Facultad de Ingeniería de la Universidad de Buenos Aires

Ingeniería en informática

Materia: Técnicas de Diseño - 75.10

Fecha de entrega: 13 de junio de 2024

Grupo #2

Integrantes:

Alumno	Padrón	Mail
Paula Brück	107533	pbruck.ext@fi.uba.ar
Clara Ruano Frugoli	106835	cruano@fi.uba.ar
Francisco Ezequiel Martínez	108460	femartinez@fi.uba.ar
Ramiro Gestoso	105950	rgestoso@fi.uba.ar

Descripción de la aplicación

Se construyó una aplicación que permite monitorear los valores de criptomonedas de un usuario (con una cuenta de Binance), y definir reglas de compra y de venta que se apliquen según las condiciones o variaciones del mercado.
El sistema permite monitorear la variación de todos los pares de cotización de monedas relevantes para el usuario. Se interpretó como relevante para el usuario cualquier par de cotizaciones que aparezcan en las reglas brindadas por el mismo.
Las reglas son completamente configurables, permitiendo combinar los distintos tipos de condiciones o reglas indicadas en el enunciado.
La aplicación notifica al usuario a través de Discord y Slack cuando se realiza una operación, indicando el nombre de la regla ejecutada, además de la moneda comprada o vendida junto con su cantidad.
La aplicación hace uso del websocket de Binance para obtener los updates de las monedas relevantes asi como tambien hace uso de la API REST para interactuar con la wallet y el mercado

Requerimientos a implementar

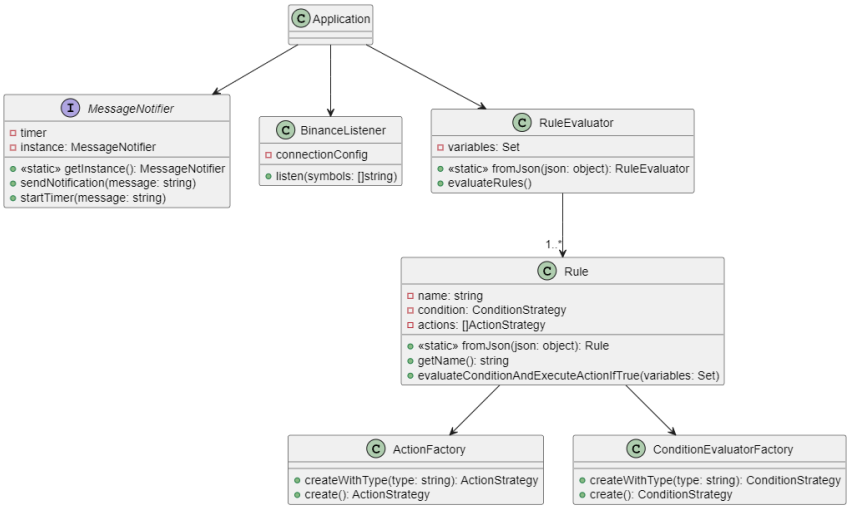
Como es pedido en el enunciado, la aplicación permite:

1. Configurar qué moneda va a operar y monitorear sus valores.
2. Ejecutarse varias veces para operar con distintas reglas y monedas.
3. Configurar reglas que digan cuándo comprar y cuándo vender en base a las variaciones de la moneda.

