Universidade do Minho
Escola de Engenharia

Sistema Multiagente de Gestão de Portfólio

Agentes e Sistemas Multiagente
Mestrado em Engenharia Informática

Grupo 12

Gabriela Santos Ferreira da Cunha - pg53829

Millena de Freitas Santos - pg54107

Nuno Guilherme Cruz Varela - pg54117



maio, 2024

Conteúdo

1 Introdução

No âmbito da unidade curricular de Agentes e Sistemas Multiagente, foi-nos primeiramente proposta a descrição e caracterização de um dado domínio tendo em consideração a aplicação de tecnologias de agentes e sistemas multiagente. Neste sentido, o tópico escolhido pelo grupo recaiu sobre os mercados financeiros, tendo sido previamente realizada uma investigação sobre o mesmo. Para a conceção e implementação de um sistema multiagente nesta área, o objetivo do grupo passou por criar um sistema multiagente de gestão de portfólio de criptomoedas.

2 Contextualização e Motivação

Nos últimos anos, o mercado de criptomoedas tem crescido exponencialmente, tornando-se uma das classes de ativos mais populares para investimentos. As criptomoedas ganharam destaque devido ao seu potencial de retorno elevado e à promessa de uma nova era de finanças descentralizadas. No entanto, este mercado é conhecido pela sua extrema volatilidade e dinamismo, o que apresenta desafios significativos para os investidores. A necessidade de informações rápidas e precisas para tomar decisões informadas é crucial, dado que, numa questão de minutos, podem ocorrer mudanças repentinas nas condições de mercado.

Dentro deste cenário, as *memecoins* surgiram como uma subcategoria notavelmente volátil e especulativa, destacando-se pela sua capacidade de capturar a imaginação e o entusiasmo das comunidades *online*. A popularidade das *memecoins* é amplamente alimentada por seu apelo cultural e pela influência das redes sociais, especialmente o Twitter/X, onde um simples *tweet* de uma figura influente pode desencadear movimentos significativos nos preços. As *memecoins*, portanto, representam um fenómeno único no mercado de criptomoedas, sendo bastante sensíveis às tendências e influências externas.

Neste contexto, a criação de um sistema que invista automaticamente pode oferecer uma solução viável para aproveitar oportunidades de lucro e gerir riscos de forma eficaz. Com movimentos de preço rápidos e imprevisíveis, este sistema pode reagir instantaneamente a *tweets* e tendências relevantes. Para além disso, a sua capacidade de análise rápida e ajuste conforme parâmetros definidos pelo utilizador oferece uma abordagem racional e adaptável diante das flutuações desse mercado altamente volátil e influenciado pelas redes sociais.

2.1 Objetivos

- **Maximização de retornos financeiros:** Maximizar os retornos de investimentos em ativos de um utilizador, garantindo a tomada de decisões inteligentes e oportunas que resultem em lucros substanciais;

- **Redução de riscos:** Identificar e mitigar, de forma eficaz, os riscos associados ao investimento - geralmente relacionados com a volatilidade inerente - e proteger o capital do cliente contra perdas significativas;
- **Economia de tempo e esforço:** Economizar o tempo e esforço do cliente, automatizando tarefas tediosas e complexas e permitindo a estes dedicar menos tempo à análise e tomada de decisões e mais tempo a outras atividades.

2.2 Abordagem

- **Distribuição de responsabilidades:** Implementação de uma arquitetura multiagente que permita a execução paralela de várias tarefas, distribuindo as responsabilidades por vários agentes, aumentando a eficiência e a escalabilidade do sistema;
- ***Scraping* de *influencers*:** Estabelecimento de um sistema para rastrear e analisar *tweets* de influenciadores-chave definidos pelo utilizador;
- **Execução automatizada de transações:** Execução automática de transações de compra e venda de ativos com base nos sinais gerados pela análise de tendências e seguindo os parâmetros de investimento definidos;
- **Personalização de parâmetros:** Criação de uma interface amigável e intuitiva que permita aos utilizadores configurar influenciadores, monitorizar o desempenho do portfólio e ajustar estratégias de investimento e definir parâmetros de risco, garantindo que o sistema atenda às suas necessidades individuais.

3 Sistema

3.1 Arquitetura

Relativamente à arquitetura proposta na figura 1, foram utilizados 5 diferentes tipos de agentes: Broker, Caller, Manager, Collector e Mapper. O agente Manager é a peça fundamental do sistema que possui todas as informações relativas ao portfólio de criptomoedas e histórico das transações efetuadas. Posteriormente, iremos abordar o papel de cada agente e o funcionamento do sistema como um todo.

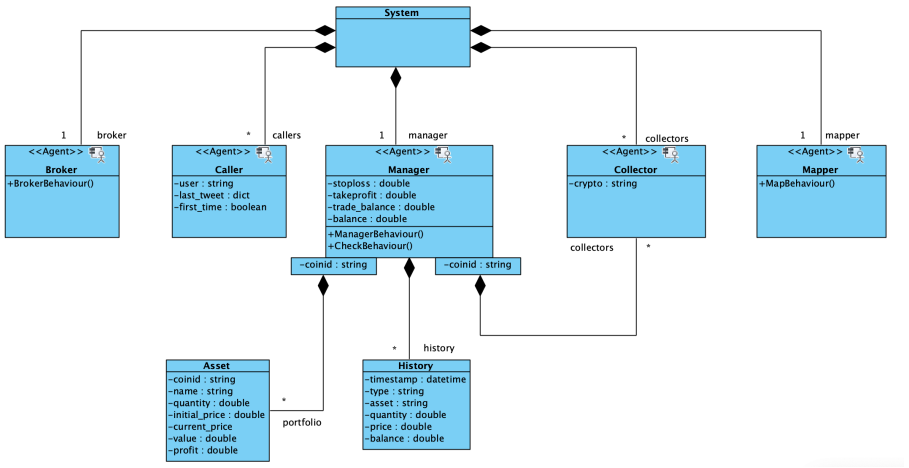


Figura 1: Diagrama de classes.

3.2 Funcionalidades

- Adição e remoção de *influencers*;
- Recolha de *tweets* dos *influencers* e extração de criptomoedas mencionadas;
- Recolha de *market data* através de uma dada API;
- Compra e venda de criptomoedas de acordo com a decisão do sistema;
- Configuração manual dos parâmetros *takeprofit*, *stoploss* e *trade balance*;
- Acesso ao portfólio que contém todas as criptomoedas possuídas pelo utilizador, bem como a quantidade, o preço atual, o valor total e a percentagem de lucro;
- Acesso ao histórico das decisões de compra e venda tomadas pelo sistema;
- Gestão de risco e maximização de lucros.

3.3 Dependências

Twitter *WebScraping* API

Para efetuar o *tracking* de cada *influencer* no X foi utilizada a biblioteca `twitter-api-client`. Esta biblioteca exige que a versão do Python seja igual ou superior à 3.10.10, o que leva a um problema de incompatibilidade entre versões do Python por conta do ambiente SPADE configurado, que utiliza uma versão anterior. Desta forma, não foi possível incorporar este comportamento num dado *behaviour*, tendo sido necessário o desenvolvimento de uma API Flask

para integrar esta funcionalidade.

CoinMarketCap API

Para recolher os dados das criptomoedas, foi utilizada a API do CoinMarketCap. Esta API é bastante completa, permitindo que se aceda a uma variedade de dados relacionados ao mercado de criptomoedas, como preços em tempo real, volumes de negociação, capitalização de mercado, histórico de preços, informações sobre *exchanges*, entre outros. O plano básico tem um limite de 30 pedidos por minuto e 10 mil por mês, limites os quais consideramos suficientes para o estado atual do sistema. No entanto, considerando a necessidade de escalabilidade da solução tanto no número de *influencers* a seguir como no número de ativos com investimento do utilizador, seria importante estender estes limites, por forma a preservar a utilidade do sistema.

3.4 Agentes

Nesta secção, exploramos os componentes fundamentais do sistema multiagente desenvolvido. Cada agente dentro deste ecossistema desempenha funções específicas que contribuem para a eficácia geral do sistema, desde a recolha de dados até à execução de transações financeiras. Os agentes são projetados para operar de forma coordenada, garantindo que as informações sejam precisas e as ações de mercado sejam executadas de forma oportuna e conforme as estratégias estabelecidas. Em seguida, são detalhadas as responsabilidades e o papel de cada agente.

Caller

Este agente é responsável pelo *tracking* de uma determinada conta do Twitter, verificando se há ou não novos *tweets* e, em caso afirmativo, recolhendo o *ticker* presente no *tweet*.

Mapper

Este agente possui a função de mapear o *ticker* recebido na CoinMarketCap API. Um *ticker* pode ser comum a várias criptomoedas, sendo necessário encontrar o identificador único da criptomoeda em questão, por forma a garantir que o sistema opera sempre sobre o mesmo ativo.

Collector

Este agente é responsável por recolher e atualizar periodicamente as informações relativas ao desempenho, liquidez e valor de uma dada criptomoeda, recorrendo à CoinMarketCap API.

Broker

Este agente atua como intermediário entre os investidores e o mercado financeiro, efetuando a compra ou venda de uma dada criptomoeda. Neste caso, é feita uma simulação destas ações mas considerando uma integração, por exemplo, com uma corretora, este agente seria responsável por interagir com a mesma para efetuar as transações.

Manager

Este agente desempenha um papel crucial no sistema, atuando como um coordenador central encarregue pela orquestração das interações e fluxos de trabalho entre os restantes agentes. A responsabilidade associada ao processo de tomada de decisão recai sobre este agente, sendo ele que ordena a compra ou venda de um determinado ativo, com base nas informações do portfólio que vão sendo atualizadas conforme a informação proveniente da comunicação com outros agentes. Neste sentido, este agente armazena os valores de *stoploss* e *takeprofit* definidos pelo utilizador, bem como o saldo global do portfólio, o saldo particular a cada ativo e o histórico de compras e vendas do sistema. Em suma, o Manager integra e sincroniza as ações dos agentes para alcançar objetivos comuns dentro do sistema.

3.5 Comportamentos

Nesta secção, detalhamos os comportamentos específicos de cada agente dentro do sistema. Cada comportamento é meticulosamente projetado para responder a estímulos específicos, garantindo que o sistema seja proativo e eficiente na captura de oportunidades de mercado e na gestão de riscos. A seguir, exploramos como cada comportamento contribui para o funcionamento geral do sistema.

CallerBehaviour

Este é um comportamento periódico invocado pelo agente Caller, responsável por recolher os *tweets* mais recentes de um dado *influencer*. Desta forma, este comportamento é executado a cada 20 segundos, onde é feita uma requisição para obter o *tweet* mais recente. Sempre que é detetado um novo *tweet*, é feito um processamento de linguagem para encontrar o *ticker* no *tweet*, que é posteriormente comunicado ao Manager caso esteja presente.

MapBehaviour

Este é um comportamento cíclico, o que implica a execução contínua à espera de mensagens. Ao receber uma `request` *performative*, é recolhido o identificador único da criptomoeda na CoinMarketCap API. Para isto, primeiramente são recolhidas todas as criptomoedas com esse *ticker* e, posteriormente, é selecionada a moeda com o maior volume de negociação nas últimas 24h.

Como foi explicado, um *ticker* pode estar associado a várias criptomoedas e o volume de negociação é um indicador importante para encontrar a criptomoeda em questão pois reflete a atividade e o interesse dos investidores nela. Quando um influenciador menciona um *ticker* específico e o volume de negociação associado a essa criptomoeda é alto, isto geralmente indica que há muitos compradores e vendedores ativos, ou seja, há um grande interesse na moeda naquele momento.

CollectBehaviour

Este comportamento é um *periodic behaviour* invocado pelo agente Collector e executado a cada minuto. Em cada ciclo, é realizado um acesso à CoinMarketCap API para recolher o identificador, o nome, o preço em dólares, o volume de negociação nas últimas 24h e a capitalização de mercado de uma determinada criptomoeda.

BrokerBehaviour

Este é o comportamento cíclico do agente Broker que, ao receber uma mensagem, compra ou vende um determinado ativo, de acordo com a *performative*. Tanto para a compra como para a venda, são comunicados os detalhes da transação como a quantidade de moeda comprada/vendida, o preço atual e o montante a deduzir ou acrescentar ao saldo.

CheckBehaviour

Este comportamento é um *periodic behaviour* invocado pelo Manager que é executado a cada 30 segundos. Durante cada execução, é feita uma verificação para cada moeda do portfólio. Se o lucro da moeda for maior ou igual ao limite estabelecido para realização de lucro (*takeprofit*) ou menor ou igual ao limite de perda aceitável (*stoploss*), ele instrui a venda do ativo. Este comportamento foca-se na gestão de risco e realização de lucros de acordo com critérios predefinidos.

ManagerBehaviour

Este comportamento é um *cyclic behaviour* que consiste na lógica principal do Manager, esperando pelo contacto de outros agentes. Ao receber uma mensagem, são despoletados comportamentos diferentes consoante a *performative*:

- **call_inform:** Redireciona a mensagem com a intenção de mapear a informação recebida;
- **collector_inform:** Atualiza o preço de um ativo no portfólio com base nas informações recebidas;
- **mapper_confirm:** Caso o ativo não esteja no portfólio, é iniciado o processo para adicionar o ativo ao portfólio, instruir a compra do ativo e registar um agente para recolher os dados;

- **mapper_failure:** Tratamento de erros na eventualidade de não haver um mapeamento do *ticker* para uma criptomoeda;
- **buy_confirm ou sell_confirm:** Atualiza o saldo global e o histórico de compras e vendas;
- **buy_failure ou sell_failure:** Tratamento de erros caso o Broker não consiga efetuar a ação de compra/venda de uma criptomoeda.

3.6 Comunicação

Após a especificação do papel dos agentes e do respetivo comportamento que adotam, é essencial explorarmos as interações que surgem entre eles. Desta forma, esta secção detalha a comunicação existente no sistema, isto é, os tipos de mensagens que são trocadas, imprescindíveis para coordenar ações e tomar decisões informadas. As *performatives* utilizadas nas várias mensagens trocadas entre os agentes são ilustradas pelo diagrama de colaboração representado na figura 2.

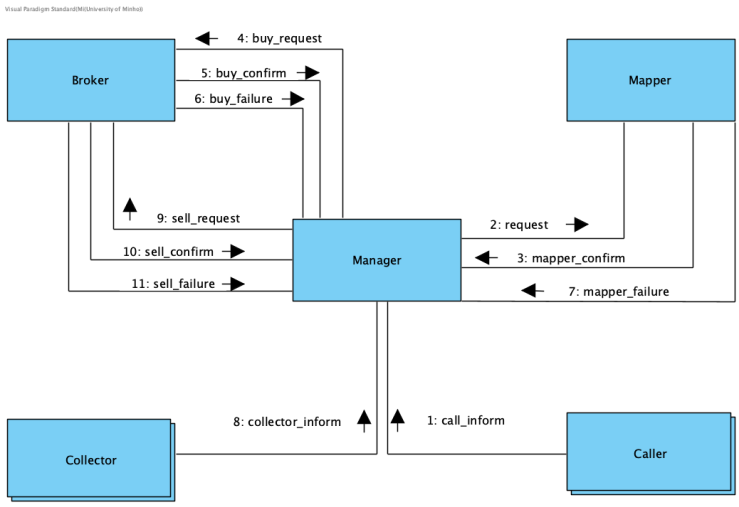


Figura 2: Diagrama de colaboração.

Todas as mensagens trocadas pelos agentes são constituídas por objetos devidamente codificados e descodificados aquando do envio e receção, respetivamente. As *performatives* utilizadas são, maioritariamente, pertencentes às normas da FIPA. No entanto, houve a necessidade de criar novas *performatives* para melhorar clareza e distinguir diferentes comportamentos nos agentes.

As mensagens que usam a *performative* 1: `call_inform` possuem como remetente o agente Caller e o destinatário é o agente Manager. São constituídas pelo objeto Call e o seu propósito é enviar a informação recolhida sobre uma criptomoeda ao detetar novos *tweets* dos *influencers* sobre esta.

Ao receber esta mensagem, o Manager envia o seu conteúdo ao agente Mapper com a *performative* 2: `request` para solicitar a verificação e atualização, caso necessário, do *coinid* e do nome da criptomoeda referente ao *ticker* extraído do *tweet*. O Mapper responde, então, ao Manager com o mesmo objeto Call que contém o id da criptomoeda que possui o *ticker* extraído do *tweet* e a *performative* utilizada é a 3: `mapper_confirm`. Caso a criptomoeda não seja encontrada ou caso ocorra algum erro, a *performative* de resposta correspondente é a 7: `mapper_failure`. Caso o Manager receba esta mensagem que indica um erro, não irá continuar com a sequência de ações que faria num caso normal.

Ao receber a resposta positiva do Mapper, o Manager envia um pedido de compra da criptomoeda ao agente Broker com o objeto Trade como o seu conteúdo e *performative* 4: `buy_request`. O conteúdo deste objeto corresponde ao id da criptomoeda e o valor na moeda corrente que pretende comprar. O Broker simula a compra e responde com a quantidade e o preço a que conseguiu realizar a transação. Caso a compra tenha sido realizada com sucesso, a *performative* é 5: `buy_confirm`. Caso contrário, responde com 6: `buy_failure`, indicando uma falha na ação de compra.

Caso o Manager receba a mensagem de erro ao efetuar a compra, ele remove a criptomoeda do seu portfólio. Caso receba a mensagem de sucesso, o agente inicia um agente Collector, atualiza o histórico e os dados como o valor, preço e quantidade referentes à criptomoeda comprada.

O agente Collector envia mensagens de 60 em 60 segundos ao Manager com a *performative* 8: `collector_inform` e o objeto Data como o seu conteúdo. Este objeto contém informações atualizadas sobre o desempenho, liquidez e o valor de uma dada criptomoeda. O Manager ao receber esta mensagem, utiliza a informação recebida para atualizar o seu portfólio.

Por fim, o agente Manager possui um comportamento periódico responsável por utilizar os parâmetros definidos nas configurações para avaliar a possibilidade de venda de uma criptomoeda. Caso opte por vender, envia uma mensagem ao Broker com a *performative* 9: `sell_request` e o objeto Trade, mencionado acima. O Broker, ao receber este pedido, simula a venda da criptomoeda e responde com o mesmo objeto como conteúdo da mensagem. Caso a venda tenha ocorrido com sucesso, a *performative* utilizada é a 10: `sell_confirm` e, ao receber esta resposta, o Manager atualiza o seu saldo, o seu portfólio e o histórico de decisões.

Detalhados os agentes, os seus comportamentos e a comunicação existente entre eles, elaboramos vários diagramas por forma a ilustrar e esclarecer alguns procedimentos mais críticos e complexos no nosso sistema. O primeiro diagrama, representado na figura 3, modela, através de um diagrama de sequência, o processo de compra de um ativo desde a *call* do agente responsável pelo *scraping* de um determinado *influencer* até à compra do mesmo.

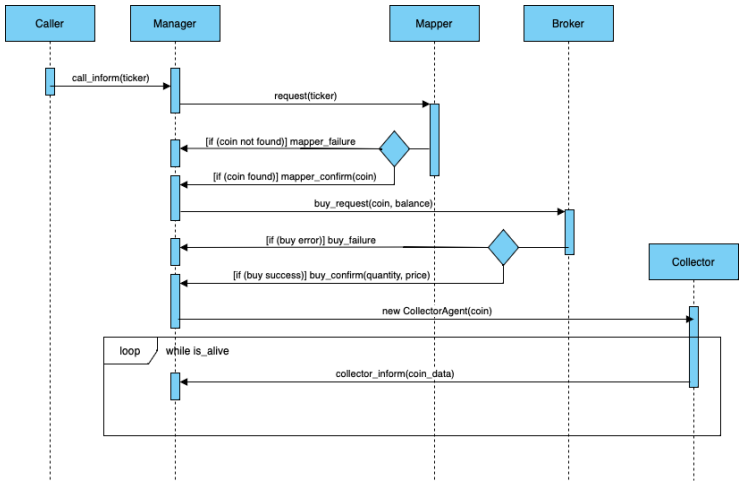


Figura 3: Diagrama de sequência.

O segundo diagrama, representado na figura 4, modela, através de um diagrama de atividades, o processo de tomada de decisão por parte do Manager e venda de um ativo.

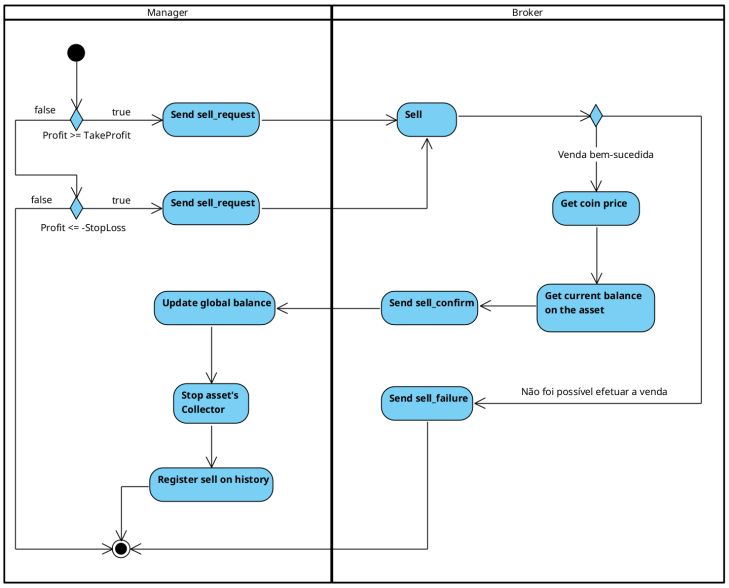
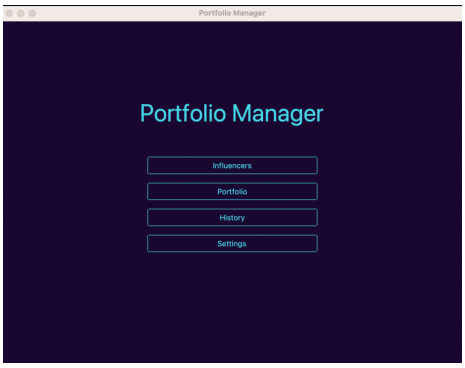


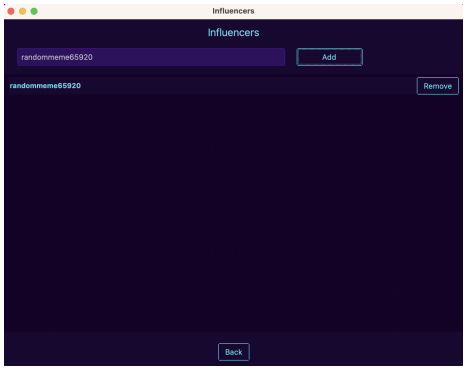
Figura 4: Diagrama de atividades.

4 Interface Gráfica

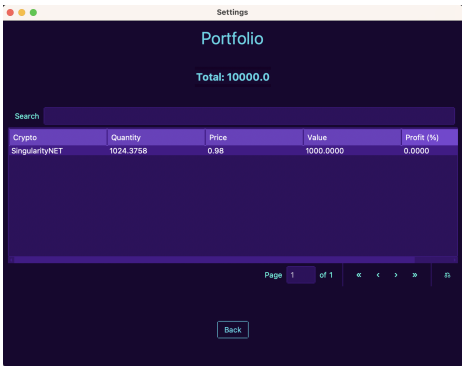
Para o desenvolvimento da interface gráfica foi utilizada a biblioteca `ttkbootstrap` que estende a biblioteca padrão `tkinter` para a criação de interfaces mais modernas. Esta interface é constituída por uma página inicial que funciona como um menu através do qual podemos prosseguir para diversas páginas com diferentes funcionalidades. Nestas páginas encontram-se uma dedicada à adição e remoção de influenciadores, uma para monitorizar o portfólio, outra para visualizar o histórico de compra e venda de ativos e uma última para definir as configurações associadas à gestão de risco.



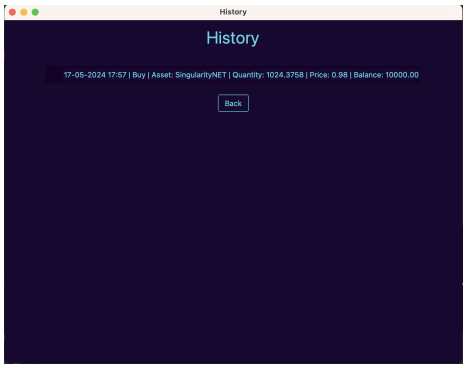
(a) Página inicial



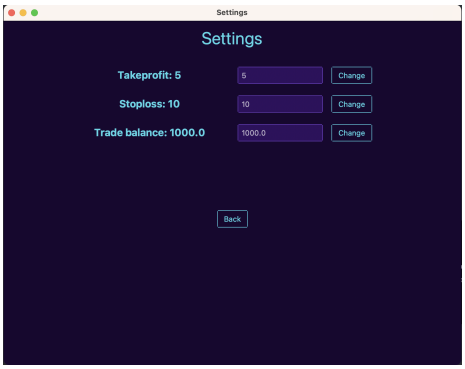
(b) *Influencers*.



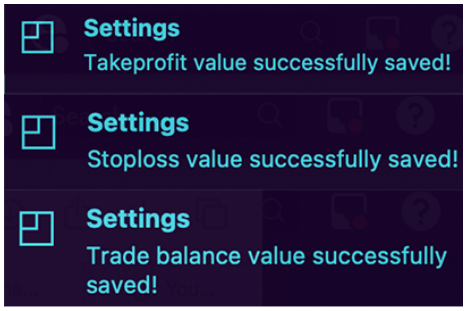
(c) Portfólio.



(d) Histórico de compras e vendas.



(e) Configurações.



(f) Notificações recebidas ao modificar as configurações.

5 Limitações e Trabalho Futuro

Com vista a obter uma aplicação completa para a posterior disponibilização no mercado, foram identificados alguns tópicos a serem implementados.

Por razões de ordem financeira e por questões de facilidade na demonstração, optou-se por simular a compra e venda de criptomoedas diretamente na aplicação utilizada neste projeto. É importante salientar que essa simulação não reflete a realidade do mercado, não envolvendo investimento efetivo por parte do utilizador. Para garantir transações reais e a gestão de uma carteira com fundos reais, recomenda-se a integração da aplicação com uma corretora ou diretamente com a *blockchain*. Este procedimento asseguraria uma experiência alinhada com os objetivos reais dos investidores.

A integração com uma base de dados assume um papel crucial na garantia da persistência dos dados, possibilitando ao utilizador retomar as suas atividades de investimento a partir do ponto em que as interrompeu e fornecendo acesso a estatísticas detalhadas relacionadas às decisões e transações efetuadas pelos agentes. A disponibilidade de estatísticas de desempenho da carteira é essencial para mostrar a evolução do valor ao longo do tempo, permitindo a análise de ganhos, perdas e o desempenho global em diferentes períodos.

A inclusão de execução contínua do SPADE em segundo plano é um aspeto a ser considerado para aprimorar a aplicação. Desta forma, a administração e o investimento passam a ocorrer de forma contínua, mesmo na ausência de interação direta do utilizador com a interface gráfica da aplicação, o que é essencial tendo em conta o objetivo de economia de tempo e esforço por parte do cliente. Isto garante também uma maior agilidade na resposta às mudanças e oportunidades do mercado, assegurando uma gestão dinâmica e pro-ativa do portfólio.

Relativamente ao *scraping* de *influencers*, seria pertinente adicionar um nível de processamento de linguagem aos *tweets*, tendo em conta que estes podem ter

uma conotação diferente à qual estamos a inferir de momento, que se trata de incentivar a compra da criptomoeda. Muitas vezes, as criptomoedas encontram-se nas tendências por motivos menos positivos e seria importante incluir a análise de sentimento no sistema, por forma a distinguir estas situações.

Por fim, a tomada de decisão pode ser melhorada ao combinar técnicas exploradas no trabalho de investigação como o *Deep Q-Learning*, que consiste numa combinação entre aprendizagem profunda e aprendizagem por reforço. Esta abordagem oferece uma estrutura robusta para a tomada de decisões autónomas, capacitando os agentes a aprender com a experiência e a otimizar suas estratégias de investimento de forma adaptativa. Assim, podemos esperar uma melhoria significativa na capacidade de tomar decisões que maximizem os retornos e minimizem os riscos para o utilizador, contribuindo assim para uma experiência de investimento mais sólida e confiável.

6 Conclusão

Este projeto propôs um sistema multiagente de gestão de portfólio de criptomoedas, utilizando tecnologias de agentes para automatizar e otimizar o processo de investimento neste mercado altamente volátil. O sistema foi projetado para maximizar retornos, reduzir riscos e economizar tempo e esforço dos investidores, através da implementação de uma arquitetura multiagente robusta e funcionalidades que permitem a análise e a execução automática de transações.

Refletindo sobre o trabalho elaborado, o grupo encontra-se satisfeito com a solução desenvolvida. Apesar das vantagens oferecidas pelo sistema, temos consciência de que este teria de sofrer alterações significativas, referidas neste relatório, para realmente possuir utilidade e tornar-se viável para uso generalizado. Ainda assim, ao nível do sistema multiagente, consideramos que os objetivos foram cumpridos, tendo sido implementado um sistema com uma arquitetura cuja qual consideramos bem definida, composta por vários agentes que comunicam entre si através de *performatives* que seguem as normas FIPA-ACL, sempre que possível. Para além disso, é importante realçar que são utilizados objetos nestas trocas de mensagens, assim como proposto no enunciado. Quanto aos agentes envolvidos no sistema, podemos contar com, pelo menos, 5 agentes, mas este número é escalável consoante o número de *influencers* e ativos, sendo que a criação de vários agentes é feita em *runtime*.