Министерство образования Республики Беларусь

Учреждение образования

Белорусский государственный университет информатики и радиоэлектроники

Институт информационных технологий

Факультет повышения квалификации и переподготовки

Кафедра «Микропроцессорные системы и сети»

Дисциплина:

“Средства визуального программирования приложений”

Пояснительная записка по курсовому проекту

***Программное средство “Портфолио криптовалют”***

2021

РЕФЕРАТ

Пояснительная записка содержит 24 страниц, 19 рисунков.

C#, VISUAL STUDIO, Rider, UML, figma MVVM, .NET FRAMEWORK, EntityFramework, SQL-Server.

**Цель работы** — спроектировать и реализовать ПО для учета и анализа криптовалют в портфеле.

Результатом работы является десктопное приложение, обеспечивающее учет и менеджмент криптовалют и транзакций.

Реализация программы выполнена в средах разработки JetBrains Rider 2021 и Microsoft Visual Studio 2019, C#10, .NET 5.0, Sql Server 2017 с использованием системы контроля версий git и веб-сервиса github, десктопный git-клиент - Fork 1.68,

Пояснительная записка к дипломному проекту выполнена в Google Docs.

**Оглавление**

[Введение](#_wjt39yrnxwo9) **4**

[1. Описание предметной области и определение требований к системе с точки зрения предметной области](#_rl7hfjl1ifzb) **7**

[1.1 Описание предметной области](#_dcgqd2cudsw7) **7**

[1.2 Требования к программной системе с точки зрения предметной области](#_pxsd0zkgdtfe) **7**

[2. Постановка задачи и обзор методов её решения](#_cdrjaiw5tl1z) **8**

[2.1 Постановка задачи](#_1ij0aan6wayq) **8**

[3. Модели представления системы и их описание](#_xbo2sbn9l768) **9**

[4. Разработка приложения](#_tqychqou3k6t) **10**

[4.1 Описание интерфейса](#_y41g6i2gchjw) **10**

[4.2 Архитектура проекта](#_c5nz9sj6tcyd) **13**

[4.3 Характеристика программы](#_yj1dkqpzyokq) **18**

[5. Заключение](#_98xwemerg95d) **23**

[6. Список использованных источников](#_tiacobk42jmx) **24**

### 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.

Задачи решаемые в рамках реализации ИС:

1. Снижение человеческого фактора при внесении данных
2. Реализация учета
3. Реализация необходимых статистических отчетов
4. Анализ и наглядное представление данных

Основной целью курсового 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:

* Монеты, внесенные в ИС
* Транзакции добавления и обмена монет

1. Обеспечивать отказоустойчивость при ошибочных данных введенных пользователем
2. Использовать базу данных доступную другим модулям при расширении ИС
3. Приложение должно выводить следующие данные:

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

Для реализации программы применены следующие программные продукты:

1. 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.

1. Среда разработки JetBrains Rider. Применение обусловлено необходимостью обеспечения удобной разработки в быстрой среде с отличной инспекций кода, с наличием различных контекстных действий и рефакторингов, удобной отладкой.
2. Система управления базами данных Microsoft SQL Server 2017.

Применение обусловлено необходимостью использования базы данных совместимой с модулями ИС.

1. Среда моделирования RationalRose 2003.

Применение обусловлено необходимостью создания моделей и представлений предметной области для дальнейшего использования при разработке программы.

1. Онлайн-сервис для проектирования и прототипирования Figma

### 1. Описание предметной области и определение требований к системе с точки зрения предметной области

### 1.1 Описание предметной области

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

### 1.2 Требования к программной системе с точки зрения предметной области

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

### 2. Постановка задачи и обзор методов её решения

### 2.1 Постановка задачи

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 приложения
5. Тестирование работы базы данных, приложения

Используя подход «Code First» создать базу данных, содержащую две таблицы, связанные отношением «многие-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 разделения логики приложения и отображения.

Для реакции модели представления на действия пользователя использовать механизм команд. Один из методов доступа к данным сделать асинхронным. Составить UML диаграмму классов для спроектированной системы.

