IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

# PA 2021/22

Programação Avançada

**Enunciado do Projeto de Época Normal**

# Rede Logística

## 1. Objetivo do Projeto

Pretende-se desenvolver um programa em Java que utilize um grafo como modelo de dados, segundo os princípios da orientação a objetos e com utilização de padrões de software.

**O programa consiste numa aplicação gráfica** na qual o utilizador visualiza, manipula e obtém informação diversa sobre uma rede logística.

O projeto será entregue em 3 fases, ver secção 5 – Deliverables.

## 2. Problema

Uma rede logística consiste na localização de centros de distribuição movimentações de produtos entre esses centros. A um centro de distribuição daremos o nome de "**hub**" e a uma ligação (e sua distância) entre dois *hubs* a designação de "**rota**" – ver figura 1**; cada rota é bidirecional**.

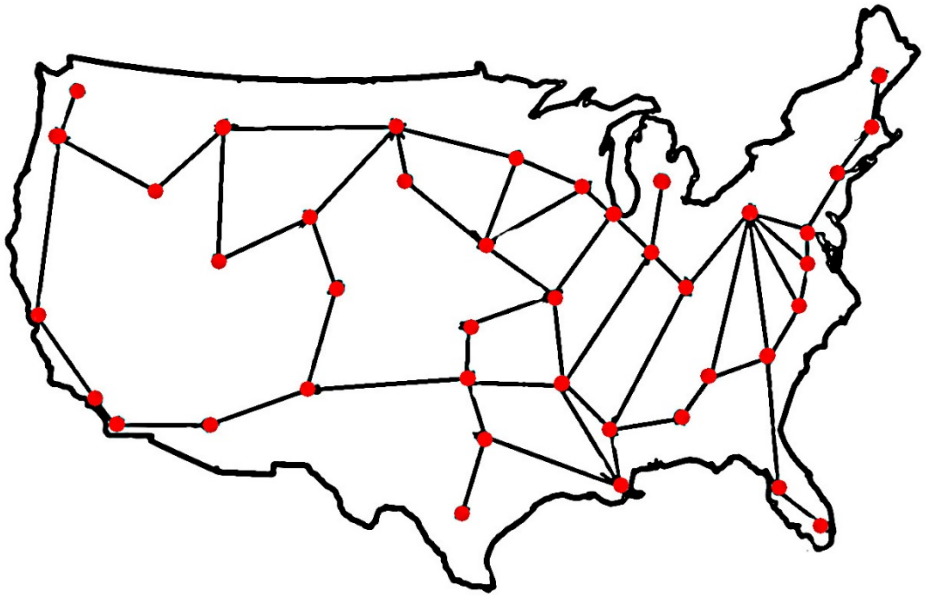

**Figura 1 – Rede logística.**

Esta rede permitirá, entre outras coisas, saber como transportar algo do hub A para o hub B.

Intuitivamente é fácil perceber que uma rede pode ser modelada através de um grafo: vértices consistem nos hubs e as arestas consistem nas rotas.

### 2.1 Dados de entrada

A rede (grafo) será obtida através da importação de *datasets* - **cada dataset consiste num conjunto de ficheiros** que contêm:

1. Nomes das cidades onde os hubs estão localizados;
2. População de cada cidade;
3. Para efeitos de representação gráfica, as coordenadas de ecrã de cada uma das cidades. Isto permitirá visualizar corretamente a localização relativa dos vários hubs a nível geográfico.
4. Rotas disponíveis (distâncias) entre hubs – na forma de uma matriz de distâncias.

Portanto, a importação de uma rede (grafo) consistirá no parse de 3 ficheiros (1,2 e 4) e a sua representação gráfica necessitará adicionalmente de outro (3).

Os dados de entrada são explicados no repositório git (ver secção seguinte), em particular no ficheiro README.