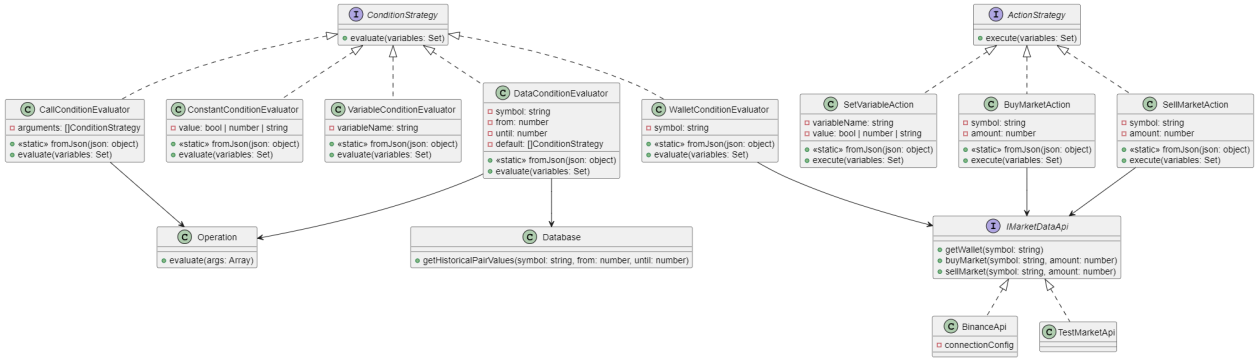
Modelo 4+1

Vista lógica

A nivel general la aplicación se ve conformada por las siguientes clases:

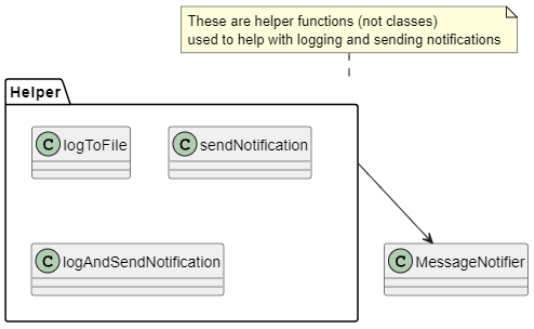


A continuación se presenta una vista más detallada de las condiciones y acciones en las reglas:



Paquete Helper

Un paquete adicional nos provee funciones útiles para realizar logs y notificar por distintos medios, siendo las alternativas actuales Slack y Discord. Estos paquetes son fácilmente extensibles para poder aceptar otro tipo de notificadores, por ejemplo.



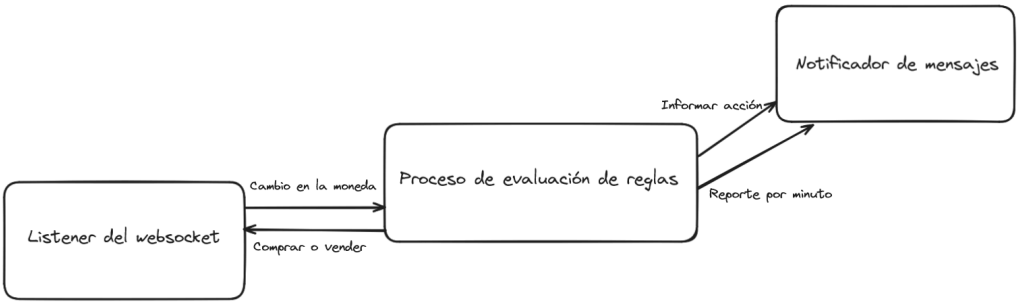
Vista física

Creemos que correr el ejecutable sobre una máquina local bastará para el objetivo deseado. También podría utilizarse una máquina en la nube.

Vista de procesos

Los procesos que estarán corriendo en conjunto son:

- El listener del websocket
- El proceso de evaluación de reglas
- El notificador de mensajes



Vista de casos de uso

Escenario 1:
Se lee del socket una variación de suba del BTC frente al USDT de un 0.1%, que coincide con el mínimo límite fijado de variación porcentual. Se evalúa una regla que evalúa si la variación fue mayor o igual al 0.1%. Como esto se cumple, se compra BTC con USDT con saldo suficiente. Se envía una notificación vía Discord con los detalles de la compra.

Escenario 2:
Se lee del socket una variación de suba del BTC frente al USDT de un 0.1%, que coincide con el mínimo límite fijado de variación porcentual. Se evalúa una regla que evalúa si la variación fue mayor o igual al 0.1%. Como esto se cumple, se compra BTC con USDT. Como no hay saldo suficiente, la compra no se realiza. Se envía una notificación vía Discord indicando la situación.

Escenario 3:
Se lee del socket una variación de suba del BTC frente al USDT de 0.05% con respecto al último valor guardado. Como el diferencial está por debajo de 0.1% se ignora el evento y no se almacena el valor leído del socket en la base de datos.

