



ARQUITETURAS DE SOFTWARE

Ferramenta de Market Trading (Primeiro trabalho
prático)

Relatório

Descrição das diversas fases de desenvolvimento do projeto de software, seguindo um
template próprio para o efeito: arc42.

Conteúdo

Introduction and Goals	3
Requirements Overview	4
Quality Goals	8
Stakeholders	11
Architecture Constraints.....	12
Restrições Técnicas	12
Restrições de Organização.....	13
System Scope and Context.....	14
Business Context	14
Technical Context.....	15
Solution Strategy	17
Building Block View	18
Whitebox Overall System.....	19
AppESS (blackbox).....	20
YahooFinance (blackbox).....	20
MySQL DB (blackbox)	20
Building Blocks	21
AppESS (whitebox).....	21
YahooFinance (whitebox).....	21
Runtime View	22
Consulta do Valor de Ativo	23
Ver Portfolio	24
Adicionar ativo ao Portfolio	25
Abrir Venda	26
Deployment View	27
Cross-cutting Concepts	29
Modelo de Domínio	29
Persistência	41
Interface Gráfica	41
Sessões	41
Segurança.....	41
Proteção	41
Comunicação.....	41

Validade dos Dados	41
Tratamento de Exceções e Erros.....	42
Internacionalização	42
Migração	42
Testabilidade	42
Build-Managment.....	42
Design Decisions	43
Uso do Sistema de Gestão de Base de Dados MySQL.....	43
Uso do JSwing para o desenvolvimento da Interface Gráfica	43
Uso da API Yahoo Finance para os valores reais dos ativos.....	44
Quality Requirements	45
Árvore de Qualidade	48
Cenários de Qualidade	48
Risks and Technical Debts.....	49
Glossary	50
Apêndices.....	51
Mockups iniciais	51
Menu Inicial	51

Introduction and Goals

Durante o desenvolvimento de software, qualquer que seja o seu tipo, surge a necessidade de manter um equilíbrio e um diálogo entre ambas as partes, cliente e fornecedor. No entanto, não são raras as ocasiões, onde o cliente tem de aceitar uma opinião externa de alguém com experiência no ramo, sacrificando algumas das suas ideias pessoais. Não obstante, o fornecedor (engenheiro) vê-se muitas das vezes obrigado a corresponder, a determinados pedidos feitos ou até mesmo exigidos por clientes, não tendo qualquer tipo de voto na matéria, mesmo que na sua maneira de ver, a forma mais eficaz de atacar o problema não seja exatamente essa, i.e., a pedida pelo cliente.

Independentemente das divergências que possam surgir – e irão inevitavelmente surgir – é essencial que as pessoas envolvidas no projeto, levem as mesmas num espírito construtivo, visto que todos estão a trabalhar para um objetivo comum: criar um software funcional, útil, agradável e que satisfaça as necessidades para as quais foi criado.

No que diz respeito ao problema em mãos, fica claro que o objetivo principal passa por concretizar as ideias acima referidas, de modo a que no final de exaustivas horas de reuniões e de desenvolvimento, esteja preparado um software de plataforma de negociação.

Este tipo de plataformas permite aos investidores interessados gerir de forma mais cómoda, rápida e ágil, os seus investimentos num universo variado de mercados. Escusado será dizer, que todas as funções básicas, e até mesmo legais, deste tipo de plataformas vão estar presentes, de maneira a que seja possível categorizar o produto como uma plataforma de negociação.

O que irá diferenciar o produto de outros que já existem no mercado, são os pequenos toques pessoais que iremos incluir, juntamente com alguma correção a determinado mecanismo que não é do nosso agrado quando analisamos outras plataformas deste género. Finalmente, as exigências pessoais do cliente ESS Ltd, irão conferir, naturalmente, a este produto uma identidade própria.

A inclusão das funcionalidades base deste tipo de software aliadas aos toques pessoais desenvolvidos pela equipa juntamente com o cliente, irão permitir que ESS Ltd se estabeleça como o verdadeiro standard no mercado das aplicações de trading. A nossa ambição pessoal é tornar a empresa ESS Ltd, na escolha número 1 dos utilizadores na hora de decidir, ou sugerir uma aplicação de trading.

Requirements Overview

“Um requisito é uma capacidade que o sistema deve ter, de maneira a dar resposta a uma necessidade do utilizador.”

A descrição dos requisitos que o sistema vai apresentar, servirá como ferramenta para classificar o produto, inserindo o mesmo numa determinada categoria.

Concretamente, ao falarmos de uma plataforma de trading, imediatamente pensamos em opções de compra e venda de ativos, não pensamos, por exemplo, numa opção que mantenha o registo de batimentos cardíacos do utilizador.

A descrição textual de requisitos, não só ajuda imenso, quando chegar à altura de programar, mas permite também perceber se cliente e fornecedor estão na mesma página, no que diz respeito ao que é esperado do sistema.

Tendo em conta o contexto do problema e após uma análise cuidada, achamos indispensável a presença dos seguintes requisitos:

ID do Requisito	Nome	Personagens	Descrição
ReqF1	Login/Logout	Utilizadores Anónimos	Utilizadores Anónimos: Devem ser capazes de efetuar a autenticação na página desenvolvida para tal.
ReqF2	Registo	Utilizadores Anónimos	Utilizadores Anónimos: Devem ser capazes de efetuar registo no sistema, recebendo um plafond inicial para negociarem ou investirem.
ReqF3	Consulta do Valor de Ativos	Traders Investidores	Traders e Investidores: Os traders e investidores devem ser capazes de consultar os valores dos ativos que a qualquer momento, quer sejam do tipo ação (Google, Apple...), quer sejam do tipo commodities (Ouro, Prata...).
ReqF4	Abrir Compra	Traders Investidores	Traders e Investidores: Devem ser capazes de abrir uma ação de compra de um determinado ativo.
ReqF5	Abrir Venda	Traders Investidores	Traders e Investidores: Devem ser capazes de abrir uma ação de venda de um determinado ativo.
ReqF6	Ver Portfolio	Traders Investidores	Traders e Investidores: Devem ser capazes de consultar o seu portfolio a qualquer momento em “tempo real”. Do portfolio, devem constar os ativos que estão a ser negociados naquele instante, por aquele trader ou investidor. Deve ser possível a consulta das perdas ou ganhos em cada ativo que conste do portfolio.

Tabela 1 Requisitos Funcionais

ID do Requisito	Nome	Personagens	Descrição
ReqF7	Fechar Compra	Traders Investidores	Traders e Investidores: Devem ser capazes de, a qualquer momento, fechar a opção de compra do ativo “X”.
ReqF8	Fechar Venda	Traders Investidores	Traders e Investidores: Devem ser capazes de, a qualquer momento, fechar a opção de venda do ativo “X”.
ReqF9	Adicionar ao Portfólio	Traders Investidores	Traders e Investidores: Devem ser capazes de adicionar ao seu portfolio, a informação relativa a um ativo “X” – a qualquer momento - permitindo monitorizá-lo separadamente dos outros.
ReqF10	Remover do Portfólio	Traders Investidores	Traders e Investidores: Devem ser capazes de remover do seu portofólio, a informação relativa a um ativo “X” – a qualquer momento – assim que o entenderem.
ReqF11	Adicionar à Watchlist	Traders Investidores	Traders e Investidores: Devem ser capazes de adicionar um novo ativo à sua watchlist, para além daqueles já pré-definidos.
ReqF12	Remover da Watchlist	Traders Investidores	Traders e Investidores: Devem ser capazes de remover um ativo da sua watchlist.
ReqF13	Adicionar Fundos	Traders Investidores	Traders e Investidores: Devem ser capazes de adicionar tantos fundos quanto entenderem ao seu Plafond.

ReqF11	Menu Visitante	Utilizadores não autenticados	Utilizadores não autenticados: Devem ser capazes de abrir um Menu, que apenas lhes permite ver a tabela com os ativos e os respetivos valores de mercado, não estando qualquer outra operação disponível neste modo.
ReqF12	Alerta de Ativos	Traders Investidores	Traders e Investidores: Devem ser capazes de definir alertas para cada ativo, recebendo deste modo uma notificação textual, indicativa de que o ativo ultrapassou o valor que definiram.

Tabela 2 Requisitos Funcionais

Quality Goals

Atributos de Qualidade, por vezes também referidos como requisitos não funcionais, podem ser definidos como sendo uma especificação das características que preferencialmente, o sistema deve possuir, sem que no entanto, comprometam nenhuma das capacidades do sistema, se por algum motive forem excluídas, ou não implementadas no produto final.

Consistem sobretudo em medidas de performance, ou até mesmo toques pessoais que o cliente pretende dar ao produto, de modo a torná-lo mais apelativo em termos estéticos.

Tendo em conta o contexto onde nos encontramos, ESS Ltd entendeu como sendo essencial garantir os seguintes atributos de qualidade:

ID do Requisito	Nome	Personagens	Descrição
ReqNF1	Limite de Perda	Traders Investidores	Traders e Investidores: Devem ser capazes de definir um limite de perda máxima, a partir do qual o sistema deve fechar a ação de venda do ativo "X", se o valor de perda estimado atingir o valor definido pelo utilizador.
ReqNF2	Limite de Lucro	Traders Investidores	Traders e Investidores: Devem ser capazes de definir um limite de lucro máximo, a partir do qual o sistema deve fechar a ação de venda do ativo "X", se o valor de lucro estimado atingir o valor definido pelo utilizador.

Tabela 3 Requisitos N/ Funcionais

No que diz respeito às ambições extra de quem desenvolveu este projeto, é expectável e desejável que o mesmo apresente as seguintes qualidades depois de operacional:

ID	Nome	Descrição
1	Eficiência	O programa deve ser, rápido e ao mesmo tempo preciso e exato, ao realizar as operações que oferece ao utilizador. Adicionar ou remover ativos do portfolio devem ser operações triviais. A abertura e fecho das compras/vendas de ativos devem ser operações rápidas, e por operarem valores sensíveis que influenciarão os investimentos dos utilizadores na forma de ganhos ou perdas desses investimentos, devem ser operações que manipulam de maneira correta e sem falhas os valores com os quais vão trabalhar.
2	Atratividade	O programa deve ser atrativo o suficiente em termos de interface gráfica, para cativar o utilizador imediatamente. Todos os botões devem estar devidamente legendados, para que se perceba imediatamente o seu propósito. Deve haver uma certa preocupação estética com as componentes que constituem as janelas dos diversos menus, nomeadamente, o seu tamanho, que deve ser em todos os casos adequado, consoante o número de operações associadas aquela janela, as cores, ou imagens de fundo, que devem ser chamativas, mas sem interferir com a legibilidade do texto ou dos botões do menu e o próprio texto que deve ter um tamanho de letra adequado e cor que contraste com aquelas usadas para decoração do fundo de modo a ser o mais legível possível. A apresentação gráfica dos resultados das operações, por exemplo de inserção ou remoção de um ativo do portfolio do utilizador, deve ter resultados estéticos significativos e que agradem ao utilizador, nomeadamente um menu próprio para a apresentação destes resultados que cumpra com os requisitos enumerados acima no que toca às cores e texto envolvidos.
3	Compreensibilidade	A arquitetura e requisitos usados no projeto devem fáceis o suficiente de entender e aplicar para que seja implementada uma solução eficiente e que faça uma boa gestão dos recursos disponíveis em JAVA.
4	Interoperabilidade	A aplicação deve ser capaz de trabalhar com APIs externas (sem qualquer tipo de impedimento), nomeadamente aquela usada para a recolha dos valores de mercado dos ativos e a própria base de dados, de maneira a que não surjam problemas ou exceções com isso.

