading.Tasks;  
using Portfolio.WEB.Models;  
using Portfolio.DAL.Data;  
using Portfolio.DAL.Entities;  
  
namespace Portfolio.WEB.Controllers  
{  
 public class ProductController : Controller  
 {  
 int \_pageSize;  
 ApplicationDbContext \_context;  
 *//private ILogger \_logger;* public ProductController(ApplicationDbContext context */\*, ILogger<ProductController> logger\*/*) {  
 \_pageSize = 3;  
 \_context = context;  
 *//\_logger = logger;* }  
 [Route("Catalog")]  
 [Route("Catalog/Page\_{pageNo}")]  
 public IActionResult Index(int? group, int pageNo = 1)  
 {  
 *//\_logger.LogInformation($"info: group={group}, page={pageNo}");* var coinsFiltered = \_context.Coins.Where(c => !group.HasValue || c.CoinId == group.Value);  
 *// Поместить список групп во ViewData* ViewData["Groups"] = \_context.CoinGroups;  
 *// Получить id текущей группы и поместить в TempData* ViewData["CurrentGroup"] = group ?? 0;  
  
 var model = ListViewModel<Coin>.*GetModel*(coinsFiltered, pageNo, \_pageSize);  
 *//if (Request.Headers["x-requested-with"].ToString().ToLower().Equals("xmlhttprequest"))  
 // return PartialView("\_listpartial", model);* if (Request.IsAjaxRequest())  
 return PartialView("\_listpartial", model);  
 else  
 return View(model);  
 }  
  
 public IActionResult UserProducts()  
 {  
 throw new NotImplementedException(); } }}

using Microsoft.AspNetCore.Http;  
  
namespace Portfolio.WEB.Extensions  
{  
 public static class RequestExtensions  
 {  
 public static bool IsAjaxRequest(this HttpRequest request)  
 {  
 return request  
 .Headers["x-requested-with"]  
 .Equals("XMLHttpRequest");}}}  
using Microsoft.AspNetCore.Http;  
using Newtonsoft.Json;  
using System;  
namespace Portfolio.WEB.Extensions  
{  
 public static class SessionExtensions  
 {  
 public static void Set<T>(this ISession session, string key,  
 T item)  
 {  
 var serializedItem = JsonConvert.*SerializeObject*(item);  
 session.SetString(key, serializedItem);  
 }  
 public static T Get<T>(this ISession session, string key)  
 {  
 var item = session.GetString(key);  
 return item == null  
 ? Activator.*CreateInstance*<T>() *// default(T)* : JsonConvert.*DeserializeObject*<T>(item);}}}  
 using Microsoft.AspNetCore.Builder;  
namespace Portfolio.WEB.Middleware  
{  
 public static class AppExtensions  
 {  
 public static IApplicationBuilder UseFileLogging(this IApplicationBuilder app)  
 => app.UseMiddleware<LogMiddleware>(); }}

using Microsoft.AspNetCore.Http;  
using Microsoft.Extensions.Logging;  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Threading.Tasks;  
namespace Portfolio.WEB.Middleware  
{  
 public class LogMiddleware {  
 RequestDelegate \_next;  
 ILogger<LogMiddleware> \_logger;  
 public LogMiddleware(RequestDelegate next,  
 ILogger<LogMiddleware> logger) {  
 \_next = next;  
 \_logger = logger; }  
 public async Task Invoke(HttpContext context) {  
 await \_next.Invoke(context);  
 if (context.Response.StatusCode != StatusCodes.*Status200OK*) {  
 var path = context.Request.Path + context.Request.QueryString;  
 \_logger.LogInformation($"Request {path} returns status code {context.Response.StatusCode.ToString()}");}}}}

using System.Collections.Generic;  
using System.Linq;  
using Portfolio.DAL.Entities;  
  
namespace Portfolio.WEB.Models  
{  
 public class Cart  
 {  
 public Dictionary<int, CartItem> Items { get; set; }  
 public Cart()  
 {  
 Items = new Dictionary<int, CartItem>();  
 }  
 */// <summary>  
 /// Количество объектов в корзине  
 /// </summary>* public int Count {  
 get  
 {  
 return Items.Sum(item => item.Value.Quantity); } }  
 */// <summary>  
 /// Amount,$  
 /// </summary>* public int Calories  
 {  
 get {  
 *// return Items.Sum(item => item.Value.Quantity \*  
 // item.Value.Coin.Calories);* return Items.Sum(item =>   
 item.Value.Quantity \* item.Value.Coin.Price); } } */// <summary>  
 /// Добавление в портфель  
 /// </summary>  
 /// <param name="coin">добавляемый объект</param>* public virtual void AddToCart(Coin coin)  
 {  
  
