**Mestrado em**

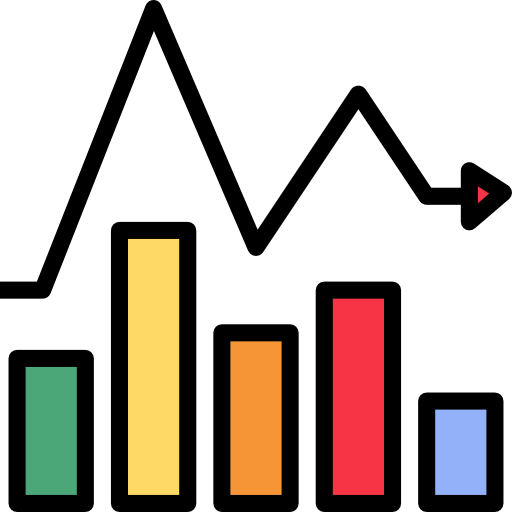
**Engenharia Informática**

**1º ano | 1º Semestre**

**Arquiteturas de Software**

**Relatório de Projeto 1º parte**

**(entrega intermédia)**

****

Francisco Saraiva | PG39287

Ano Letivo 2019/2020

Braga, 24 de Outubro 2019

**Conteúdos**

[**Introdução** 2](#_Toc22813927)

[**1.** **Contexto do problema** 3](#_Toc22813928)

[**2.** **Tecnologias** 4](#_Toc22813929)

[**3.** **Análise** 5](#_Toc22813930)

[**3.1.** **Domínio** 5](#_Toc22813931)

[**3.2.** **Arquitetura** 9](#_Toc22813932)

[**3.3.** **Cenários de atributos de qualidade** 14](#_Toc22813933)

[**4.** **Implementações futuras** 16](#_Toc22813934)

**Ilustrações**

[**Figura 1- Diagrama de Domínio 5**](#_Toc22813939)

[**Figura 2- Diagrama de Use Cases 6**](#_Toc22813940)

[**Figura 3 - Mockups da aplicação 7**](#_Toc22813941)

[**Figura 4- Diagrama de Navegação 8**](#_Toc22813942)

[**Figura 5- Diagrama de Classes 9**](file:///C:\Users\PC\Documents\UMinho\Arquitetura%20de%20Software\Projeto\1fase\projeto\Report.docx#_Toc22813943)

[**Figura 6- Modelo Relacional 10**](file:///C:\Users\PC\Documents\UMinho\Arquitetura%20de%20Software\Projeto\1fase\projeto\Report.docx#_Toc22813944)

[**Figura 7- Diagrama de Comportamento - Criar Long CFD 11**](#_Toc22813945)

[**Figura 8- Diagrama de Comportamenot – Fechar CFD Manualmente 12**](#_Toc22813946)

[**Figura 9 – Diagrama de Comportamento – Fecho CFD Automático 13**](#_Toc22813947)

[**Figura 10 - Cenário Performance 14**](#_Toc22813948)

[**Figura 11 - Cenário Disponibilidade 15**](#_Toc22813949)

[**Figura 12 - Cenário Confiabilidade 15**](#_Toc22813950)

# **Introdução**

Este documento serve de suporte e detalha a análise efetuada e os aspetos técnicos de implementação da aplicação web **ESS Trading Platform**, para a componente curricular de projeto da cadeira de arquiteturas de software.

Nesta entrega intermédia foi proposta arranjar uma solução para o problema de oferecer uma plataforma de negociação para permitir a investidores ou traders de gerir posições no mercado financeiro por CFDs (*Contract For Differences*). Elaboração passou por uma análise e levantamento de funcionalidades para a plataforma, desenvolvimento de mockups e visualização das várias funcionalidades, identificar 3 a 4 funcionalidades principais, atributos de qualidade e condicionantes e a elaboração de diagramas para representar a estrutura da aplicação.

# **Contexto do problema**

A ESS Ltd, pretende criar um produto para negociação de CFDs através de uma plataforma de negociação onde utilizadores (investidores e traders) abrem, fecham e gerem posições no mercado financeiro onde podem de vários ativos financeiros escolher para ingressar na compra e venda de CFDs, dos quais acções, *commodities* como ouro e petróleo, índices ou até moedas.