Tabela 4 Atributos de Qualidade

De modo a traduzir de uma forma um pouco mais simpática os requisitos acima descritos passo a apresentar uma representação gráfica do diagrama Use Case, que traduz essencialmente o que foi sendo descrito por palavras, no que às funcionalidades da aplicação e a quem se destinam diz respeito:

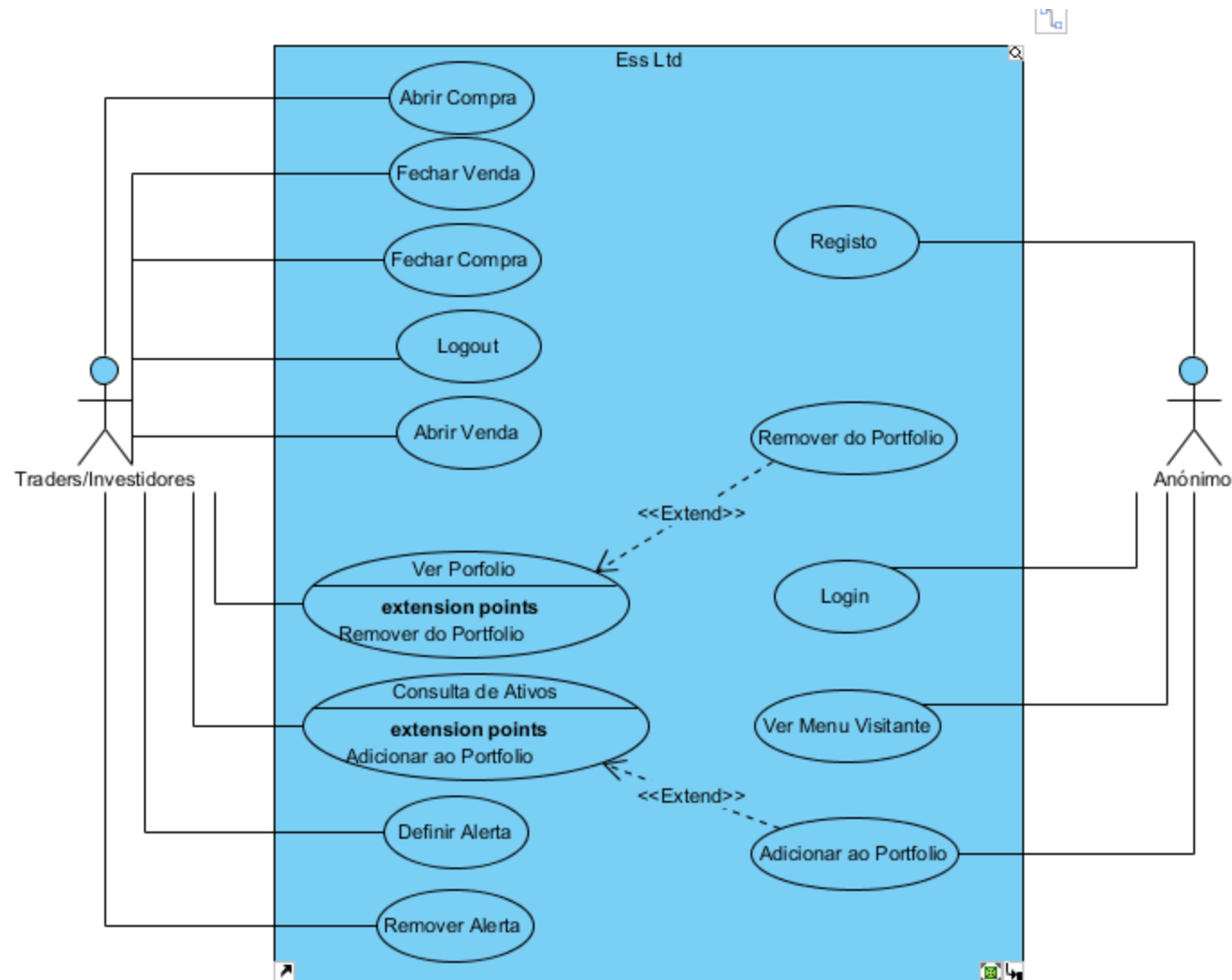


Figura 1 Diagrama de Use Case

Stakeholders

De seguida passamos a apresentar as pessoas mais importantes no contexto deste projeto, assim como os seus papeis:

Papel/Nome	Contacto	Expectativas
<i>João Nuno Gomes Rodrigues de Almeida</i>	Joaonuno.almeida01@gmail.com	<ul style="list-style-type: none">- Deve conhecer a arquitetura.- Deve tomar decisões acerca do desenvolvimento do sistema e da arquitetura.
<i>ESS Ltd</i>	essltd@toocomplicated.com	<ul style="list-style-type: none">- Tem de estar convencido com a arquitetura.- Tem de trabalhar com a arquitetura e com o sistema.- Necessita da documentação para trabalhar com a arquitetura/sistema.
<i>Utilizadores</i>	Diversos	<ul style="list-style-type: none">- Deve poder usufruir de todos as funcionalidades oferecidas pela aplicação, com confiança e satisfação.

Tabela 5 Stakeholders

Architecture Constraints

De seguida apresentamos as restrições associadas ao projeto, e a uma pequena explicação do porquê das ter incluído.

Restrições Técnicas

ID da Restrição	Nome	Descrição
ResT1	Restrição de SO	O sistema deve ser desenvolvido de modo a funcionar de maneira equivalente nos três principais sistemas operativos – Windows, MacOS e Linux.

Tabela 6 Restrições Técnicas

Restrições de Organização

ID da Restrição	Nome	Descrição
Res01	Equipa	João Nuno Gomes Rodrigues de Almeida
Res02	Prazo de entrega	22 de outubro de 2017
Res03	Open Source	O sistema será oferecido numa modalidade Open Source.

Tabela 7 Restrições de Orgazinação

System Scope and Context

De seguida passamos a apresentar os ambientes onde o projeto ESS Ltd se encontra, assim como as eventuais dependências externas do mesmo.

Business Context

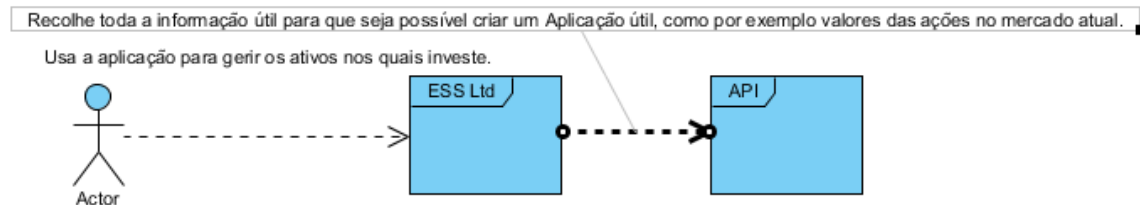


Figura 2 Diagrama UML: Business Context

Traders/Investidores – Utilizadores que pretendem gerir os ativos onde têm dinheiro investido. A aplicação permite que estes invistam em novos ativos, fechem o investimento em ativos, onde abriam um investimento previamente. Permite criar um conceito de Portfolio, de modo a guardar os ativos mais importantes para o Trader/Investidor, e que este pretende monitorizar mais de perto.

ESS Ltd – Gere os ativos de um utilizador registado. Permite a compra e venda de ações, substituindo o trabalho de um corretor. Permite criar um Portfolio com as preferências do utilizador em termos de ativos, permitindo que o mesmo esteja mais atento às flutuações de mercado dos mesmos.

API Externa – Permite que sejam retirados valores reais dos ativos no mercado atual.

Technical Context

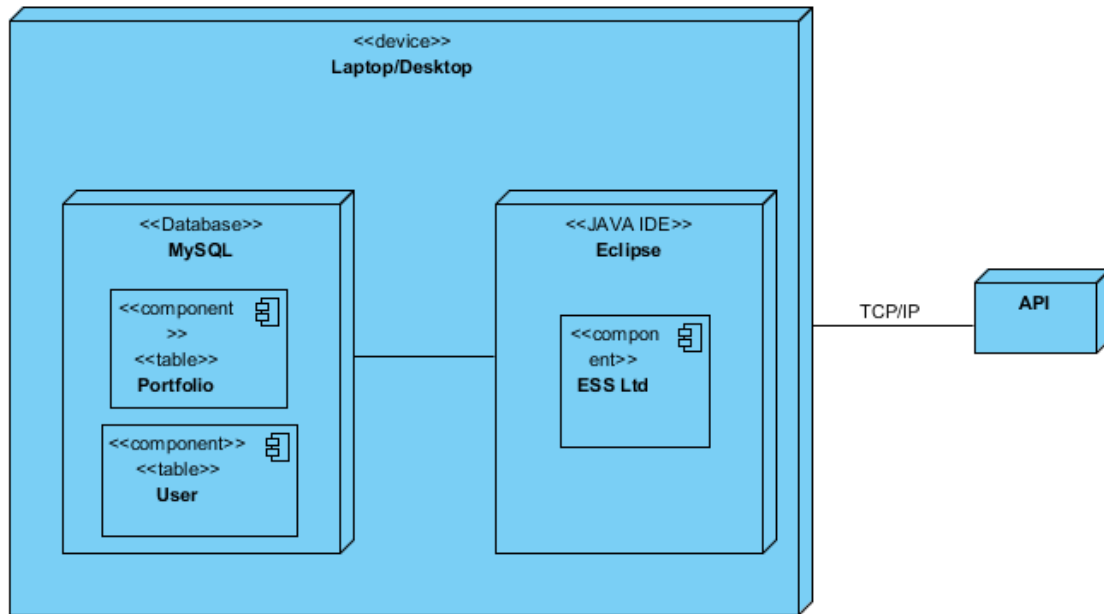


Figura 3 Diagrama UML: Technical Context

Os dois componentes principais:

Backend (API e MySQL)

MySQL – A base de dados permite guardar informação importante para o normal funcionamento da aplicação, nomeadamente os dados de registo dos utilizadores, e os seus respetivos Portfolios com os dados de cada ativo. A ligação à base de dados é local.

API – Fornece o valor real de mercado (NASDAQ) dos ativos que vão ser geridos em ESS Ltd. A taxa de atualização destes valores – infelizmente, por se tratar de uma API Open Source – é de um dia. A ligação é regida sobre o protocolo TCP/IP.

Frontend (ESS Ltd)

O Frontend, trata-se da interface da aplicação ESS Ltd desenvolvida em Java Swing, que corre sobre um IDE de Java, como por exemplo o Eclipse.

Interface	Canal
Valores dos ativos, extraídos da API externa.	Protocolo TCP/IP
Interface da aplicação	Java Swing

Tabela 8 Frontend

Solution Strategy

A aplicação ESS Ltd, tenta usar as ferramentas da linguagem de programação JAVA de modo a obter um programa funcional, e que apresente todas as capacidades a que se propôs.

O desenvolvimento e implementação da solução irá ser conseguido recorrendo a um IDE de JAVA, nomeadamente o Eclipse, devido às muitas qualidades destas ferramentas, sobretudo no que diz respeito à simplicidade e automatização do processo de programação.

O uso destes IDEs permite também, desenvolver de uma maneira bastante automatizada e consequentemente simples, uma interface gráfica apelativa e bastante completa. O JAVA Swing esconde a complexidade envolvida no desenvolvimento gráfico dos botões, tabelas, caixas de texto... gerando esse código automaticamente. A atribuição de ações a cada uma destas instâncias é facilitada devido à simplicidade na codificação das mesmas.

Os dados são uma componente com uma grande importância nesta aplicação. A API externa (yahoofinance) irá fornecer os valores reais dos ativos, em tempo real, para criar uma aplicação confiável e segura de se negociar. O sistema de gestão de base de dados MySQL irá jogar um papel importante durante a vida útil desta aplicação, visto que irá armazenar tudo o que diz respeito aos dados de registo dos utilizadores e os ativos dos seus respetivos portfolios. A importância destes dados levou a optar por um modelo relacional como o MySQL, garantindo deste modo a segurança e consistência dos mesmos.

A segurança desta aplicação, não é um foco a que se tivesse dado muita importância. O sistema de autenticação apenas requer uma password por utilizador, ainda que a mesma tenha de ter um tamanho superior a 8 caracteres, sem obrigatoriedade de inserção de qualquer um dos tipos existentes dos mesmos, mas não impedindo nenhum tipo de carácter de ser usado.

Building Block View

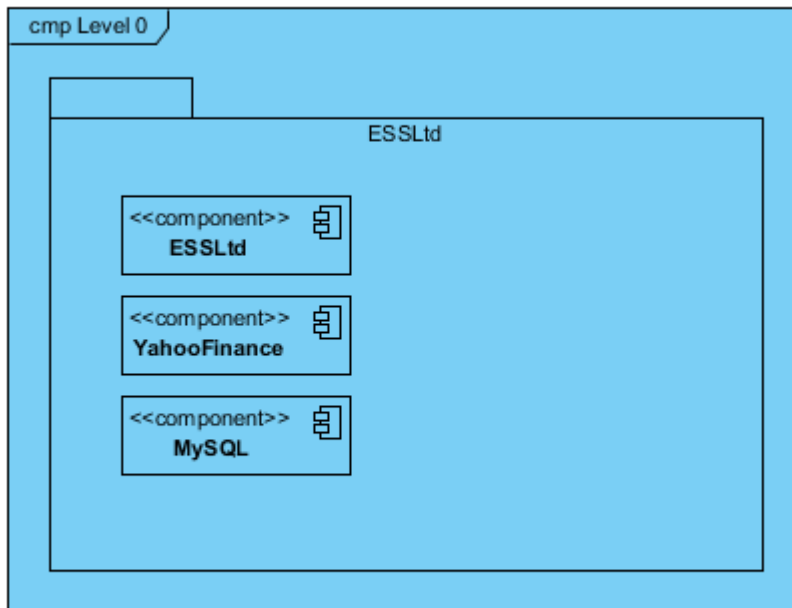


Figura 4 Diagrama UML: Building Block View

A aplicação ESS Ltd contém duas partes importantes de se serem destacadas: a aplicação em si (e a sua API) e YahooFinance, a API externa, usada neste projeto para retirar valores reais de valores dos ativos no mercado.