 *// если объект есть в портфель  
 // то увеличить количество* if (Items.ContainsKey(coin.CoinId))  
 Items[coin.CoinId].Quantity++;  
 *// иначе - добавить объект в портфель* else  
 Items.Add(coin.CoinId, new CartItem  
 {  
 Coin = coin,  
 Quantity = 1 }); }  
 */// <summary>  
 /// Удалить объект из корзины  
 /// </summary>  
 /// <param name="id">id удаляемого объекта</param>* public virtual void RemoveFromCart(int id)  
  
 {  
 Items.Remove(id);  
 }  
 */// <summary>  
 /// Очистить корзину  
 /// </summary>* public virtual void ClearAll()  
 {  
 Items.Clear(); } }  
 */// <summary>  
 /// Клас описывает одну позицию в корзине  
 /// </summary>* public class CartItem  
 {  
 public Coin Coin { get; set; }

public int Quantity { get; set; } }}

using System;  
using System.Collections.Generic;  
using System.Linq;  
namespace Portfolio.WEB.Models{  
 public class ListViewModel<T> : List<T> where T : class  
 {  
 public int TotalPages { get; set; }  
 public int CurrentPage { get; set; }  
 private ListViewModel(IEnumerable<T> items,  
 int total,  
 int current) : base(items) {  
 TotalPages = total;  
 CurrentPage = current; }  
 */// <summary>   
 /// Получить модель представления списка объектов   
 /// </summary>   
 /// <param name="list">исходный список объектов</param> /// <param name="current">номер текущей страницы</param> /// <param name="itemsPerPage">кол. объектов на странице</param> /// <returns>объект класса ListViewModel</returns>* public static ListViewModel<T> *GetModel*(IEnumerable<T> list, int current, int itemsPerPage) {  
 var items = list  
 .Skip((current - 1) \* itemsPerPage)  
 .Take(itemsPerPage)  
 .ToList();  
 var total = (int)Math.*Ceiling*((double)list.Count() /  
 itemsPerPage);  
 return new ListViewModel<T>(items, total, current); } }}  
namespace Portfolio.WEB.Models  
{ public class MenuItem  
 { public bool IsPage { get; set; } = false;  
public string Area { get; set; } = "";  
public string Action { get; set; } = "";  
public string Controller { get; set; } = "";  
public string Page { get; set; } = "";  
public string Active { get; set; } = "";  
public string Text { get; set; } = ""; }}

