

Relatório

Exposição e Explicação

Diogo Sá Dias – 1161605

Duarte Dias - 1190539

ESINF | 2021-22

Índice

[Introdução 2](#_Toc89093577)

[Sprint 1 3](#_Toc89093578)

[Diagrama de Classes 3](#_Toc89093579)

[Análise de Complexidade 3](#_Toc89093580)

[Sprint 2 4](#_Toc89093581)

[Diagrama de Classes 4](#_Toc89093582)

[Análise de Complexidade 4](#_Toc89093583)

[[US201] As a Port manager, I which to import ports from a text file and create a 2D-tree with port locations. 4](#_Toc89093584)

[[US202] As a Traffic manager, I which to find the closest port of a ship given its CallSign, on a certain DateTime. 5](#_Toc89093585)

# Introdução

Breve explicação do Diagrama de Classes:

Inicialmente, foi pensado fazer vários controllers e cada um dos mesmos iria implementar uma US. Posteriormente optamos por fazer um só controller que designa-mos de MainController, no qual esse fará a função de todos os outros controllers anteriormente mencionados.

Basicamente, é importado através da classe Import, um ficheiro csv, no qual possui todas as informações dos navios. Esses navios que são guardados numa árvore ShipTree que possui os navios. Cada navio tem uma árvore de movimentos que guarda todos os atributo dinâmicos do navio, ou seja, os relacionados com a posição do navio.

Complexidade das US’s: varia desde logn até n ao quadrado

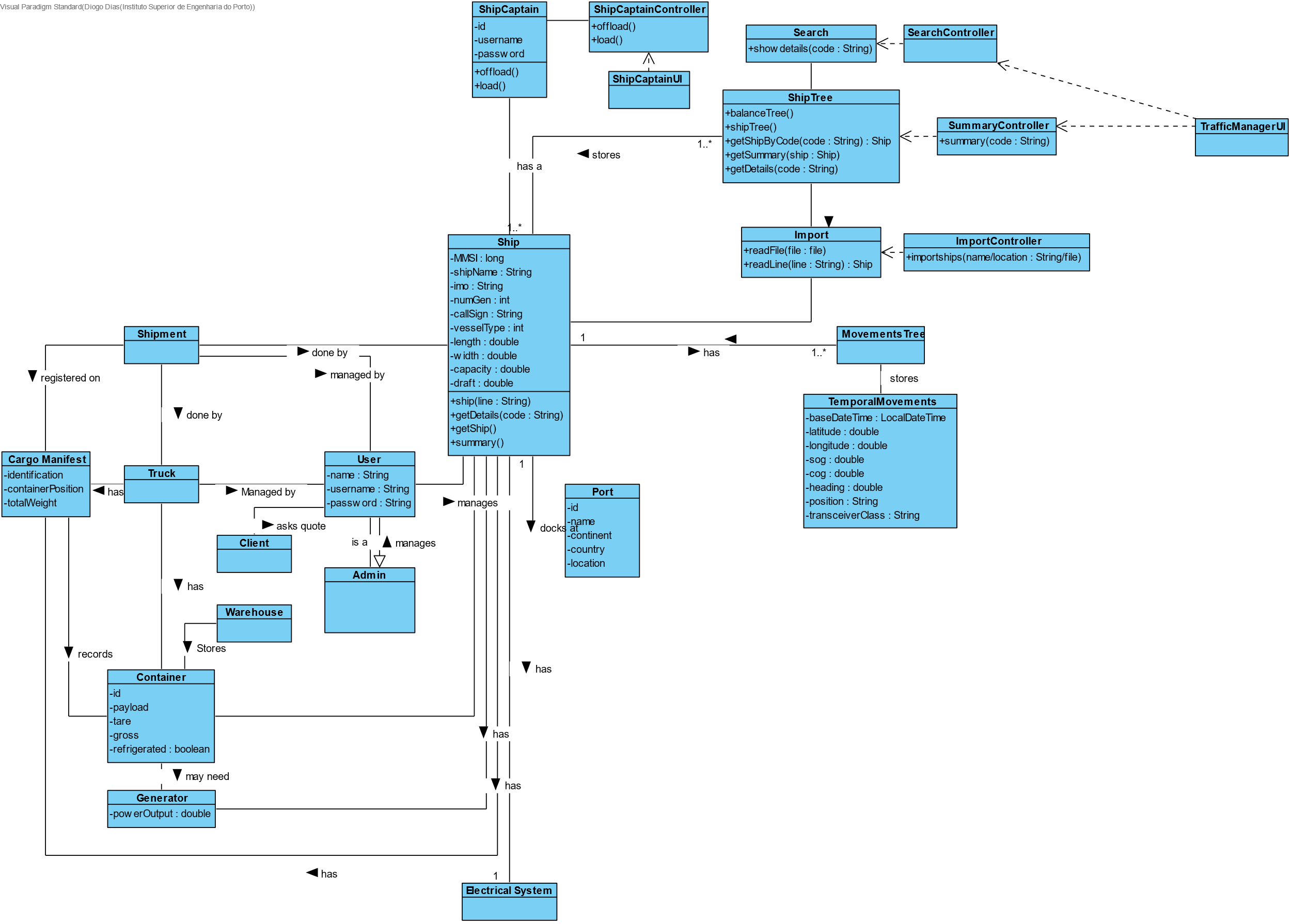
# Sprint 1

## Diagrama de Classes

## Análise de Complexidade

# Sprint 2

## Diagrama de Classes



## Análise de Complexidade

### [US201] As a Port manager, I which to import ports from a text file and create a 2D-tree with port locations.

De acordo com o pedido no enunciado, o que se pretende nesta funcionalidade é receber informação de um ficheiro e inserir a mesma numa estrutura adequada. Para isso teremos um processo que irá percorrer todas as linhas do documento indicado. Depois iremos a cada uma das linhas construir o objeto Porto e de seguida inserimos na árvore 2D-Tree.