Whitebox Overall System

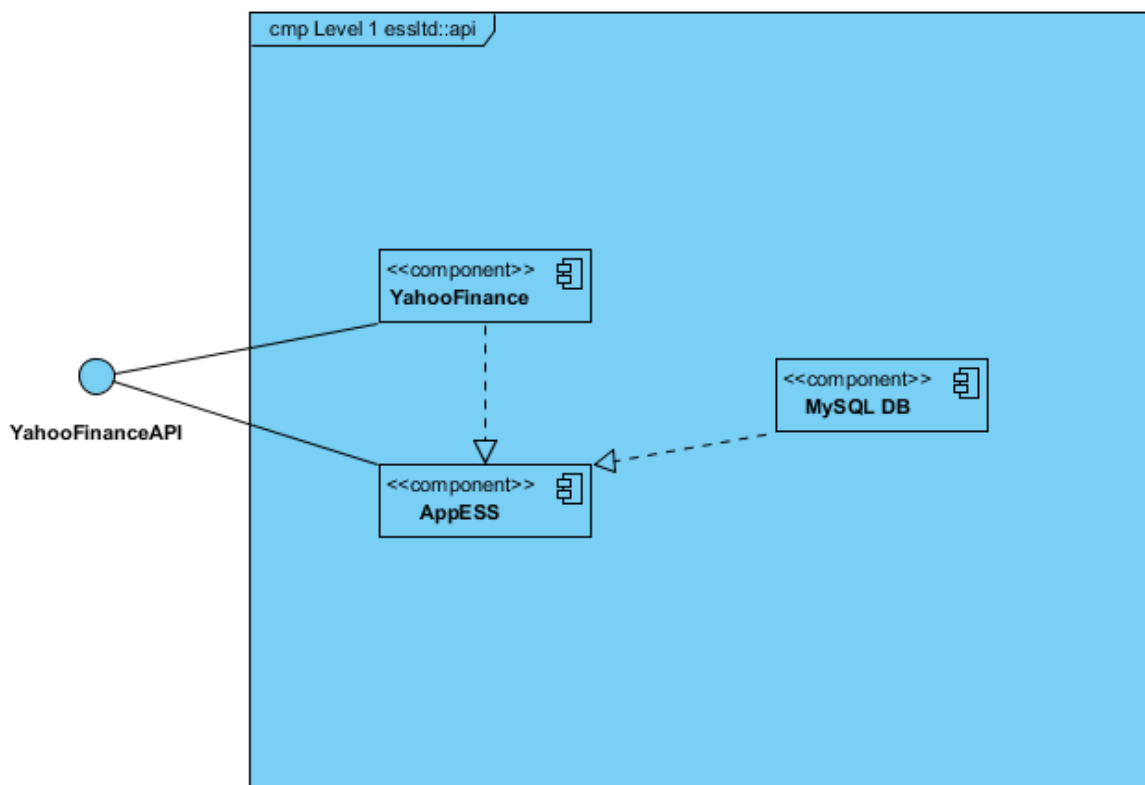


Figura 5 Diagrama UML: Whitebox

Contained Building Blocks

AppEss	Gere todo o sistema, incluindo registos e autenticações dos utilizadores e os seus portfolios assim como, ativos e as suas negociações. Vai buscar os dados dos ativos que disponibiliza em tempo real.
YahooFinance	Contém os métodos necessários para serem retiradas informações relevantes acerca dos ativos a serem negociados. Por exemplo, valores de mercado, percentagens da variação do valor do ativo, valor de uma ação dos ativos envolvidos.
MySQL DB	Guarda a informação relevante do sistema. Registo de utilizadores, portfolios, ativos negociados e as suas informações (valores de venda e compra,

	montantes investidos...).
--	---------------------------

Tabela 9 Contained Building Blocks

Important Interfaces

Interfaces	Descrição
YahooFinance API	Contém os métodos que permitem à aplicação retirar dados que dizem respeito a informações de mercado dos ativos. Por exemplo, preços de compra e venda, valores em tempo real dos preços de uma ação dos vários ativos no mercado onde os mesmos estiverem inseridos (maioritariamente NASDAQ).

Tabela 10 Interfaces Importantes

AppESS (blackbox)

Trata-se de uma API, que gere praticamente todo o sistema de ESS Ltd, nomeadamente a criação de contas no sistema, a abertura de negociações pelos ativos, respetiva manutenção de um sistema de controlo dos ativos a serem negociados num determinado momento, em qualquer uma das duas modalidades – compra ou venda - e criação de um portfolio de ativos “favoritos”. Contém também os métodos de gestão da base de dados, ou seja, está encarregue de guardar toda a informação referida anteriormente, numa base de dados desenvolvida para o efeito.

YahooFinance (blackbox)

Alberga os métodos retirados de uma API com o mesmo nome, e que estão a ser utilizados em AppESS, que tal como foi acima referido, trata da gestão e da realização de operações com um efeito prático no sistema.

MySQL DB (blackbox)

Guarda as informações mais importantes do sistema, como por exemplo, os dados de registo dos utilizadores, os ativos que o mesmo tem no portfolio, a associação entre os portfolios e os utilizadores registados, e as negociações em curso mapeadas com o utilizador que as negoceia.

Building Blocks

AppESS (whitebox)

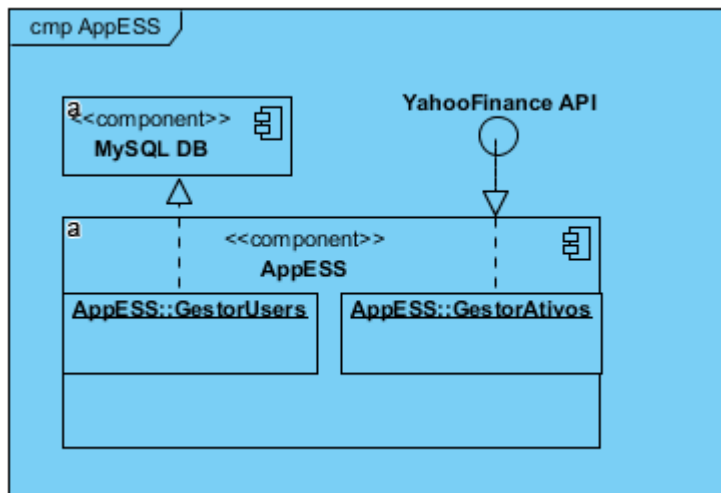


Figura 6 Diagrama UML: AppESS (whitebox)

GestorUsers e GestorAtivos são responsáveis por aceder e guardar dados referentes a utilizadores registados e ativos retirados da API YahooFinance, respetivamente.

YahooFinance (whitebox)

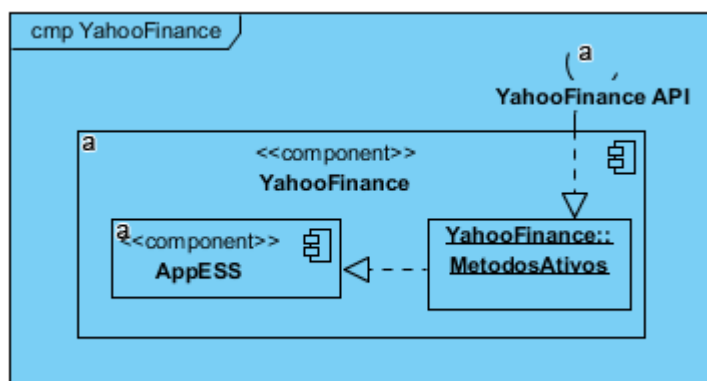


Figura 7 Diagrama UML: YahooFinance (whitebox)

Responsável por guardar os métodos, retirados diretamente da API YahooFinance, necessários para concretizar ações acerca dos ativos que o sistema necessita para que sejam cumpridos os requisitos a que se propôs.

Runtime View

Devido à grande quantidade de funcionalidades, mais ou menos, simples que se pretendem implementar neste projeto, acabei por escolher apenas 3 principais funcionalidades que têm mais alguma complexidade associada, sendo assim mais interessante desenvolver o seu desenrolar de eventos.