### 3. 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 1.

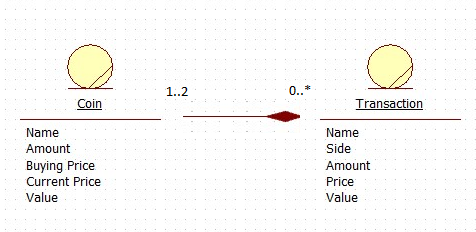


Рисунок 1 - Диаграмма сущностей предметной области

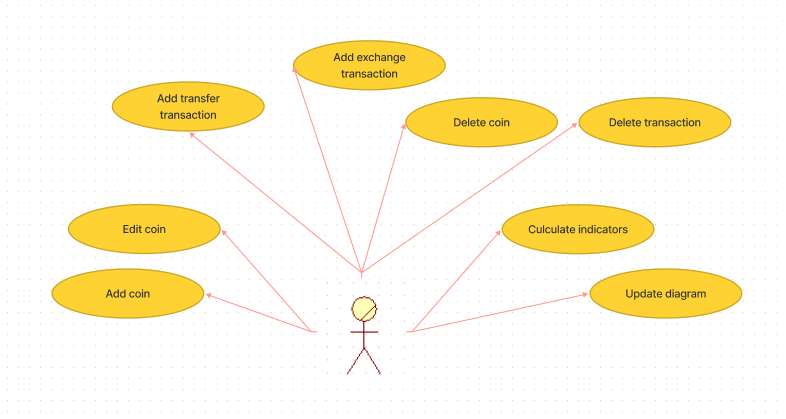
Основные функции, выделенные для разрабатываемой системы представлены нв диаграмма вариантов использования (Рис.2).

Рисунок 2 - Диаграмма вариантов использования

В результате бизнес анализа выявлены следующие сущности:

1. Монета
2. Транзакция

Отношения между сущностями и атрибуты отражены выше в диаграмме сущностей предметной области. Диаграмма классов приведена на Рисунке 6. Отношения между классами: к одной монете может относится несколько транзакций, одна транзакция может содержать от 1 до 2 монет.

### 4. Разработка приложения

### 4.1 Описание интерфейса

Приложение должно быть реализовано на языке C#. Графический интерфейс клиентского приложения должен быть реализован 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:

* главное окно приложение, предоставлющее возможность просмотра всех монет в виде таблицы, каждая строка которого представляет собой блок с информацией о конкретной монете, а именно:

1. сокращенное наименование;
2. иллюстрация в виде рисунка;
3. количество монеты;
4. средняя цена покупки;
5. последняя цена по курсу;
6. объем в USD;
7. разница относительно покупки и последней цены по курсу в USD
8. разница относительно покупки и последней цены по курсу в %
9. дата добавления.

* дополнительная панель главного окна предоставлющая возможность просмотра всех транзакций, каждая строка которого 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;

* 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)
* окно для ввода информации о новой монете, для добавления его в базу данных
* окно для изменения информации о новой монете, для изменения в базе данных
* окно для ввода информации о новой транзакции изменения количества выделенной монеты
* окно для ввода информации о новой транзакции обмена одной монеты на другую

Главное окно приложения будет содержать следующие элементы:

* диаграмма портфолио с пропорциональным представлением каждой монеты
* таблицу рассчитанных индикаторов по портфелю
* кнопка “Add coin”
* кнопка “Edit coin”
* кнопка “Recount coin”
* кнопка “Add transfer transaction”
* кнопка “Add exchange transaction”
* кнопка “Delete coin”
* 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».

### 4.2 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 на рис. 4. Оригинальная схема из статьи о Clean Architecture представлена на рис. 3.

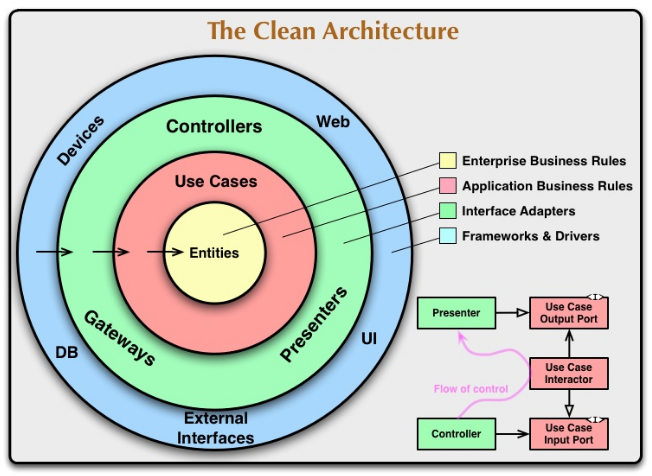


Рисунок 3. Чистая архитектура

Архитектура проекта представлена на рис. 4. Архитектура проекта с зависимостями представлена на рис. 6.