Estas plataformas geralmente possuem corretores financeiros que tratam da transação, gratuitamente ou com uma taxa aplicável a um número de posições mínimas por mês. Nesta plataforma pretende-se puder aceder a uma lista de variados ativos financeiros e dar a possibilidade ao utilizador de comprar ou vender ativos atráves de contratos de diferença, denominados de CFDs. Numa posição em que o utilizador pretende **comprar** um ativo (pelo preço base + margem de corretor) quando sobe, irá ser criado um CFD de **long** com intuito de vir a lucrar com a subida de preço do ativo, pelo inverso quando um ativo vir a descer de preço o utilizador irá querer lucrar **vendendo** o ativo com um CFD de **short**. Em CFDs o intuito de lucrar passa pela diferença do preço base em iniciou o contrato com a oscilação do preço do ativo no mercado, em que quanto maior a diferença maior o lucro. Os utilizadores podem também num CFD escolher uma quantidade para ter maior aproveitamento do investimento no ativo.

Os requisitos/funcionalidades propostas do projeto são:

* Manter valores dos ativos a ser negociados via CFDs;
* Registo de contas na plataforma com um *plafond* inicial;
* Utilizadores podem abrir CFDs sobre ativos disponíveis, de compra e venda;
* CFDs estão ligados a serem definidos com limites de ganho e de perda estipulados pelos utilizadores (*Take profit* e *Stop loss*);
* CFDs podem ser fechados a qualquer momento por acção dos utilizadores;
* Utilizadores podem monitorizar em tempo real os seus portfolios de CFDs e visualizar o valor atual dos ativos adquiridos

Com base na análise foi escolhido desenvolver este projeto numa plataforma web para dar aos utilizadores as funcionalidades pretendidas.

# **Tecnologias**

Para o desenvolvimento deste projeto, foi decidido criar uma plataforma web em **Node.js** com o intuito de oferecer as funcionalidades de gestão de CFDs. A linguagem de programação utilizada é **Typescript**, uma linguagem open source da família do **Javascript**, que oferece um paradigma orientado a objetos, com classes e tipagem estrita nos atributos das entidades ao contrário de Javascript.

Sendo o projeto desenvolvido em Node.js há vários pacotes necessários ao funcionamento do mesmo que foram incorporados para a solução sendo os principais:

* **Typescript**, para incorporar a linguagem em no esquema Node;
* **Pug**, template engine para as views html do website;
* **Sqlite3**, para a persistência dos dados do website
* **TypeORM**, para fácil manipulação e comunicação com a base de dados através dos modelos;
* **Express.js**, para arquitetura e estrutura do servidor e gerenciamento das rotas REST para os vários pedidos do website

Outras tecnologias auxiliares são **Boostrap** para funcionalidades e estilos do frontend do website e **AXIOS** para auxiliar na comunicação entre o cliente e servidor.

# **Análise**

# **Domínio**

Primeiro passo para analisar o problema é tirar dos requisitos e funcionalidades já colocadas as várias entidades que pertencem no domínio. Temos **Utilizadores**, que criam **CFDs** de **Compra** ou de **Venda** entre **Ativos Financeiros** dos quais são de vários **Tipos de Ativos**.

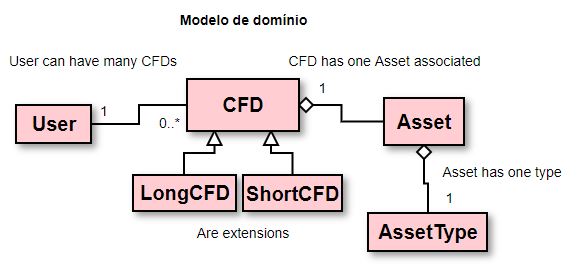


Figura 1- Diagrama de Domínio

A plataforma irá ser composta por utilizadores que interagem com os ativos criando CFDs, assim um utilizador pode criar vários contratos de diferença, que por si estão associados a um Ativo, que podem ser de vários tipos (acções, ouro, moedas…). Dentro dos CFDs existem dois tipos, o Long e o Short que são generalizações do CFD.

