МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет-изданий)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии и стандарты проектирования»

Тема «Торговая платформа Lucky Currency»

Исполнитель

студент 2 курса группы 6 Ярмолик Максим Александрович

Руководитель работы ассистент Радиванович Д.А.

Курсовой проект защищён с оценкой

Председатель Пацей Н.В

Минск 2021

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc73124035)

[1. Аналитический обзор литературы 4](#_Toc73124036)

[2. Анализ требований к программному средству и разработка функциональных требований 6](#_Toc73124037)

[3. Проектирование программного средства 8](#_Toc73124038)

[3.1 Проектирование архитектуры приложения 8](#_Toc73124039)

[3.2 Проектирование базы данных 10](#_Toc73124040)

[3.3 Проектирование логики сценариев использования 12](#_Toc73124041)

[4. Создание (реализация) программного средства 13](#_Toc73124042)

[4.1 Структура пакетов приложения 13](#_Toc73124043)

[4.2 Реализация сущностей приложения 13](#_Toc73124044)

[4.3 Реализация ViewModel 14](#_Toc73124045)

[4.4 Реализация представления 16](#_Toc73124046)

[5. Тестирование, проверка работоспособности и анализ полученных   
результатов 17](#_Toc73124047)

[6. Руководство по использованию 20](#_Toc73124048)

[ЗАКЛЮЧЕНИЕ 24](#_Toc73124049)

[Приложение А 25](#_Toc73124050)

[Приложение Б 26](#_Toc73124051)

[Приложение В 27](#_Toc73124052)

[Приложение Г 28](#_Toc73124053)

ВВЕДЕНИЕ

Начиная примерно с 2016 года, в нашу жизнь прочно вошла криптовалюта. По мере роста популярности новой технологии развивался трейдинг криптовалют. И сегодня торговля, биткоином, эфиром, альткоинами вызывает интерес у многих профессиональных трейдеров и инвесторов. Возникли специализированные электронные биржи, занимающиеся торговлей исключительно альтернативными деньгами.

Торговая платформа «Lucky Currency» – это программное средство, позволяющее торговать криптовалютными деривативами на бирже Bybit (www.bybit.com) посредством API-ключа.

Для начала работы необходимо зарегистрироваться и закрепить за своим аккаунтом API-ключ, ведь проще запомнить логин и пароль, нежели набор символов (API-ключ). Хранение всех пользователей будет осуществлено в базе данных, и при авторизации или регистрации будет непосредственное обращение к ней.

В ходе использования этой программы, пользователи смогут:

получать информацию о текущем балансе, открытых позициях, активных ордерах, закрытых позициях;

торговать криптовалютными линейными деривативами.

В основы приложения положено применение принципов ООП, архитектурного паттерна MVVM, паттернов проектирования и работы с БД.

В качестве интерфейса прикладного программирования был выбран обширный API-интерфейс — Windows Presentation Foundation (WPF), предназначенный для создания настольных программ с графически насыщенным пользовательским интерфейсом.

1. Аналитический обзор литературы

Для выполнения этого курсового проекта не обошлось без использования дополнительной литературы. Таковой является информация на сайте https://metanit.com/. На этом сайте имеется много информации, нужной для разработки этого курсового проекта. Также использовалась информация с сайта http://professorweb.ru/ и, конечно же, статьи с https://habr.com/ru/ были также очень полезны.

Функционал, необходимый для торговли, смотрел на web-версии биржи https://www.bybit.com/trade/usdt/BTCUSDT. Краткий обзор сервиса представлен ниже.

Свечной график, отражающий историю торгов (рисунок 1.1).



Рисунок 1.1 - Свечной график

Книга ордеров, отражающая ордера на бирже, с ценой близкой к текущей, и последние сделки (рисунок 1.2)

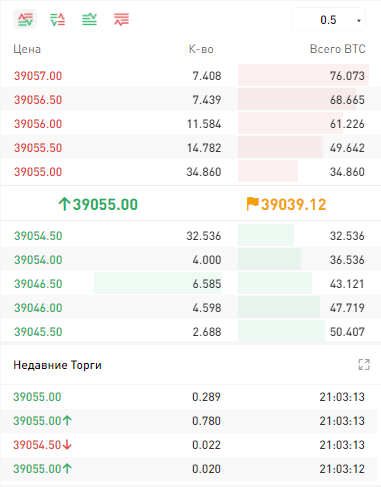


Рисунок 1.2 - Книга ордеров и последние сделки на бирже

Информация об открытых позициях, активных ордерах, закрытых позициях (рисунок 1.3).

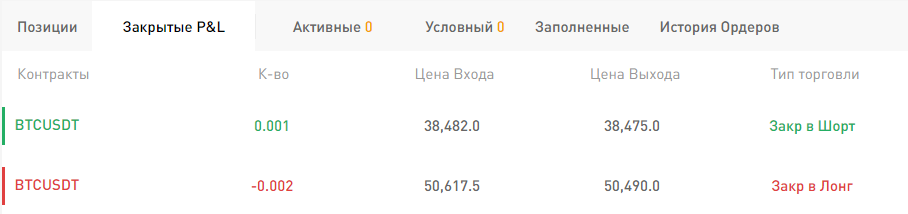


Рисунок 1.3 - Закрытые позиции

Панель для создания ордера, позволяющая открыть/закрыть позицию лимитным/рыночным ордеров на покупку/продажу (рисунок 1.4)

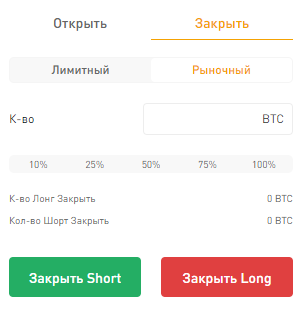


Рисунок 1.4 - Панель для создания ордера

У биржи нет desktop-версии, поэтому я решил это дело исправить. Также можно добавить кнопки с дополнительным функционалом. К примеру, кнопку, позволяющую выставить лимитный ордер по последней цене в книге ордеров (очень полезно, если не хочется платить комиссию бирже, но при этом необходимо как можно скорее открыть/закрыть позицию).

1. Анализ требований к программному средству и разработка функциональных требований

Программный продукт должен быть реализовать на объектно-ориентированном языке программирования C# с использованием технологии WPF. Windows Presentation Foundation (WPF) — система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML.

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

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

Также существует урезанная версия CLR, называющаяся WPF/E, она же известна как Silverlight.

База данных была разработана в Microsoft SQL Server.

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

SQL Server характеризуется такими особенностями как:

производительность. SQL Server работает очень быстро;

надежность и безопасность. SQL Server предоставляет шифрование данных;

простота. С данной СУБД относительно легко работать и вести администрирование.

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

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос на языке SQL должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.

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

Для осуществления связи между базой данных и приложением на C# необходим посредник, которым будет являться технология Entity Framework. Она основана на платформе .NET Framework и предназначена для работы с данными.

Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Эта технология предоставляет нам набор классов, через которые мы можем отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций.

Entity Framework предполагает три возможных способа взаимодействия с базой данных:

database first: Entity Framework создает набор классов, которые отражают модель конкретной базы данных;

model first: сначала разработчик создает модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере;

code first: разработчик создает класс модели данных, которые будут храниться в базе данных, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы.

В курсовом проектировании для разработки был выбран первый подход.

Приложение необходимо реализовать с использованием шаблона проектирования MVVM и с организацией взаимодействия с базой данных.