Deste modo, ler a linha e construir o Porto tem complexidade 1 e inserir na 2d-Tree tem complexidade . Como tal a criação desta árvore irá ter no seu pior cenário de complexidade temporal.

Diogo Sá Dias - 1161605

### [US202] As a Traffic manager, I which to find the closest port of a ship given its CallSign, on a certain DateTime.

De acordo com o enunciado pretende-se encontrar o porto mais próximo usando a 2D-Tree anteriormente criada. Para isso implementei um método da 2D-Tree designado de findNearestNeighbour. Esse método vai calcular a distância entre o ponto dado e um nó da 2d-Tree, começando pela raiz e continuando, recursivamente por ambas os ramos da 2d-Tree. No final guarda a informação do nó mais próximo.

Assim esta funcionalidade vai ter, no melhor caso complexidade e, caso a dimensão da árvore for alta ou a forma como os nós da árvore for especialmente organizada, a complexidade será n.

Duarte Dias - 1190539

# Sprint 3

## Diagrama de Classes

## Análise de Complexidade

### [US301] As a Traffic manager, I which to import data from countries, ports, borders and sea distances from the database to build a freight network.

O Enunciado pedia para através de uma busca a Base de Dados recolhermos a informação necessária para construir uma Grafo que fosse representativo do modelo de transporte que temos em mãos. Dessa forma decidimos dividir a tarefa de criação e construção da tarefa de recolha dos dados. Inicialmente, e devido a abordagem pretendida neste semestre – TDD – escolhemos começar pela parte da construção do Grafo.

Nesta fase construção e depois de analisarmos a US, onde é pedido que Portos e Países ou Cidades estejam ligados de forma indistinta decidimos criar o grafo e inserir de imediato todos os vértices através de uma lista de Portos e de uma lista de Cidades. Este método tem de **complexidade**  pois **insere** numa estrutura de matriz de Adjacências cada um dos Vértices.

De seguida e com todos os vértices inseridos decidimos ir inserindo os vértices pela ordem que consideramos mais apropriada: Primeiro as fronteiras entre capitais, de seguida a ligação entre porto mais próximo e Capital, fazendo em penúltimo lugar a ligação entre os Portos de cada país e finalizando na ligação que cada porto tem com n Portos de outro país.

Dessa forma primeiramente, verificamos se existem cidades no grafo e de seguida verificamos se o país se encontra na lista de fronteiras. Se verificarmos que essa fronteira existe, inserimos esta aresta. Para isso precisamos de receber uma lista de fronteiras do utilizador tal como as listas que recebemos antes. Esta formato, na verdade, por conveniência do grupo foi o de uma TreeMap que tem no seu keySet os países e no seu valor uma lista de países com quem faz fronteira. Sendo assim a **complexidade deste algoritmo** será da dimensão do keyset do TreeMap vezes a dimensão do tamanho máximo da lista por 1 – valor da inserção de uma aresta na Matriz de Adjacências. Será assim no pior caso de .

Para a criação de ligação entre capital e o Porto mais próximo recorremos ao calculo da distancia mais curta entre a cidade e todos os Portos do País até obtermos o mínimo valor de distância. Desta forma precisamos apenas inicialmente de obter uma lista de todos os portos do país. De seguida percorremos essa lista até encontrar os mínimos e adicionamos esse vértice. Sendo assim teremos de fazer isto para todos os vértices que são cidades, percorrendo todos os vértices que são Portos e pertencem aquele país. A **complexidade** deverá ser **.**

Antes de finalizar as últimas conexões fazemos a ligação de todos os portos de um país. Em que para cada porto presente no grafo, verificamos uma lista de portos do país. A não ser que os portos sejam o mesmo criamos a ligação. A **complexidade** é **.**

Para finalizar adicionamos as n ligações a portos de outros países a cada Porto. Para isso utilizamos o método semelhante ao Floyd-Warshall. Primeiramente criamos uma lista de portos do grafo. De seguida com a distância entre portos e os portos criamos um grafo auxiliar com as distâncias entre portos que não tenham no grafo principal ligação. De seguida com a ajuda deste grafo auxiliar vamos saber quais os portos mais próximos e adicionar um a um aumentando de arestas até n dos dois vértices. Este algoritmo tem **complexidade:**

**Sendo assim US tem complexidade de:**

Diogo Sá Dias - 1161605

### [US302] As a Traffic manager I wish to colour the map using as few colours as possible.

### [US303] As a Traffic manager I wish to know which places (cities or ports) are closest to all other places (closeness places).

O enunciado pedia para, através da utilização do grafo anteriormente construído, obter os locais que eram mais próximos a todos os outros locais, ordenados por continente. Assim, a ideia é construir 5 subgrafos, cada um deles relacionado a um continente, de modo a ser possível dividir todos os locais pelos seus respetivos continentes. Para isso a ideia é usar o algoritmo de Dijkstra que tem como complexidade (E) \* log(V) no melhor dos casos. No entanto o algoritmo do shortestPath que encontra o caminho mais curto entre dois vértices usa o algoritmo de Dijkstra V vezes, o que significa que a complexidade passa a ser V \* (E) \* log(V).

Para saber a média entre os caminhos mais curtos, este algoritmo do shortestPath tem de ser chamado V vezes, o que faz com que a complexidade final seja V\*V\*E\*log(V) no melhor dos casos e da maneira como se pretende fazer a US.

Duarte Dias - 1190539