# 1. Терминология

## **Common**

**Currency (Валюта)** — валюта, тип денег. Например, доллар США или биткоин.

**Currency pair (Валютная пара)** — пара валют, которая означает отношение одной валюты к другой. Как и в простом математическом отношении порядок имеет значение. Например, доллар США к биткоину.

**Symbol (Символ)** — символьное обозначение валюты или валютной пары. Например, USD или USDBTC.

**Platform (Платформа)** — система(, которая является основой для разной деятельности). Программная платформа — это программная система, обычно со своим API.

**Exchange (Биржа)** — учреждение для заключения финансовых и коммерческих сделок. Часто биржа реализуется как программная платформа, доступная через интернет в любой точке мира.

## **Development**

**Connector (Коннектор)** — в широком смысле это класс или группа классов, через которые мы подключаемся к API отдельных платформ, чтобы выполнить какое-либо действие. В нашем случае такие коннекторы разбиваются на клиенты, которые приводят REST и WS-интерфейсы отдельных платформ к единому интерфейсу на уровне кода, то есть реализованному в классе, а также на коннекторы в узком смысле, которые используют клиенты и реализуют более сложную логику использования API платформ, которая включает реконнекты, обработку ошибок, фиксирование скачанных и пропущенных данных, которые нужно скачать позже и так далее.

**Client (Клиент)** — класс, через который мы в программе обращаемся к другим внешним API: REST и WS API бирж, например. Другими словами, он преобразует внешний общий API в API на уровне классов конкретного языка программирования. Обычно, через одинаковый такой программный интерфейс можно обращаться к разным платформам, собственные API которых отличаются.

# 

# Интерфейс ITestConnector

Интерфейс, модели Trade и Candle доступны по [ссылке](https://drive.google.com/file/d/1RuY1PQs2esq_7hxsalORC-4-st1MdY8y/view?usp=drive_link) . Это упрощенный вариант интерфейса, если вам не будет хватать входных параметров, можно их дополнить но с объяснением, почему Вы так сделали.

Так-же, если посчитаете некоторые параметры лишними, можете их убрать, но опять же с объяснением (Можно в комментариях кода)

# 2. Тестовое задание

Необходимо реализовать коннектор под исходный интерфейс(пункт 3) на C# (Class Library), а так-же покрыть его интеграционными тестами, или сделать в отдельном проекте простой вывод на GUI Framework (WPF) на основе паттерна MVVM, не WinForms.

Что должно быть в этом коннекторе:

1. Класс клиента для REST API биржи Bitfinex, который реализует 2 функции:
   * Получение трейдов (trades)
   * Получение свечей (candles)
   * Получение информации о тикере (Ticker)
2. Класс клиента для Websocket API биржи Bitfinex, который реализует 2 функции:
   * Получение трейдов (trades)
   * Получение свечей (candles)

API Bitfinex [(ссылка на API)](https://docs.bitfinex.com/v2/docs) . Использовать версию API v2,

Также, реализовать расчет : на балансе есть 4 разных криптовалюты: 1 BTC, 15000 XRP, 50 XMR и 30 DASH. Необходимо вывести общий баланс портфеля в каждой из перечисленных валют в USDT, BTC, XRP, XMR и DASH (\* т.е. сложить и/или конвертировать) и вывести полученные результаты в Datagrid WPF

**Допущения**

1. Можно использовать API любой известной биржи (Binance, Bybit and etc)
2. Можно вместо WPF использовать asp.net (любой удобный). Winform или консоль по условию задачи не допускается.
3. Будет дана оценка на использование ИИ, если будет выполнено полное копирование, то задача не будет засчитана.

# На что будем смотреть в первую очередь

1. Простота и понятность реализации;
2. Качество кода;
3. Возможность легкого масштабирования системы;
4. Работоспособность коннектора, чтобы выполнял свои функции
5. Желательно иметь историю комитов в репозиторий, чтобы мы могли оценить последовательность выполнения задачи