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

Программное средство предоставляет пользователю следующие функциональные возможности:

регистрация пользователя;

авторизация пользователя;

получение информации о текущем балансе;

получение информации об открытых позициях;

получение информации об активных ордерах;

получение информации о закрытых позициях;

получение информации о книге ордеров;

получение информации о списке последних сделок;

открытие позиции лимитным/рыночным ордером на продажу/покупку;

закрытие позиции лимитным/рыночным ордером на продажу/покупку.

Наглядно это видно на UML-диаграмме возможностей в приложении А.

1. Проектирование программного средства
   1. Проектирование архитектуры приложения

Архитектура программного обеспечения — совокупность важнейших решений об организации программной системы. Архитектура включает:

* выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;
* соединение выбранных элементов структуры и поведения во всё более крупные системы;
* архитектурный стиль, который направляет всю организацию — все элементы, их интерфейсы, их сотрудничество и их соединение.

Для удовлетворения проектируемой системы различным атрибутам качества применяются различные архитектурные шаблоны (паттерны). В разрабатываемом приложении используется архитектурный шаблон Model-View-ViewModel (MVVM).

На рисунке 3.1 представлена диаграмма, которая показывает общую структуру приложения в рамках шаблона MVVM.

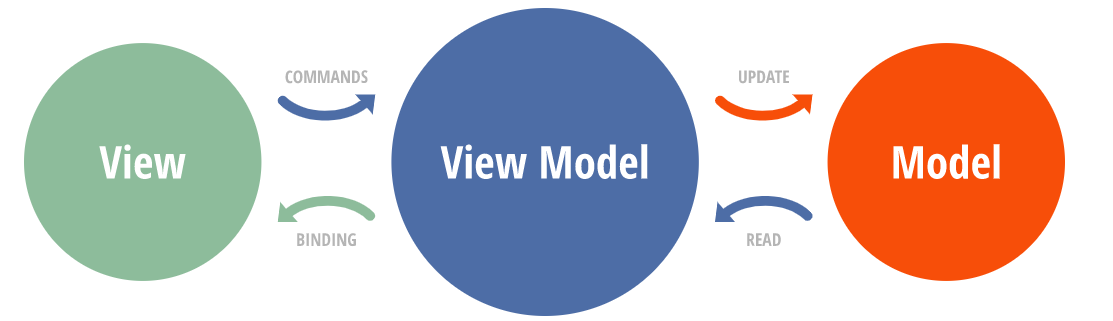


Рисунок 3.1 – Структура шаблона MVVM

Шаблон MVVM имеет три основных слоя: модель, которая представляет бизнес-логику приложения, представление пользовательского интерфейса, и представление-модель, в котором содержится вся логика построения графического интерфейса и ссылка на модель, поэтому он выступает в качестве модели для представления.

View или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Так как пользовательский интерфейс и качество его реализации играет далеко не последнее место в конечном результате, разработка эффективного интерфейса, приятного и удобного для конечного пользователя, является важной задачей. Поэтому для хорошего проектирования View необходимо понять, как пользователь будет взаимодействовать с приложением. Для этого была составлена схема на рисунке 3.2, на которой представлен принцип работы приложения с точки зрения пользователя.

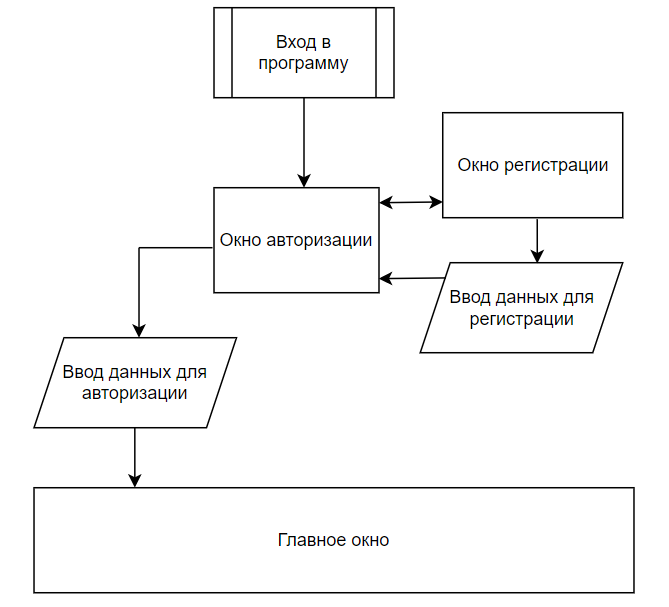


Рисунок 3.2 – Принцип работы приложения

При запуске приложения у нас появляется окно авторизации, которое требует ввода логина и пароля для начала работы.

В случае, если вас нету своей учётной записи, её можно создать, нажав на кнопку «Sign up» в окне авторизации. Вы будете направлены в окно регистрации, где вам нужно будет заполнить все поля, здесь вы укажите ваш API-ключ. После успешной регистрации, вы вернетесь в окно авторизации. Таким образом вы привяжите ваш API-ключ к аккаунту, теперь вам достаточно будет запомнить логин и пароль, для входа в аккаунт, что намного проще, нежели запоминание набора символов (API-ключа).

После ввода логина и пароля в соответствующих полях, проверяется правильность ввода данных.

В случае неверного ввода данных, выведется соответствующее сообщение. В ином случае окно авторизации закроется и откроется главное окно приложения.

ViewModel или модель представления связывает модель и представление через механизм привязки данных. Она содержит Модель, преобразованную к Представлению, а также команды, которыми может пользоваться Представление, чтобы влиять на Модель.

Model представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения.

Для разработки приложения в качестве сервиса для хранения данных была выбрана СУБД Microsoft SQL Server. На диаграмме размещение на рисунке 3.3 показано графическое представление инфраструктуры приложения.

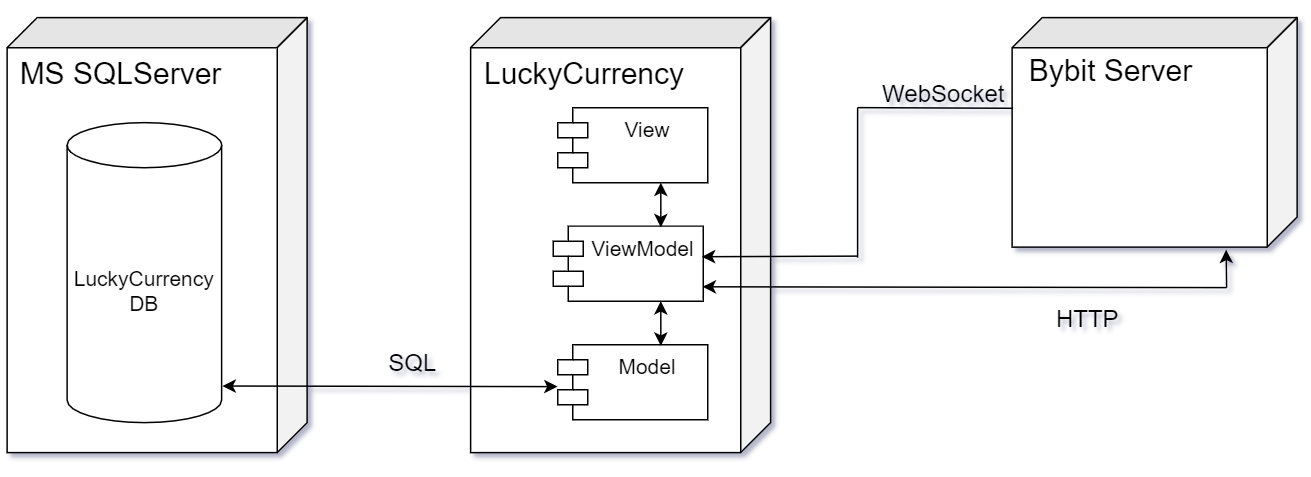


Рисунок 3.3 – Диаграмма размещения

Связь приложения с базой данных Microsoft SQL Server происходит с помощью SQL-команд.