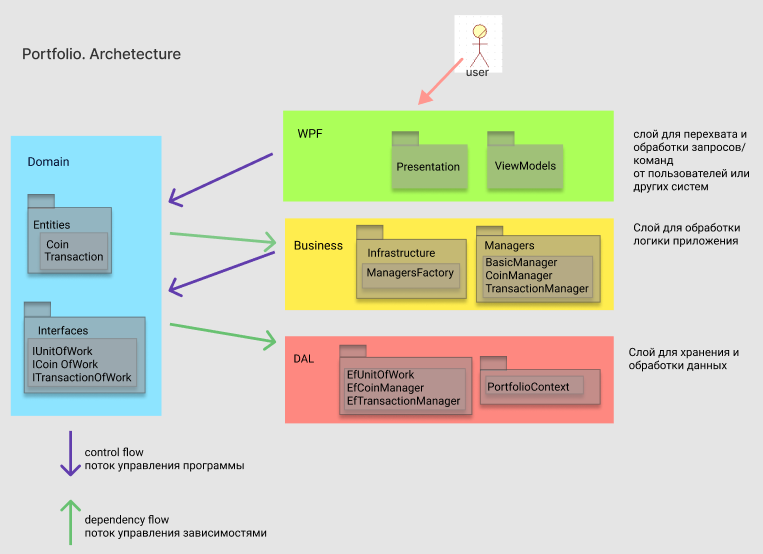


Рисунок 4. Архитектура проекта

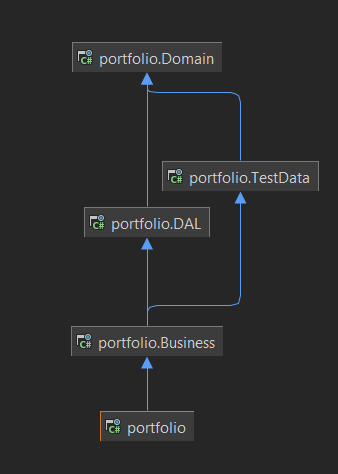
Библиотека Domain содержит описание сущностей, а также интерфейсы IRepository (хранилище данных), IUnitOfWork и ICoinClonable (рис.6)

Рисунок 5 - Зависимости проекта.

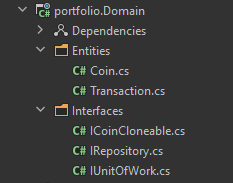


Рисунок 6 - Слой Domain

Диаграмма классов приведена на Рисунке 7. Отношения между классами: к одной монете может относится несколько транзакций, одна транзакция может содержать несколько монет.

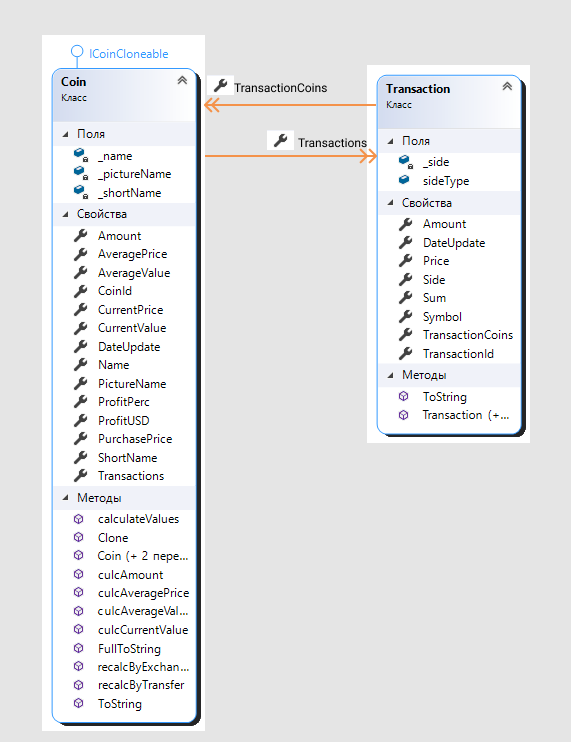


Рисунок 7 - Диаграмма классов

Полученная схема базы данных приведена на Рисунке 8.

