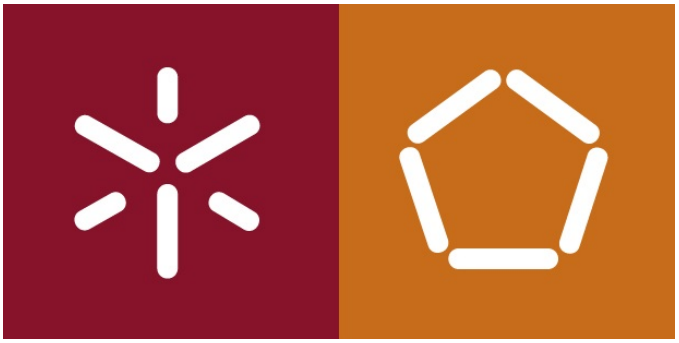
Universidade do Minho

Mestrado Integrado em Engenharia Informática -
Engenharia de Sistemas de Software



ESS Trading Platform

Relatório do Trabalho Prático 1 - Parte 1

Arquitetura de Software

Francisco Freitas

A81580

Pedro Freitas

A80975

December 12, 2019

Contents

1 Introdução

No âmbito da unidade curricular de Arquitetura de Software foi-nos proposto um trabalho que consistirá duas fases. Nesta primeira fase teremos de implementar uma solução a um problema de um enunciado, sem dar muita enfase nem ter muito cuidado com a arquitetura da solução. Isto para depois numa segunda fase podermos ter uma arquitetura cuidada e fundamentada com a matéria lecionada na Unidade Curricular.

Assim durante este relatório vamos explicar o enunciado, algumas decisões tomadas e mostrar o resultado obtido desta primeira fase de trabalho.

2 ESS Trading Platform

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.

Nesta plataforma baseamo-nos em contratos CFD (Contract For Differences), que é estabelecida entre duas partes : um "comprador"(long) e um "vendedor"(short). Existem dois tipos destes contratos que é definido quando se estabelece o contrato. Quando o utilizador acha que o valor atual de um ativo vai subir ele estabelece um **contrato de compra**, onde o lucro desse contrato é dado por: *ValorVendaFutura - ValorCompraAtual*. Quando o utilizar acha que um valor atual de um ativo vai descer estabece um **contrato de venda**. Neste tipo de contrato o lucro que vai obter é dado por *ValorVendaAtual - ValorCompraFutura*.

Esta negociação não implica que tenhamos o produto em causa. Por exemplo, podemos negociar uma ação de uma entidade sem de facto a termos.

Assim esta plataforma terá de seguir alguns requisitos pré-estabelecidos:

- A plataforma deverá ser responsável por manter os valores dos ativos a serem negociados via CFD's
- A plataforma deverá permitir a abertura de contas a investidores com plafond inicial para investimento.
- A plataforma deverá permitir que investidores abram posições (CFDs) sobre ativos disponíveis, quer seja de compra ou de venda. Também deverá ser possível definir valores para fecho de contrato : valores de *Take profit* e de *Stop Loss*.
- A plataforma deverá permitir aos investidores monitorizar em tempo real o seu portfolio de CFDs e para cada um visualizar o valor atual do ativo adquirido.

3 Principais Funcionalidades

O objetivo da plataforma é gestão de posições (CFD-Contract For Differences),isto é,abrir, fechar e gerir posições no mercado financeiro referentes a ativos,como por exemplo ações e commodities. Para esse efeito a plataforma tem que suportar as seguintes Funcionalidades.

- Registar Utilizadores
- Comprar e vender contratos
- Monitorizar em 'tempo real' do portfólio de CFDs de cada utilizador
- Visualizar o valor atual dos ativos adquiridos pelos Utilizadores

4 Atributos de qualidade

O sistema desenvolvido permite com alguma facilidade a modificação da interface gráfica do Utilizador.