using Microsoft.AspNetCore.Identity;  
using System.Collections.Generic;  
using System.Linq;  
using System.Threading.Tasks;  
using Portfolio.DAL.Data;  
using Portfolio.DAL.Entities;  
namespace Portfolio.WEB.Services{  
 public class DbInitializer  
 {  
 public static async Task *Seed*(ApplicationDbContext context,  
 UserManager<ApplicationUser> userManager,  
 RoleManager<IdentityRole> roleManager)  
 {  
 *// создать БД, если она еще не создана* context.Database.EnsureCreated();  
 *// проверка наличия ролей* if (!context.Roles.Any())  
 { var roleAdmin = new IdentityRole  
 { Name = "admin",  
 NormalizedName = "admin"  
 }; *// создать роль admin* await roleManager.CreateAsync(roleAdmin); }  
 *// проверка наличия пользователей* if (!context.Users.Any()) {  
 *// создать пользователя user@mail.ru* var user = new ApplicationUser  
 {  
 Email = "user@mail.ru",  
 UserName = "user@mail.ru"  
 };  
 await userManager.CreateAsync(user, "123456");  
 *// создать пользователя admin@mail.ru* var admin = new ApplicationUser  
 { Email = "admin@mail.ru",  
 UserName = "admin@mail.ru" };  
 await userManager.CreateAsync(admin, "123456");  
 *// назначить роль admin* admin = await userManager.FindByEmailAsync("admin@mail.ru");  
 await userManager.AddToRoleAsync(admin, "admin"); }  
 *//проверка наличия групп объектов* if (!context.CoinGroups.Any())  
 {  
 context.CoinGroups.AddRange(  
 new List<CoinGroup>  
 {  
 new CoinGroup { GroupName = "Indeterminate" },  
 new CoinGroup { GroupName = "DeFi" },  
 new CoinGroup { GroupName = "Derivatives" },  
 new CoinGroup { GroupName = "Oracles" },  
 new CoinGroup { GroupName = "Smart Contract" },  
 new CoinGroup { GroupName = "Stablecoin" }  
 });  
 await context.SaveChangesAsync(); } *// проверка наличия объектов* if (!context.Coins.Any())  
 {  
 context.Coins.AddRange(  
 new List<Coin>  
 {  
 new Coin  
 {  
 CoinName = "ETH",  
 CoinLongName = "Ethereum",  
 Description =  
 "Ethereum is a decentralized open-source blockchain " +  
 "system that features its own cryptocurrency, Ether. " +  
 "ETH works as a platform for numerous other cryptocurrencies, " +  
 "as well as for the execution of decentralized smart contracts.",  
 Price = 2612,   
 Amount = 1,  
 AmountUSD = 2612,  
 Image = "1027.png",  
 CoinGroupId = 5  
 },  
 new Coin  
 {  
 CoinName = "USDT",  
 CoinLongName = "Tether",  
 Description = "USDT is a stablecoin (stable-value cryptocurrency) " +  
 "of USDT in circulation.",   
 Price = 1,   
 Amount = 1,  
 AmountUSD = 1,  
 Image = "825.png",  
 CoinGroupId = 6  
 }  
 });  
 await context.SaveChangesAsync();}}}}  
using Microsoft.AspNetCore.Razor.TagHelpers;  
using Microsoft.AspNetCore.Routing;  
namespace Portfolio.WEB.TagHelpers  
{  
 [HtmlTargetElement(tag: "img", Attributes = "img-action, img-controller")]  
 public class ImageTagHelper : TagHelper  
 {  
 public string ImgAction { get; set; }  
 public string ImgController { get; set; }  
 LinkGenerator \_linkGenerator;  
   
 public ImageTagHelper(LinkGenerator linkGenerator)  
 {  
 \_linkGenerator = linkGenerator;  
 }  
   
 public override void Process(TagHelperContext context, TagHelperOutput output)  
 {  
 var url = \_linkGenerator.GetPathByAction(ImgAction, ImgController);  
 output.Attributes.Add("src", url);}}}  
using Microsoft.AspNetCore.Mvc.Rendering;  
using Microsoft.AspNetCore.Razor.TagHelpers;  
using Microsoft.AspNetCore.Routing;  
namespace Portfolio.WEB.TagHelpers  
{  
 public class PagerTagHelper : TagHelper  
 {  
 LinkGenerator \_linkGenerator;  
 *// номер текущей страницы* public int PageCurrent { get; set; }  
 *// общее количество страниц* public int PageTotal { get; set; }  
 *// дополнительный css класс пейджера* public string PagerClass { get; set; }  
 *// имя action* public string Action { get; set; }  
 *// имя контроллера* public string Controller { get; set; }  
 public int? GroupId { get; set; }  
   
 public PagerTagHelper(LinkGenerator linkGenerator)  
 {  
 \_linkGenerator = linkGenerator; }  
   