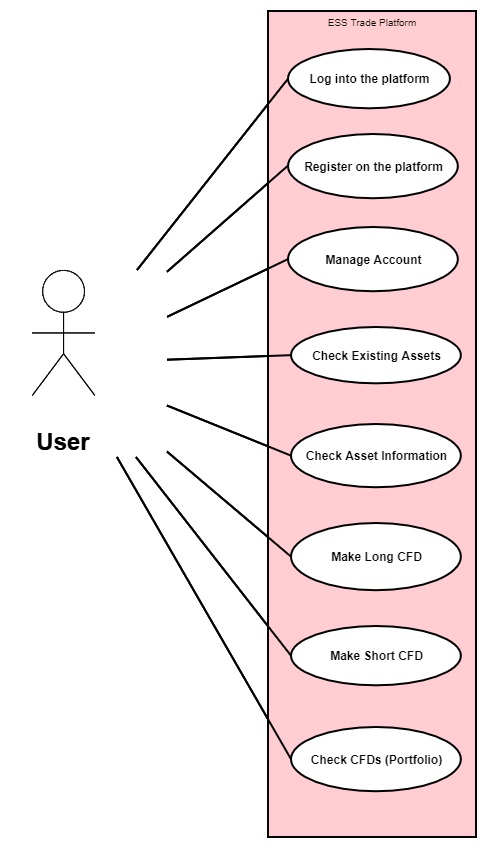


Figura 2- Diagrama de Use Cases

As funcionalidades passam por um sistema de autenticação que faz o login e o registo de novos utilizadores, verificação de informação e dos dados dos vários ativos existentes e a gestão dos CFDs do utilizador comprando e vendendo ativos e seguir o progresso dos contratos no portfolio.

Organizando com as funcionalidades foi desenvolvido os seguintes mockups para a ideia do aspeto e do funcionamento dos requisitos da aplicação, com algumas páginas com várias funcionalidades na mesma.