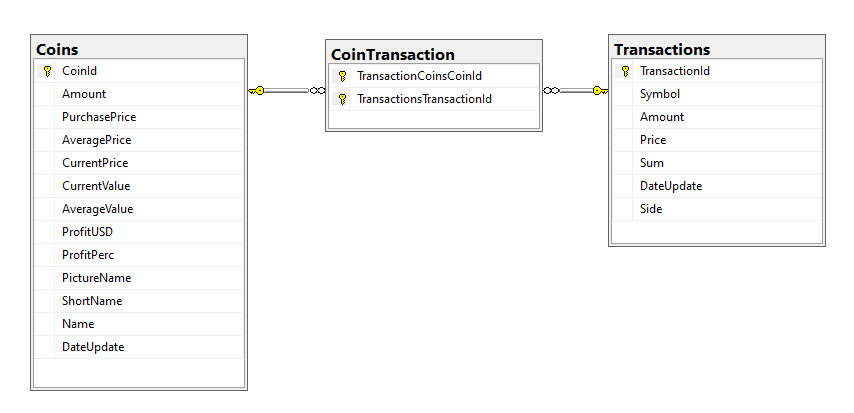


Рисунок 8 - Схема базы данных

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

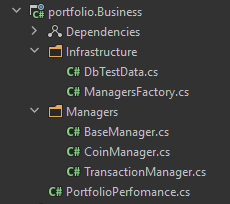


Рисунок 9. Слой Business

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

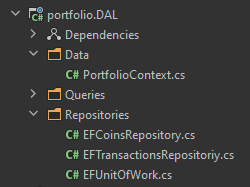


Рисунок 10. Слой DAL

### 

### Рисунок 11 - Структура основного проекта приложения

### 4.3 Характеристика программы

После запуска программы пользователь работает с главным окном (рис. 12). Главное окно приложения предоставляет возможность просмотра всех монет в виде таблицы на вкладке Coins, на вкладке Transaction всех транзакций (рис.13), на вкладке Coins profile - диаграмма отражающая отношение монеты к общему объема портфеля (рис.14). Отображение информации в панели Coin Transaction (рис.12, п.3) реализовано в виде MasterSlave , когда при выборе в списке строки одной таблицы (рис.12, п.2) автоматически отображается содержимое подчиненной таблицы).

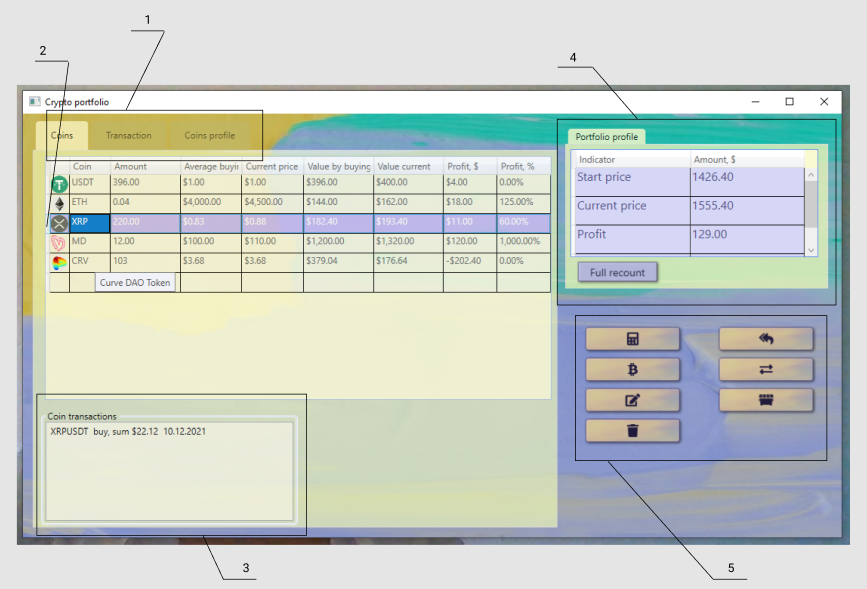


Рисунок 12 - Главное окно приложения: 1 - вкладки переключения на таблицу с монетами, таблицу с транзакциями и диаграмму; 2 - строка, отражающая основные данные о монете; 3 - панель транзакций, относящихся к выбранной монете; 4 - рассчитанные индикаторы и кнопка для пересчета; функциональные кнопки: “Recount coin”, “Add coin”, “Edit coin”, “Delete coin”, “Add transfer-transaction”, “Add exchange transaction”, , “Delete transaction”.