Consulta do Valor de Ativo

sd Consulta de Valor de Ativos

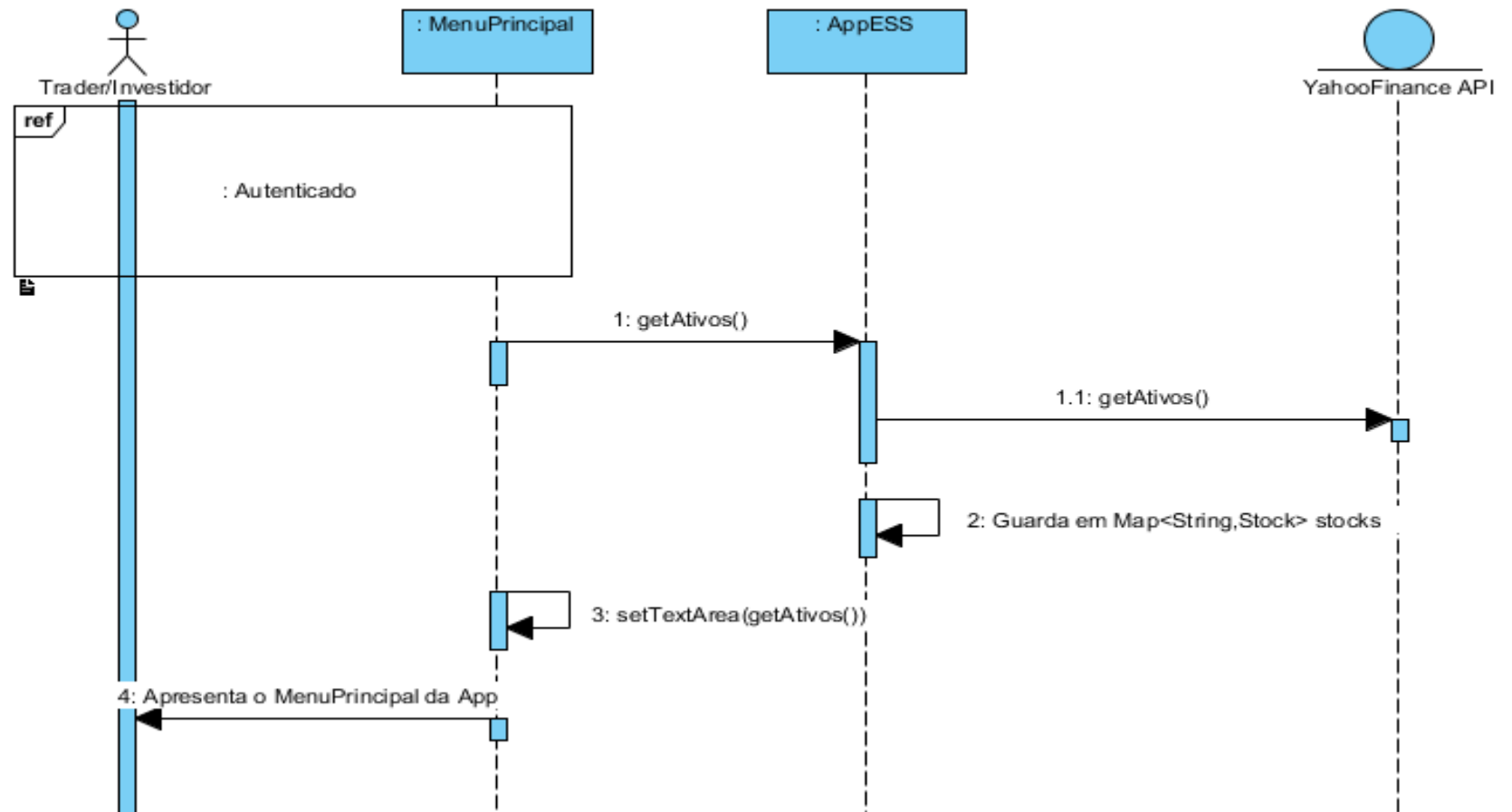


Figura 8 Diagrama UML: Consulta do Valor de Ativo

Ver Portfolio

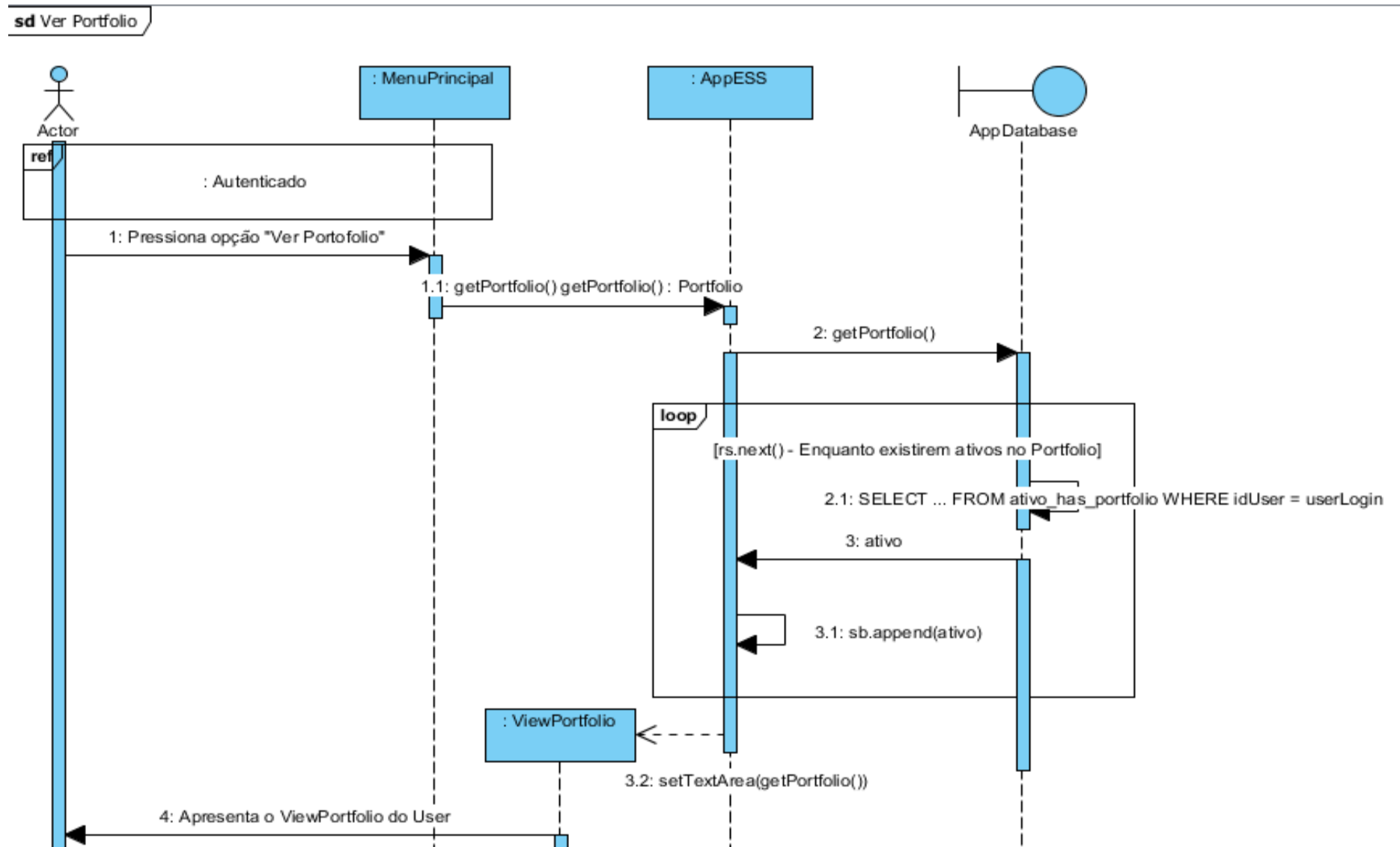


Figura 9 Diagrama UML: Ver Portfolio

Adicionar ativo ao Portfolio

sd Adicionar ativo ao Portfolio

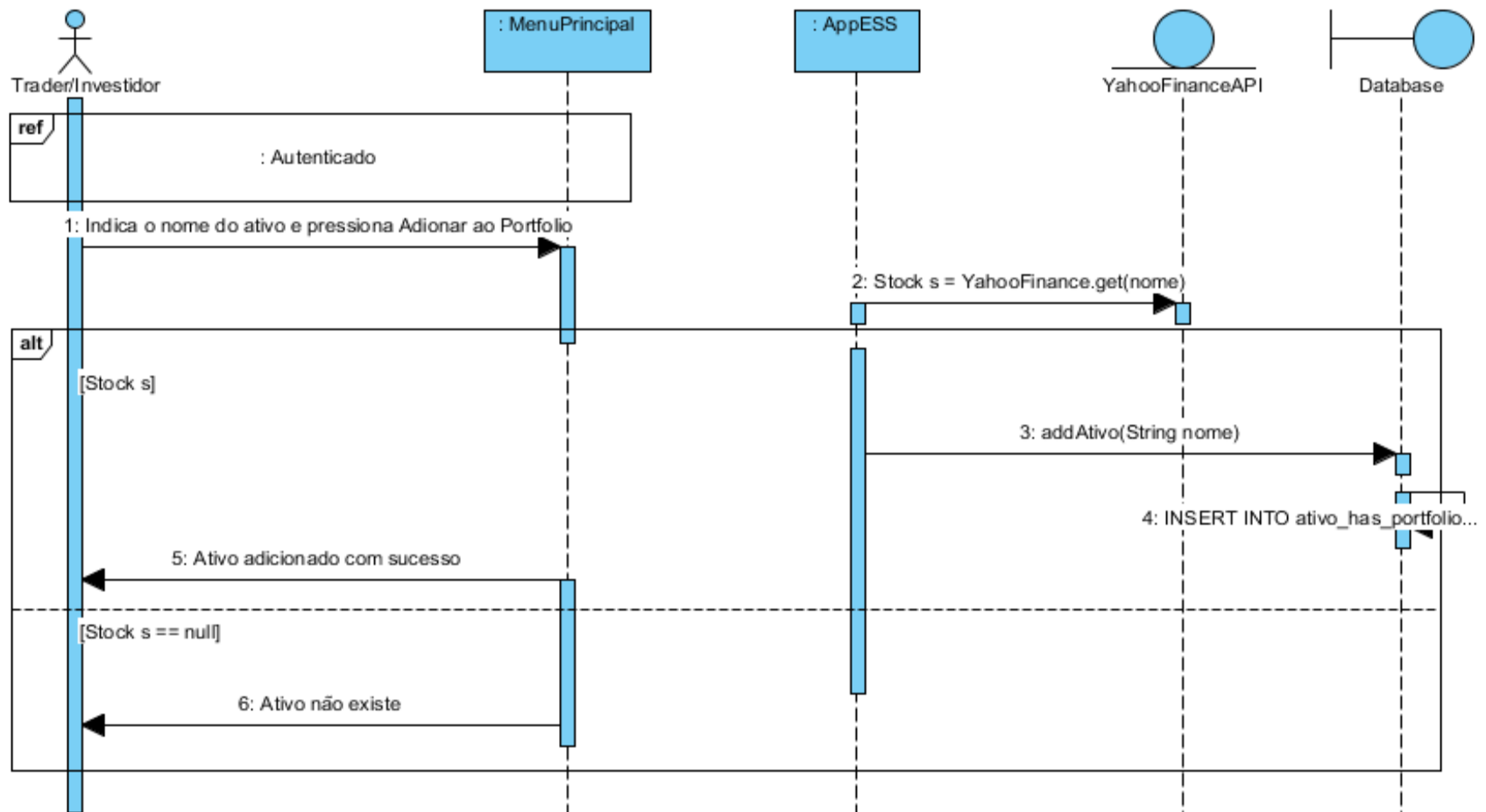


Figura 10 Diagrama UML: Adicionar ativo ao Portfolio

Abrir Venda

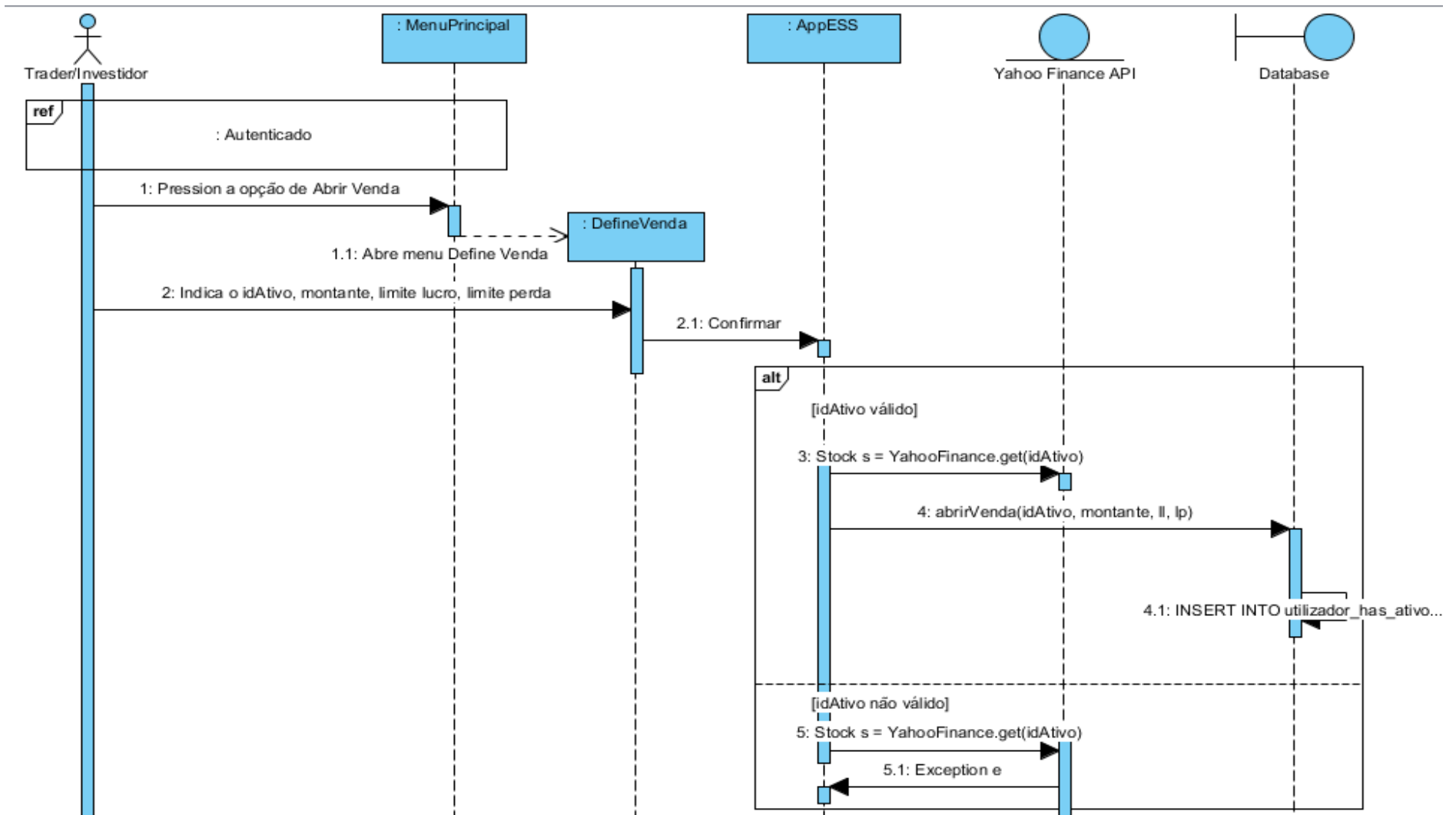


Figura 11 Diagrama UML: Abrir Venda

Deployment View

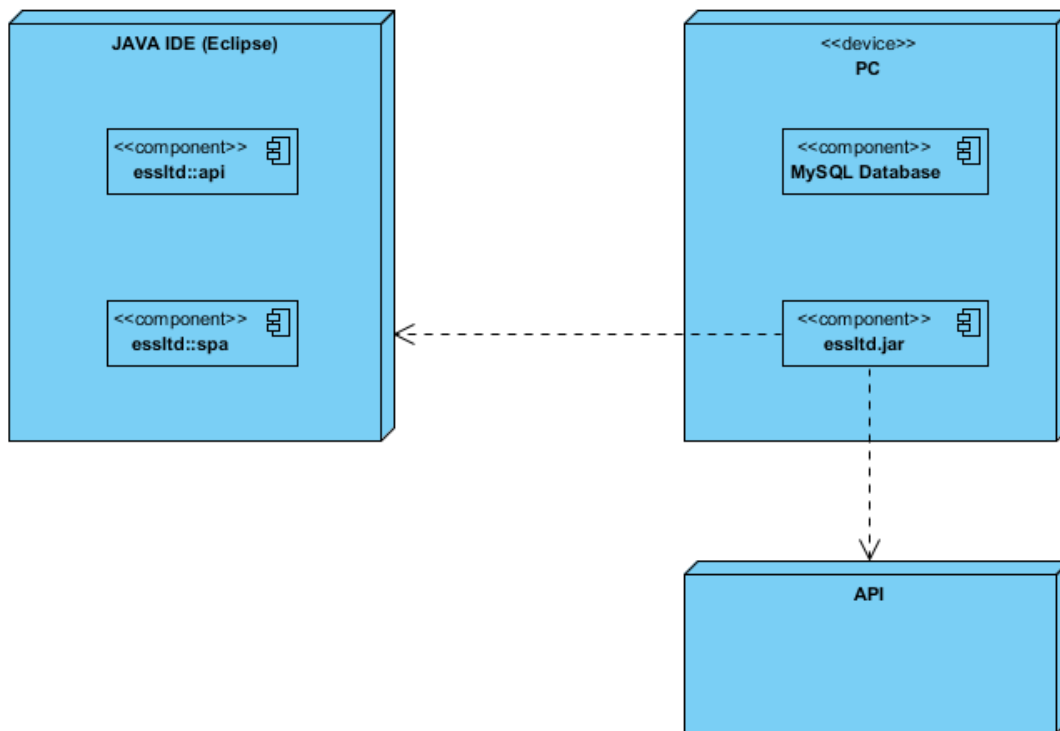


Figura 12 Diagrama UML: Deployment View

Nodos/Artifacts	Descrição
JAVA IDE (Eclipse)	Ambiente de desenvolvimento JAVA, onde a codificação do sistema ocorre, nessa mesma linguagem na sua versão JDK 8.
PC	Computador pessoal, com ligação à internet, onde a aplicação corre, juntamente com a base de dados que lhe irá dar suporte.
essltd.jar	Executável JAVA que permite que a aplicação corra em qualquer tipo de plataforma, i.e., sistema operativo. A ser executada a partir do ambiente de

	desenvolvimento Eclipse.
YahooFinance API	Uma API disponível para sistemas de Trading, que permita retirar informações reais acerca dos valores de mercado dos ativos.

Tabela 11 Deployment View

Cross-cutting Concepts

Modelo de Domínio

Visto que a aplicação ESS Ltd, será altamente dependente do uso de dados será necessário criar um diagrama entidade-relacionamento (ER-Diagram) para escolher um pouco melhor com que tipo de dados a aplicação irá trabalhar. O mesmo modelo será usado para desenvolver a base de dados MySQL, que irá tratar da manutenção desses dados.

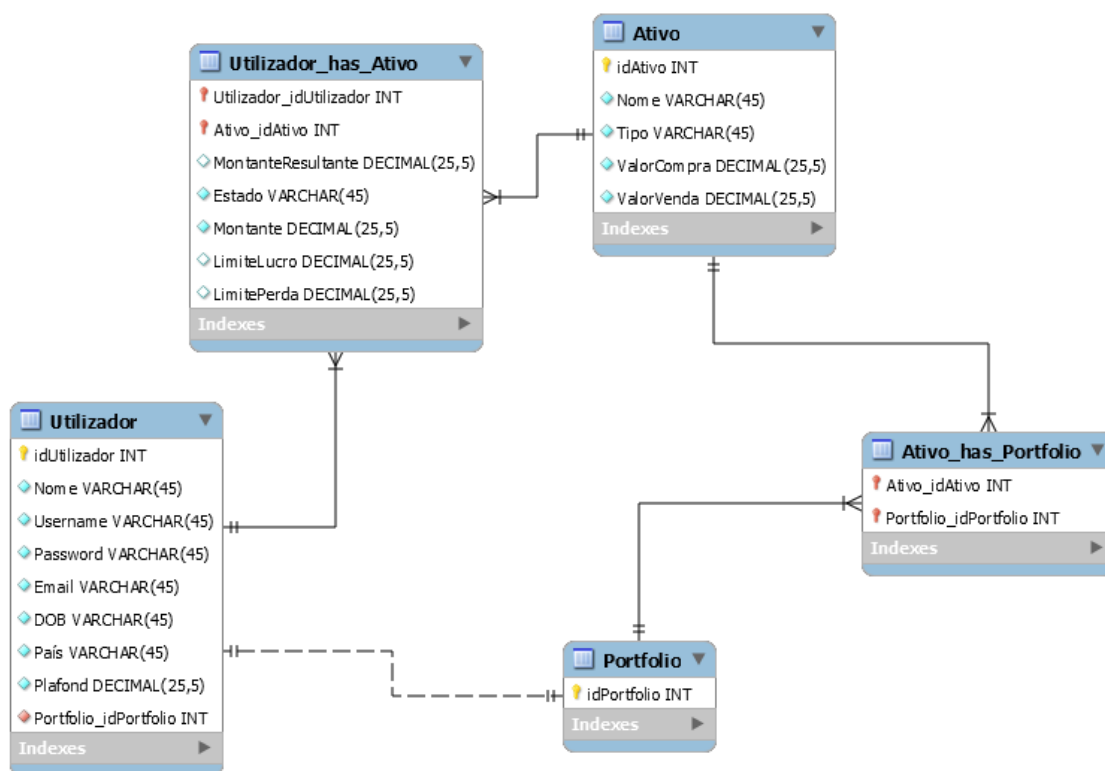


Figura 13 Modelo Lógico da Base de Dados

Nome da Tabela	Descrição
Utilizador	Contém a informação relativa ao utilizador, nomeadamente os seus dados de registo (Nome, Username, Password, DOB, Email e País), assim como uma referência ao identificador do seu Portfolio.

Portfolio	Contém apenas os identificadores dos diversos Portfolios de Utilizadores, que estão guardados no sistema.
Ativo	Contém a informação dos ativos, como por exemplo nome, tipo (índice, commodity...). Ajuda a manter informação acerca dos ativos guardados no Portfolio de cada Utilizador. Guarda informação sempre que um ativo esteja a ser negociado, nomeadamente o valor de venda, o valor de compra, o montante investido pelo utilizador, assim como os limites de perda e de lucro, que deverão servir como indicar de modo a fechar a compra ou venda do ativo imediatamente de forma automática caso o montante virtual estimado ultrapassar algum dos mesmos.
Ativo_has_Portfolio	Contém as relações entre os identificadores do Portfolios e dos ativos. Deste modo é possível perceber quais são os ativos que fazem parte de um determinado Portfolio.
Utilizador_has_Ativo	Contém as relações entre os identificadores dos Utilizadores e dos Ativos. Deste modo, é possível perceber que Ativos estão a ser negociados por determinado Utilizador, assim como os Utilizadores que estão a negociar um determinado Ativo.

Tabela 12 Dicionário de dados da Base de Dados

Modelo de Domínio (2ª Fase)

Devido ao requisito funcional que surgiu nesta segunda fase do projeto, e de modo a facilitar um pouco a implementação dessa mesma funcionalidade, foram feitas ligeiras alterações à BD MySQL utilizada na primeira fase, nomeadamente: a adição de uma tabela “alerta”, que associa aos utilizadores do sistema, os ativos que os mesmos pretendem receber notificações. O resultado final é apresentado da seguinte forma:

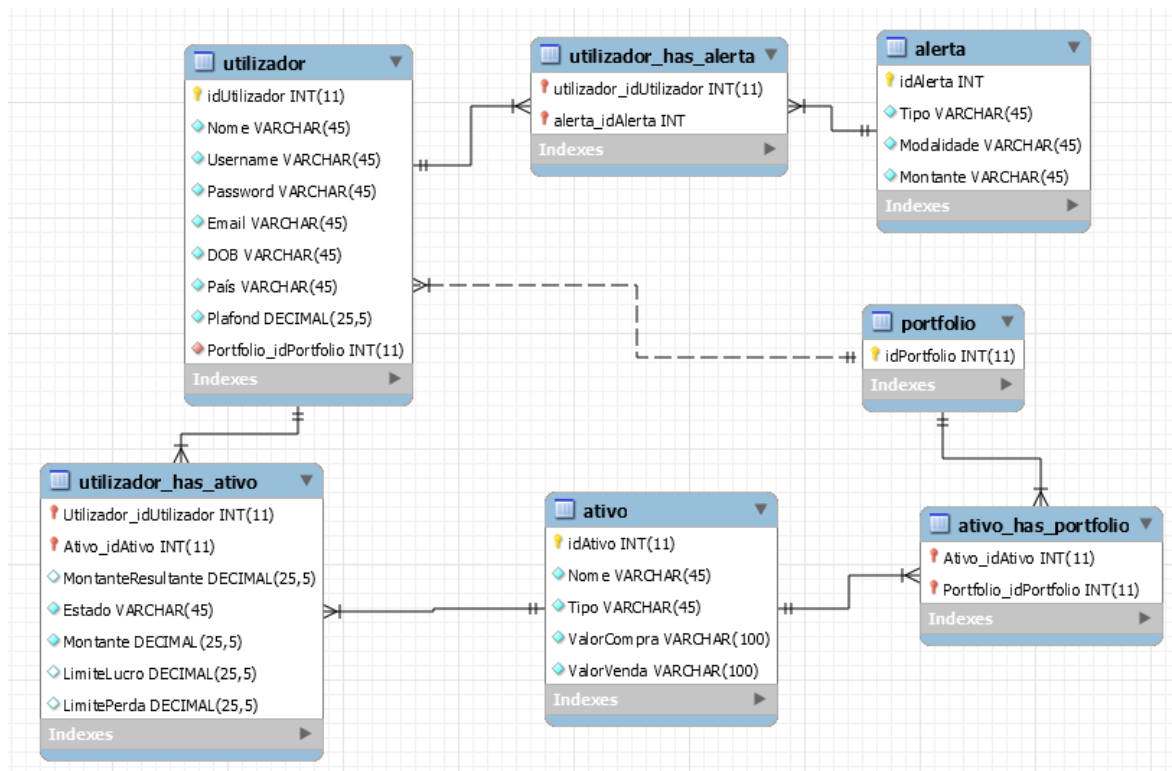


Tabela 13 Nova Base de Dados

Desta alteração, derivam as seguintes novas tabelas:

Nome da Tabela	Descrição
Alerta	Contém um id que atua como chave primária. O tipo trata-se do código NASDAQ do ativo, a modalidade trata-se da especificação de qual o valor das modalidades ser controlado, e o montante trata-se do montante limite a partir do qual o utilizador deve ser notificado.
Utilizador_has_Alerta	Contém a relação entre o utilizador e os alertas que tem registados.

No seguimento deste modelo, surge o seguinte modelo de domínio. Na classe AppESS, ou seja a classe da qual fazem parte os métodos que fazem o trabalho de calcular resultados, de modo a que o programa satisfaça todas as funcionalidades, não estão especificados todos os métodos por motivos de apresentação, fica portanto uma versão mais reduzida, apenas com uma listagem de alguns métodos mais relevantes para esta segunda fase, mas que permite ver sem problemas, a estrutura arquitetural implementada na mesma, de modo a conseguir resultados efetivos acerca da presença do padrão observador no programa:

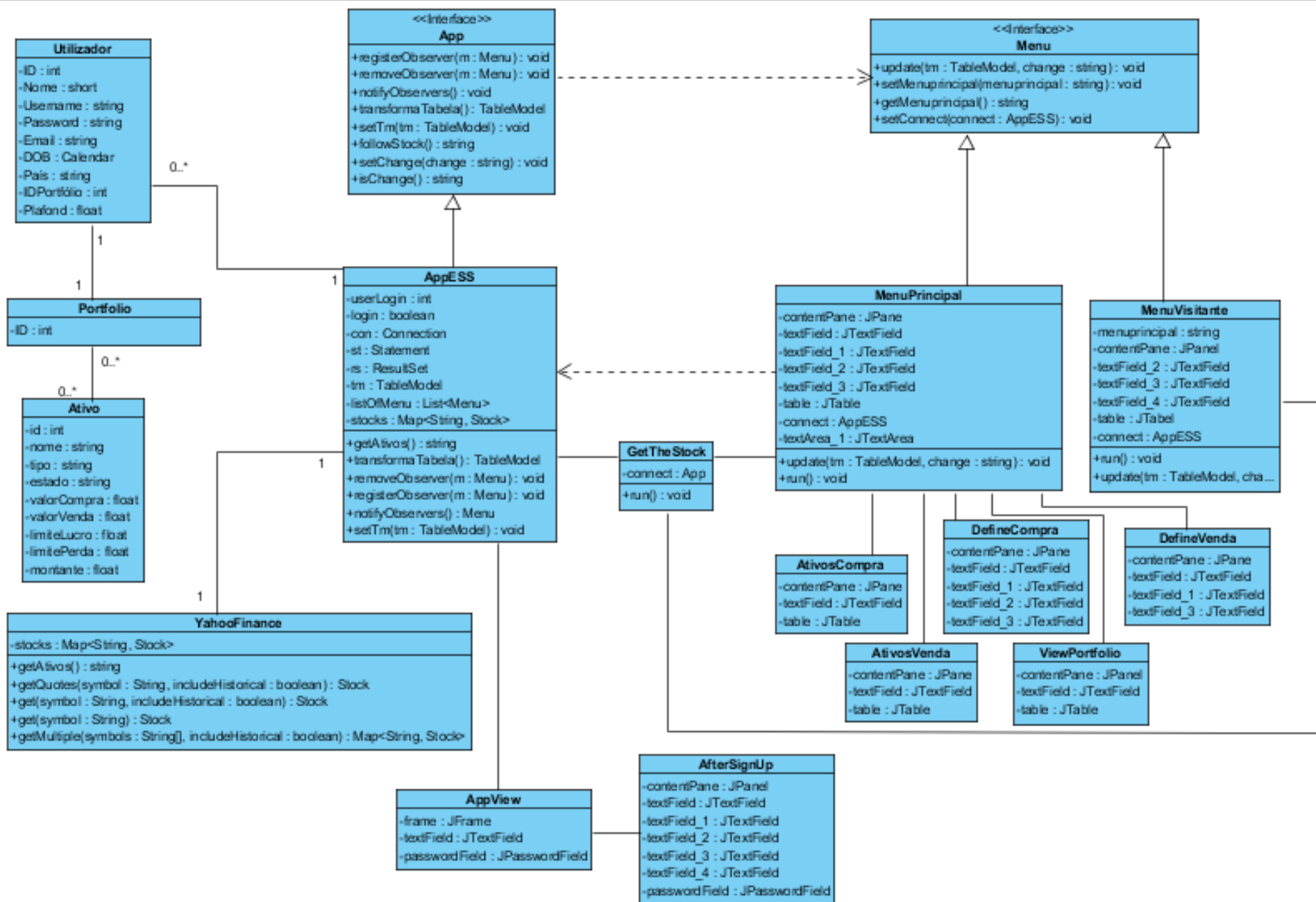


Figura 14 Diagrama UML: Modelo de Domínio

Nome	Descrição
AppESS	Terá como função controlar tudo o que diz respeito a autenticação, e menus.
Utilizador	Terá toda a informação referente a um utilizador, nomeadamente dados de registo e dados pessoais dentro da aplicação como por exemplo o seu Plafond.
Ativo	Terá todos os dados sobre a listagem de ativos que estão a ser negociados por um Utilizador, num determinado instante. Terá os dados específicos de cada um dos ativos a serem negociados.
Portfolio	Terá a listagem dos ativos marcados como importantes, por cada utilizador.
YahooFinance	Terá os métodos retirados diretamente da API com o mesmo nome, e que serão úteis para operações relacionadas com os ativos.
AppView	Menu de abertura da aplicação, o seu conteúdo será aquele indispensável a um desenvolvimento com elementos JSwing.
AfterSignUp	Menu de criação de conta da aplicação o seu conteúdo será aquele indispensável a um desenvolvimento com elementos JSwing.
MenuPrincipal	Menu que aparece após uma autenticação bem-sucedida, o seu conteúdo será aquele indispensável a um desenvolvimento com elementos JSwing.
AtivosVenda	Menu de ativos que o utilizador está a negociar em modo de Venda, o seu conteúdo será aquele indispensável a um

	desenvolvimento com elementos JSwing.
AtivosCompra	Menu de ativos que o utilizador está a negociar em modo de Compra, o seu conteúdo será aquele indispensável a um desenvolvimento com elementos JSwing.
ViewPortfolio	Menu de ativos no Portfolio do utilizador, o seu conteúdo será aquele indispensável a um desenvolvimento com elementos JSwing.
DefineCompra	Menu onde o utilizador deve indicar qual o ativo, o montante, o limite de lucro e o limite de perda que pretende associar a uma nova abertura de opção de compra, o seu conteúdo será aquele indispensável a um desenvolvimento com elementos JSwing.
DefineVenda	Menu onde o utilizador deve indicar qual o ativo, o montante, o limite de lucro e o limite de perda que pretende associar a uma nova abertura de opção de venda, o seu conteúdo será aquele indispensável a um desenvolvimento com elementos JSwing.
MenuVisitante	Menu para utilizadores não registados, não necessita de autenticação prévia, apenas mostra a tabela com os valores dos ativos.
GetTheStock	Classe onde é implementada a Thread que de 5 em 5 segundos, dá indicação de que devem ser notificados todos os “Observadores”, pois os valores dos ativos mudaram.
App	Interface que as classes dos “Observados” implementam.

Menu	Interface que as classes “Observadoras” implementam.
------	--

Tabela 14 Descrição do Modelo de Domínio

Métodos da classe “AppESS”

Nome	Descrição
criarConta	Efetua o registo de um novo utilizador no sistema.
fazLogin	Autentica um utilizador no sistema.
logout	Termina a sessão de um utilizador.
getAtivos	Apresenta a lista dos ativos.
getPortfolio	Apresenta o Portfolio do utilizador autenticado.
abrirVenda	Abre uma opção de venda num determinado ativo.
abrirCompra	Abre uma opção de compra num determinado ativo.
addAtivo	Adiciona um determinado ativo ao portfolio.
removeAtivo	Remove um determinado ativo do portfolio.
getPlafond	Devolve o Plafond atual do utilizador autenticado.
getAtivosCompra	Devolve a lista de ativos em que o utilizador investiu na opção de compra.
getAtivosVenda	Devolve a lista de ativos em que o utilizador investiu na opção de venda.
fechaCompra	Fecha o investimento em determinado ativo no modo compra.

fechaVenda	Fecha o investimento em determinado ativo no modo compra.
addWatchlist	Adiciona um novo ativo à watchlist.
removeWatchlist	Remove um ativo da watchlist.
addFundos	Acrescenta fundos.
resultSetToTableModel	Converte um ResultSet num formato Table Model.
transformaTabela	Transforma a lista de ativos numa Table Model.
getPlafondInvestido	Devolve o montante que o utilizador tem investido em ações num determinado instante.
getPlafondInvestidoModalidade	Devolve o montante que o utilizador tem investido em ações num determinado instante, consoante a modalidade em que esse dinheiro está investido.
updateCompra	Atualiza, em tempo real, o montante resultante do investimento realizado em determinado ativo, na modalidade de compra.
updateVenda	Atualiza, em tempo real, o montante resultante do investimento realizado em determinado ativo, na modalidade de venda.
registerObserver	Regista um objeto como observador, de determinado observado.
removeObserver	Retira um objeto da lista de observadores registados, deixando de receber as atualizações sempre que esse objeto sofre uma mudança.

notifyObservers	Notifica os observadores registados, da mudança ocorrida no observado.
addAlerta	Permite ao utilizador receber notificações assim que um ativo atinja o valor que o mesmo determinar de antemão.
followStock	Permite devolver as notificações ao utilizador em tempo real.
unfollowStock	Permite ao utilizador, cancelar a receção de notificações referentes a um determinado ativo.

Tabela 15 Descrição dos métodos da classe AppESS

Métodos da classe “YahooFinance”

Nome	Descrição
getAtivos	Apresenta a lista de ativos.
getQuotes	Apresenta, para cada ativo, a informação relativa a valores de preço, compra, venda, valores de fecho etc.
get	Vai buscar a informação atual de um ativo em particular.
get	Vai buscar a informação atual de um ativo em particular com a opção de devolver juntamente todo o seu histórico.
getMultiple	Vai buscar a informação atual de vários ativos ao mesmo tempo, com a opção de devolver juntamente o histórico de cada um desses ativos.

Tabela 16 Descrição dos métodos da classe YahooFinance

Métodos da classe “MenuPrincipal”

Nome	Descrição
update	Atualiza a tabela de ativos, caso o menu esteja registrado como Observador.

Tabela 17 Descrição dos métodos da classe MenuPrincipal

Métodos da classe “MenuVisitante”

Nome	Descrição
update	Atualiza a tabela de ativos, caso o menu esteja registrado como Observador.

Tabela 18 Descrição dos métodos da classe MenuVisitante

Persistência

Tal como já vem sendo a ser referido neste relatório, a persistência está ao cargo de uma base de dados relacional, nomeadamente o MySQL. Esta base de dados irá armazenar os dados dos utilizadores e ativos.

A aplicação será capaz de aceder à base de dados através de conetores JDBC.

Interface Gráfica

A interface gráfica para todos os utilizadores será desenvolvida em JSwing, com recurso ao IDE Eclipse (versão Oxygen) para ajudar na codificação da mesma.

Sessões

As sessões são geridas pela classe AppESS, localmente.

Segurança

O único sistema de segurança que o ESS Ltd apresenta, é a autenticação via uma password apenas. Confiamos nos protocolos de segurança do TCP/IP para garantir a mesma na hora de ir buscar dados com a API externa.

Proteção

Nenhum aspeto nesta aplicação coloca em risco os utilizadores.

Comunicação

A aplicação não é persistente o suficiente para conseguir guardar o estado atual e reiniciar a aplicação exatamente nesse estado.

Validade dos Dados

Ainda que o a base de dados, sobretudo por ser relacional, trate de garantir que os datatypes, valores nulos, intervalos estão corretos e a consistência está sempre garantida em todos os momentos, existem alguns dados onde vale a pena um esclarecimento adicional acerca da sua validação:

1. Não pode haver utilizadores com o mesmo nome de utilizador;
2. Se existir uma perda após o fecho da opção de compra/venda de determinado ativo, o plafond do utilizador deve ser inferior ao anterior;
3. Só existem dois estados possíveis para cada ativo: VENDA e COMPRA, que correspondem às opções de negócio oferecidas aos utilizadores;

4. Os tipos de ativo possíveis são: índice, commodity, ações e moeda.

Tratamento de Exceções e Erros

O tratamento de exceções e erros, está ao cargo da base de dados, que deve apenas aceitar dados que mantenham a base de dados consistente em qualquer momento. A base de dados deve encarregar-se de garantir que as transações que ocorrerão na base de dados não tornarão a mesma inconsistente. Falhas de hardware, não são consideradas, i.e., não existe sistema de backup, nem algo do género, logo podem ocorrer perdas de dados, caso algo a este nível falhe.

Internacionalização

A aplicação apresenta uma mistura de dois idiomas, com o Português como língua principal e com a presença de alguns termos em Inglês.

Migração

O ESS Ltd é uma aplicação construída de raiz.

Testabilidade

Até à data nenhum teste foi feito à aplicação.

Build-Management

A aplicação pode continuar a ser desenvolvida sem problemas noutro IDE Java que não o Eclipse. A interface pode continuar a ser desenvolvida em qualquer outro IDE, desde que suporte JSwing. A base de dados pode ser transferida desde o MySQL, para qualquer outro motor, desde que o mesmo adote também ele uma abordagem relacional, após as devidas conversões semânticas.

Design Decisions

Uso do Sistema de Gestão de Base de Dados MySQL

Problema:

A importância e a quantidade de dados que circulam nesta aplicação, implicava o recurso a SGBD, para que os mesmos ficassem mais seguros e organizados.

Restrições:

Os dados devem ser fáceis de aceder.

Não será implementado um sistema de backup, logo o SGBD deve ser fiável.

Assunções:

Os SGBD relacionais dão preferência à consistência em relação à disponibilidade dos dados.

Alternativas Consideradas:

Armazenamento dos dados num ficheiro do formato .docx, .xml, .txt...

Decisão:

Recurso ao SGBD MySQL, para fazer o trabalho de base de dados. Por ser um SGBD que adota um modelo do tipo relacional satisfaz as condições que considere necessário para que tudo se processe sem problemas.

Uso do JSwing para o desenvolvimento da Interface Gráfica

Problema:

Era necessário que a aplicação tivesse uma interface gráfica apelativa, pois estamos perante uma aplicação que depende muito dessa interface, para a concretização das tarefas a que se propõe.

Restrições:

Deve ser simples e fácil de perceber por parte dos utilizadores.

Assunções:

Interfaces pesadas comprometem a usabilidade e a capacidade de resposta da aplicação.

Alternativas Consideradas:

Desenvolver a aplicação sobre a linha de comandos, i.e., com pouca, ou até mesmo, nenhuma interface gráfica.

Decisão:

Recurso ao JSwing, por ser fácil de trabalhar, e por ter muitos automatismos implementados, no que diz respeito à facilidade na codificação da interface. É capaz de criar interfaces bastante apelativas.

Uso da API Yahoo Finance para os valores reais dos ativos

Problema:

Era necessário encontrar solução para atribuir valores aos valores de mercado dos ativos a serem negociados, caso contrário seria impossível negociar sobre os mesmos na forma de investimentos monetários. Era necessário de igual maneira conseguir encontrar os próprios ativos.

Restrições:

Deve ser possível arranjar valores de compra e venda para cada ativo assim como a lista dos próprios ativos.

Assunções:

APIs externas podem comprometer a aplicação e podem ser difíceis de trabalhar.

Valores aleatórios não traduzem verdadeiramente as flutuações de mercado.

Alternativas Consideradas:

Reunir os valores de mercado de um número fixo de ativos, e de seguida, tendo em conta esses valores, para cada um dos ativos determinar um desvio-padrão adequado, de modo a calcular um intervalo fixo, onde valores aleatórios de venda e compra seriam gerados para cada ativo, dentro das fronteiras desse mesmo intervalo.

Decisão:

Recurso a uma API externa chamada Yahoo Finance, pela simplicidade de aplicação dos seus métodos ao projeto, por traduzir mais verdadeiramente o que se passa nos mercados – não se trata de uma mera simulação - e sobretudo pela sua alta taxa de atualização dos valores nos mercados reais (refresh rate).

Uso do padrão observador

Problema:

Estamos perante um projeto, e mais especificamente, uma aplicação, que requer constantes atualizações dos dados que apresenta, pois só assim é atingida a credibilidade pretendida no seu uso. Em particular, no meu caso, apresento uma aplicação com uma interface gráfica que usa muitos Menus diferentes, para melhor organização, mas que têm em grande parte, uma característica comum: a presença de ativos, cujos valores devem ser atualizados frequentemente. Em suma, muitos casos em que a solução para o problema é a mesma, pois os elementos presentes são equivalentes entre si, assim como os seus constituintes.

Restrições:

Não deve ser expectável uma mudança radical da arquitetura da aplicação. Deve existir uma solução que permita apenas realizar um trabalho de “adaptação” da versão antiga.

Assunções:

Padrões de desenho podem ser difíceis de adaptar ao projeto. Trata-se de uma questão de sorte, no sentido, em que o processo de adaptação pode ser simples, ou no extremo oposto, pode deixar de ser um processo de adaptação e passar a um processo de remodelação, caso a primeira implementação (que não se preocupava com padrões) seja algo demasiado estrito, e que não permita mudanças.

Alternativas Consideradas:

Outros padrões de desenho, como por exemplo, Iterador.

Decisão:

Optei por usar o padrão observador, por se tratar de um padrão simples e eficiente. A minha aplicação é extremamente dependente da interface gráfica. No entanto, apesar de complicada, uma das decisões tomadas na primeira fase do projeto, no que diz respeito à interface, deu frutos. Essa decisão foi a de criar Menus separados, para cada uma das operações/funcionalidades da aplicação. Deste modo, cada execução de determinada operação traduz-me num novo Menu, independente dos outros. Falo de operações, como a de abrir compra/venda, ver ativos investidos etc...

Consequentemente, surgirão inúmeros cenários, onde vamos ter vários objetos equivalentes entre si, os Menus, que partilham a mesma informação, os ativos.

A implementação de um padrão observador, permitiu juntar os Menus, que no meu entender se podem considerar como observadores, dos mesmos componentes, ou seja dos ativos. Para além de acabar com estaticidade do programa, pois passa a existir uma atualização automática dos Menus, cada vez que existe uma alteração dos seus

constituintes, é criada uma homogeneidade na informação que circula nos diferentes Menus evitando disparidades, pois todos vão passar a ser notificados com os mesmos conteúdos e uma melhor organização na própria arquitetura, com maior facilidade na inserção de novas funcionalidades daí em diante.

Tal como já se pode entender, no padrão observador clássico teríamos uma estrutura de Subjects e Observers. Aqui neste caso específico AppESS e os Menus: Principal e Visitante (mas também outros do mesmo tipo, Menu, que no futuro queiramos incluir), vão ocupar esses papéis respetivamente. Desta feita posso dizer que as alterações que foram feitas ao projeto da fase anterior, em termos estruturais passaram por:

- i. Criação de duas interfaces: App e Menu, que AppESS e MenuPrincipal e MenuVisitante, passam a implementar respetivamente;
- ii. Criação da classe GetTheStock, para que de 5 em 5 segundos (valor modificável), seja lançada uma indicação de que devem ser notificados todos os Menus de modo a sofrerem um update, da sua tabela e respetivos alertas;
- iii. Criação dos métodos presentes em qualquer projeto que adote o padrão observador, e essenciais ao seu mesmo funcionamento (registerObserver, notifyObservers, update, etc.);
- iv. Criação de uma lista de Menus em AppESS, para serem todos notificados do novo update, criação de uma instância do tipo AppESS em cada Menu, para o registar como Observador;
- v. Alteração da base de dados para correta implementação do novo requisito de alertas;
- vi. Alteração do método update. Deste modo cada Menu registado como observador, passa não só a receber, a nova tabela atualizada, mas também uma String com a informação textual organizada por linhas, dos alertas definidos pelo utilizador em questão;
- vii. Criação de um Menu para visitantes, de modo a confirmar o correto registo por parte do sistema na listagem de Menus Observadores.