Связь с биржей Bybit поддерживается посредством HTTP запросов и WebSocket соединения.

* 1. Проектирование базы данных

Проектирование баз данных — процесс создания схемы базы данных и определения необходимых ограничений целостности.

Основные задачи проектирования базы данных:

* обеспечение хранения в БД всей необходимой информации;
* обеспечение возможности получения данных по всем необходимым запросам;
* сокращение избыточности и дублирования данных;
* обеспечение целостности базы данных.

Проектирование базы данных проводится в два этапа: концептуальное и логическое проектирование.

Концептуальное (инфологическое) проектирование — построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. В результате этого этапа создаётся ER-модель. Такая модель создаётся без ориентации на какую-либо конкретную СУБД и модель данных.

Основными понятиями ER-модели являются: сущность, связь и атрибут.

Сущность – это реальный или представляемый объект, информация о котором должна сохраняться и быть доступна.

Связь – это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Эта ассоциация обычно является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь).

Атрибут сущности – это любая детaль, которая служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности.

Логическое проектирование — создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели данных логическая модель — набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи. На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

Логическая модель базы данных представлена на рисунке 3.4.

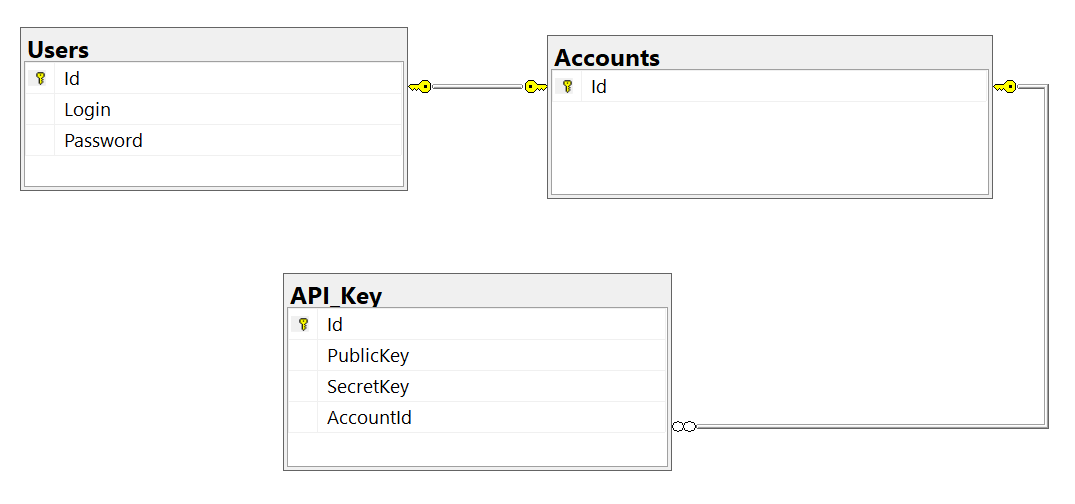


Рисунок 3.4 – Логическая модель базы данных

В базе данных содержится 3 таблицы:

* Users;
* Accounts;
* API\_Key.

Таблица Users хранит информацию о пользователях приложения. Она содержит столбцы:

Id;

Login;

Password.

Столбцы Password и Login – данные необходимые для авторизации.

Таблицы Users и Accounts имеют связь один ко одному. Столбец Id в таблице Accounts одновременно является первичным и внешним ключом, указывающим на Id в таблице Users.

Таблицы Accounts и API\_Key имеют связь один ко многим, столбец AccountId в таблице API\_Key является внешним ключом, указывающим на Id в таблице Accounts.

Таблица API\_Key хранит информацию об API-ключе и имеет следующие столбцы:

Id;

PublicKey;

SecretKey;

AccountId.

* 1. Проектирование логики сценариев использования

При проектировании приложения были рассмотрены различные сценарии использования. Наглядно это видно на UML-диаграмме последовательности в приложении Б.

На диаграмме видно, что для начала работы необходимо авторизироваться, после чего открывается главное окно, которое взаимодействует с Bybit Server. Главное окно имеет следующий функционал:

получение информации о текущем балансе;

получение информации об открытых позициях;

получение информации об активных ордерах;

получение информации о закрытых позициях;

получение информации о книге ордеров;

получение информации о списке последних сделок;

открытие позиции лимитным/рыночным ордером на продажу/покупку;

закрытие позиции лимитным/рыночным ордером на продажу/покупку.

1. Создание (реализация) программного средства
   1. Структура пакетов приложения

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

Пакетная структура проекта показана на рисунке 4.1

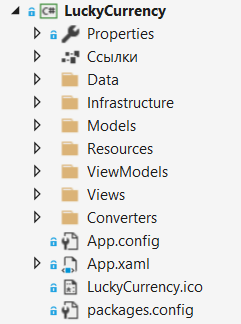


Рисунок 4.1 – Пакетная структура проекта

Более подробно структура проекта описана в таблице 4.

Таблица 4.1 – Описание структурных пакетов проекта

|  |  |
| --- | --- |
| Имя пакета | Описание |
| Models | описание сущностей |
| Infrastructure | содержит основные классы, используемые внутренней логикой (к примеру, реализацию команды WPF) |
| Converters | вспомогательные классы-конвертеры |
| Data | работа с базой данных |
| Resources | файлы стилей |
| Views | представления |
| ViewModels | основная бизнес-логика |
| App.config | файл конфигурации приложения |
| App.xaml | общие ресурсы приложения |

В этой таблице приведены основные логические составляющие пакеты, которые используются в работе приложения.

* 1. Реализация сущностей приложения

В данном программном решении был использован подход Code-First. При данном подходе модель EDMX не используется. Создание базы данных происходит из созданной вручную модели объектов C#.

Диаграмма классов UML для сущностных классов представлена на рисунке 4.2.

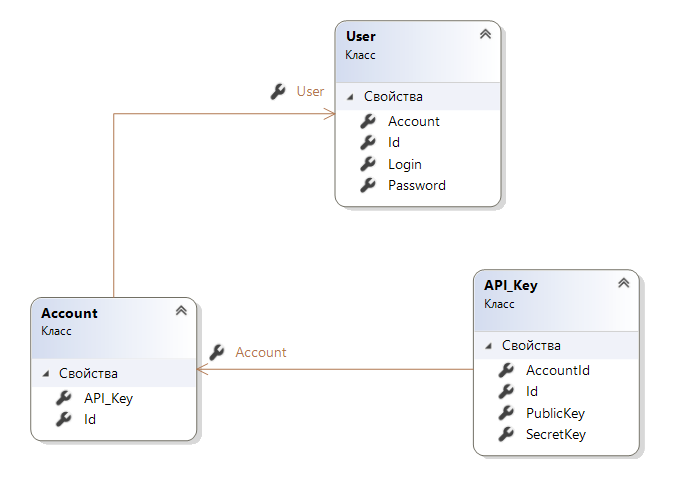


Рисунок 4.2 – UML диаграмма классов-сущностей.