## 2.2 Repositório Git

O repositório *template* será fornecido através do Git Classroom, após registo do grupo de trabalho na plataforma. Para tal, siga as instruções disponibilizadas no Moodle.

Este repositório contém os dados de entrada e uma breve explicação dos mesmos.

# 3. Requisitos

Aqui apresentam-se as funcionalidades esperadas da aplicação e alguns detalhes a ter em conta.

## 3.1 Funcionalidades

### 3.1.1 Importação e visualização da rede logística

Deverá ser possível importar um dataset, gerar o modelo (grafo) correspondente – a rede logística, e visualizá-la na aplicação – ver exemplo da figura 1.

### 3.1.2 Cálculo de métricas

A aplicação deverá exibir, ou permitir consultar a pedido do utilizador, as seguintes métricas:

1. Número de hubs na rede;
2. Número de rotas na rede;
3. Centralidade dos hubs – lista de tuplos < hub, número de hubs adjacentes>; esta lista deve ser apresentada decrescentemente por número de hubs adjacentes;
4. Top 5 hubs "mais centrais" – os primeiros 5 hubs da lista anterior;
5. Número de (sub-)redes logísticas (ver secção 3.2 – Detalhes de implementação).

As métricas 1, 2, 3 e 5 podem ser apresentadas em modo de texto; a métrica 4 tem de ser apresentada através de um gráfico de barras (cada barra é um hub, a "altura" das barras o número de hubs adjacentes).

### 3.1.3 Adição e remoção de rotas e disponibilização de funcionalidade de Undo

Deve ser possível adicionar e remover rotas entre dois hubs em tempo-real (i.e., durante a execução da aplicação) num modelo importado.

Não é permitida a existência de duas rotas simultâneas entre o mesmo par de hubs.

Deve ser disponibilizada a funcionalidade de Undo a qualquer uma destas operações (adicionar/remover).

A visualização da rede e métricas têm de ser atualizadas para o conjunto de rotas resultante.

### 3.1.4 Exportar rotas (matriz de distâncias)

Deve ser possível exportar as rotas de um modelo (e.g., após adicionar/remover rotas). O formato deste ficheiro deve coincidir com o ficheiro 4 da secção 2.1 – Dados de entrada.

Este ficheiro deverá poder ser importado posteriormente, da mesma forma que qualquer outro.

### 3.1.5 Cálculo de caminhos mais curtos entre um par de hubs

Para simular o transporte de algo de um hub para outro, a aplicação deverá permitir encontrar o "caminho mais curto" entre qualquer par de hubs escolhidos pelo utilizador.

Deve ser apresentado ao utilizador a rota encontrada (lista sequencial de hubs) e a distância total desse caminho.

Será bonificada a apresentação visual do caminho encontrado.

### 3.1.6 Par de hubs mais distante

Deve ser possível obter o caminho entre o dois hubs que distam mais um do outro (sempre através dos caminhos mais curtos).

Será bonificada a apresentação visual do caminho encontrado.

### 3.1.7 Hubs que distam N rotas do hub H

Deverá ser possível obter a lista de hubs que distam no máximo N rotas do hub H; H e N são escolhidos pelo utilizador.

## 3.2 Detalhes de implementação

Relativamente às funcionalidades anteriores:

(3.1.1)
A visualização do modelo será feita através da biblioteca JavaFXSmartGraph - https://github.com/brunomnsilva/JavaFXSmartGraph.

(3.1.2)
Pretende-se o "número de componentes" do grafo. A página da *Wikipédia* apresenta um possível algoritmo em linguagem natural (ver https://en.wikipedia.org/wiki/Component_(graph_theory)) ; no caso onde a rede seja representada por um grafo bipartido (ver https://en.wikipedia.org/wiki/Bipartite_graph) esse valor será diferente de 1.

(3.1.6)
É uma aplicação "gananciosa" (*greedy*) da funcionalidade/algoritmo em 3.1.5.

(3.1.7)
É uma aplicação da travessia *breadth-first* com origem em H (mas "limitada").