### 

Рисунок 13 - Главное окно приложения, вкладка Transaction

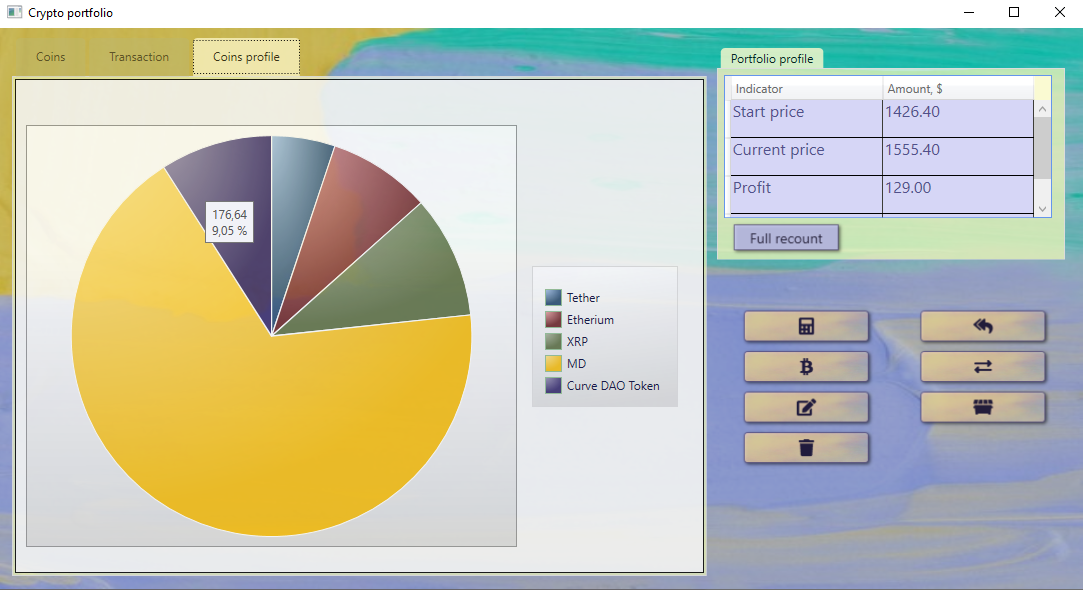


Рисунок 14 - Главное окно приложения, вкладка Coins profile

Окно для ввода информации о новой монете будет вызываться по клику на кнопке «Add coin» (рис. 15)

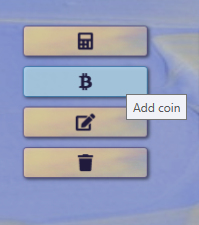
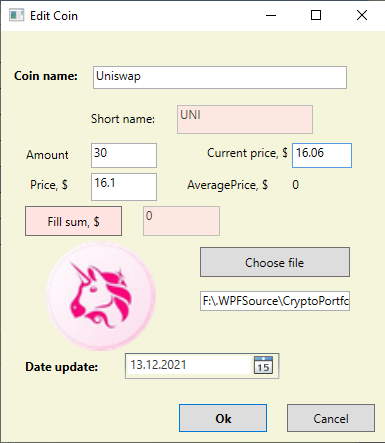
 

Рисунок 15 - Окно добавления и редактирования монеты

Окна для ввода информации о новых транзакциях вызывается по клику на кнопке «Add transfer-transaction» (рис. 16) или «Add exchange transaction» (рис. 17).