* 1. Реализация ViewModel

Слой view-model – промежуточное звено, которое связывает представление и данные. Пакетная реализация данного звена представлена на рисунке 4.3.

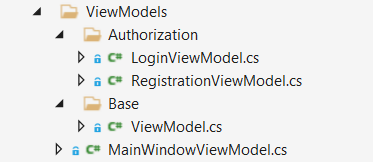


Рисунок 4.3 – Пакетная реализация ViewModel

Диаграмма классов UML для классов view-model представлена на рисунке 4.4.

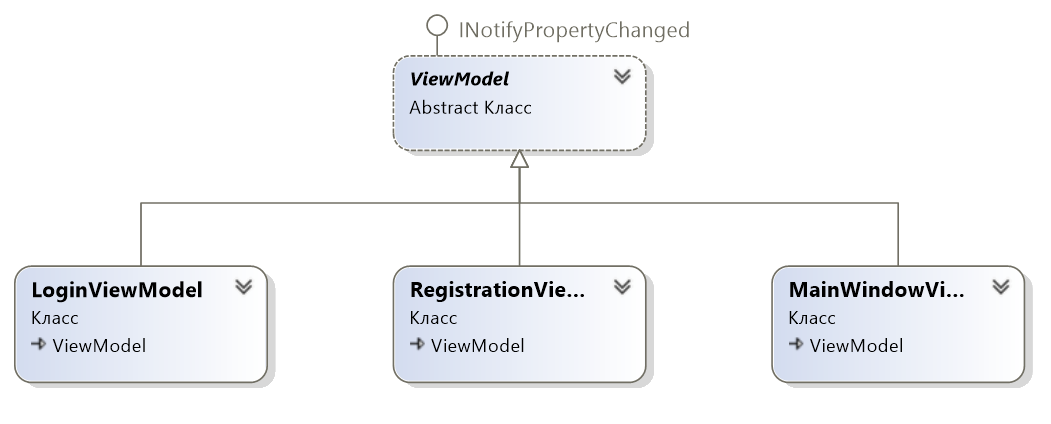


Рисунок 4.4 – UML диаграмма классов view-model

Для возможности связать данные из view-model через привязки с представлением – был реализован интерфейс INotifyPropertyChanged, в родительском классе ViewModel. Для его реализации, нужно объявить свойство PropertyChanged и метод OnPropertyChanged(). Для того, чтобы начать отслеживание свойства, необходимо вызывать метод OnPropertyChanged(“Имя свойства”). Код реализации предоставлен в приложении В.

Для удобной работы с данными, используется паттерн Repository. Repository - позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой. Так как работа с данными, ведётся через один контекст, был реализован паттерн UnitOfWork. Который реализует интерфейс IDisposable – за счёт этого происходит подчистка данных после работы с контекстом. Код реализации UnitOfWork представлен в приложении Г.

UML диаграмма классов входящих в состав паттернов Repository и UnitOfWork представлена на рисунке 4.5

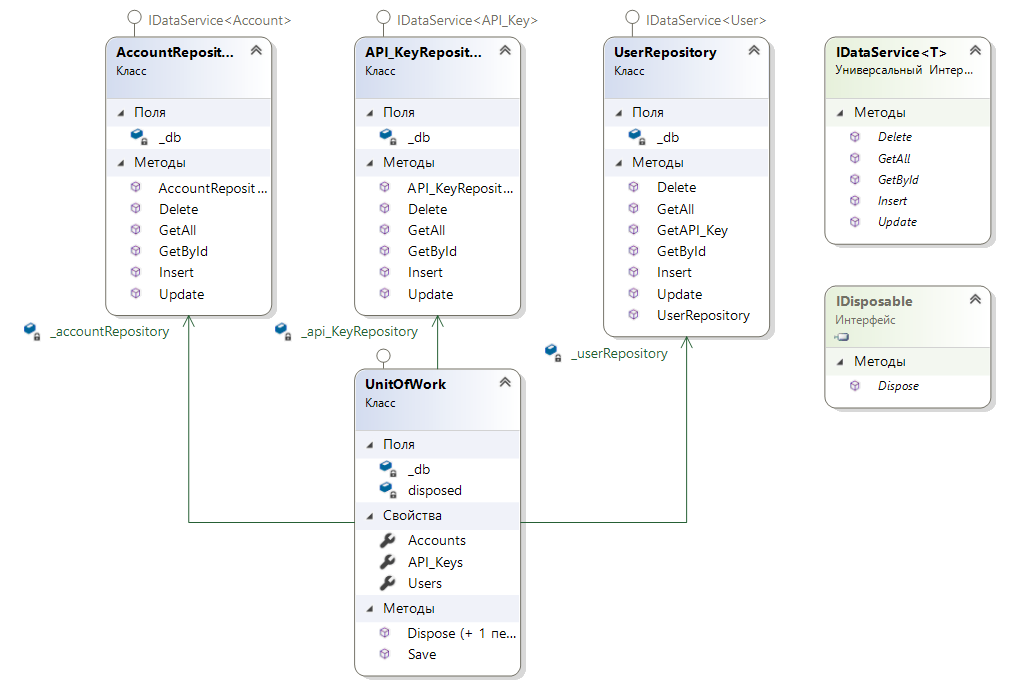


Рисунок 4.5 – Реализация паттернов Repository и UnitOfWork

* 1. Реализация представления

Структура пакета View, в котором хранятся файлы представлений на языке разметки xaml, а также соответствующие им файлы на языке C#, показана на изображении 4.6.

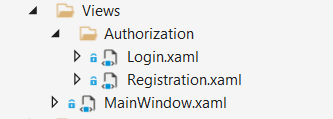


Рисунок 4.6 – Структура пакета View

В этом пакете находятся все «окна» и «страницы», которые мы видим, используя приложение. Назначение каждого файла описано в таблице ниже (таблица 4.1)

Таблица 4.1 – Описание файлов View

|  |  |
| --- | --- |
| Название | Краткое описание |
| Login.xaml | окно авторизации |
| Registration.xaml | окно регистрации |
| MainWindow.xaml | главное окно |

1. Тестирование, проверка работоспособности и анализ полученных результатов

При тестировании данного приложения, были применены сценарии, которые могли бы привести к ошибке. В этой главе мы рассмотрим некоторые такие сценарии и посмотрим на их обработку.

В момент регистрации/авторизации, возможна такая ситуация, в которой пользователь ничего не ввёл. Обработка данного сценария приведена на рисунке 5.1.

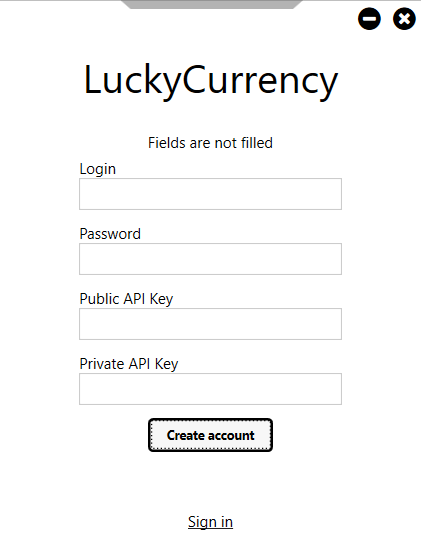


Рисунок 5.1 – Поля не заполнены

Обработка ситуации ввода неверного логина или пароля представлена на рисунке 5.2.