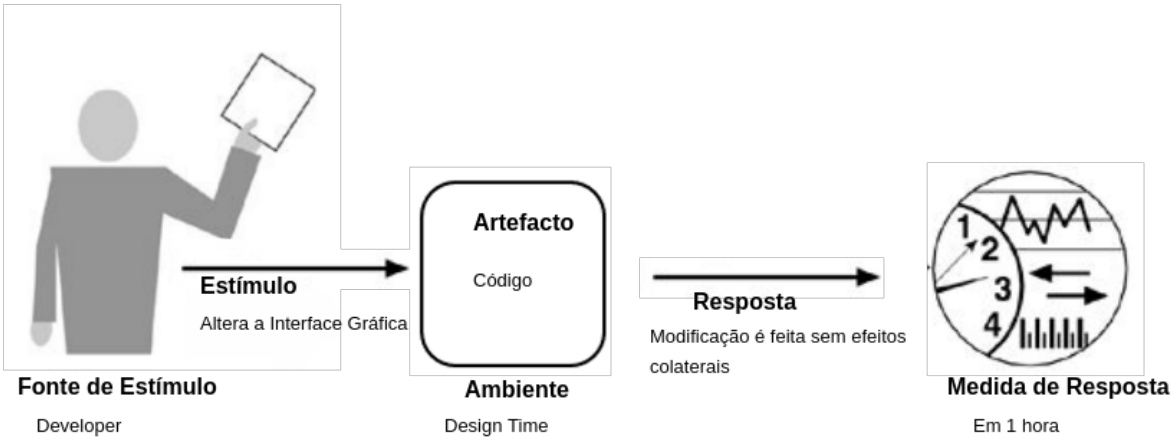


Figure 1: Cenário concreto de Facilidade de modificação

O sistema apresenta alto desempenho uma vez que utiliza multiplas Threads.

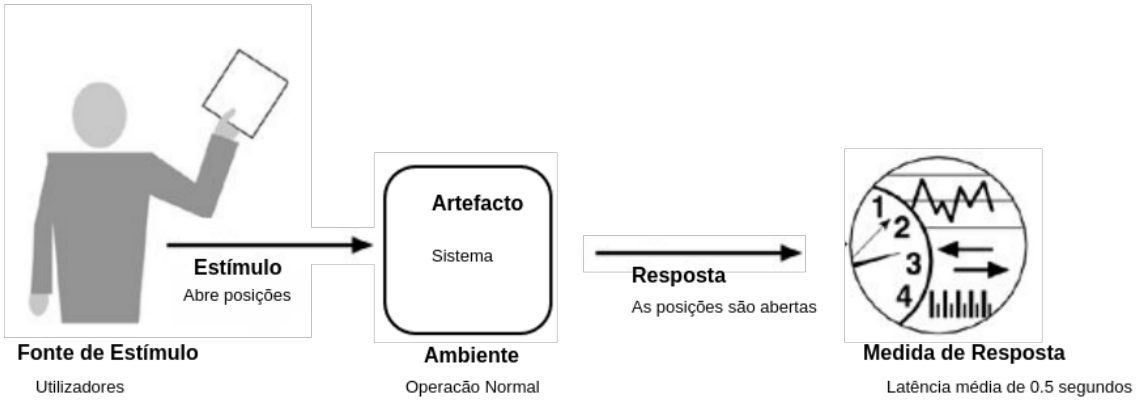


Figure 2: Cenário concreto de Desempenho

Através da autenticação o sistema confere proteção à conta do Utilizador e a todos os seus dados.

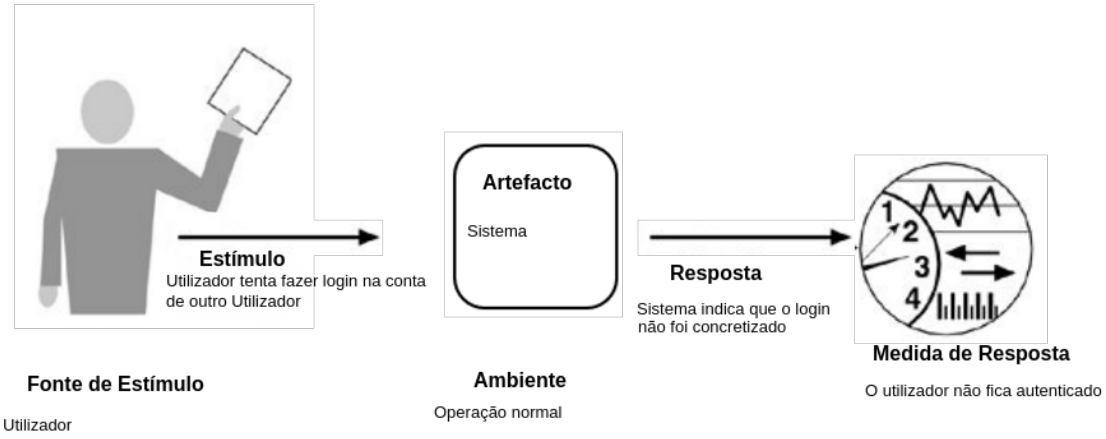


Figure 3: Cenário concreto de Segurança

5 Condicionantes

Condicionante	Descrição
Implementação em java	A implementação da plataforma será feita em java. Posto isto, existem restrições provenientes da línguagem.
Execução por terminal ou IDE	A plataforma não apresenta interface gráfica para o utilizador. Por este motivo será executada na linha de comandos do terminal ou num IDE que suporte Java
Imparcialidade	A plataforma terá que ser suportada nos diferentes sistemas operativos

6 Estratégia de Solução

Depois de uma análise detalhada do problema e da construção do modelo de domínio verificou-se que as entidades principais da ESS Ltd são o Utilizador, Contrato, Ativo e o Registo.