Em traços gerais:

- Deve haver uma separação clara entre classes que representam o modelo (e contêm a lógica de negócio) e as classes que representam a interface gráfica – utilize corretamente os *packages* para esta distinção.
- A interface gráfica (as classes) deve ser unicamente responsável por obter os dados do utilizador e mostrar resultados – não deverá conter "algoritmos" das funcionalidades.
- Pense "atomicamente" – uma funcionalidade tem "entradas" e "saídas": codifique métodos que implementem o algoritmo. Desta forma é fácil entender o papel esperado da interface gráfica.
- Na versão funcional a consola "não existe" – tudo tem de ser apresentado através da interface gráfica.
- No seguimento do disposto no ponto anterior, a versão funcional deve poder ser lançada através de "duplo-clique" no ficheiro JAR gerado.
- É esperada a aplicação adequada de padrões de software.

# 4 Relatório e Documentação

## 4.1 Documentação

Todo o código deve ser documentado utilizando **Javadoc**. A documentação deve ser gerada em formato HTML e entregue junto com o projeto para fácil consulta.

## 4.2 Relatório

O relatório deverá conter (para além da capa) as seguintes secções:

1. Tipos Abstratos de Dados
    - A apresentação do(s) ADT(s) implementado(s) - descrição da interface Java e da implementação obtida;
2. Diagrama de classes
    - diagramas ilegíveis serão contabilizados com 0 valores;
3. Padrões de software
    - deverá indicar para cada um dos padrões utilizados, a descrição do mesmo, qual a razão de esse ter sido selecionado e de que forma as classes do padrão foram mapeadas para a solução concreta;
4. Refactoring:
    - Uma tabela com os tipos de bad-smells detetados, quantas situações deste tipo foram detetadas e técnica de refactoring aplicada para a sua correção.
    - Um exemplo de cada correção efetuada, com código antes e após refactoring.

# 5 Deliverables e datas de entrega

O projeto contempla 3 fases de entrega e respetivos *deliverables*:

## 5.1 Milestone ( 29 de novembro de 2021)

Consiste na entreag de um projeto IntelliJ IDEA contendo uma aplicação gráfica que:

1. Contenha uma implementação de Graph utilizando a estrutura de dados "lista de adjacências";
2. Contenha testes unitários adequados para a implementação do ponto 1.
3. Permita importar um dataset e construir o modelo (grafo) respetivo;
4. Permita visualizar o modelo obtido através da biblioteca JavaFXSmartGraph.
5. Mockup da interface gráfica idealizada para a aplicação.
6. Ficheiro README.md contendo um resumo do trabalho / código desenvolvido.

Este *deliverable* será alvo de uma apresentação de 10min por parte do grupo, ao qual será atribuída uma nota no intervalo [0, 5].

## 5.2 Projeto Funcional ( 10 janeiro de 2022 )

Consiste na submissão de:

1. uma aplicação gráfica [Projeto IntelliJ IDEA] que resolva o problema proposto na secção 2 e que obedece às funcionalidades solicitadas na secção 3.

Projetos não-funcionais não serão avaliados e não acedem à entrega seguinte.

A partir desta fase não serão contabilizadas funcionalidades adicionais nem correção das mesmas.

Este *deliverable* será alvo de uma apresentação de 10min por parte do grupo, na qual serão anotadas as funcionalidades existentes.

## 5.3 Projeto Final ( 28 janeiro de 2022 )

Consiste na submissão de:

1. uma aplicação gráfica [Projeto IntelliJ IDEA] que resolva o problema proposto na secção 2 e que obedece às funcionalidades solicitadas na secção 3.
    - Permitida melhoria da estrutura do código após ter sido realizado *refactoring* ao mesmo.

2. documentação e relatório – ver secção 5.

A adição de outras novas funcionalidades e/ou correção de problemas anteriores não serão contabilizadas.