Uso do padrão arquitetural por camadas e Model View

Apesar de não ter sido muito explorado da minha parte, foi tomada a decisão de recorrer a um padrão arquitetural de uma camada (layer) pois todo o projeto assenta desde início, de raiz, apenas num package.

Consequentemente, devido ao uso do JSwing como ferramenta para o desenvolvimento da interface gráfica, fica patente a presença do padrão arquitetural Model View, no projeto.

Quality Requirements

Árvore de Qualidade

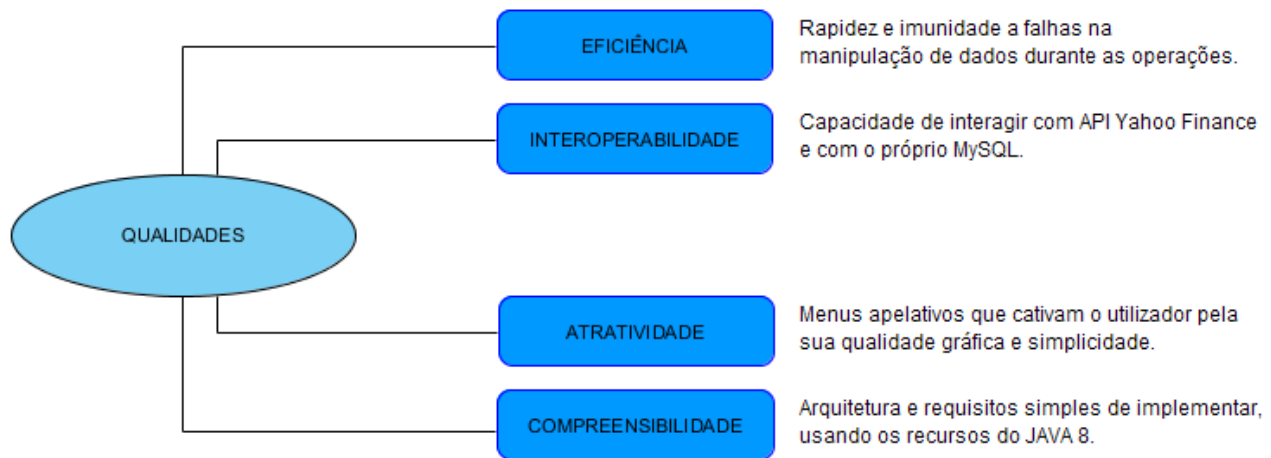


Figura 15 Árvore de Qualidade

Cenários de Qualidade

Eficiência – Todas as operações devem ser concretizadas na sua totalidade em menos de 1s. Deve ser admitido como compreensível/aceitável a ocorrência de 1 falha a cada 100000 (cem mil) operações.

Interoperabilidade – As operações devem todas implicar uma transação na base de dados possível de se provar, assim como os valores dos ativos devem corresponder aqueles que constam no serviço Yahoo Finance, quando comparados.

Atratividade – A aplicação deve cativar os utilizadores de modo a atingir uma meta de 5000 (cinco mil) downloads no primeiro mês de comercialização. Deve haver uma receptividade positiva por parte desses mesmos utilizadores. Os utilizadores habituados a aplicações e ao mundo do trading devem cometer no máximo 1 erro a cada 10 operações, comprovando deste modo que tudo o que diz respeito a especificações dos botões e as funções dos mesmos foi devidamente conseguido.

Risks and Technical Debts

O maior risco associado à aplicação ESS Ltd, é a perda dos dados inseridos na base de dados, este risco é mitigado através da realização periódica de cópias de segurança da mesma.

Glossary

Nome	Descrição
IDE	Sigla para ambiente de Desenvolvimento Integrado (Integrated Development Enviroment).
JAVA	Linguagem de programação orientada aos Objetos.
MySQL	Sistema de gestão de base de dados relacional.
JSwing	Ferramenta que permite contruir GUIs para programas desenvolvidos em JAVA.
GUI	Interface gráfica do utilizador (Graphical User Interface).
API	Interface de programação de Aplicações (Application Programming Interface).
Eclipse	Nome do IDE usado no desenvolvimento deste projeto.

Tabela 19 Glossário

Apêndices

Mockups iniciais

De seguida passo a apresentar os primeiros mockups, desenvolvidos para a interface gráfica da aplicação, juntamente com o resultado final efetivo para cada um dos menus:

Menu Inicial



Figura 18 Mockup Menu Inicial

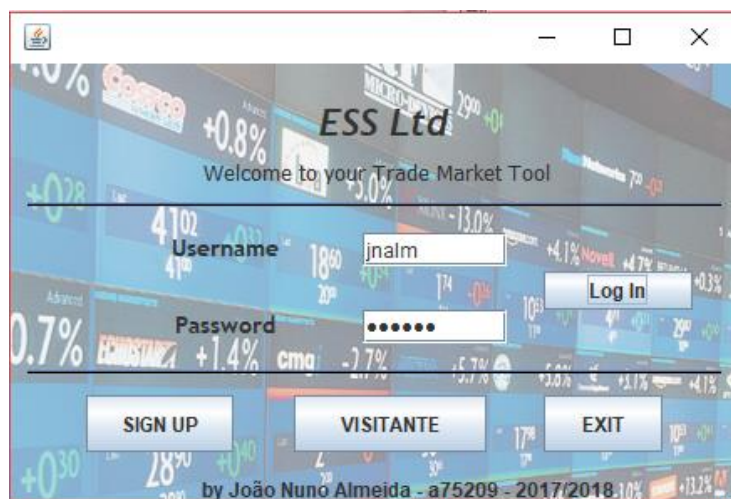


Figura 17 Menu Inicial

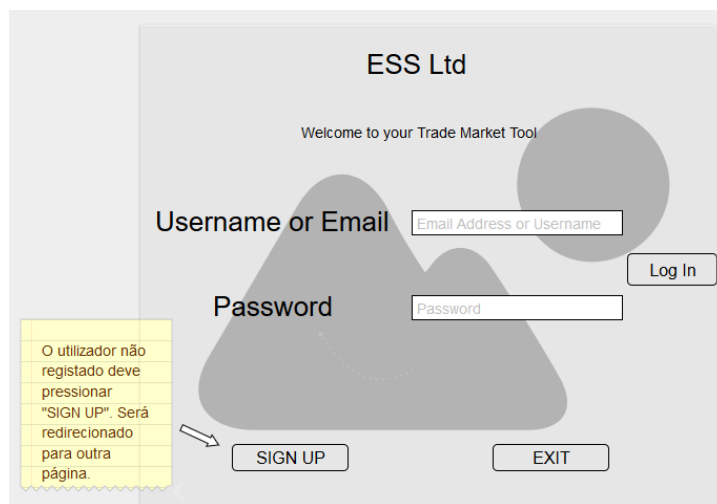
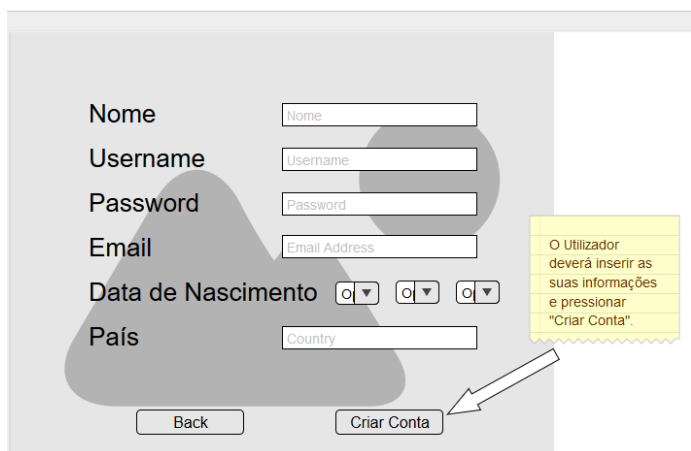


Figura 16 Mockup Menu Inicial

Sign Up



Nome

Username

Password

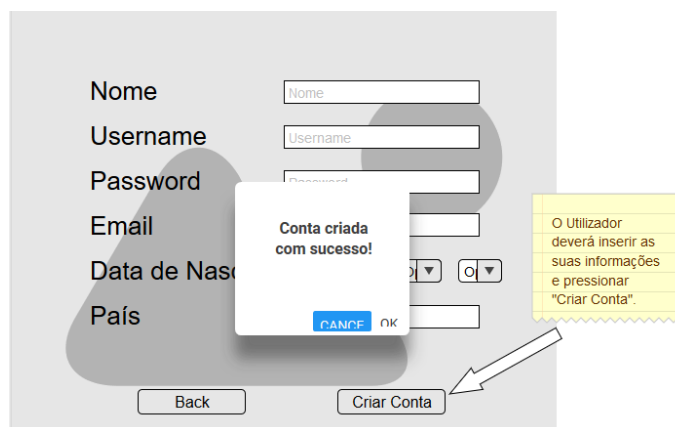
Email

Data de Nascimento

País

O Utilizador deverá inserir as suas informações e pressionar "Criar Conta".

Figura 20 Mockup Sign Up



Nome

Username

Password

Email

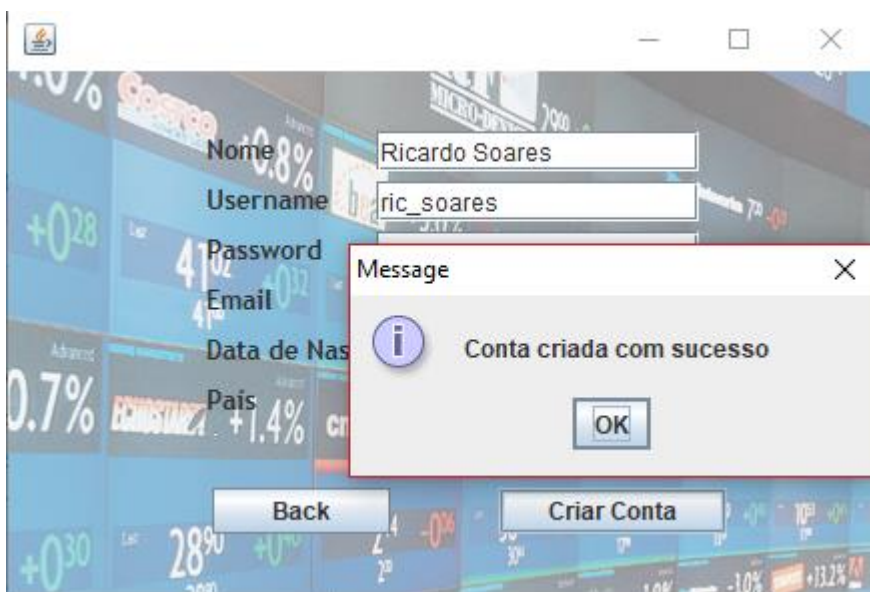
Data de Nascimento

País

Conta criada com sucesso!

O Utilizador deverá inserir as suas informações e pressionar "Criar Conta".