O Ativo contém as informações relevantes sobre as ações e *commodities* a serem negociadas que vão sendo atualizadas,em tempo real, com os dados recolhidos da API. Sempre que um utilizador comprar ou vender contratos vai-se ao Ativo buscar os valores de compra e venda deste e associa-se um destes ao contrato. Caso o utilizador queira consultar um valor passado do Ativo terá que aceder aos seus registos ou contratos referentes a esse ativo.

O Contrato é criado pela necessidade de guardar informações acerca das compras/vendas efetuadas sobre ativos por um utilizador. Por isso guarda consigo o identificador do utilizador e do ativo, bem como a quantidade de ativo e o valor pelo o qual este foi adquirido. Para conseguir distinguir contratos de venda e contratos de compra foi adicionado um booleano. Para que o utilizador consiga distinguir no seu portfólio contratos fechados e contratos abertos é seguida a mesma metodologia acrescentando um booleano (*true* se fechado, *false* se aberto). Nesta primeira versão quando um contrato é criado, o utilizador tem que definir sempre o Stop Loss e o Take Profit.

O Registo foi criado com intuito do utilizador poder ver os seus contratos fechados e ter uma noção da variação do valor de ativos já adquiridos. Para esse efeito o registo tem um identificador, o identificador do utilizador e o identificador do contrato.Para além disso contém o lucro/perda realizado com o fecho do contrato.

Depois de uma análise detalhada dos requisitos da aplicação, chegou-se á conclusão que é necessário ter uma conexão servidor-cliente. O cliente manda mensagens ao servidor (pedidos). Estas mensagens são interpretadas pelo servidor enviando também uma mensagem como resposta para o cliente. Desta forma é possível o serviço pode ser utilizado por vários clientes em simultâneo.

Para manter os valores dos ativos sempre atualizados foi criada uma classe que através de Threads vai fazendo alterações na base de dados no preço de compra e venda do ativo.

7 Modelação

Nesta secção vamos falar sobre a modelação e opções tomadas ao longo da realização do projeto.

7.1 Modelo de Domínio

O Modelo de Domínio captura as entidades relevantes para o sistema e a relação entre eles.

7.1.1 Diagrama de Modelo de Domínio

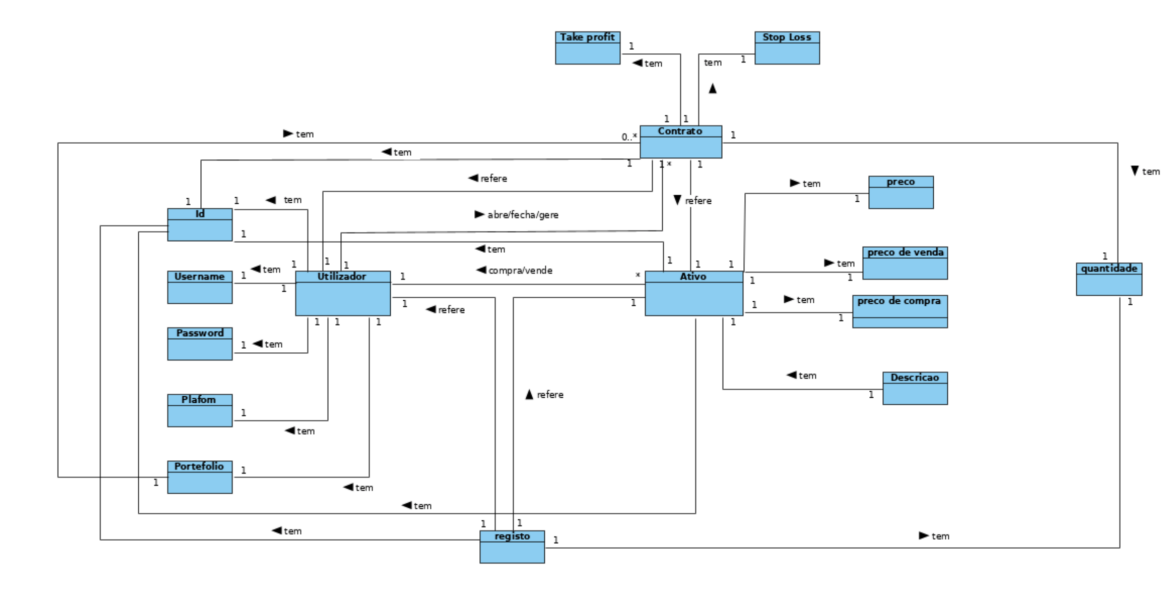


Figure 4: Modelo de Domínio

7.1.2 Entidades Relevantes

Ativo

Esta entidade representa um **ativo** em si. Sendo uma plataforma de Tradding, esta entidade vai ter um id(único) que diz respeito a um ativo, assim como preço de venda e de compra e ainda a descrição.

Utilizador

Esta entidade representa um utilizador (**trader**) da plataforma. Esta entidade também terá associado um id(único), um username e uma password, um planfom inicial e todo o portefólio de contratos efetuados.