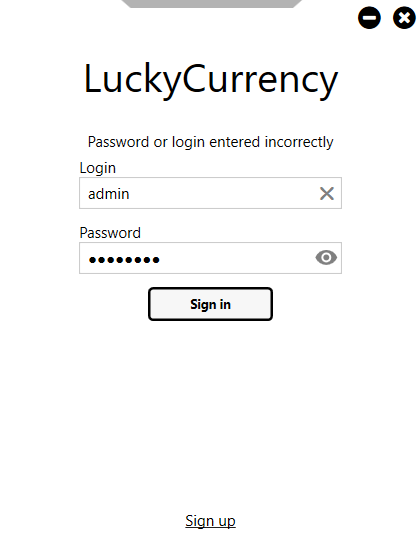


Рисунок 5.2 – Неверный логин или пароль

Попытка создать пользователя с повторяющимися данными также обрабатывается в приложении, демонстрация на рисунке 5.3.

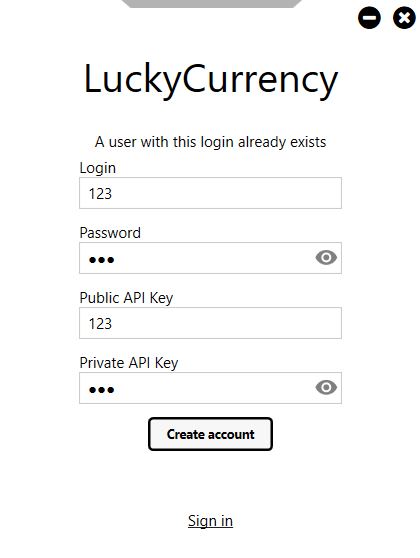


Рисунок 5.3 – Регистрация пользователя с повторяющимся логином

Также было проведено тестирование работы валидации внутри приложения. К примеру, в случае невалидного API-ключа, пользователь будет об этом уведомлен, демонстрация на рисунке 5.4.

При попытке выставления ордера с невалидным ключом, также будет уведомление (рисунок 5.5)

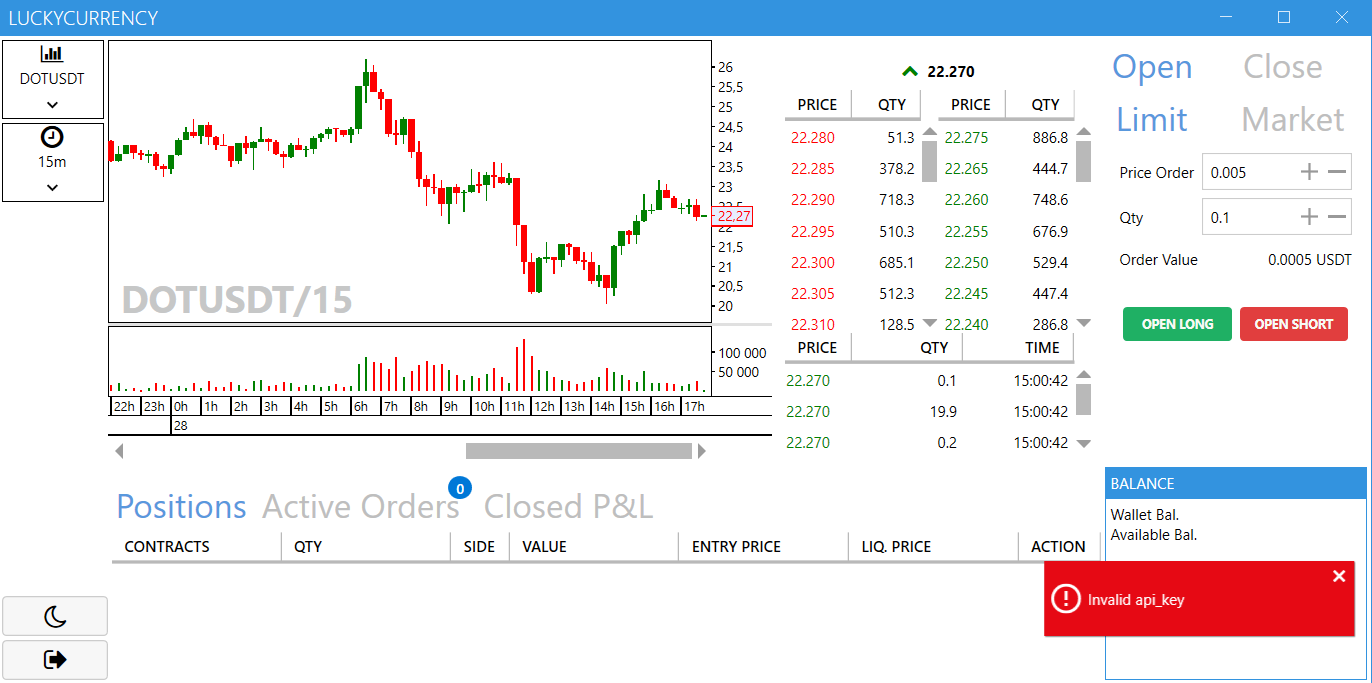


Рисунок 5.4 – Невалидный API-ключ (авторизация)



Рисунок 5.5 – Невалидный API-ключ (попытка выставления ордера)

На данном этапе были выполнены тесты на проверку работоспособности приложения, а именно на проверку валидации.

1. Руководство по использованию

После запуска приложения открывается окно авторизации (рисунок 6.1), если у вас нет аккаунта, можно зарегистрировать нового пользователя (рисунок 6.2). В форме регистрации необходимо ввести API-ключ, предоставленный биржей Bybit. Таким образом, после регистрации, вы закрепите за своим аккаунтом API-ключ и теперь для входа вам нужно знать только логин и пароль, а API-ключ будет браться из базы данных.

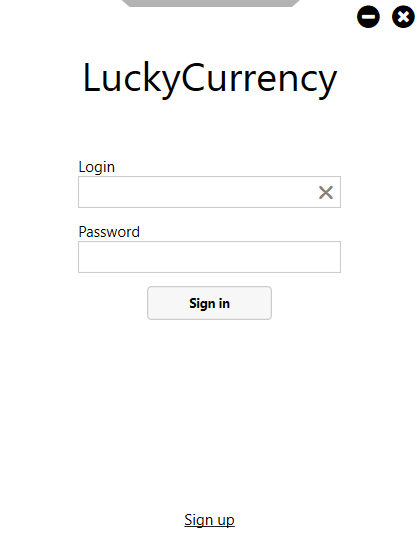


Рисунок 6.1 – Окно авторизации

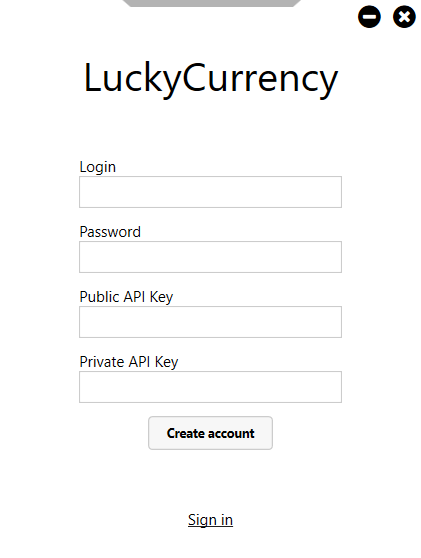


Рисунок 6.2 – Окно регистрации

После успешного входа откроется главное окно. Оно содержит боковое меню, свечной график, книгу ордеров, список последних сделок на бирже, список открытых позиций, список активных ордеров, список закрытых позиций, текущий баланс и панель для торговли (рисунок 6.3).