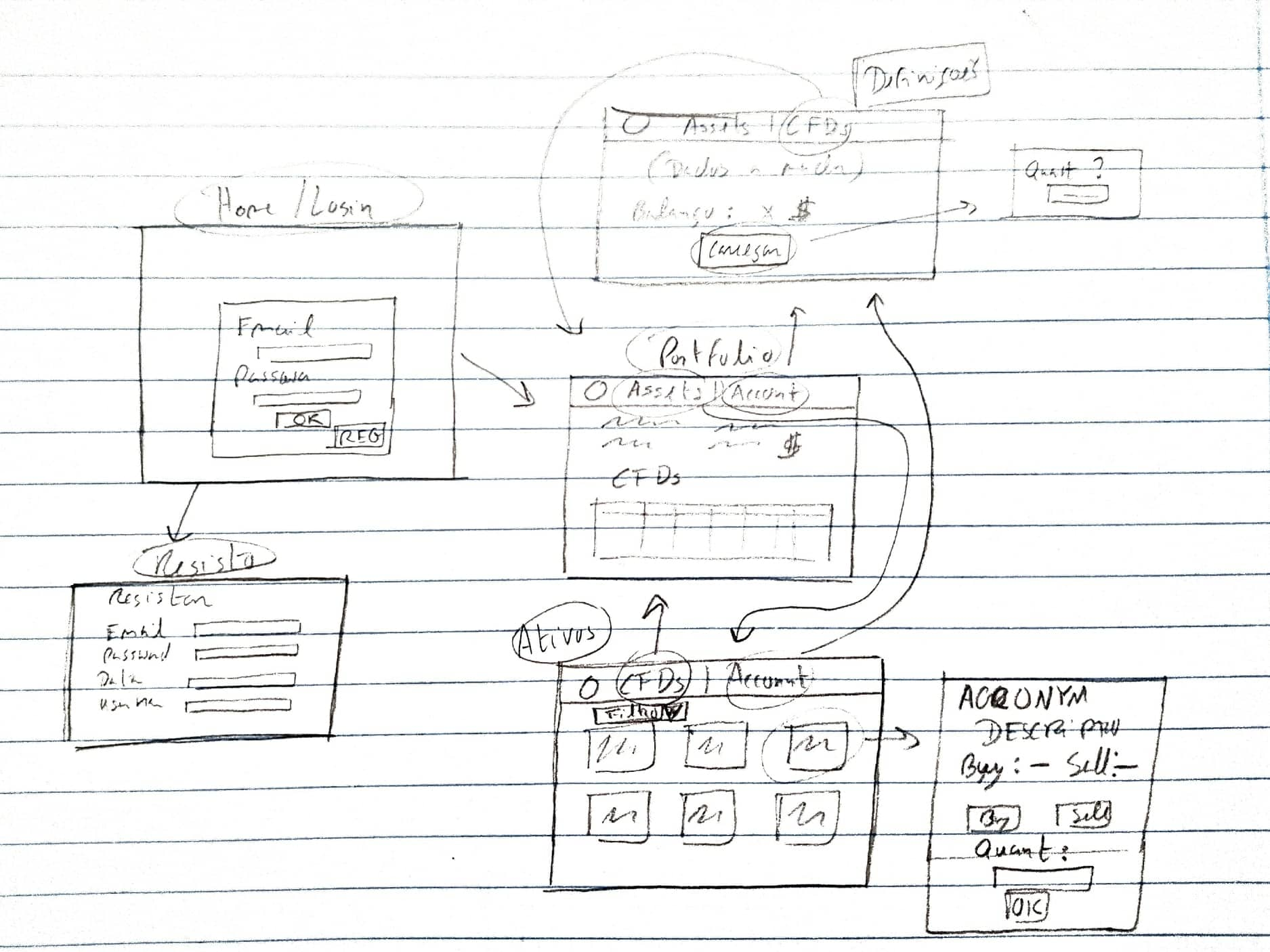


Figura 3 - Mockups da aplicação

O seguinte diagrama de navegação apresenta o mapa do website e dos caminhos de navegação.

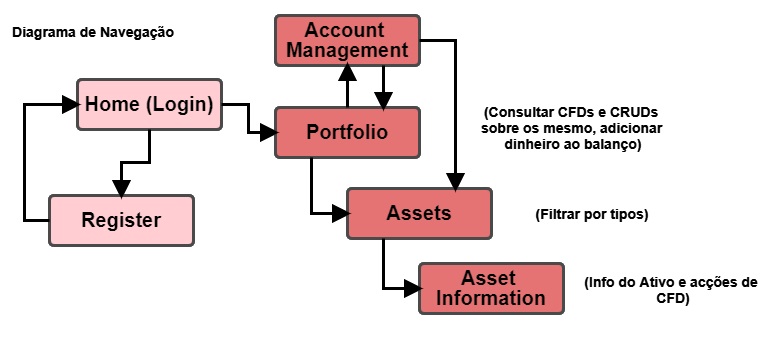


Figura 4- Diagrama de Navegação

# **Arquitetura**

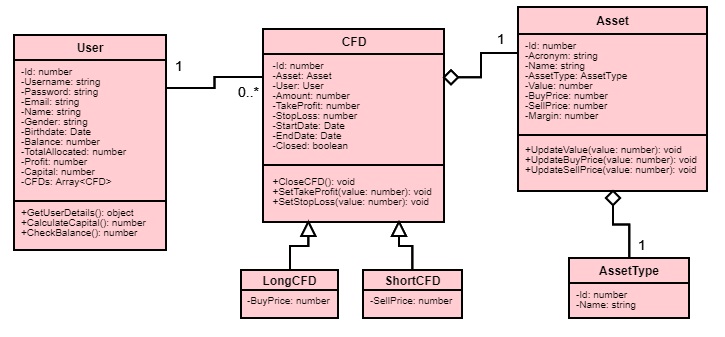
Com base no domínio do modelo temos o seguinte diagrama de classes com as seguintes propriedades adaptado a Typescript.

Figura 5- Diagrama de Classes

O utilizador é composto por atributos gerais de identificação como o username, nome verdadeiro email, data de nascimento e outros. Ligados à plataforma tem o **Balance** ligado ao dinheiro atual que tem na plataforma, o **TotalAllocated** referente ao total de dinheiro colocado nos seus CFDs, o **Profit** o dinheiro faturado com os CFDs, **Capital** o somatório de todos os outros campos e **CFDs** onde se guarda numa lista de Arrays do tipo CFD.

CFD por si é constítuido pelo **Asset** ativo a que se refere no contrato, **User** o user referente que o contrato foi criado, **Amount** a quantidade do ativo adquirida neste contrato, **TakeProfit** e **StopLoss** são os atributos de limites para fecho automático dos CFDs, **StartDate** e **EndDate** referem-se ao ínicio e fim do contrato e **Closed** indica se foi ou não fechado já o CFD. Temos também as generalizações LongCFD e ShortCFD, referente à compra e venda dos ativos, respetivamente. Ambas herdam da classe mãe CFD mas com o campo **BuyPrice** e **SellPrice** respetivamente a indicar o preço por unidade na abertura do contrato.