Figura 19 Mockup Sign Up



Nome

Username

Password

Email

Data de Nascimento

País

Message

Conta criada com sucesso

Figura 21 Sign Up

Menu Principal

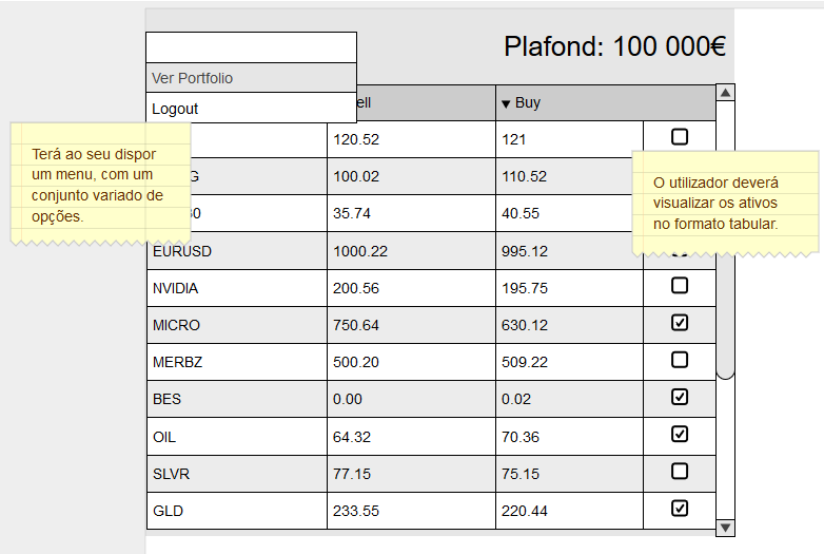


Figura 22 Mockup Menu Principal

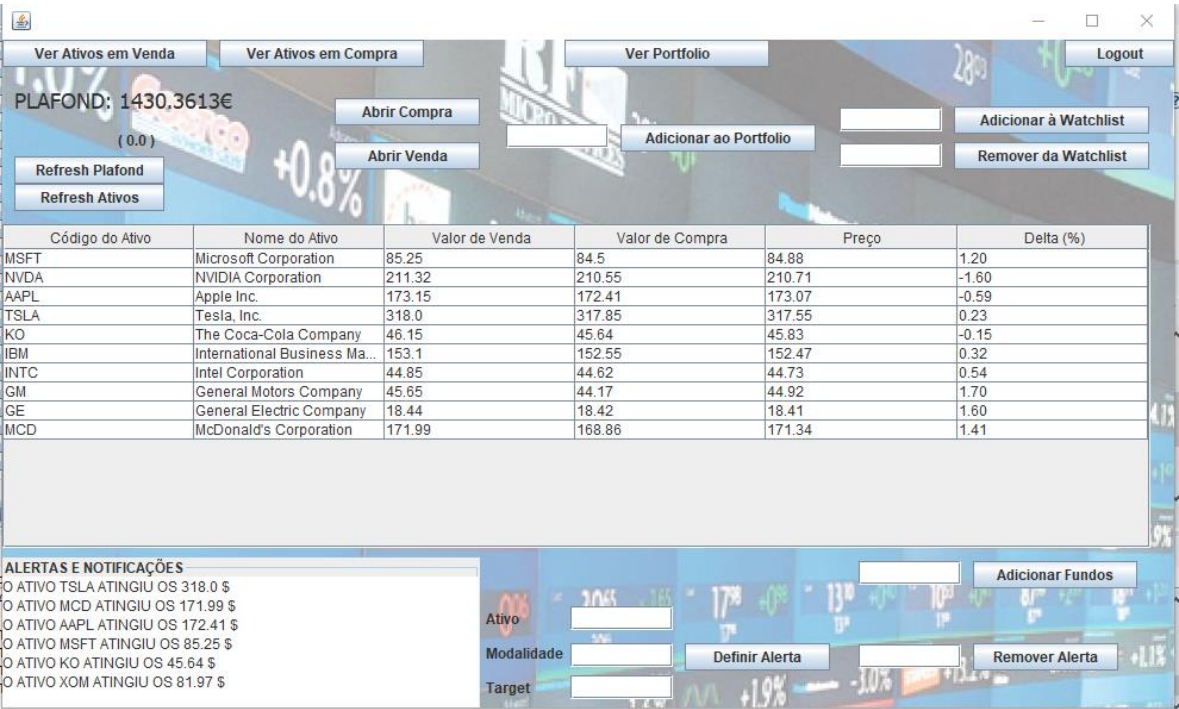


Figura 23 Menu Principal

Menu Visitante



Código do Ativo	Nome do Ativo	Valor de Venda	Valor de Compra	Preço	Delta (%)
MSFT	Microsoft Corporation	85.25	84.5	84.88	1.20
NVDA	NVIDIA Corporation	211.32	210.55	210.71	-1.60
AAPL	Apple Inc.	173.15	172.41	173.07	-0.59
TSLA	Tesla, Inc.	318.0	317.85	317.55	0.23
KO	The Coca-Cola Company	46.15	45.64	45.83	-0.15
IBM	International Business Ma...	153.1	152.55	152.47	0.32
INTC	Intel Corporation	44.85	44.62	44.73	0.54
GM	General Motors Company	45.65	44.17	44.92	1.70
GE	General Electric Company	18.44	18.42	18.41	1.60
MCD	McDonald's Corporation	171.99	168.86	171.34	1.41

Figura 24 Menu Visitante

Abrir Venda/Abrir Compra

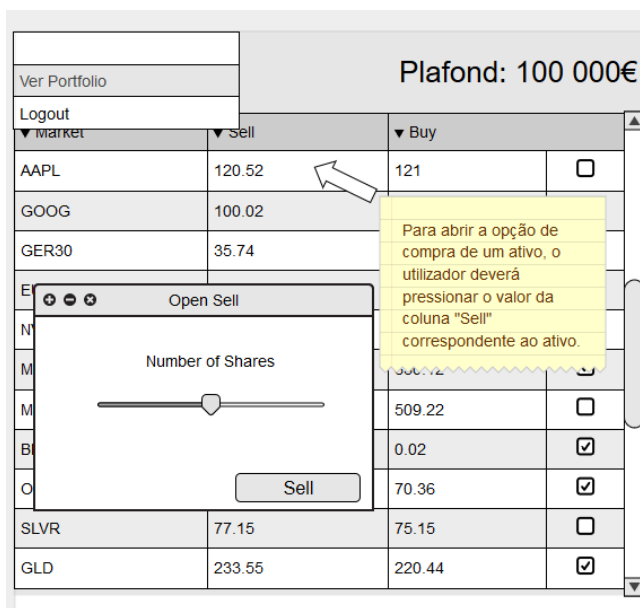


Figura 25 Mockup Abrir Venda

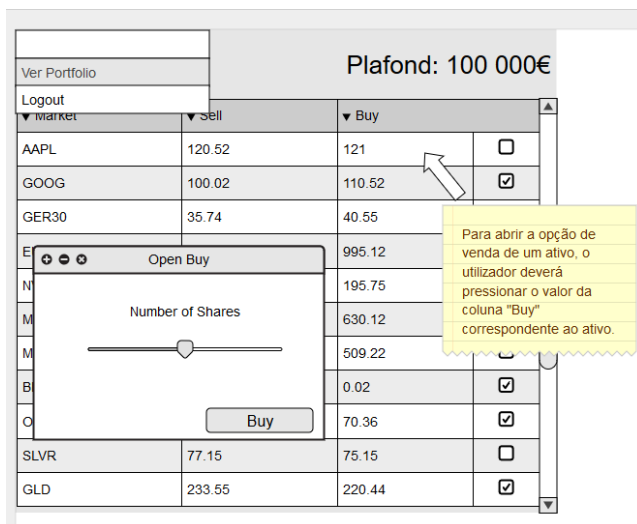


Figura 26 Mockup Abrir Compra

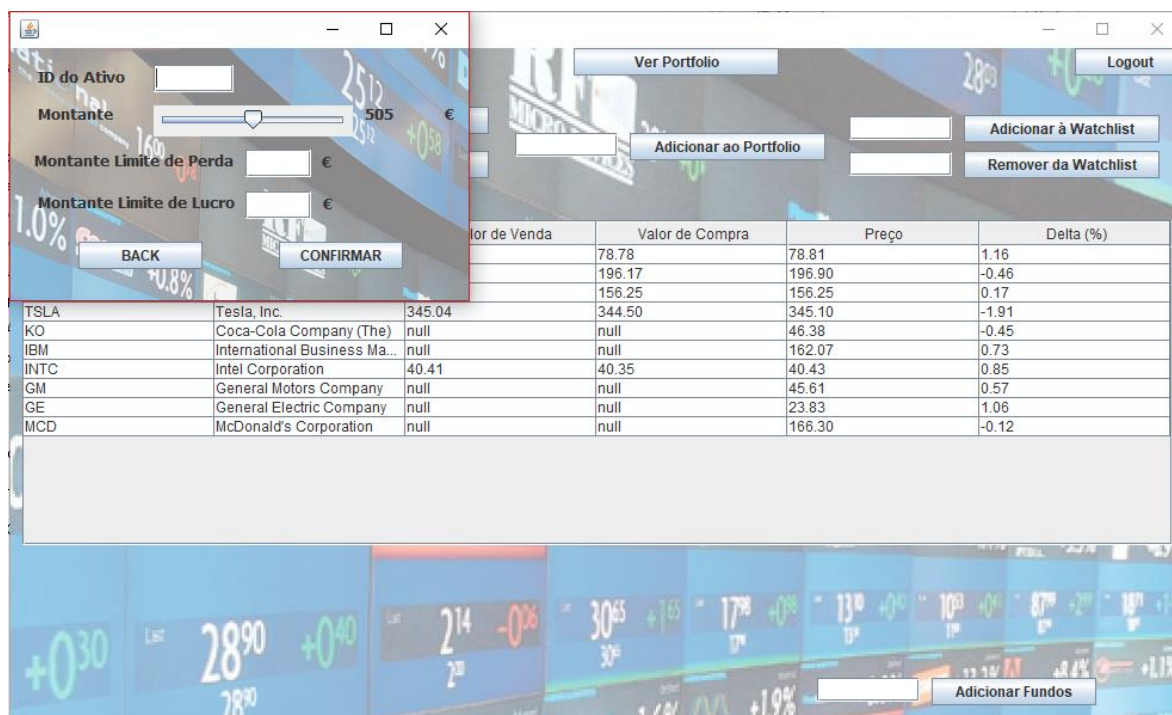


Figura 27 Abrir Compra/Abrir Venda

Ver Portfolio

Plafond: 100 000€			
Ver Portfolio			
Logout			
▼ Market	▼ Sell	▼ Buy	
AAPL	12		<input type="checkbox"/>
GOOG	10		<input checked="" type="checkbox"/>
GER30	35		<input type="checkbox"/>
EURUSD	1000.22	995.12	<input checked="" type="checkbox"/>
NVIDIA	200.56	195.75	<input type="checkbox"/>
MICRO	750.64	630.12	<input checked="" type="checkbox"/>
MERBZ	500.20	509.22	<input type="checkbox"/>
BES	0.00	0.02	<input checked="" type="checkbox"/>
OIL	64.32	70.36	<input checked="" type="checkbox"/>
SLVR	77.15	75.15	<input type="checkbox"/>
GLD	233.55	220.44	<input checked="" type="checkbox"/>

Figura 28 Mockup Ver Portfolio

Plafond: 100 000€			
▼ Market	▼ Sell	▼ Buy	
AAPL	120.52	121	<input checked="" type="checkbox"/>
GOOG	100.02	110.52	<input checked="" type="checkbox"/>
GER30	35.74	40.55	<input checked="" type="checkbox"/>

Figura 29 Mockup Ver Portfolio

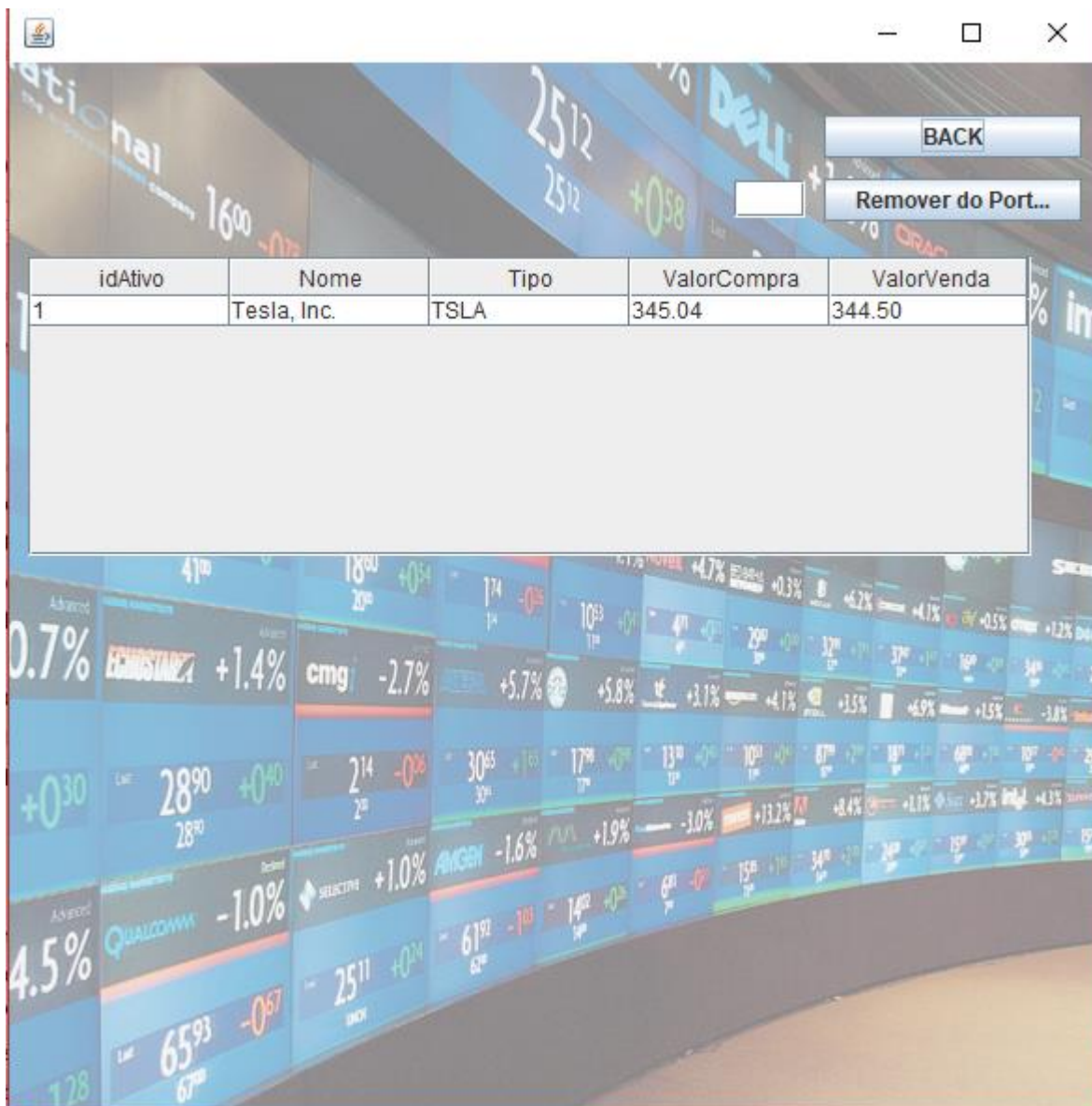


Figura 30 Ver Portfolio

Funcionalidades não implementadas

No relatório da fase anterior, foi colocada aqui neste tópico uma explicação para a não implementação do sistema de fecho automático de um contrato caso o mesmo ultrapassa-se os limites de lucro ou perda definidos pelo utilizador no momento da abertura do contrato.

Nesta segunda fase a minha intenção é desta feita informar o leitor e utilizador, que essa funcionalidade foi agora implementada e está pronta a ser usada em qualquer momento.