Registo

Esta entidade representa o histórico de contratos fechados. Esta entidade refere-se às entidades envolvidas num contrato (um **utilizador** e um **ativo**). Além dos id's destas duas entidades, teremos também um id próprio, a quantidade e o lucro obtido.

Contrato

Por último temos a entidade **Contrato** que é o foco de toda a plataforma. Cada contrato refere-se à posição tomada de um **Utilizador** em relação a um **Ativo**. Esta posição é caracterizada pelo tipo de

contrato tomado (compra ou venda), assim como preco, lucro máximo e perda máxima, a quantidade e o estado da mesma.

7.2 Use Cases

Os Use Cases da nossa plataforma é uma forma de representarmos as principais funcionalidades que queremos implementar no sistema. Assim foi necessária uma análise ao enunciado do problema e depois de alguma pesquisa sobre o funcionamento geral de uma plataforma de Tradding foi possível construir este diagrama.

7.2.1 Diagrama de Modelo de Use Cases

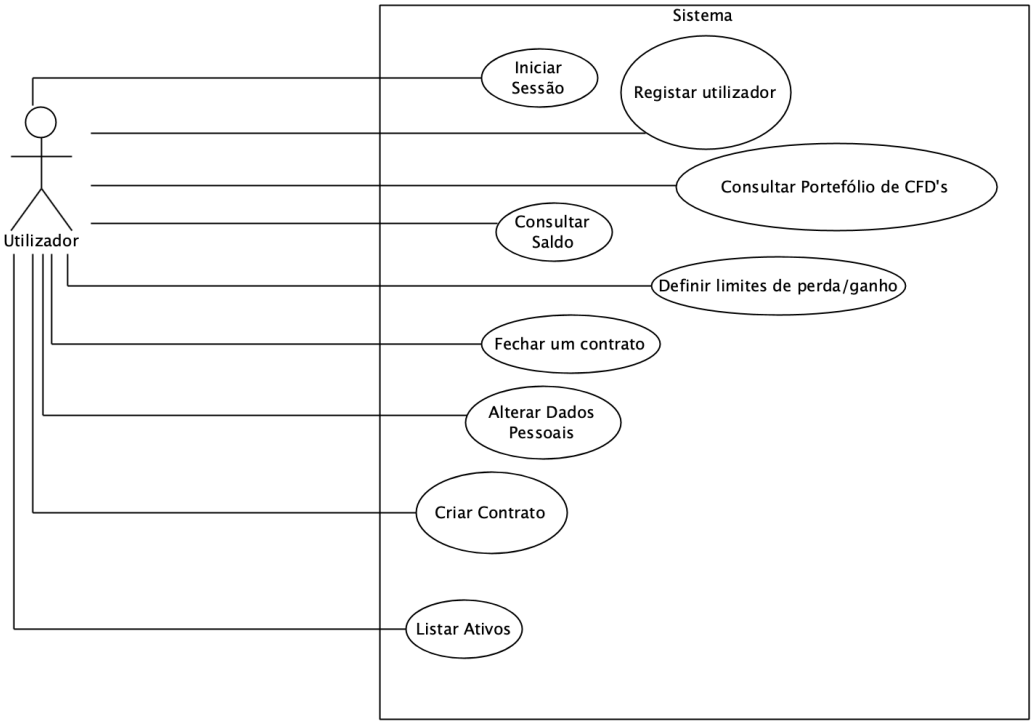


Figure 5: Diagrama de Modelo de Use Cases

7.3 Diagramas de Comportamento

Para representar o comportamento de algumas funcionalidades decidimos usar os diagramas de sequência mais detalhadamente(sub-sistemas e implementação).

7.3.1 Fechar Contrato

Esta funcionalidade representa quando o utilizador quer fechar um contrato, quer seja de venda ou de compra.