 public override void Process(TagHelperContext context, TagHelperOutput output)  
 {  
 *// контейнер разметки пейджера* output.TagName = "nav";  
 *// пейджер* var ulTag = new TagBuilder("ul");  
 ulTag.AddCssClass("pagination");  
 ulTag.AddCssClass(PagerClass);  
 for (int i = 1; i <= PageTotal; i++)  
 {  
 var url = \_linkGenerator.GetPathByAction(Action,  
 Controller,  
 new  
 {  
 pageNo = i,  
 group = GroupId == 0 ? null : GroupId  
 });  
 *// получение разметки одной кнопки пейджера* var item = GetPagerItem(  
 url: url, text: i.ToString(),  
 active: i == PageCurrent,  
 disabled: i == PageCurrent  
 );  
 *// добавить кнопку в разметку пейджера* ulTag.InnerHtml.AppendHtml(item);  
 }  
 *// добавить пейджер в контейнер* output.Content.AppendHtml(ulTag);  
 }  
 */// <summary>  
 /// Генерирует разметку одной кнопки пейджера  
 /// </summary>  
 /// <param name="url">адрес</param>  
 /// <param name="text">текст кнопки пейджера</param>  
 /// <param name="active">признак текущей страницы</param>  
 /// <param name="disabled">запретить доступ к кнопке</param>  
 /// <returns>объект класса TagBuilder</returns>* private TagBuilder GetPagerItem(string url, string text,  
 bool active = false,  
 bool disabled = false)  
 {  
 *// создать тэг <li>* var liTag = new TagBuilder("li");  
 liTag.AddCssClass("page-item");  
 liTag.AddCssClass(active ? "active" : "");  
 *//liTag.AddCssClass(disabled ? "disabled" : "");  
  
 // создать тэг <a>* var aTag = new TagBuilder("a");  
 aTag.AddCssClass("page-link");  
 aTag.Attributes.Add("href", url);  
 aTag.InnerHtml.Append(text);  
 *// добавить тэг <a> внутрь <li>* liTag.InnerHtml.AppendHtml(aTag);  
 return liTag; }}}}  
@{  
 ViewData["Title"] = "Crypto portfolio";  
  
}  
  
<div align="right">  
 <p></p>  
 <p>  
 Safe and Simple <br/>  
 All in One Place <br/>  
 Crypto Portfolio  
 </p>  
</div>  
<div>  
 <h2>  
 <p></p>  
 <p class="lead">  
 Управление крипто- и DeFi портфолио  
 </p>  
 </h2>   
</div>  
  
<h2></h2>

@model Coin  
  
<div class='card m-2 p-1 text-center col-4'>  
 <img src="~/Images/@Model.Image"  
 alt='NoImage'  
 class="card-img-top" />  
 <div class="card-body">  
 <h5 class="card-title">  
 @Model.CoinName  
 </h5>  
 <p class="card-text">  
 @Model.Description  
 </p>  
 <div class="card-text badge badge-secondary">  
 <h6>@Model.AmountUSD USD</h6>  
 </div>  
 @{ *// Получение текущего адреса* var request = ViewContext.HttpContext.Request; var returnUrl = request.Path +  
 request.QueryString.ToUriComponent();  
 }  
 *<!--Разметка кнопки добавления в корзину-->* <p class="mt-2">  
 <**a asp-action**="Add"  
 **asp-controller**="Cart"  
 **asp-route-id**="@Model.CoinId"  
 **asp-route-returnUrl**="@returnUrl"  
 class="btn btn-primary">  
 <i class="fa-solid fa-hexagon-plus"></i> Add to portfolio  
 </**a**>  
 </p>  
 </div>  
</div>

@model ListViewModel<Coin>  
  
@foreach (var item in Model)  
{  
 <**partial name**="\_ListItemPartial" **model**="@item" />  
}

@model ListViewModel<Coin>  
@{  
 ViewData["Title"] = "Меню";  
 var categories = ViewData["Groups"] as IEnumerable<CoinGroup>;   
 int currentGroup = (int)ViewData["CurrentGroup"];  
 var text = currentGroup != 0  
 ? categories  
 .FirstOrDefault(g => g.CoinGroupId == currentGroup)?.GroupName  
 : "Все";  
}  
<div class="row">  
 <div class="col-2">  
 <div class="dropdown mt-2">  
 <**a** class="btn btn-secondary dropdown-toggle" **asp-action**="Index" **asp-controller**="Product" role="button"  
 id="dropdownMenuLink"  
 data-toggle="dropdown"  
 aria-haspopup="true" aria-expanded="false">  
 @text </**a**>  
 <div class="dropdown-menu"  
 aria-labelledby="dropdownMenuLink">  
 <**a** class="dropdown-item"  
 **asp-action**="Index"  
 **asp-controller**="Product">Все</**a**>  
 @foreach (var item in categories)  
 {  
 <**a** class="dropdown-item"  
 **asp-action**="Index"  
 **asp-controller**="Product"  
 **asp-route-group**="@item.CoinGroupId" **asp-route-pageNo**="1">@item.GroupName</**a**>  
 }  
 </div>  
 </div>  
 </div>  
 <div class="col-10">  
 <div class="card-group">  
  
 <div class="card-group" id="list">  
 <**partial name**="\_ListPartial" **model**="@Model" />  
 </div> </div> <**pager page-current**="@Model.CurrentPage"  
 **page-total**="@Model.TotalPages"  
 **action**="Index"  
 **controller**="Product" **group-id**="@currentGroup"></**pager**>  
 </div></div>

@model IEnumerable<MenuItem>  
<div class="navbar-nav">  
 @foreach (var item in Model)  
 {  
 @if (item.IsPage)  
 {  
 <**a** class="nav-item nav-link @item.Active"  
 **asp-area**="@item.Area"  
 **asp-page**="@item.Page">  
 @item.Text  
 </**a**>  
 }  
 else {  
 <**a** class="nav-item nav-link @item.Active"  
 **asp-controller**="@item.Controller"  
 **asp-action**="@item.Action">  
 @item.Text </**a**> } }</div><!DOCTYPE html>  
<html lang="en"><head> <meta charset="utf-8" />  
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
 <title>@ViewData["Title"] - Portfolio.WEBPortfolio.WEB</title>  
 <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />  
 <link rel="stylesheet" href="~/css/site.css" />  
 <script src="lib/jquery/dist/jquery.js"></script>  
 <link href="~/lib/font-awesome/css/all.min.css" rel="stylesheet" />  
</head><body> <header> <div class="container">  
 <nav class="navbar navbar-expand-md navbar-dark">  
 *<!-- меню сайта -->* <**a** class="navbar-brand" **asp-action**="Index" **asp-controller**="Home">Crypto portfolio</**a**>  
 <div class="navbar-nav">  
 @await Component.InvokeAsync("Menu")  
 </div>  
 *<!-- меню сайта - конец -->  
 <!-- Информация пользователя -->* <**partial name**="\_UserPartial" />  
 *<!-- Информация пользователя - конец -->* </nav> </div>  
 </header>  
 <main role="main" class="container">  
 @RenderBody()  
 </main>  
 <footer class="fixed-bottom">  
 <div class="navbar navbar-dark navbar-expand-md bg-dark">  
 <div class="navbar-nav ml-auto">  
 <a class="nav-item nav-link" href="https://t.me/+liaXl1l-LH5iZmUy">  
 <i class="fab fa-telegram fa-2x"></i> </a>  
 <a class="nav-item nav-link" href="https://mail.google.com/mail/u/0/#inbox?compose=new">  
 <i class="fa fa-envelope fa-2x"></i>  
 </a>  
 </div>  
 </div>  
 </footer>  
 <script src="~/lib/jquery/dist/jquery.min.js"></script>  
 <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>  
 <**script src**="~/js/site.js" **asp-append-version**="true"></**script**>  
 @RenderSection("Scripts", required: false)  
</body></htm>@using Microsoft.AspNetCore.Identity  
@inject SignInManager<IdentityUser> SignInManager  
@inject UserManager<IdentityUser> UserManager  
<ul class="navbar-nav">  
@if (SignInManager.IsSignedIn(User))  
{  
 <li class="nav-item">  
 <**a** class="nav-link text-dark" **asp-area**="Identity" **asp-page**="/Account/Manage/Index" title="Manage">Hello @User.Identity.Name!</**a**>  
 </li>  
 <li class="nav-item">  
 <**form** class="form-inline" **asp-area**="Identity" **asp-page**="/Account/Logout" asp-route-returnUrl="@Url.Action("Index", "Home", new { area = "" })">  
 <button type="submit" class="nav-link btn btn-link text-dark">Logout</button>  
 </**form**> </li>}  
else{ <li class="nav-item"> <**a** class="nav-link text-dark" **asp-area**="Identity" **asp-page**="/Account/Register">Register</**a**>  
 </li>  
 <li class="nav-item">  
 <**a** class="nav-link text-dark" **asp-area**="Identity" **asp-page**="/Account/Login">Login</**a**>  
 </li>  
}  
</ul>@using Microsoft.AspNetCore.Identity;  
@inject SignInManager<ApplicationUser> signInManager  
@if (signInManager.IsSignedIn(User))  
{ *<!-- Информация пользователя -->* @await Component.InvokeAsync("Cart")  
 <div class="dropdown ml-4 nav-color">  
 <div class="dropdown-toggle" id="dropdownMenuButton" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">  
 <img img-action="GetAvatar"  
 img-controller="Image"  
 width="30" alt="User"  
 class="rounded bg-light" /> </div>  
 <div class="dropdown-menu" aria-labelledby="dropdownMenuButton">  
 <div class="dropdown-item-text">  
 <img img-action="GetAvatar"  
 img-controller="Image"  
 width="30" alt="User"  
 class="rounded bg-light" />  
 @User.Identity.Name  
 </div>  
 <div class="dropdown-divider"></div>  
 <**a** class="dropdown-item" **asp-controller**="Product"  
 **asp-action**="UserProducts">  
   