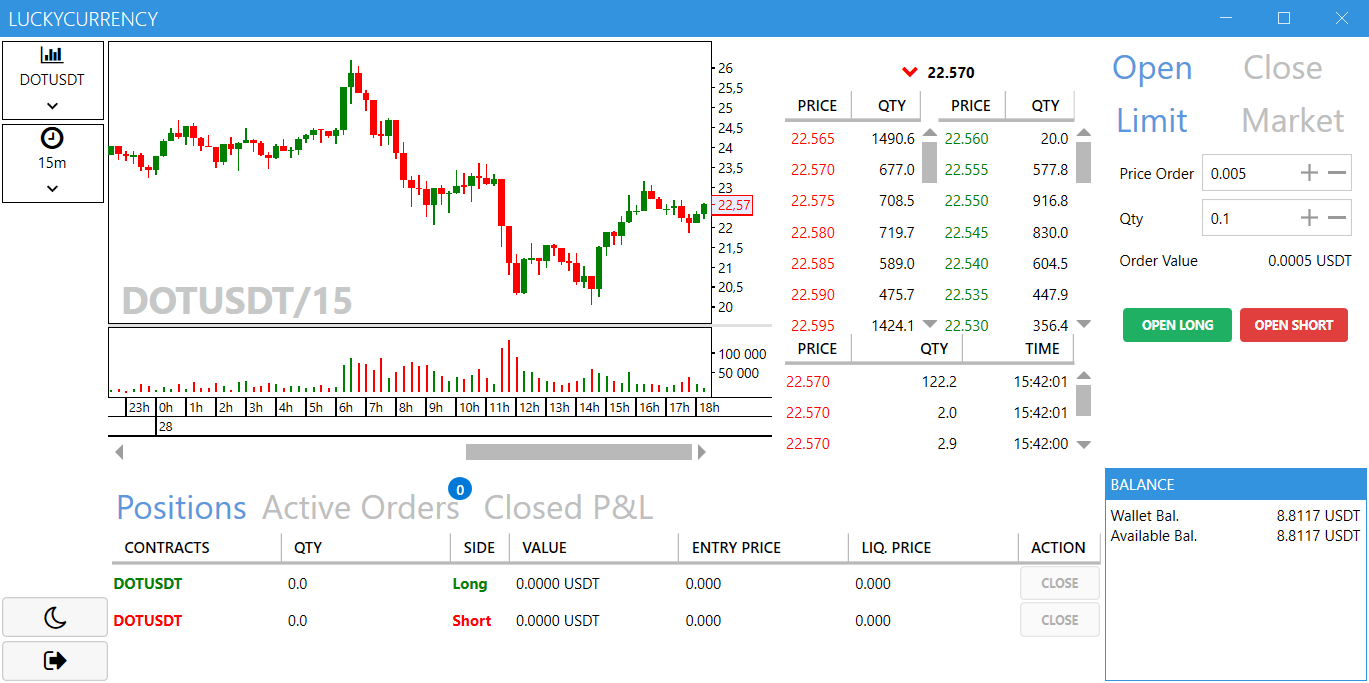


Рисунок 6.3 – Главное окно

Есть возможность открыть/закрыть позицию лимитным или рыночным ордером. При создании лимитного ордера, он будет помещен в книгу ордеров и мы будем ожидать исполнения нашего ордера, таким образом мы будем являться «мейкером». Мейкеров биржа поощряет отрицательной комиссией (-0,025%), так как они создают плотность книги ордеров. Пример создания лимитного ордера на покупку приведен на рисунке 6.4.

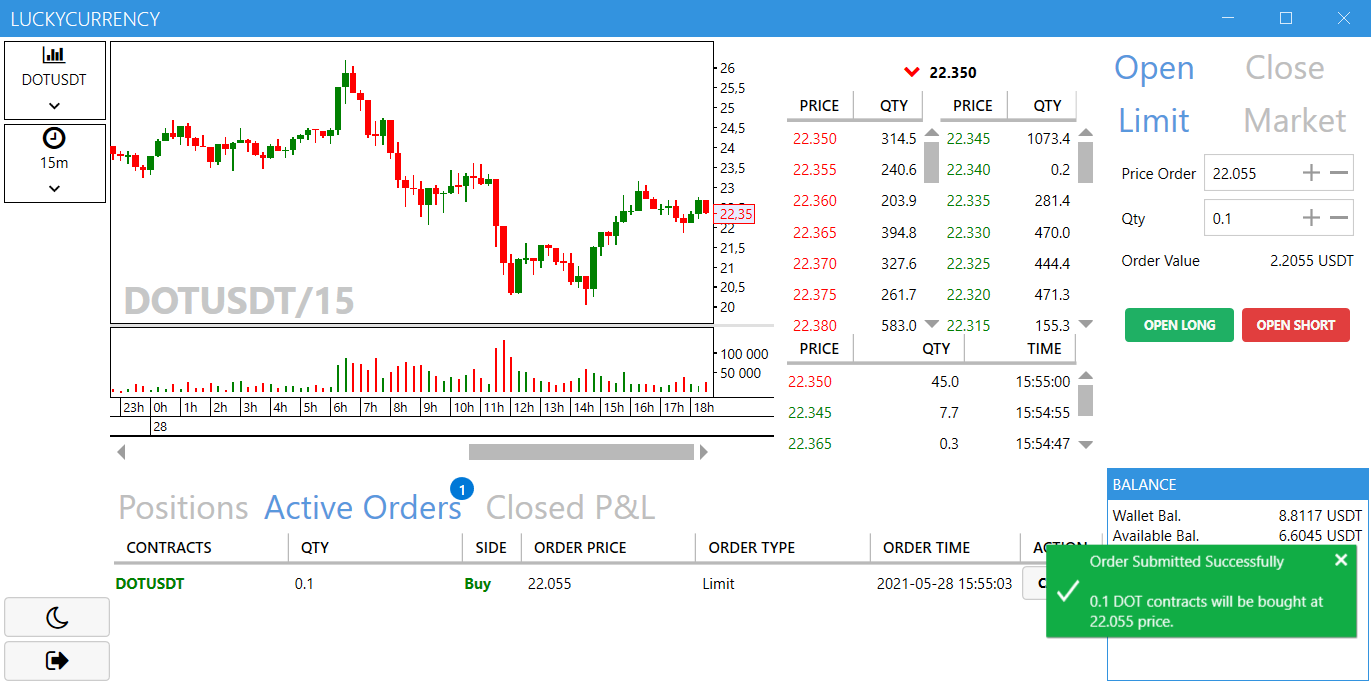


Рисунок 6.4 – Выставление лимитного ордера на покупку

Если же мы хотим выставить ордер, который сразу же будет исполнен, то наш выбор – рыночный ордер. В данной ситуации мы будем являться «тейкером». Тейкер, это тот, кто покупает или продает по рынку, он удовлетворяет ближайшие ордера в книге ордеров. Биржа будет взымать с нас комиссию (0,075%). Пример создания рыночного ордера на продажу представлен на рисунке 6.5.