Por fim o ativo representado por Asset, composto por **Acronym** o acrónimo identificador único do ativo, o **Name** que descreve o Ativo, **AssetType** referente ao tipo de ativo, **Value** representa o valor do Ativo no mercado de momento, com **BuyPrice** e **SellPrice** sendo o valor por unidade respetiva para os CFDs.

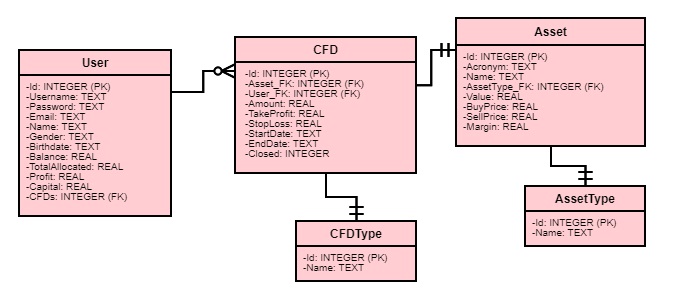
Adaptando o diagrama de classes a um modelo relacional virado a SQLite.

Figura 6- Modelo Relacional

Como a comunicação da aplicação se irá passar entre cliente e servidor, é de definir uma REST API para os vários pedidos efetuados para as ações na plataforma. Nomeadamente há o roteador, que tratará por encaminhar com base na entidade a cada controlador respetivo, significando cada entidade tem um controlador.

Foram criados também três diagramas de comportamento para três funcionalidades da plataforma:

* Criar um Long CFD
* Fecho de um CFD manual
* Fecho de um CFD automaticamente

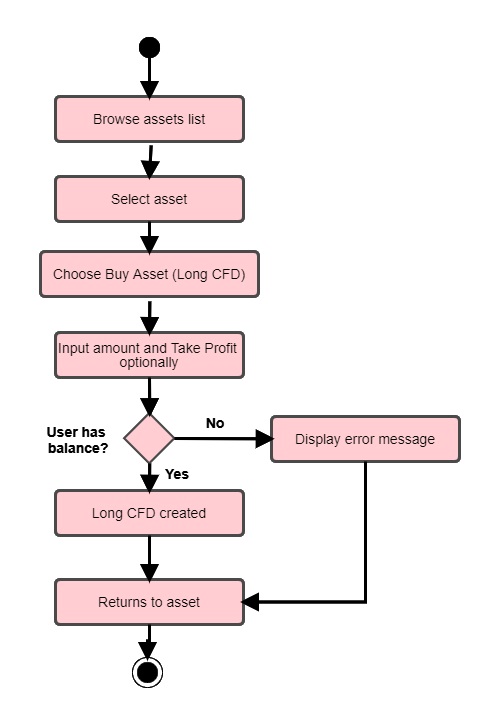


Figura 7- Diagrama de Comportamento - Criar Long CFD

Neste fluxo vemos a ordem porque o utilizador irá criar um CFD Long, em que a condição decisora aqui é se o utilizador dispõe na conta da quantia a pagar para criar o contrato. O utilizador da lista de ativos, basta selecionar sobre um para abrir o menu de compra ou venda do ativo.

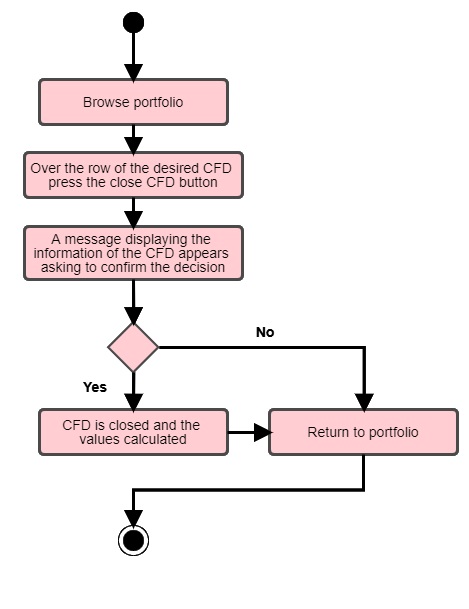


Figura 8- Diagrama de Comportamenot – Fechar CFD Manualmente

Neste fluxo vemos como o utilizador pode manualmente fechar um CFD quando quiser, através da view do seu portfolio/homepage pode diretamente da tabela de CFDs seus clicar sobre a opção de fecho do CFD, onde é lhe apresentada uma caixa de informação sobre o CFD e a confirmar a ação do utilizador.