 </**a**>  
 <**form asp-area**="Identity"  
 **asp-page**="/Account/Logout"  
 asp-route-returnurl="@ViewContext.HttpContext.Request.Path">  
 <input type="submit"  
 value="Log off"  
 class="dropdown-item" />  
 </**form**>  
 </div>  
 </div>  
 *<!-- Информация пользователя - конец -->*}else{ <ul class="nav navbar-nav ml-auto">  
 <li>  
 <**a** class="nav-item nav-link"  
 **asp-area**="Identity"  
 **asp-page**="/Account/Login">  
 LogIn  
 </**a**>  
 </li>  
 <li>  
 <**a** class="nav-item nav-link"  
 **asp-area**="Identity"  
 **asp-page**="/Account/Register">  
 Registration </**a**> </li> </ul>}

<script src="~/lib/jquery-validation/dist/jquery.validate.min.js"></script>  
<script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.min.js"></script>@using Portfolio.WEB  
@using Portfolio.WEB.Models  
@using Portfolio.DAL.Entities  
@addTagHelper \*, Microsoft.AspNetCore.Mvc.TagHelpers  
@addTagHelper Portfolio.WEB.TagHelpers.\*, Portfolio.WEB@{  
 Layout = "~/Views/Shared/\_Layout.cshtml";  
}<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="utf-8" />  
 <title>From labs</title>  
  
 <link href="lib/bootstrap/dist/css/bootstrap.css" rel="stylesheet" />  
 <link href="lib/font-awesome/css/all.css" rel="stylesheet" />  
 <link href="css/site.css" rel="stylesheet" />  
</head>  
<body>  
 <script src="lib/jquery/dist/jquery.js"></script>  
 <script src="lib/bootstrap/dist/js/bootstrap.bundle.js"></script>  
 <header> <div class="container">  
 <span class="nav-color"></span>  
 <nav class="navbar navbar-expand-md navbar-dark">  
 *<!-- меню сайта -->* <a class="navbar-brand" href="#">Home</a>  
 <div class="navbar-nav">  
 <a class="nav-item nav-link active" href="#">Page?</a>  
 <a class="nav-item nav-link" href="#">Coins</a>  
 <a class="nav-item nav-link" href="#">Portfolio</a>  
 <a class="nav-item nav-link" href="#">Administration</a>  
 </div>  
 *<!-- меню сайта - конец -->  
 <!-- Информация пользователя -->* <a href="#" class="navbar-text ml-auto">  
 00,0 руб <i class="fas fa-shopping-cart nav-color"></i> (0)  
  
 </a>  
 <div class="dropdown ml-4 nav-color">  
 <div class="dropdown-toggle" id="dropdownMenuButton"  
 data-toggle="dropdown" aria-haspopup="true"  
 aria-expanded="false">  
 <img src="images/avatar.jpg" width="30"  
 alt="User" class="rounded" />  
 </div>  
 <div class="dropdown-menu"  
 aria-labelledby="dropdownMenuButton">  
 <div class="dropdown-item-text">  
 <img src="images/avatar.jpg" width="50"  
 alt="User" class="rounded" />  
 user@mail.ru  
 </div>  
 <div class="dropdown-divider"></div>  
 <a class="dropdown-item" href="#">Мои товары</a>  
 <a class="dropdown-item" href="#">Log off</a> </div> </div> *<!-- Информация пользователя - конец -->* </nav> </div> </header>  
 <main class="container">  
 <h1>Лабораторная работа 1</h1>  
 <h2>Список 1</h2>  
 <ol type="A">  
 <li>элемент 1 списка</li>  
 <li>элемент 2 списка</li>  
 </ol>  
 <hr/>  
 <h2>Форма</h2>  
 <form>  
 <div class="form-group row">  
 <div class="col-sm-2"></div>  
 <div class="col-sm-10">  
 <label>  
 <input type="checkbox" name="Check-box-1" value="1" /> Чек-бокс 1  
 </label>  
 </div>  
 </div>  
 <div class="form-group row">  
 <div class="col-sm-2"></div>  
 <div class="col-sm-10">  
 <label>  
 <input type="checkbox" name="Check-box-2" value="2" /> Чек-бокс 2  
 </label>  
 </div>  
 </div>  
  