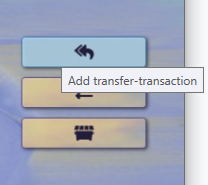
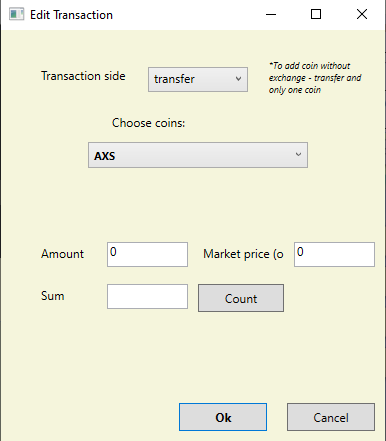
 

Рисунок 16 - Окно добавления трансфер-транзакции

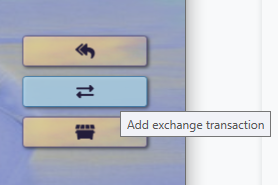
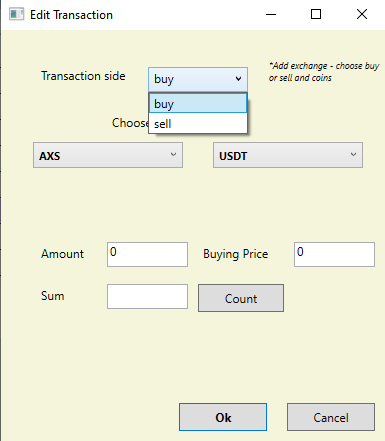
 

Рисунок 17 - Окно добавления транзакции обмена монет

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 рис. 13, пересчитанная диаграмма после добавления нескольких монет представлена на рис. 14.

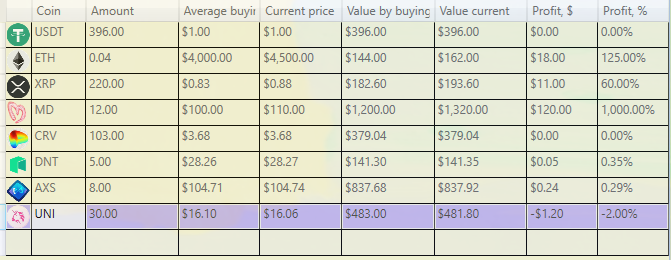


Рисунок 18 - Обновленная таблица, вкладка Coins

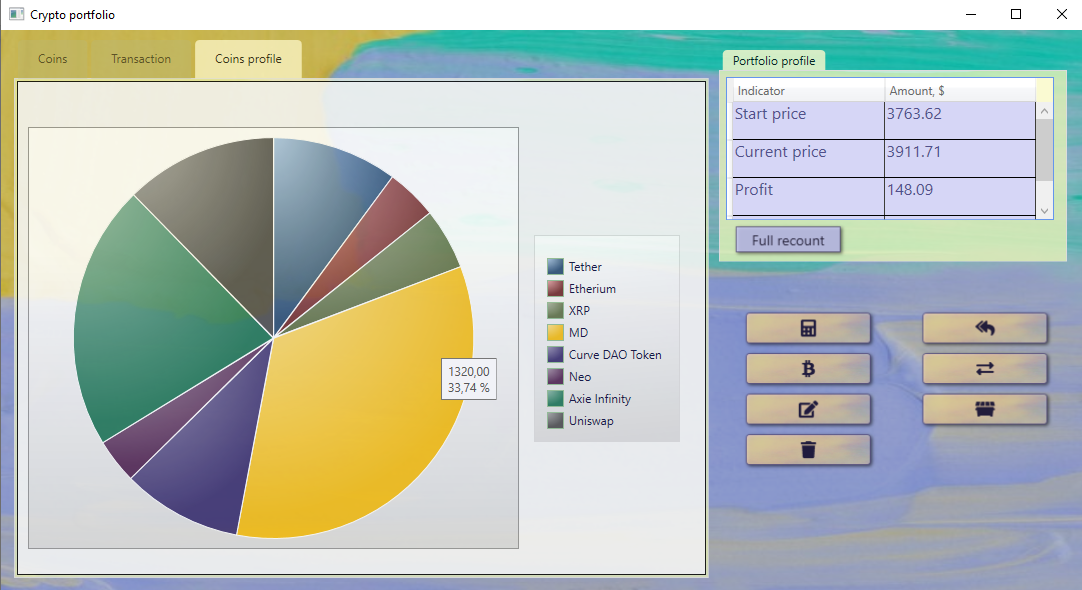


Рисунок 19 - Обновленная диаграмма на вкладке Coin profile

### 

### 5. Заключение

При работе над курсовым проектом была разработана программа для ведения учёта криптовалют. Программа 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 курсовой работы были применены знания полученные при изучении курса “Средства визуального программирования приложений”.

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

### 6. 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 <https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>, свободный. — Загл. с экрана.