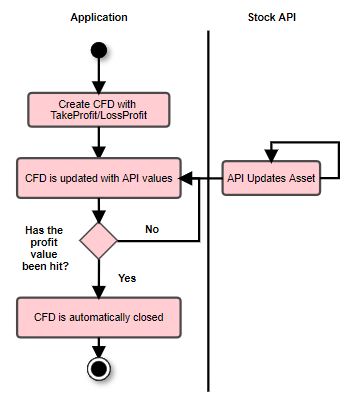


Figura 9 – Diagrama de Comportamento – Fecho CFD Automático

Neste fluxo vemos o comportamento do fecho automático do CFD após atingir o limite imposto pelo utilizador à criação do CFD, onde está dependente da atualização do ativo via API exterior, em que com os cálculos efetuados a cada mudança verifica se a meta é atingida e automaticamente fecha o CFD.

# **Cenários de atributos de qualidade**

Foram especificados 3 cenários de atributos de qualidade para a aplicação para assegurar o funcionamento dos requisitos ou o produto em geral.

Três tipos escolhidos foram:

* Performance
* Disponibilidade
* Confiabilidade

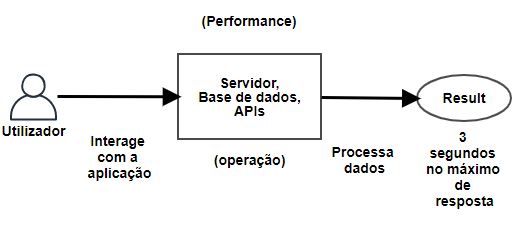


Figura 10 - Cenário Performance

Neste exemplo temos um caso normal do utilizador interagir com a aplicação e haver comunicações a vários níveis como de cliente para servidor, servidor para base de dados, ou servidor API, o que leva a tempos de espera e processamento de dados contínuo, num cenário correto a aplicação não pode demorar mais de 3 segundos a responder a um pedido.

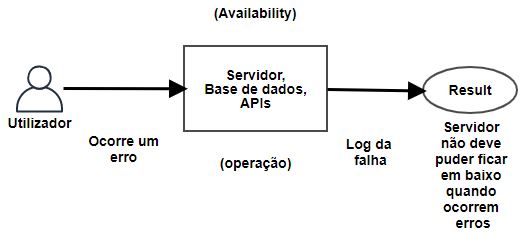


Figura 11 - Cenário Disponibilidade

No seguinte exemplo, temos um cenário onde pode ocorrer um erro tanto internamente na resolução de um pedido num bug ou a aceder à base de dados, a aplicação não pode ir abaixo e apenas reportar o erro ou fazer um log do mesmo para que se possa resolver os potenciais bugs.

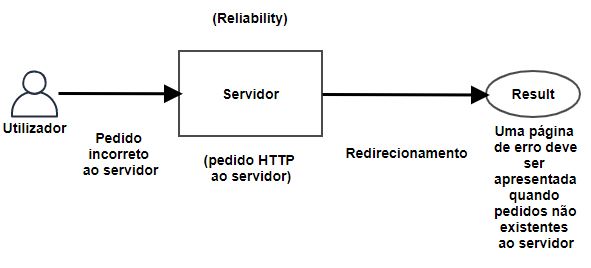


Figura 12 - Cenário Confiabilidade

Neste cenário temos um caso para utilizadores que testem a aplicação ou até mesmo algum utilizador incluí algo no pedido HTTP ao servidor que não existe, sem tratamento isto pode causar o servidor a ir abaixo, isto não pode acontecer me que no caso de ser detetada uma rota não existente é suposto redirecionar para uma página de erro automaticamente.

# **Implementações futuras**

Esta entrega intermédia ainda apresenta uma análise inicial do problema que ainda está suscetível a muitas mudanças ainda na próxima entrega e na 2º fase do projeto.

Quanto a implementações ainda não presentes ou esclarecidas, à a separação dos modelos com os controllers, roteadores para os pedidos e direcionamento dos controladores corretos às entidades, as rotas da API para os pedidos e padrões de desenho, que deverão ser implementados ao longo do desenvolvimento do projeto e alterado nos diagramas necessários do projeto.