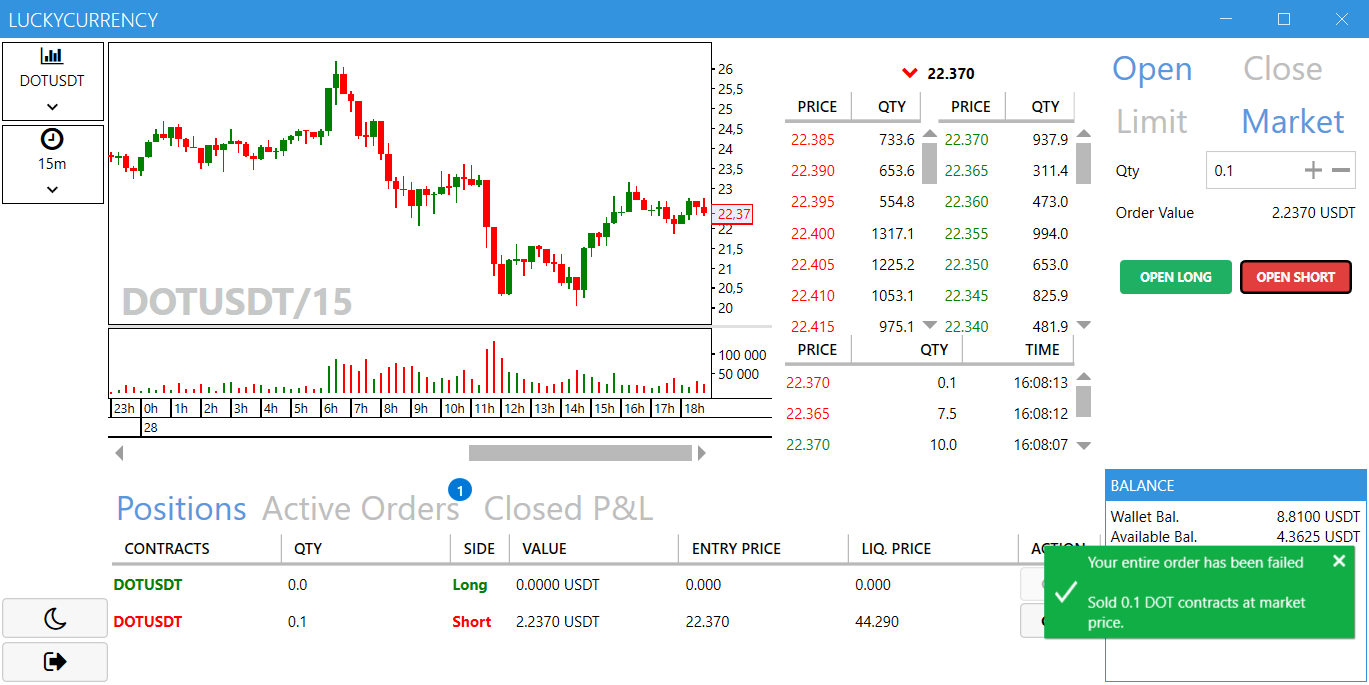


Рисунок 6.5 – Выставление рыночного ордера на продажу

После закрытия позиции (открытой рыночным ордером выше), мы можем посмотреть информацию о ней во вкладке «Closed P&L» (рисунок 6.6)

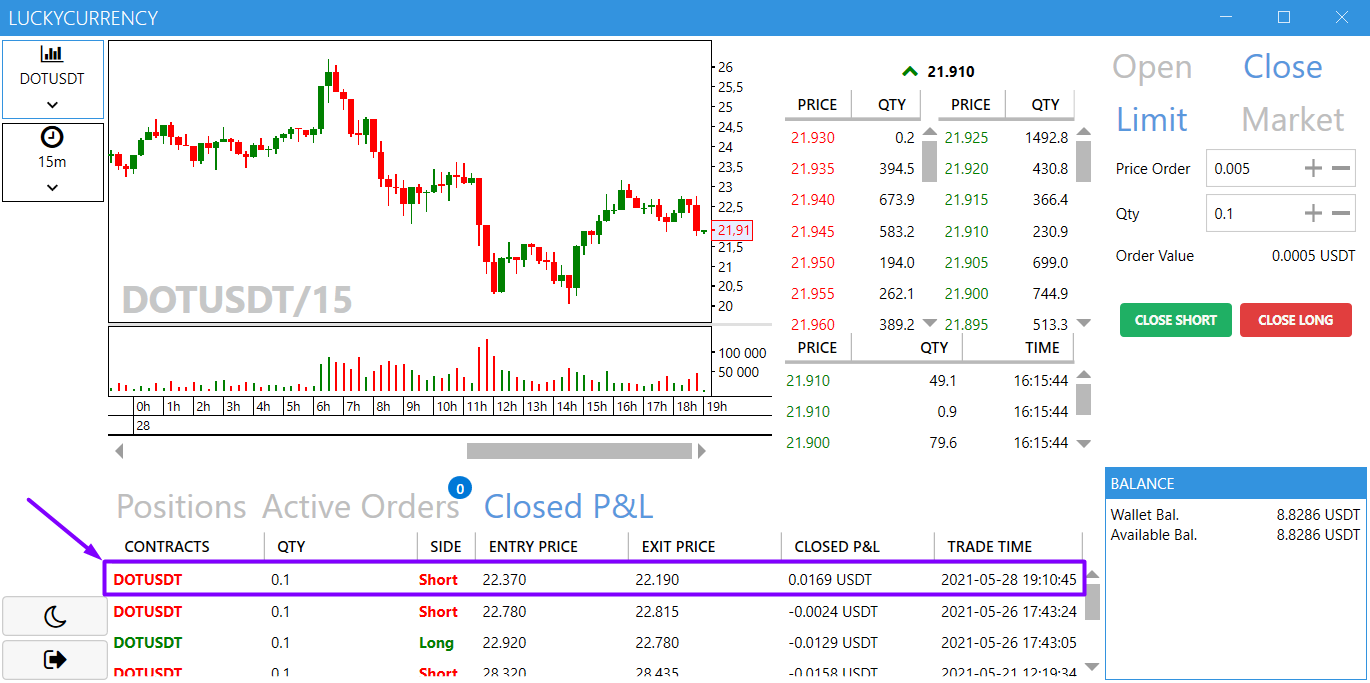


Рисунок 6.6 – Информация о закрытых позициях

ЗАКЛЮЧЕНИЕ

В данном курсовом проекте была разработана торговая платформа «Lucky Currency» при помощи языка программирования C#, API-интерфейса Windows Presentation Foundation, технологии Entity Framework и базы данных Microsoft SQL Server. При выполнении курсового проекта использовались принципы и приемы ООП.

Программное средство предоставляет пользователю следующие функциональные возможности:

регистрация пользователя;

авторизация пользователя;

получение информации о текущем балансе;

получение информации об открытых позициях;

получение информации об активных ордерах;

получение информации о закрытых позициях;

получение информации о книге ордеров;

получение информации о списке последних сделок;

открытие позиции лимитным/рыночным ордером на продажу/покупку;

закрытие позиции лимитным/рыночным ордером на продажу/покупку.