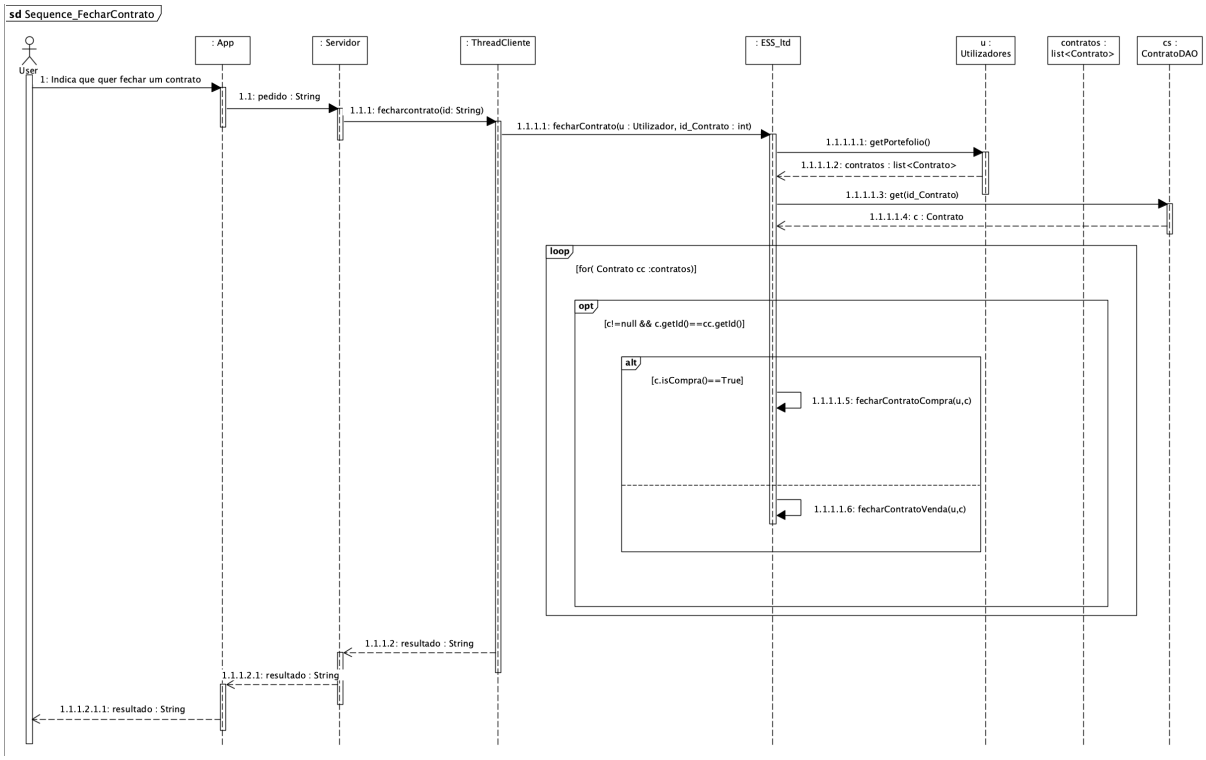


Figure 6: Diagrama de Sub-sistemas de Fechar Contrato

7.3.2 Criar Ativo

Embora esta funcionalidade seja executada em back-end sem que o utilizador tenha controlo sobre ela, achamos por bem explicar e mostrar como essa funcionalidade funciona. Assim representamo-la da seguinte maneira:

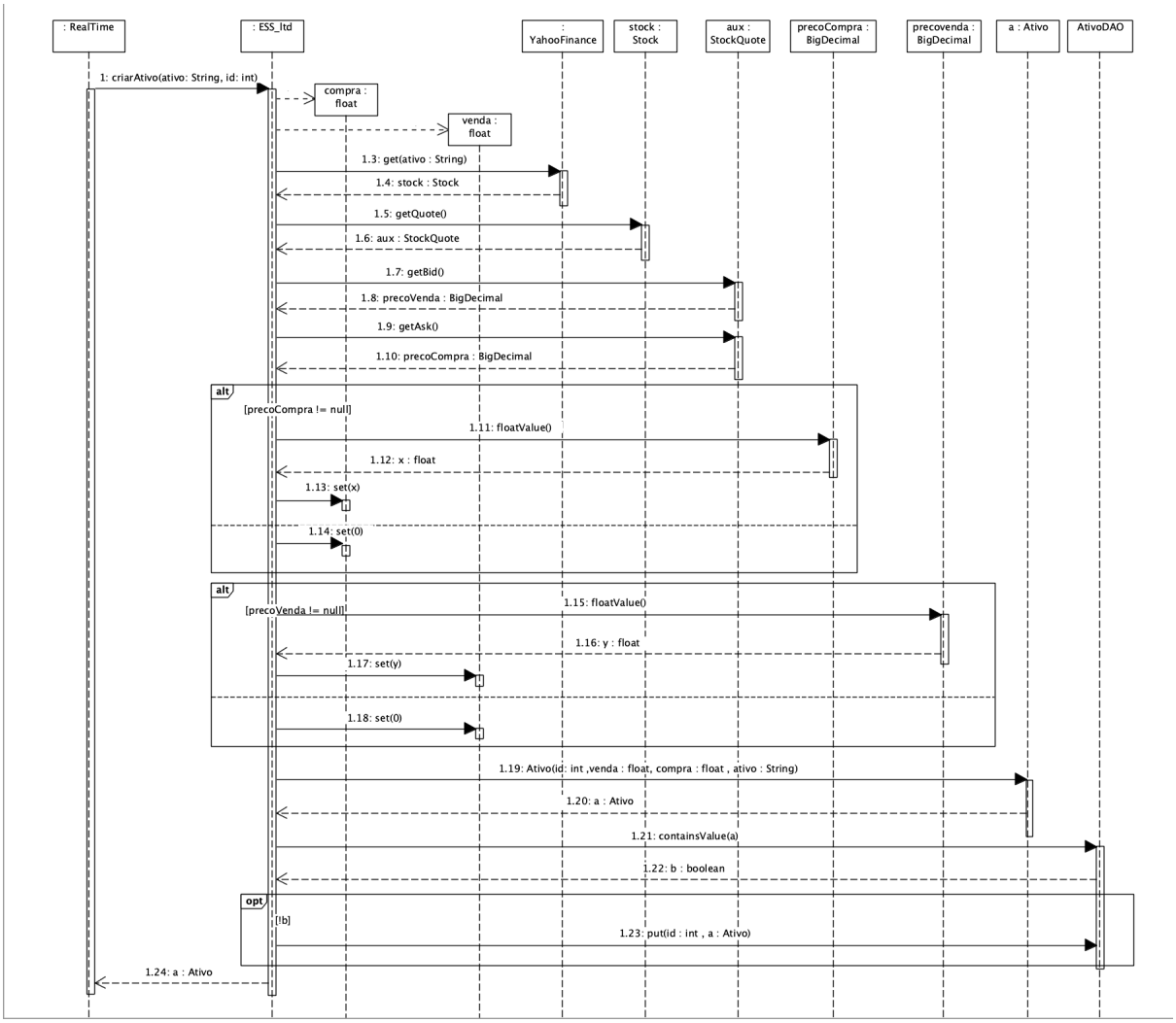


Figure 7: Diagrama de Implementação Criar Ativo

7.3.3 Listar Portefólio

Esta funcionalidade diz respeito a um utilizador que pretenda ver todo o seu portefólio de contratos feitos (quer seja fechado ou ainda em aberto):

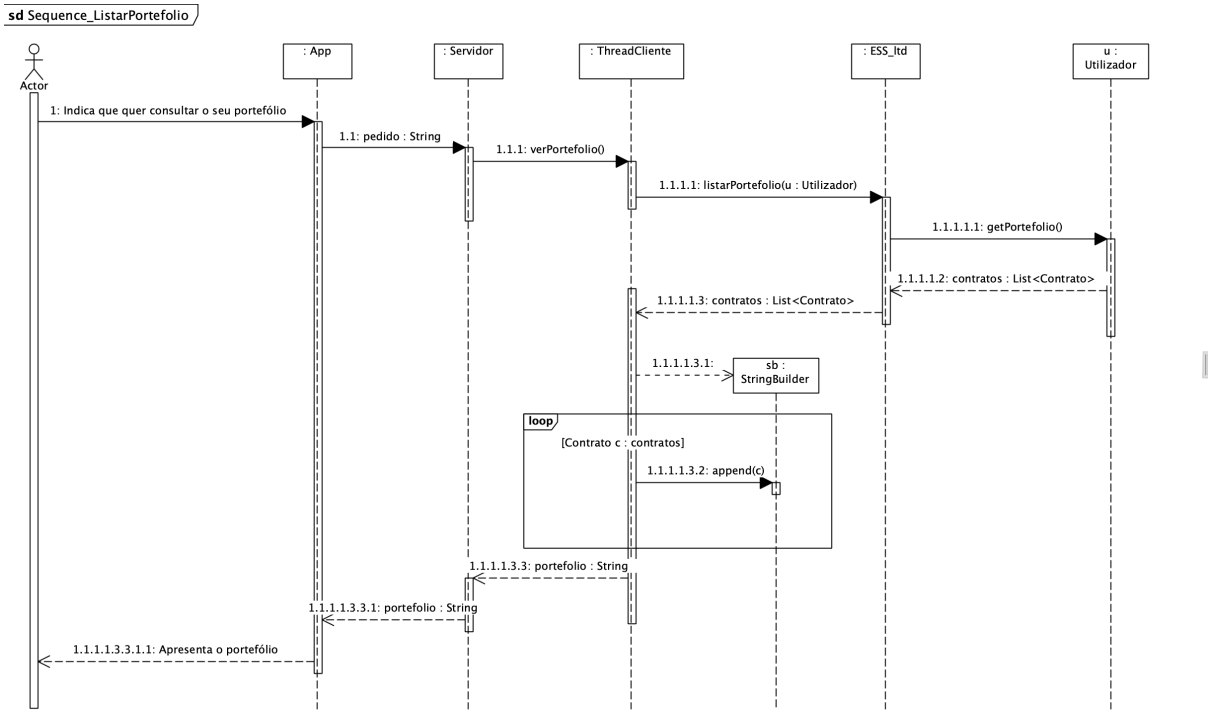


Figure 8: Diagrama de Implementação Listar Portefólio

7.4 Diagrama de Classes

Depois de termos refletido sobre o problema construímos o Diagrama de Classes da seguinte maneira:

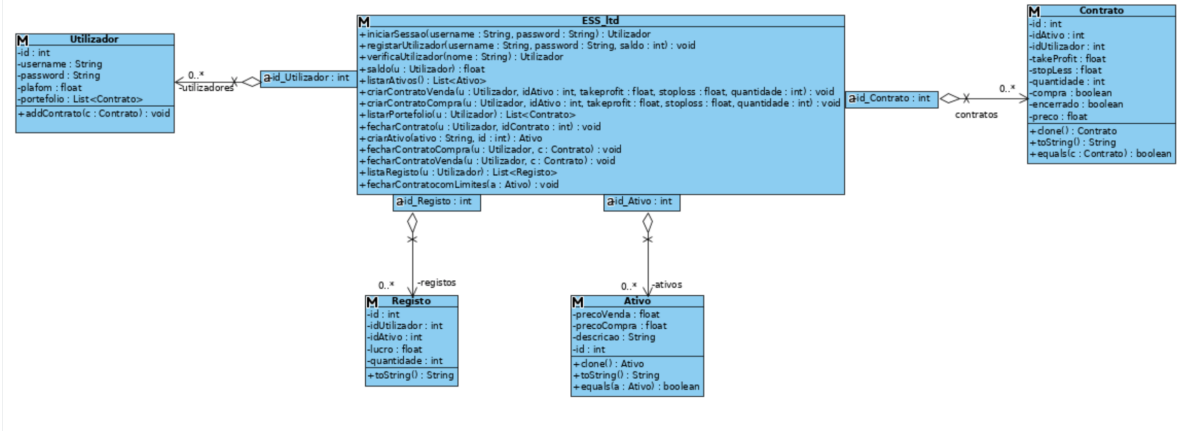


Figure 9: Diagrama de Classes

7.5 Diagrama de ORM

A pensar na implementação da solução , construímos o diagrama de ORM para mostrar como a base de dados (DAO's) interege no sistema.

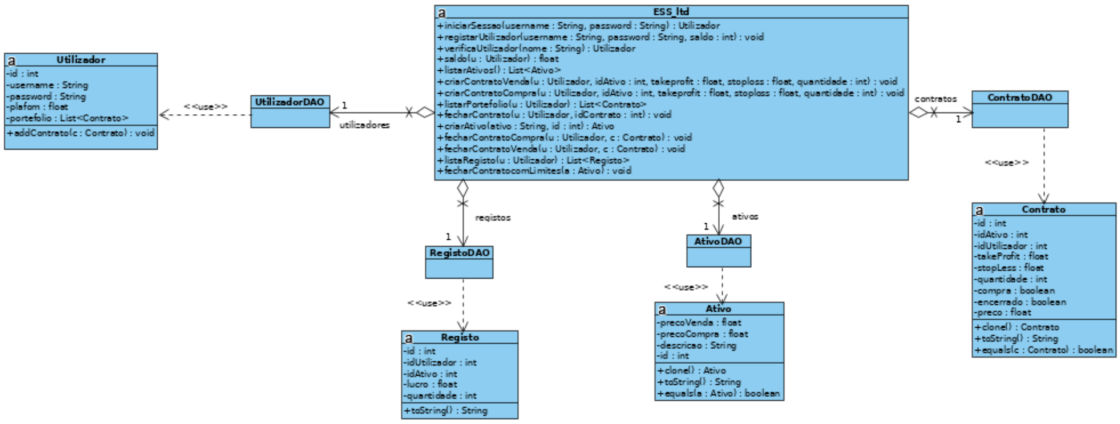


Figure 10: Diagrama de ORM

7.6 Diagrama de Packages

O diagrama de packages representa todos os pacotes (packages) necessários para o sistema e como interagem.

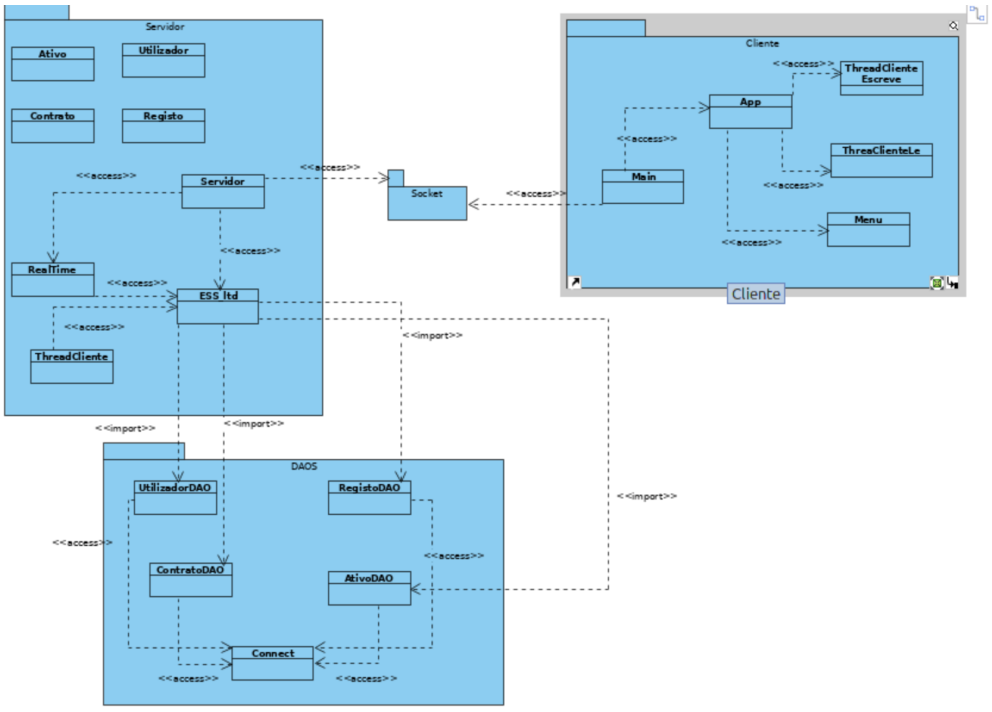


Figure 11: Diagrama de Packages

7.7 Diagrama de Instalação

Neste diagrama vemos o que é necessário para a aplicação correr corretamente na nossa máquina.

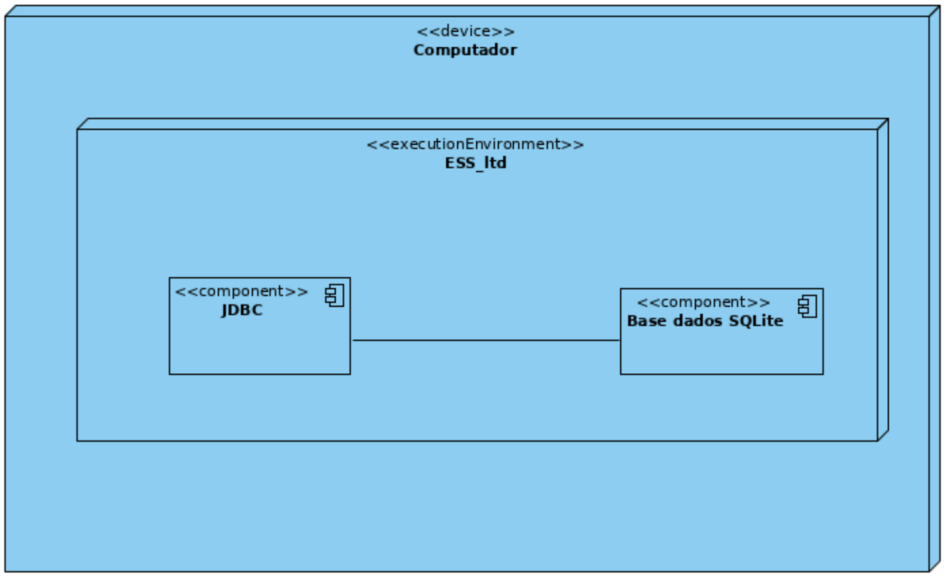


Figure 12: Diagrama de Instalação

8 Interface

Nesta primeira fase temos uma interface bastante simples e baseada no terminal. Para tal é necessário correrrmos o servidor num processo de terminal e o(os) clientes(clientes) noutro(s).

Quando correrrmos a aplicação deparamo-nos com este menu:

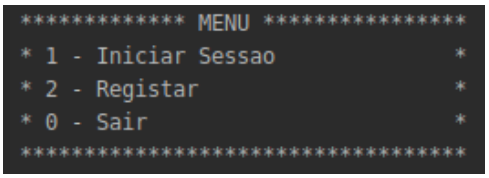


Figure 13: Menu inicial

Depois do login ser efetuado temos um menu principal que é onde podemos usufruir de todas as funcionalidades implementadas:

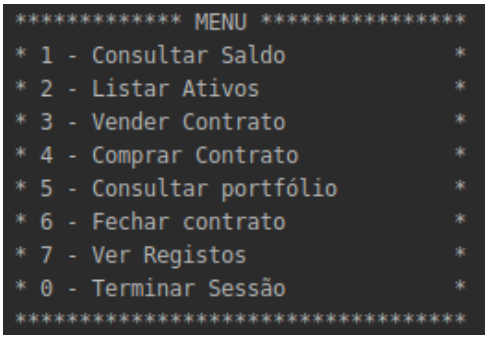


Figure 14: Menu principal

9 Conclusão

Nesta primeira fase além de fazermos o planeamento e a modelação da aplicação acabamos também por fazer uma implementação do código já fidedigna e funcional.

Na próxima fase esperamos que seja necessário pequenas alterações e adaptações do código para ir de encontro à nova arquitetura.

Assim concluímos que esta primeira fase foi concluída com sucesso e que estamos prontos e motivados para enfrentar-mos a fase seguinte.