 <div class="form-group row">  
 <div class="col-sm-2"></div>  
 <div class="col-sm-10">  
 <div class="form-check form-check-inline">  
 <input class="form-check-input" type="radio" name="Radio-btn" id="inlineRadio1" value="option1">  
 <label class="form-check-label" for="inlineRadio1">Радио-1</label>  
 </div>  
 <div class="form-check form-check-inline">  
 <input class="form-check-input" type="radio" name="Radio-btn" id="inlineRadio2" value="option2">  
 <label class="form-check-label" for="inlineRadio2">Радио-2</label>  
 </div>  
 <div class="form-check form-check-inline">  
 <input class="form-check-input" type="radio" name="Radio-btn" id="inlineRadio3" value="option3">  
 <label class="form-check-label" for="inlineRadio3">Радио-3</label>  
 </div> </div> </div> <div class="form-group row">  
 <div class="col-sm-2">  
 <label for="name">Login:</label>  
 </div>  
 <div class="col-sm-10">  
 <input type="text" id="name" name="login" class="form-control" placeholder="Введите имя" />  
 </div>  
  
 </div>  
 <div class="form-group row">  
 <div class="col-sm-2">  
 <label for="name">Password:</label>  
 </div>  
 <div class="col-sm-10">  
 <input type="password" id="pass" name="pass" class="form-control" placeholder="Введите пароль" />  
 </div>  
  
 </div>  
   
 <div class="form-group">  
 <button type="submit" class="btn btn-info">Send</button>  
 </div>  
 </form>  
  
 </main>  
 <footer class="fixed-bottom">  
 <div class="navbar navbar-dark navbar-expand-md bg-dark">  
 <div class="navbar-nav ml-auto">  
 <a class="nav-item nav-link" href="http://www.facebook.com">  
 <i class="fab fa-facebook-f fa-2x"></i>  
 </a>  
  
 <a class="nav-item nav-link" href="http://www.vk.com">  
 <i class="fab fa-vk fa-2x"></i>  
 </a>  
 <a class="nav-item nav-link" href="http://www.twitter.com">  
 <i class="fab fa-twitter fa-2x"></i>  
 </a> </div> </div> </footer></body></html>{  
 "ConnectionStrings": {  
 "DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=portfolio0;Trusted\_Connection=True;MultipleActiveResultSets=true"  
 },  
 "Logging": {  
 "LogLevel": {  
 "Default": "Information",  
 "Microsoft": "Warning",  
 "Microsoft.Hosting.Lifetime": "Information"  
 }  
 },  
 "AllowedHosts": "\*"  
}

using Microsoft.AspNetCore.Hosting;  
using Microsoft.Extensions.Configuration;  
using Microsoft.Extensions.Hosting;  
using Microsoft.Extensions.Logging;  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Threading.Tasks;  
  
namespace Portfolio.WEB  
{  
 public class Program  
 {  
 public static void *Main*(string[] args)  
 {  
 *CreateHostBuilder*(args).Build().Run();  
 }  
  
 public static IHostBuilder *CreateHostBuilder*(string[] args)  
 {  
 var host = Host.*CreateDefaultBuilder*(args);  
 host.ConfigureWebHostDefaults(webBuilder =>  
 {  
 webBuilder.UseStartup<Startup>();  
 }).ConfigureLogging(lp => { lp.ClearProviders(); lp.AddFilter("Microsoft", LogLevel.*None*); });  
 return host;  
 }  
 *//public static IHostBuilder CreateHostBuilder(string[] args) =>  
 // Host.CreateDefaultBuilder(args)  
 // .ConfigureWebHostDefaults(webBuilder =>  
 // {  
 // webBuilder.UseStartup<Startup>();  
 // }).ConfigureLogging(lp => {  
 // lp.ClearProviders();  
 // lp.AddFilter("Microsoft", LogLevel.None);  
 // });* }  
}