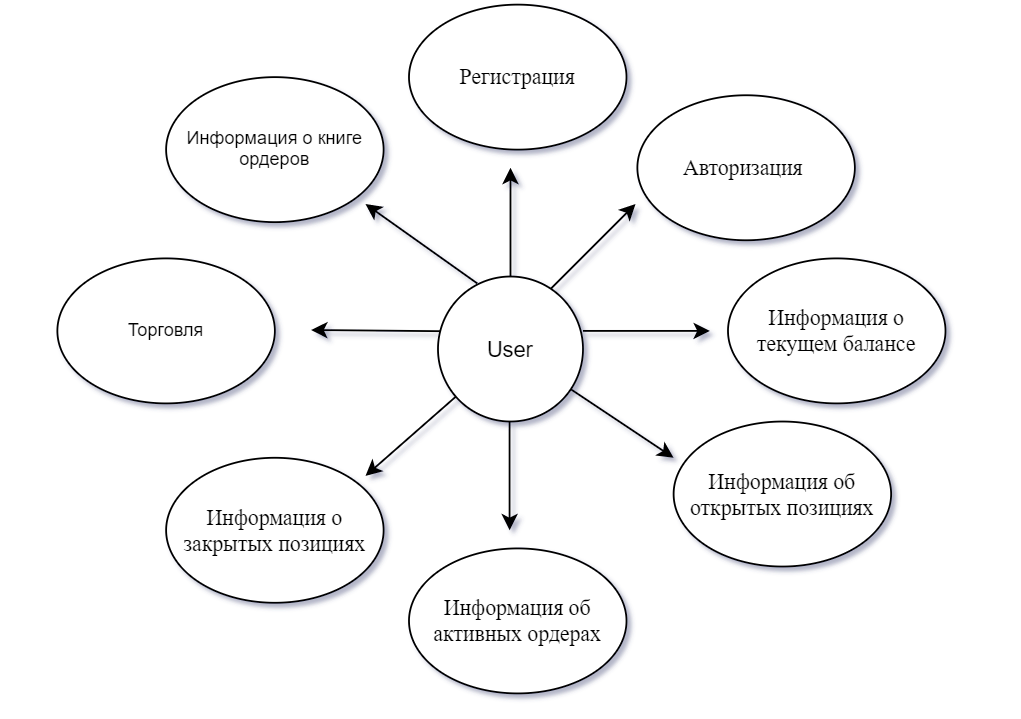
Проектирование осуществлялось по паттерну проектирования MVVM.

Для удобной работы с данными, был реализован паттерн Repository и UnitOfWork.

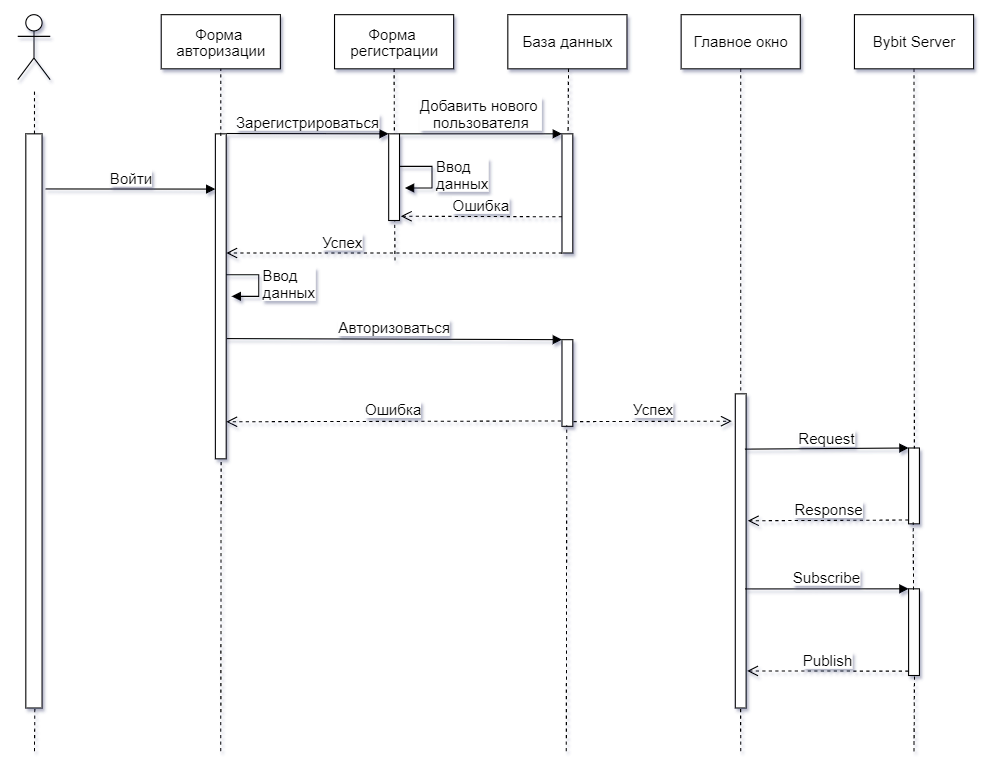
Приложение было успешно протестировано, что означает пригодность этого приложения для своей цели.

Для хранения исходного кода и удобства контроля версий проекта использовался GitHub. Ссылка на проект https://github.com/LilDev1l/LuckyCurrency

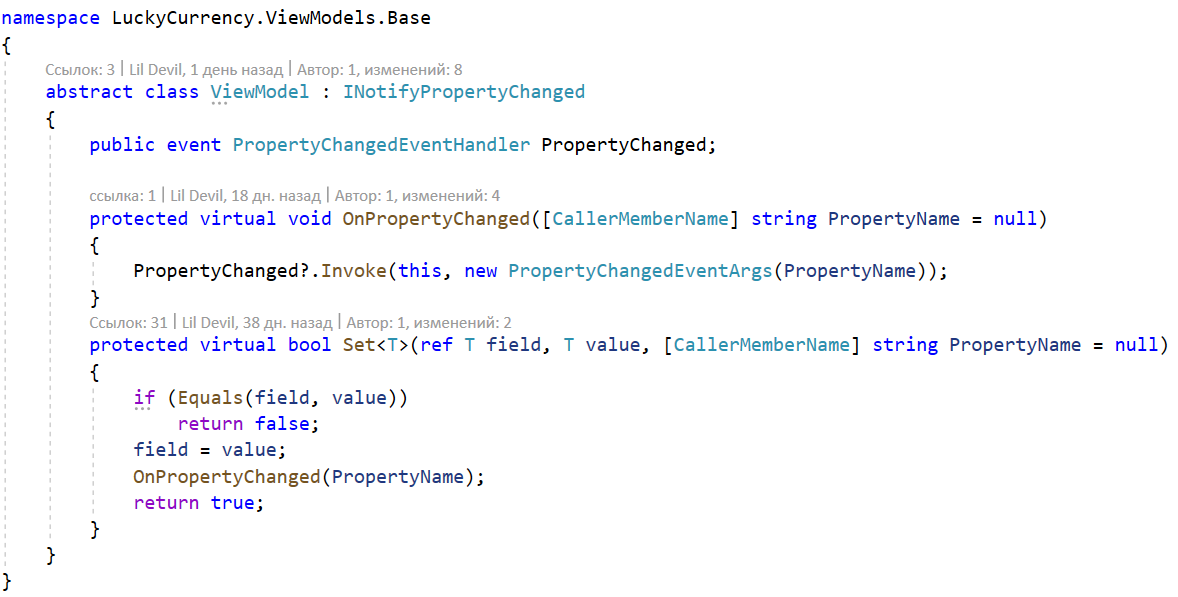
Приложение А



Приложение Б



Приложение В



Приложение Г