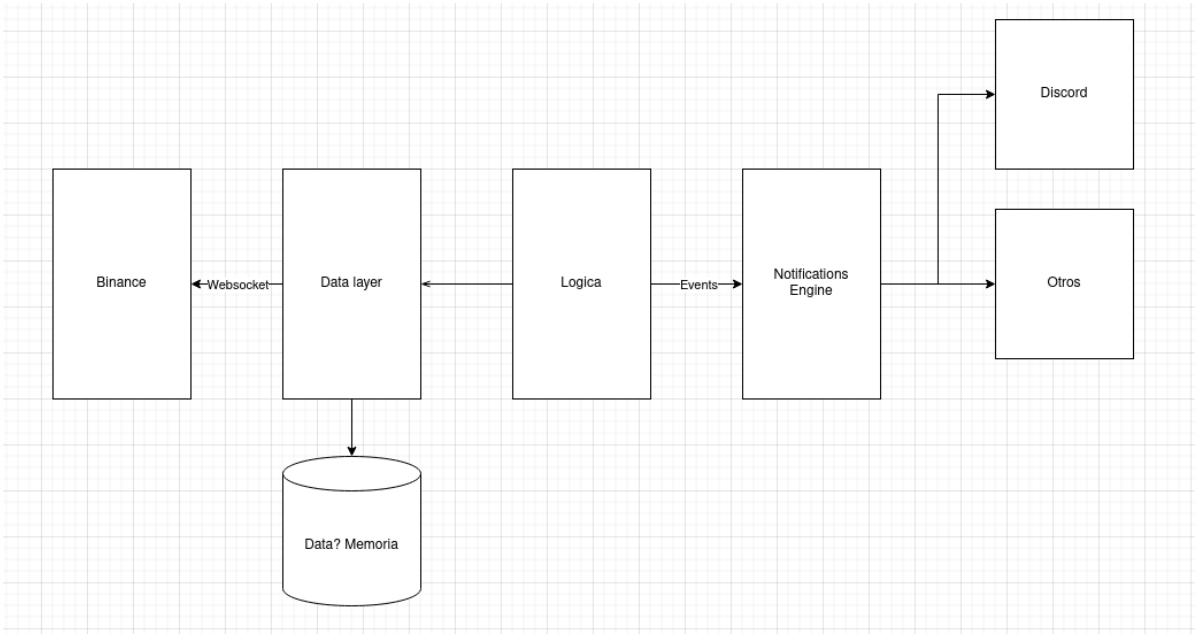
Escenario 4:
Se lee del socket una variación de suba del BTC frente al USDT de un 0.1%. Evaluamos una regla que compara el último precio al que se realizó la compra de BTC contra el precio actual. La regla indica que si el precio subió 10% vendemos una parte del BTC de la wallet para obtener ganancias. Como la regla no se cumple se ignora el evento.

Escenario 5:
Pasaron 30 segundos desde la última notificación enviada al usuario. Para que el usuario sepa que el bot sigue vivo, se envía un mensaje con el estado actual de su billetera, junto a un resumen de las últimas compras en el mercado (punto de inicio: el último resumen).
Ejemplo del resumen:

- 20 transacciones en monedas: moneda1, moneda2, etc
- Resultados finales (sumando lo comprado y restando lo vendido):
    1. moneda1: compraste X (se compró más de lo que se vendió)
    2. moneda2: vendiste Y (se vendió más de lo que se compró)
    3. moneda3: sin cambios (se vendió lo mismo de lo que se compró)

Arquitectura propuesta

Creemos que un patrón basado en eventos es una decisión adecuada para este proyecto, dado que el objetivo es hacer actualizaciones en base a los eventos reportados por Binance.
Adicionalmente, el patrón layers puede ayudarnos a obtener un código más modular y mantenible, como se puede apreciar en el siguiente diagrama:



Archivo de reglas

Una descripción del archivo de reglas se puede ver representado en el siguiente diagrama, donde se detallan los componentes (ruleset, rules, condiciones, argumentos, etc) propios del dominio del problema y sus relaciones: