Rogério Gomes Lopes Moreira, A74634

Sobre

Trabalho realizado no âmbito da Unidade Curricular de Arquiteturas de Sistema de Software, perfil de Engenharia de Sistemas de Software do Mestrado integrado de Engenharia Informática, do Departamento de Informática da Universidade do Minho.

Introduction and Goals

Quality Goals

Pretende-se com este projeto o desenvolvimento de uma plataforma para a negociação de CFDs (Contract For Differences). Entre os principais objetivos estão:

- Criação de uma plataforma que permita a negociação de CFDs;
- Permitir a interação dos utilizadores com a plataforma;
- Fácil uso e navegabilidade.

Stakeholders

Role/Name	Contact
João M. Fernandes	*jmf@di.uminho.pt*
André Ferreira	*alferreira@di.uminho.pt*
João Luís Sobral	*jls@di.uminho.pt*

Architecture Constraints

Tendo em vista os objetivos acima descritos, e partindo dos requisitos solicitados pelo cliente, elaborou-se uma lista das principais funcionalidades a desenvolver e que serão vitais para o bom funcionamento da plataforma. Listam-se então as funcionalidades:

- Registo de traders/investidores na plataforma, tendo a si associados um plafond inicial com o qual podem negociar;
- Login de traders/investidores permitindo a negociação;
- Listagem de empresas com o valor atual de mercado, o valor de compra e o valor de venda;
- O preço atual terá que ser atualizado através de uma API para permitir preços em tempo real (aproximado).
- Cada utilizador tem um saldo associado à sua conta que é calculado a partir do montante disponível para investir, o valor atualmente investido e o lucro atual.
- O utilizador começa com um saldo de 10.000$ na conta, permitindo-o investir logo que se regista.
- Permitir a abertura de CFD de compra (Buy) – o investidor compra uma participação na empresa, tendo em vista o crescimento do ativo comprado. Pode definir ainda no valor Stop Loss, um Take Profit e o número de unidades a comprar.
- Permitir a abertura de CFD de venda – o investidor compra uma participação numa empresa, com o intuito de a vender, tendo em vista desvalorização do ativo. Pode ainda definir um Stop Loss, Take Profit e o número de unidades a comprar.
- Stop Loss é o máximo de perda que um investir. Take Profit é o máximo de ganho possível para determinado CFD. As ações deverão ser vendidas assim que o seu preço for mais baixo que o Stop Loss definido aquando da compra ou quando o preço for mais alto que o Take Profit definido também no momento da compra. Para isso terá que existir uma *task* em background que vá verificando e comparando os valores dos CFD com o preço atual.
- As ações são expressas em unidades inteiras.
- Terão que existir dois tipos de preço, um preço de Buy e um preço de Sell, consoante o tipo de CFD a comprar.
- No momento da compra terá que existir uma verificação se o símbolo, correspondente a uma empresa, inserido existe na API.

- Permitir ao utilizador ter uma *WatchList* uma lista de empresas, definidas pelo próprio, e onde poderá ver o *Sell Price* e o *Buy Price*.
- Permitir ao utilizador adicionar empresas à *WatchList*.
- Quando um utilizador compra um CFD, seja ele do tipo Sell ou do tipo Buy, a empresa é adicionada automaticamente à sua *WatchList*.
- Confirmação no momento em que se está a investir se o saldo da conta do utilizador é suficiente.

Business Context

Uma plataforma de negociação é uma aplicação que permite investidores e traders abrir, fechar e gerir posições no mercado financeiro, que podem envolver compra e venda de ativos financeiros, por exemplo ações, commodities (ouro, petróleo), índices ou moeda. As plataformas de negociação são frequentemente oferecidas por corretores gratuitamente ou com uma taxa aplicável a um número de posições mínimas por mês.

Em finanças, um contrato de diferenças (CFD – Contract For Differences) estabelece-se entre duas partes, normalmente referidas como "comprador (long) e "vendedor" (short), estipulando que o vendedor pagará ao comprador a diferença entre o valor atual de um ativo e o seu valor em tempo de fecho de contrato (se a diferença for negativa, então o comprador paga ao vendedor).

A título de exemplo as ações da Google são passíveis de ficarem associadas a um CFD de compra a um valor de $920.6 (valor efetivo mais a margem do corretor) ou um CFD de venda a um valor de $919.5 (valor efetivo), valores estes definidos pela dinâmica de compra e venda de ações no mercado bolsista. Quando um comprador adquire um CFD sobre 1 unidade de ações, está a investir $920.6 tendo a expectativa que o valor da ação suba para que possa vender a um valor superior no futuro, ganhando nesta diferença.

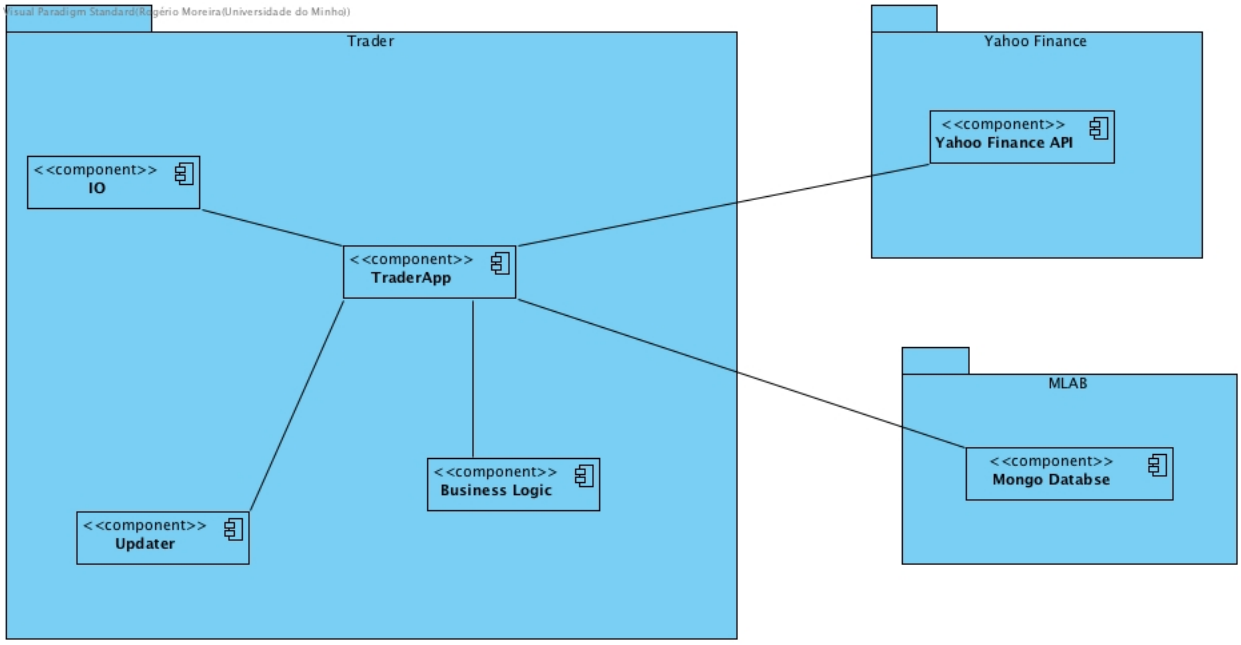
De forma inversa, uma opção de venda é usada por um investidor para ganhar na desvalorização de um ativo (ou perda caso este valorize). Através de um CFD o 'vendedor' aceita que vende um CFD a um valor de venda $919.5 e o comprará a um valor mais baixo no futuro, ganhando desta forma na diferença dos valores.

Technical Context

A solução encontrada terá que ser codificada numa linguagem de programação orientada a objetos, neste caso, Java.

Building Block View

Demonstra-se de seguida o funcionamento geral do programa.

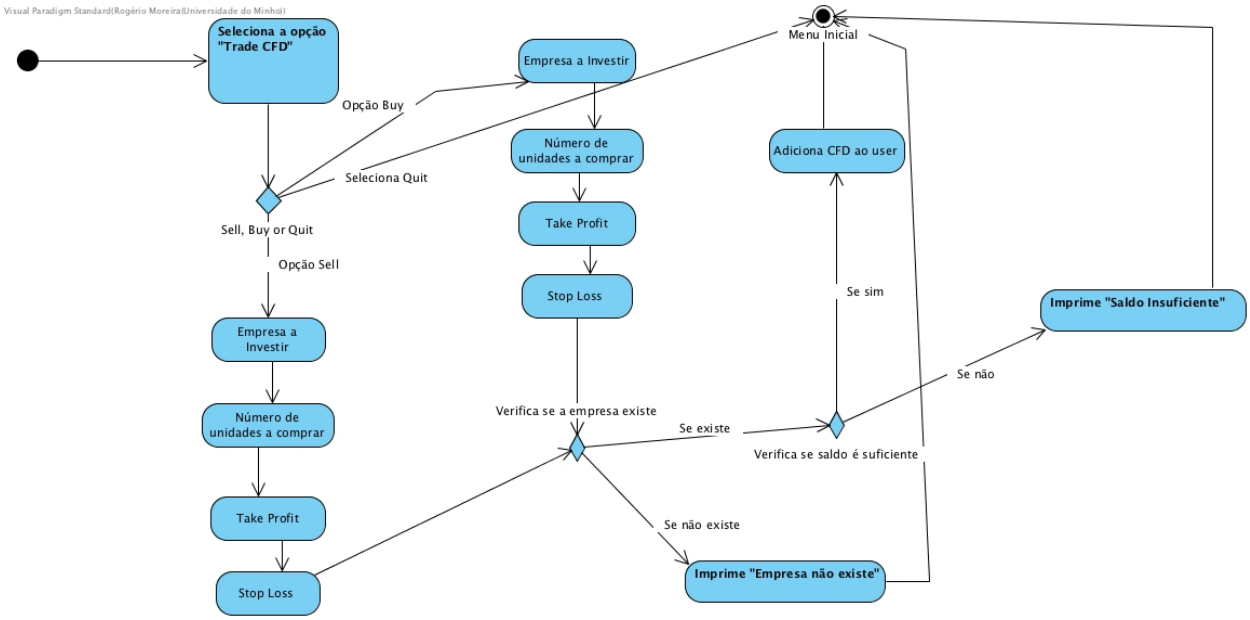


O Sistema é composto por dois módulos. Um módulo principal com o qual o utilizador interage, a interação sistema/utilizador e os tipos de dados específicos do programa em questão e um segundo módulo que faz a interação com a API do Yahoo Finance de onde são extraídos os dados do mercado. Para além disso, existe ainda a ligação à base de dados online em Mongo de onde são lidos e escritos os dados gerados pelo programa.

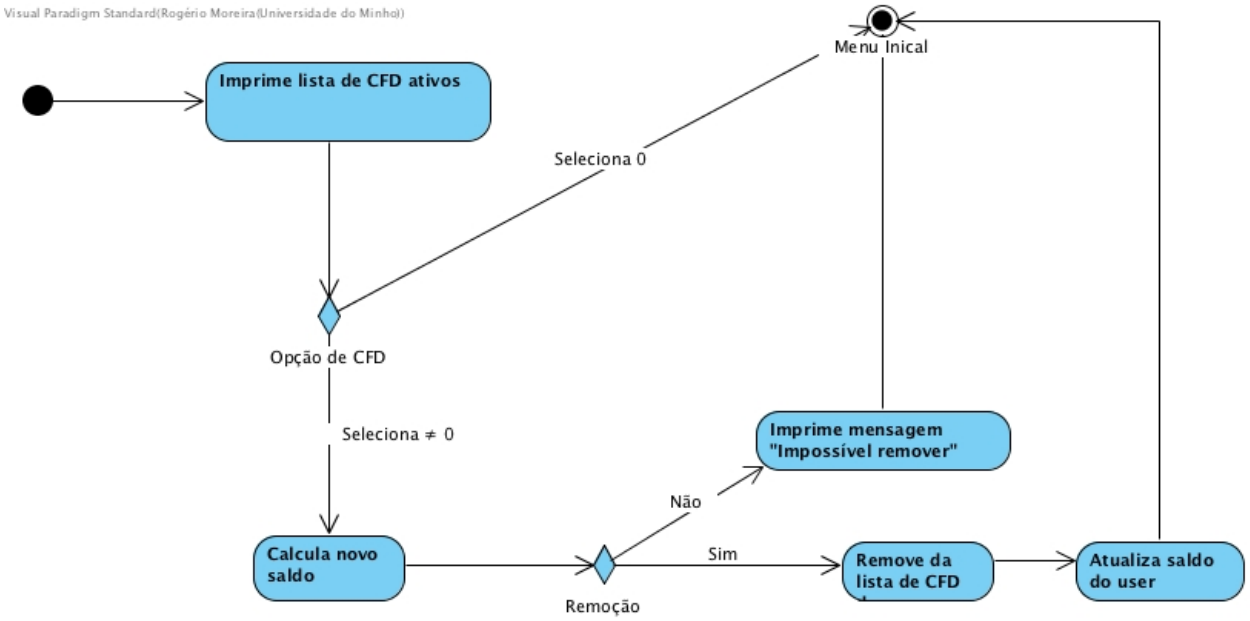
Whitebox Overall System

De seguida listam-se algumas das principais funcionalidades e os diagramas que as estruturam. Para cada funcionalidade explicita-se o seu diagrama de atividade.

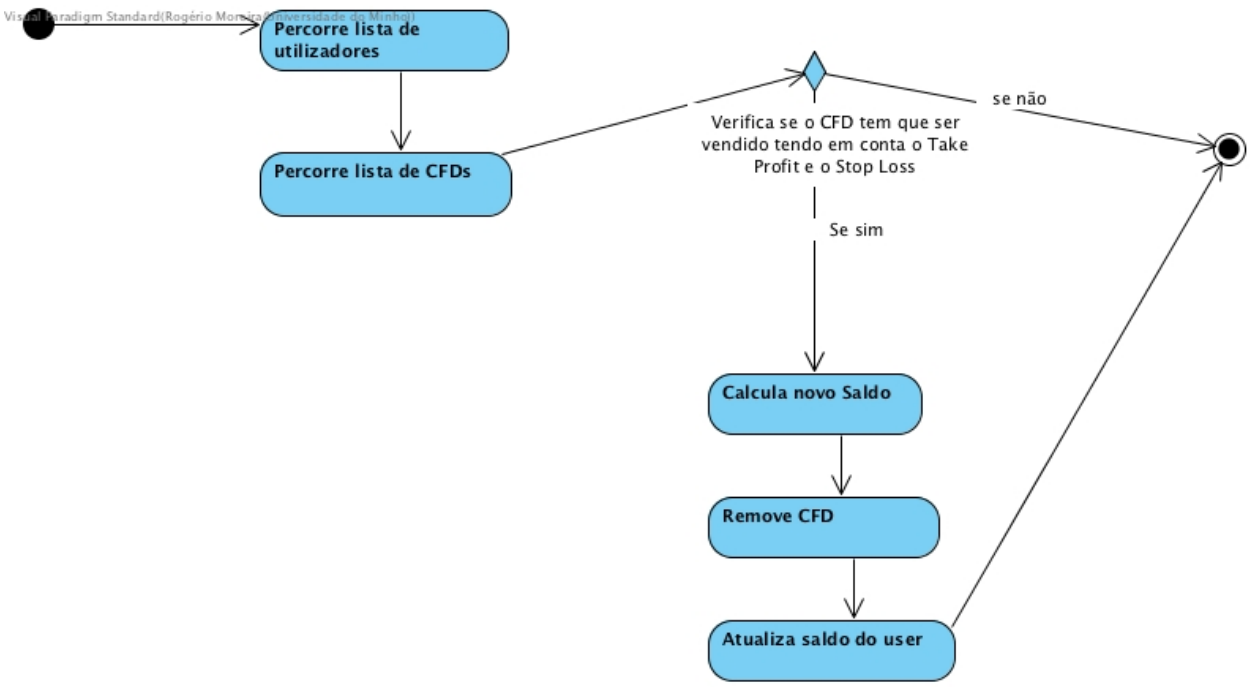
Trade



Close CFD



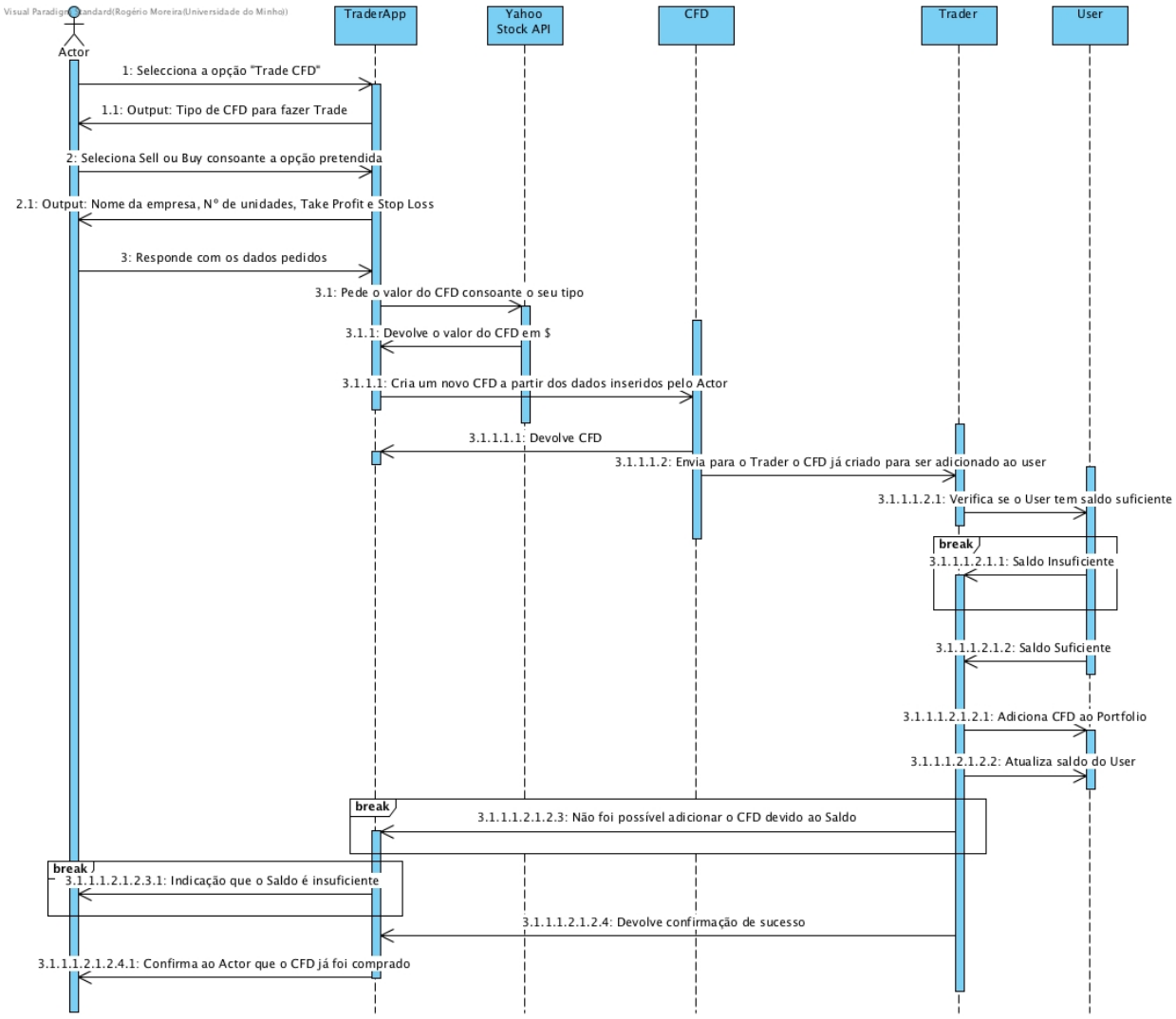
Update CFD


Percorre lista de utilizadores
Percorre lista de CFDs
Verifica se o CFD tem que ser vendido tendo em conta o Take Profit e o Stop Loss
se não
Se sim
Calcula novo Saldo
Remove CFD
Atualiza saldo do user

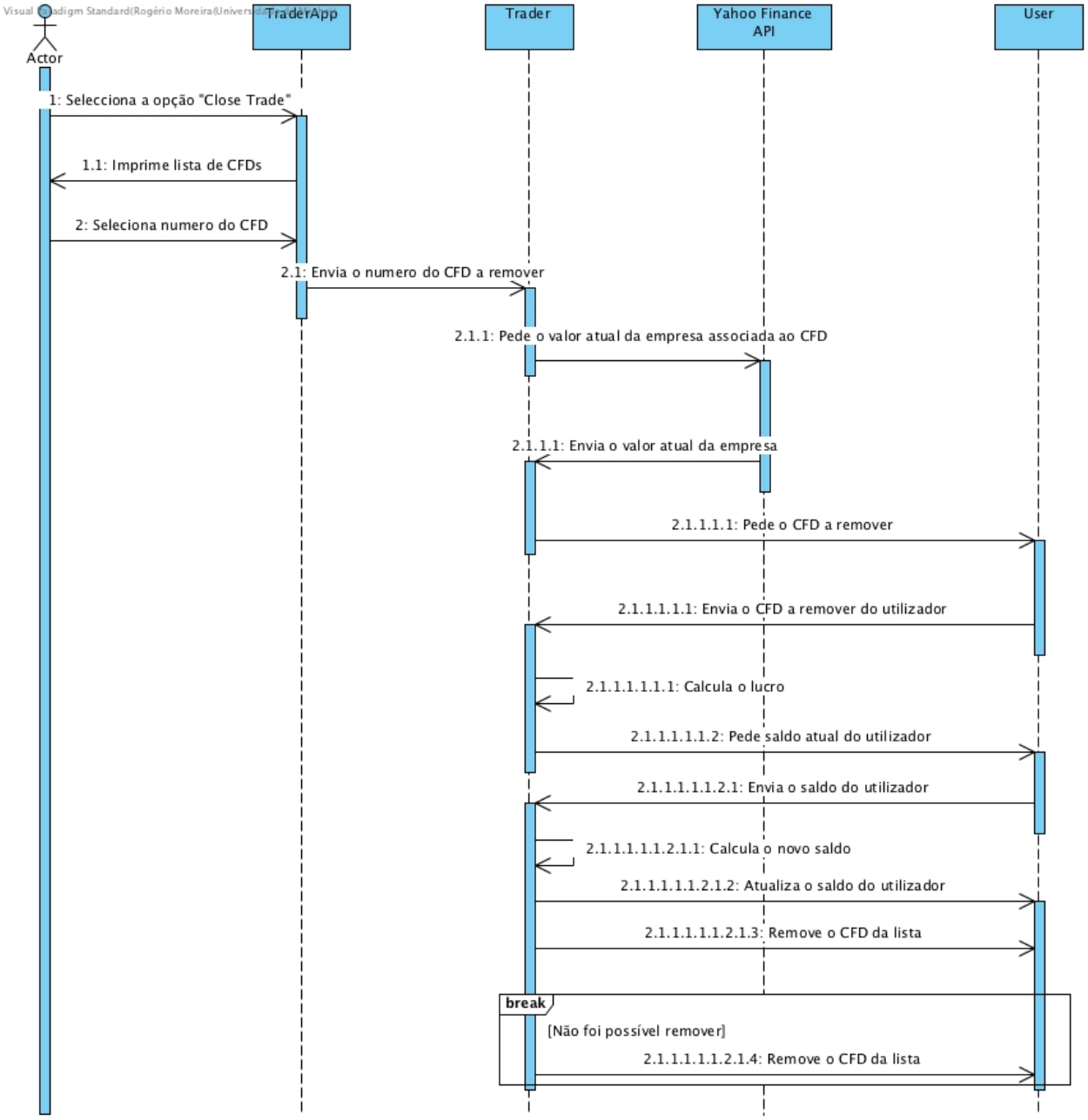
Runtime View

Em seguida apresentam-se os diagramas de sequência das principais funcionalidades e que ilustram o comportamento dentro do sistema para cada uma delas.

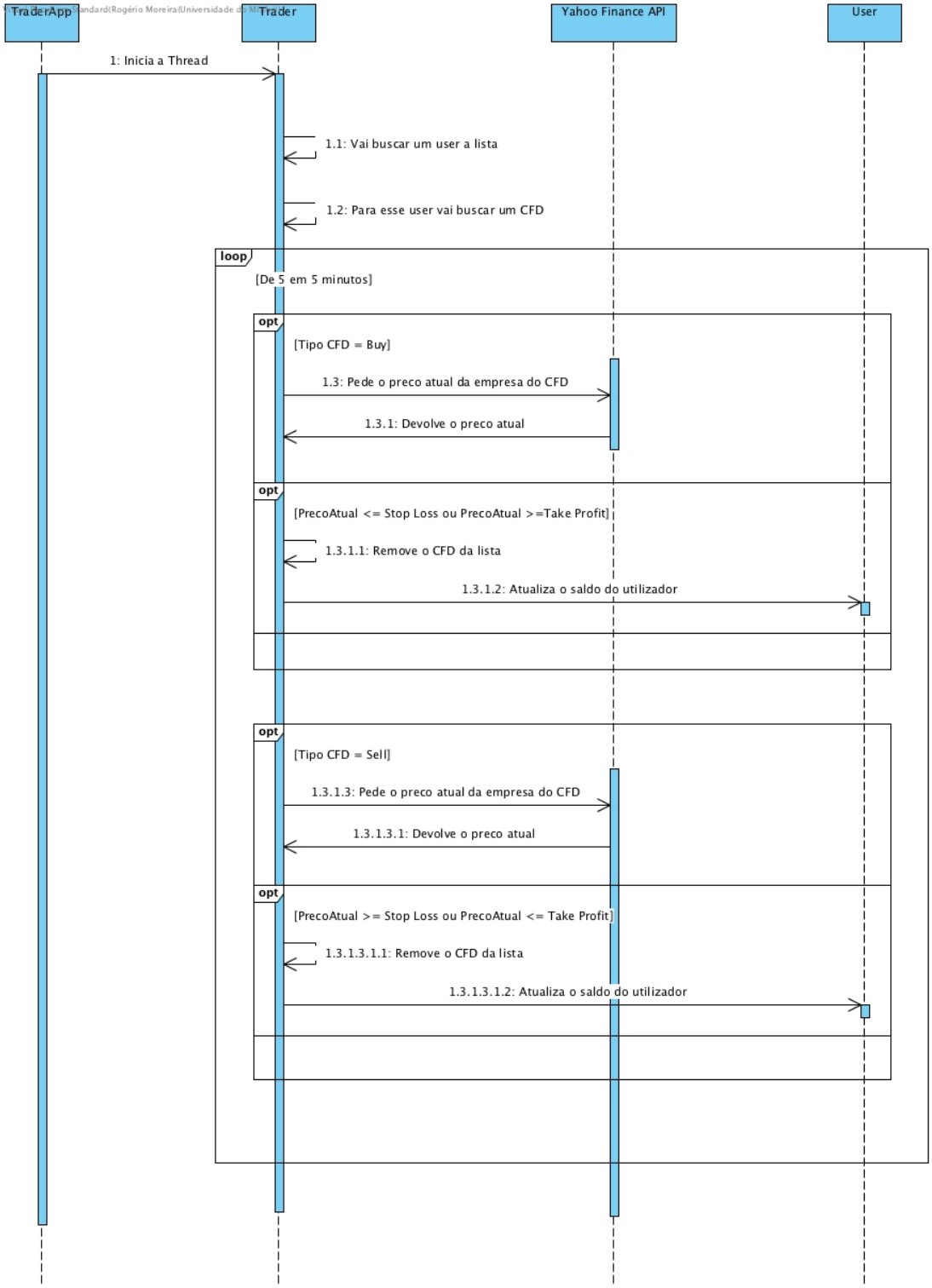
Trade



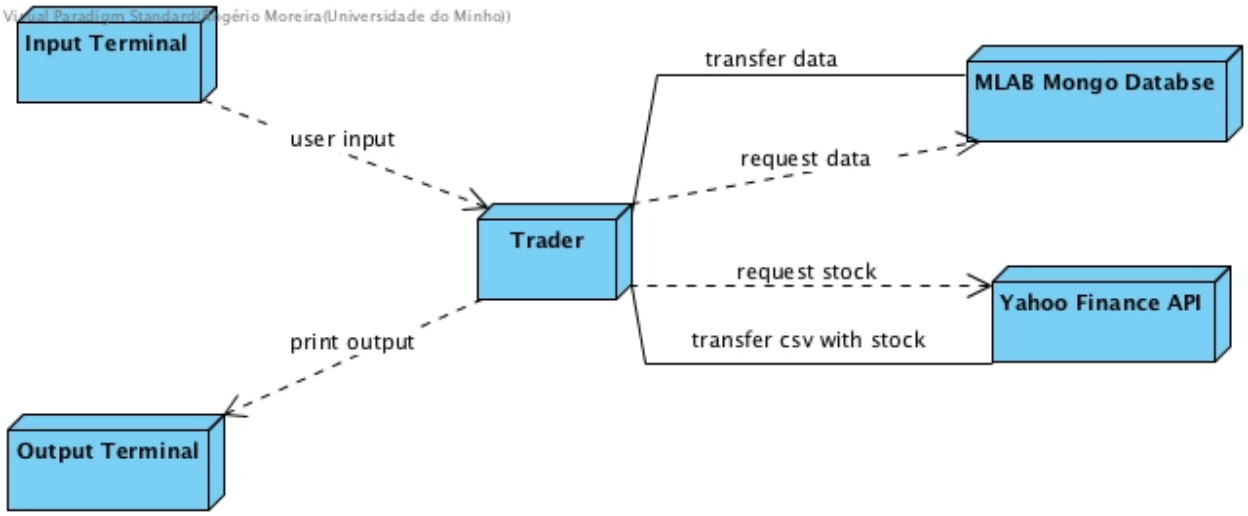
Close trade


Visual Paradigm Standard(Rogério Moreira(Universidade Lusíada))
Actor
TraderApp
Trader
Yahoo Finance API
User
1: Selecciona a opção "Close Trade"
1.1: Imprime lista de CFDs
2: Seleciona numero do CFD
2.1: Envia o numero do CFD a remover
2.1.1: Pede o valor atual da empresa associada ao CFD
2.1.1.1: Envia o valor atual da empresa
2.1.1.1.1: Pede o CFD a remover
2.1.1.1.1.1: Envia o CFD a remover do utilizador
2.1.1.1.1.1.1: Calcula o lucro
2.1.1.1.1.1.2: Pede saldo atual do utilizador
2.1.1.1.1.1.2.1: Envia o saldo do utilizador
2.1.1.1.1.1.2.1.1: Calcula o novo saldo
2.1.1.1.1.1.2.1.2: Atualiza o saldo do utilizador
2.1.1.1.1.1.2.1.3: Remove o CFD da lista
break
[Não foi possível remover]
2.1.1.1.1.1.2.1.4: Remove o CFD da lista

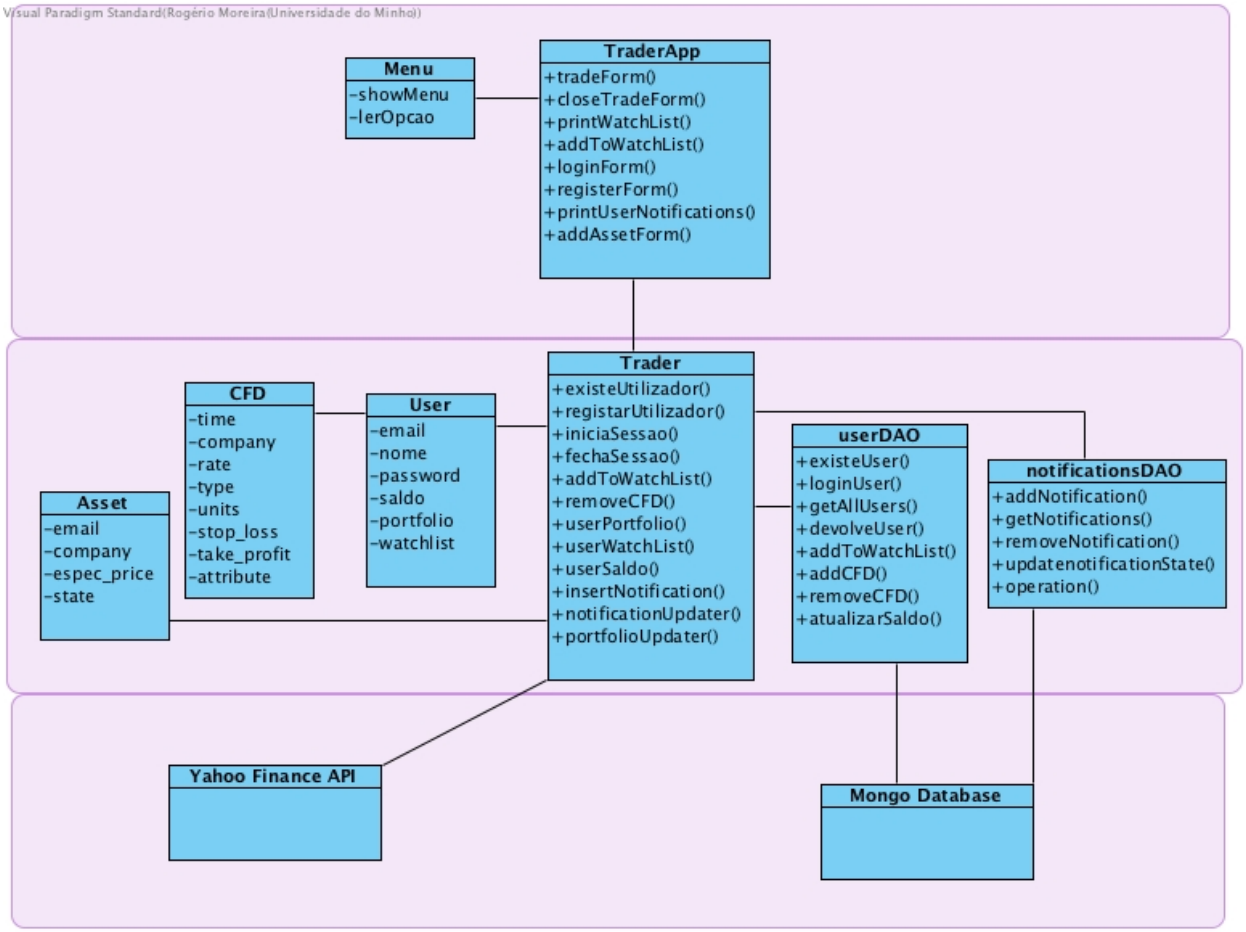
Update CFDs


TraderApp
Trader
Yahoo Finance API
User
1: Inicia a Thread
1.1: Vai buscar um user a lista
1.2: Para esse user vai buscar um CFD
loop
[De 5 em 5 minutos]
opt
[Tipo CFD = Buy]
1.3: Pede o preco atual da empresa do CFD
1.3.1: Devolve o preco atual
opt
[PrecoAtual <= Stop Loss ou PrecoAtual >=Take Profit]
1.3.1.1: Remove o CFD da lista
1.3.1.2: Atualiza o saldo do utilizador
opt
[Tipo CFD = Sell]
1.3.1.3: Pede o preco atual da empresa do CFD
1.3.1.3.1: Devolve o preco atual
opt
[PrecoAtual >= Stop Loss ou PrecoAtual <= Take Profit]
1.3.1.3.1.1: Remove o CFD da lista
1.3.1.3.1.2: Atualiza o saldo do utilizador

Deployment View



Design Decisions



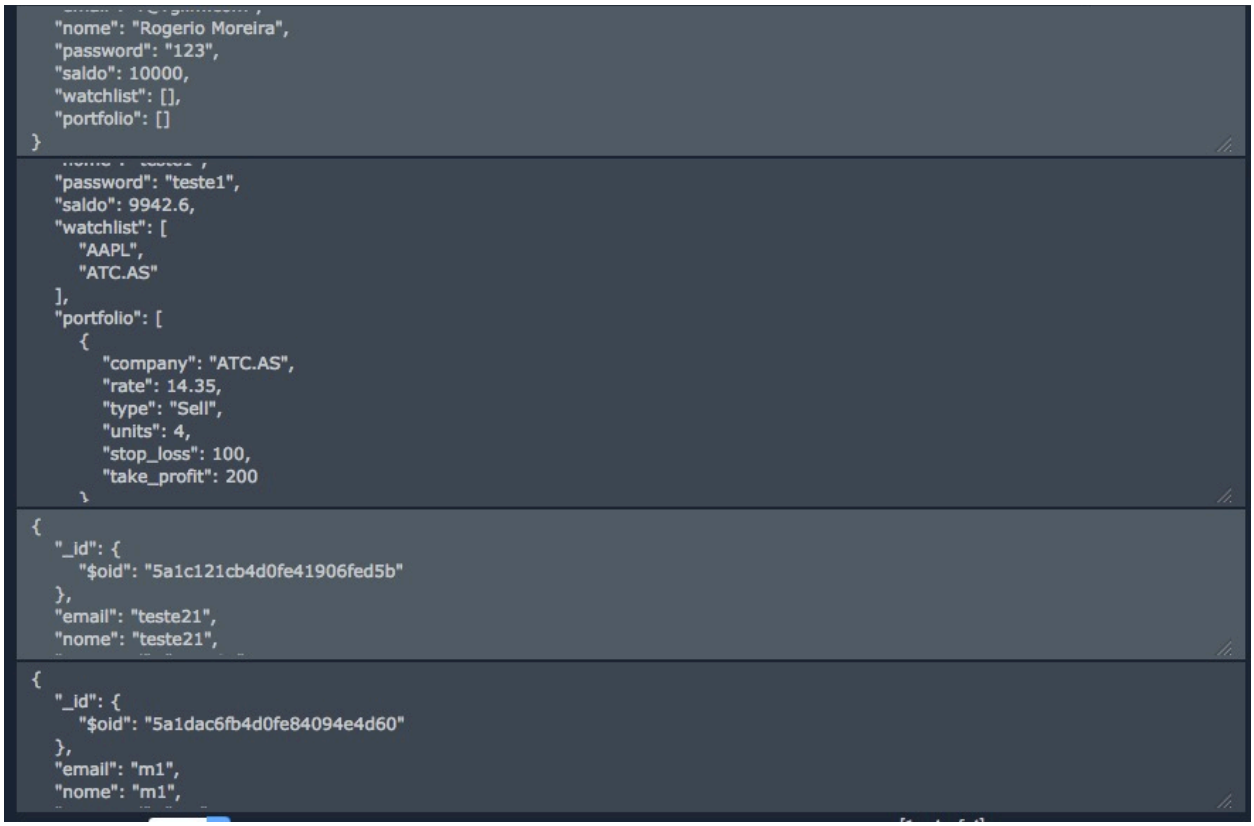
2nd Phase

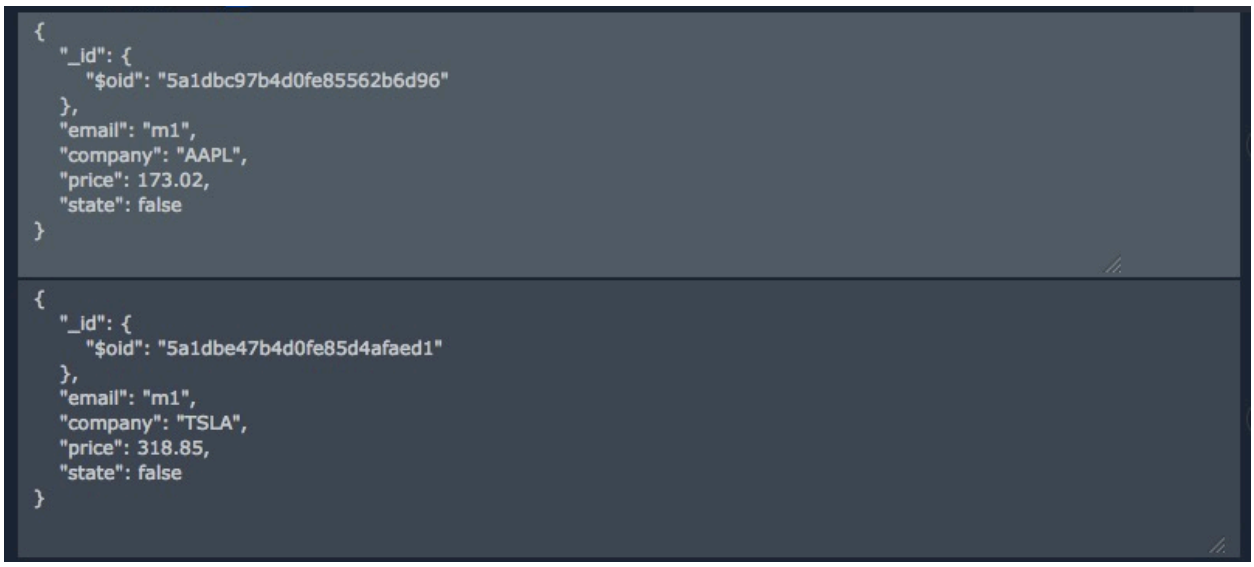
Na segunda fase do projeto foi pedido a implementação de “Design Patterns” no sistema previamente criado. De notar que, a arquitetura que o programa desenvolvido segue é Model-View-Controller, ou seja, temos uma parte de negócio do programa, uma parte de apresentação e uma base de dados para gravar todos os dados criados durante a sua execução.

Foi assim implementado o padrão Observer na arquitetura responsável por atualizar os CFDs, ou seja, agora um CFD tem um estado e logo que esse estado mude (o take profit ou o stop loss seja atingido) o CFD é vendido, atualizando de seguida o saldo do utilizador em questão.

Nesta fase foi também necessário atualizar a API da Yahoo Finance, uma vez que a anteriormente usada deixou de estar disponível. O novo módulo de API é mais robusto, já que não transfere as informações via CSV mas sim por uma ligação REST API ao Yahoo Finance, disponibilizando mais métodos para serem usados no futuro, como por exemplo o histórico de preços para determina empresa.

Para maior segurança, confiabilidade e manutenção do programa desenvolvido cheguei também à conclusão que precisaria de uma solução mais robusta para guardar os dados do que utilizando a serialização do Java. A solução anterior apesar de cumprir o prometido era vulnerável a mudanças do programa, já que a qualquer mudança os dados deixariam de ser possíveis de ser lidos. A melhor solução que encontrei foi uma base de dados No-SQL, neste caso MongoDB num serviço online. A solução foi implementada e todos os dados são lidos e escritos para a base de dados à medida que o utilizador interage com a aplicação. Para esta interação foram criados os DAOs utilizador e notifications, correspondendo às collections do MongoDB. Toda a interação entre a aplicação e a base de dados é feita através destas duas classes DAO. A collection users é responsável por guardar todos os dados de cada utilizador e a collection notifications é responsável por guardar as notificações e o seu estado.





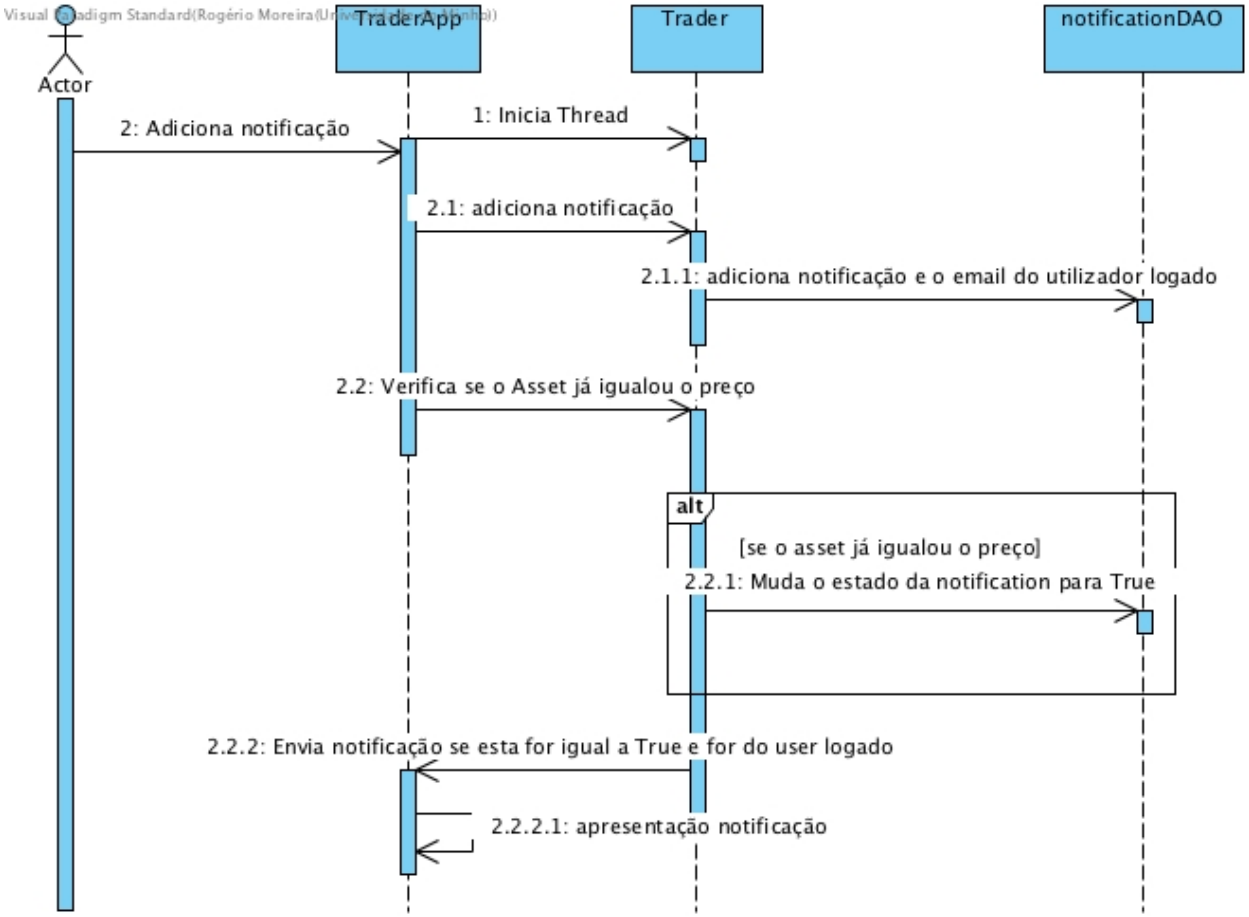
Notifications Requirement

Um dos requisitos desta etapa foi a implementação de uma nova funcionalidade ao sistema que permita aos "traders" seguir a cotação de determinados ativos, sendo notificados quando um ativo atinge um valor definido.

Para isto os módulos alterados foram:

- **Asset** – criado módulo que representa a notificação do utilizador. Este "Asset" representa o subject da Observer Pattern;
- **Trader** – adicionada a parte de negócio das notificações;
- **NotificationsDAO** – módulo que armazena as notificações criadas pelos utilizadores, para que seja possível apresentar as notificações mesmo que um utilizador termine a sessão no programa;
- **TraderApp** – módulo alterado para ser possível tanto apresentar as notificações como registar novas notificações.

Algumas funcionalidades do sistema foram reaproveitadas, tais como o módulo Trader que trata da parte de negócio das notificações e a ligação à base de dados. De seguida demonstra-se o funcionamento do novo recurso, recorrendo a um Diagrama de Sequência.



Glossary

Term	Definition
CFD	Contract for Differences
Trade	Investir em CFDs
Close trade	Vender os CFDs
Portfolio	Conjunto de CFDs de um utilizador
Asset	Preço da empresa do qual o utilizador quer ser notificado
DAO	Data Access Object