Este *deliverable* será alvo de discussão oral.

## 6 Tabela de Cotações e Penalizações

Em traços gerais a cotação total do projeto envolverá o somatório das cotações parciais dos deliverables Milestone e Projeto Final, segundo as seguintes ponderações:

- 20% - Milestone
- 80% - Projeto Final

A avaliação do trabalho será feita de acordo com os seguintes princípios:

- **Estruturação:** o programa deve ser elaborado segundo os princípios da orientação a objetos, com uma boa organização de pacotes e utilizando padrões de software sempre que adequados.
- **Correção:** o programa deve executar as funcionalidades, tal como solicitado.
- **Legibilidade e documentação:** o código deve ser escrito, formatado e comentado de acordo com o standard de programação definido para a disciplina.
- **Desempenho:** As funcionalidades devem ser implementadas da forma mais eficiente possível.

Todas as cotações serão ponderadas de acordo com os princípios acima descritos.

A tabela de cotações apresentada abaixo será aplicada ao *deliverable* 5.3 – Projeto Final.

| **Descrição** | **Cotação (valores)** |
|---|---|
| Implementação Graph com lista de adjacências e testes unitários | 2 |
| Importação e visualização da rede logística | 2 |
| Cálculo de métricas | 2 |
| Adição e remoção de rotas e disponibilização de funcionalidade de Undo | 2 |
| Exportar rotas (matriz de distâncias) | 1 |
| Cálculo de caminhos mais curtos entre um par de hubs | 2 |
| Par de hubs mais distante | 1 |
| Hubs que distam N rotas do hub H | 2 |
| Interface gráfica (usabilidade geral) | 1 |
| Utilização do versionamento Git | 1 |
| Relatório e Documentação | 3 (2+1) |
| **TOTAL** | 20 |

**Um projeto só será considerado "funcional" se implementar as funcionalidades assinaladas a amarelo.**

A seguinte tabela contém penalizações a aplicar:

| **Descrição** | **Penalização (val.)** |
|---|---|
| Existência de code smells | até 2 |
| Não utilização de padrões de software | até 3 |

## 7 Regras de elaboração e submissão

O não cumprimento das regras a seguir descritas implica uma penalização na nota do trabalho prático. Se ocorrer alguma situação não prevista nas regras a seguir expostas, essa ocorrência deverá ser comunicada ao respetivo docente de laboratório de PA e/ou ao RUC.

## Regras:

a) O Projeto deverá ser elaborado por **quatro alunos do mesmo docente de laboratório**.
   - O desvio desta regra terá de ter o consentimento expresso da RUC.

b) Os *delieverable*s serão submetidos no Moodle até às 9h da data limite respetiva;

c) A nota do Projeto será atribuída individualmente a cada um dos elementos do grupo após a discussão. As discussões poderão ser orais e/ou com perguntas escritas. As orais poderão ser feitas com todos os elementos do grupo presentes em simultâneo ou individualmente. E poderão ser feitas remotamente via plataforma zoom.

   - Os *commits* no repositório individual do grupo serão tidos em conta na avaliação individual.

d) **A apresentação de relatórios ou implementações plagiadas leva à imediata atribuição de nota zero a todos os trabalhos com semelhanças, quer tenham sido o original ou a cópia**.

e) No rosto do relatório e nos ficheiros de implementação deverá constar o número, nome e turma dos autores e o nome do docente a que se destina.

   Para tal terão que criar uma pasta com o nome: **nomeAluno1_númeroAluno1-nomeAluno2_númeroAluno2-…**, onde colocarão o ficheiro do relatório em formato **pdf** e uma pasta com o projeto IntelliJ (cópia do repositório, versão para submissão). Os alunos terão de submeter essa **pasta compactada em formato ZIP**. Apenas será permitido submeter um ficheiro.

f) As datas das discussões serão publicadas após a entrega dos trabalhos.

(fim de enunciado)