using Portfolio.WEB.Middleware;  
using Microsoft.AspNetCore.Builder;  
using Microsoft.AspNetCore.Hosting;  
using Microsoft.AspNetCore.Http;  
using Microsoft.AspNetCore.HttpsPolicy;  
using Microsoft.AspNetCore.Identity;  
using Microsoft.AspNetCore.Identity.UI;  
using Microsoft.AspNetCore.Mvc;  
using Microsoft.EntityFrameworkCore;  
using Microsoft.Extensions.Configuration;  
using Microsoft.Extensions.DependencyInjection;  
using Microsoft.Extensions.Hosting;  
using Microsoft.Extensions.Logging;  
using Microsoft.Net.Http.Headers;  
using Portfolio.WEB.Models;  
using Portfolio.WEB.Services;  
using Portfolio.DAL.Data;  
using Portfolio.DAL.Entities;  
  
namespace Portfolio.WEB  
{  
 public class Startup  
 {  
 public Startup(IConfiguration configuration)  
 {  
 Configuration = configuration;  
 }  
 public IConfiguration Configuration { get; }  
  
 *// This method gets called by the runtime. Use this method to add services to the container.* public void ConfigureServices(IServiceCollection services)  
 {  
 services.ConfigureApplicationCookie(options =>  
 {  
 options.LoginPath = $"/Identity/Account/Login";  
 options.LogoutPath = $"/Identity/Account/Logout";  
 });  
  
 services.AddDbContext<ApplicationDbContext>(options =>  
 options.UseSqlServer(  
 Configuration.GetConnectionString("DefaultConnection")));  
  
 services.AddIdentity<ApplicationUser, IdentityRole>(options =>  
 {  
 options.SignIn.RequireConfirmedAccount = false;  
 options.Password.RequireLowercase = false;  
 options.Password.RequireNonAlphanumeric = false;  
 options.Password.RequireUppercase = false;  
 options.Password.RequireDigit = false;  
 }).AddEntityFrameworkStores<ApplicationDbContext>().AddDefaultTokenProviders();  
 services.AddDistributedMemoryCache();  
 services.AddSession(opt =>  
 {  
 opt.Cookie.HttpOnly = true;  
 opt.Cookie.IsEssential = true;  
 });  
 services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.*Version\_2\_2*);  
 services.AddSingleton<IHttpContextAccessor, HttpContextAccessor>();  
 services.AddScoped<Cart>(sp => CartService.*GetCart*(sp));  
  
 *//services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireConfirmedAccount = true)  
 // .AddEntityFrameworkStores<ApplicationDbContext>();* services.AddControllersWithViews();  
 services.AddRazorPages();  
 }  
 *// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.* public void Configure(IApplicationBuilder app,  
 IWebHostEnvironment env, ApplicationDbContext context,  
 UserManager<ApplicationUser> userManager,  
 RoleManager<IdentityRole> roleManager,  
 ILoggerFactory logger)  
 {  
 logger.AddFile("Logs/log-{Date}.txt");  
  
 if (env.IsDevelopment())  
 {  
 app.UseDeveloperExceptionPage();  
 app.UseDatabaseErrorPage();  
 }  
 else {  
 app.UseExceptionHandler("/Home/Error");  
 *// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.* app.UseHsts();  
 }  
 app.UseHttpsRedirection();  
 app.UseStaticFiles();  
  
 app.UseRouting();  
  
 app.UseAuthentication();  
 app.UseSession();  
 app.UseAuthorization();  
  
 app.UseFileLogging();  
 app.UseCors(policy => policy.AllowAnyOrigin().AllowAnyMethod().WithHeaders(HeaderNames.*ContentType*));   
   
 *//context.Database.Migrate();* DbInitializer.*Seed*(context, userManager, roleManager).GetAwaiter().GetResult();  
 app.UseEndpoints(endpoints =>  
 {  
 endpoints.MapControllerRoute(  
 name: "default",  
 pattern: "{controller=Home}/{action=Index}/{id?}");  
 endpoints.MapRazorPages();  
 });  
 }  
 }  
}