Uma imagem com pessoa, interior, desfocagem

Descrição gerada automaticamenteA picture containing shape

Description automatically generated

**Relatório das funcionalidades implementadas no Sprint1 -Correções na Entrega do Sprint 2**

Projeto realizado pelo Grupo 102:

-1190701 João Sousa

- 1201942 Fábio Silva

-1190569 Filipe Morais

-1200618 Jorge Cunha

-1191576 Francisco Sampaio

Index

[Contextualização 3](#_Toc120902414)

[Class Diagram 3](#_Toc120902415)

[UC 301 4](#_Toc120902416)

[Enunciado: 4](#_Toc120902417)

[Proposta de Solução: 4](#_Toc120902418)

[Explicação: 5](#_Toc120902419)

[UC302 6](#_Toc120902420)

[Enunciado: 6](#_Toc120902421)

[Proposta de solução: 6](#_Toc120902424)

[Explicação: 7](#_Toc120902425)

[UC303 8](#_Toc120902426)

[Enunciado: 8](#_Toc120902427)

[Proposta de Solução 8](#_Toc120902432)

[Explicação: 8](#_Toc120902433)

[UC304 9](#_Toc120902434)

[Enunciado: 9](#_Toc120902435)

[Explicação: 9](#_Toc120902438)

[UC305 10](#_Toc120902439)

[Enunciado: 10](#_Toc120902440)

[Proposta de Solução: 10](#_Toc120902443)

[Explicação: 10](#_Toc120902444)

[Funções adicionais: 11](#_Toc120902445)

[VertCycle 11](#_Toc120902446)

[Proposta de solução: 11](#_Toc120902447)

[Explicação 11](#_Toc120902448)

[Proposta de solução: 11](#_Toc120902449)

[Explicação: 11](#_Toc120902450)

# Contextualização

# Class Diagram

Os diagramas de classes foi entregue em conjunto com este documento. Devido ao seu tamanho, incluí-lo neste ficheiro torná-lo-ia muito difícil de ler, razão pela qual foi entregue em separado, num formato de SVG.

Depois da leitura do enunciado do projeto, identificámos as classes base que modelam as entidades do negócio: *Pessoa, Cliente, Empresa, Produtor.* Estas classes seriam armazenadas em grafos, para tal implementamos as classes algoritmos, graphs e MapGraphs.

As classes de grafos e os seus algoritmos usadas foram as desenvolvidas durante as aulas. Optámos por usar o MapGraph, pois em deterimento do problema em questão é mais eficiente comparando com as outras opções.

A classe responsável pela importação do CSV é a *ReadFromCSV*, que depois de processar os dados, cria objetos pessoa e distancias que são armazenadas em listas para mais tarde serem utilizadas na criação dos grafos.

De forma a ter maior organização do projeto, o grupo decidiu criar um ui e um controller*, o que nos permite obter os vários métodos sem estes serem todos static.*

Por fim, foram criadas algumas classes que albergam os dados de retorno dos exercícios.

# UC 301

Enunciado:

Construir a rede de distribuição de cabazes a partir da informação fornecida nos

ficheiros. O grafo deve ser implementado usando a representação mais adequada e garantindo a

**manipulação indistinta dos clientes/empresas e produtores agrícolas** (código cliente: C, código

empresa: E, código produtor: P).

## Proposta de Solução:

Uma imagem com texto

Descrição gerada automaticamenteUma imagem com texto

Descrição gerada automaticamenteUma imagem com texto

Descrição gerada automaticamenteUma imagem com texto

Descrição gerada automaticamente

## Explicação:

No metodo lerClienteProdutor,o ficheiro é lido e armazenado na storelist para depois ser usado.

No metodo lerDistancias,o ficheiro é lido e armazenado na storelist para depois ser usado.

Apos serem lidos ambos ficheiros é usado o método createGraph que pega na lista de clientes Produtores e vai adicionando como vértices e em cada vértice adiciona o edge com o ficheiro das distâncias.

# UC302

## Enunciado:

## Verificar se o grafo carregado é conexo e devolver o número mínimo de ligações

## necessário para nesta rede qualquer cliente/produtor conseguir contactar um qualquer outro.

## Proposta de solução:

Uma imagem com texto

Descrição gerada automaticamente

Método inicialmente usado da US302:

Uma imagem com texto

Descrição gerada automaticamenteUma imagem com texto

Descrição gerada automaticamente

Método Alternativo da US 302:

## Uma imagem com texto Descrição gerada automaticamenteExplicação:

No método verificarGraphConexo, nós vamos buscar os dos vértices e fazemos um for each e em cada vértice usamos o BreadthFirstSearch para retornar um caminho e verificamos se existe pelo menos um caminho que contenha todos os vértices e se tiver o grafo é conexo.

No método numeroLigacoesDiam vamos buscar os vértices outra vez e fazemos um for each e em cada vértice usamos o vertCycles que nos ira dar todos os caminhos pelo o método depthFirstSearch e adiciona numa lista, depois procuramos a lista com o maior número de ligações e retornamos esse número.

# UC303

## Enunciado:

## Definir os hubs da rede de distribuição, ou seja, encontrar as N empresas mais próximas de

## todos os pontos da rede (clientes e produtores agrícolas). A medida de proximidade deve ser

## calculada como a média do comprimento do caminho mais curto de cada empresa a todos os

## clientes e produtores agrícolas. (small, N=3)

## Proposta de Solução

Uma imagem com texto

Descrição gerada automaticamente Uma imagem com texto

Descrição gerada automaticamente

## Explicação:

Neste método findNearestHub ,ele começa por fazer o Algoritmo minDistGraph que faz com que retorne a matrix graph (ajudas nos a converter os weights e torna a execução do programa mais rápido), logo de seguida vamos buscar os vértices e criamos um hash map.

Fazemos um for each para todos os vértices que sejam Empresa para ver as ligações com os outros que não são Empresa.

# UC304

## Enunciado:

## Para cada cliente (particular ou empresa) determinar o hub mais próximo.

## Proposta de Solução:

Uma imagem com texto

Descrição gerada automaticamente Uma imagem com texto

Descrição gerada automaticamente

## Explicação:

No método findNearestHubEachClient,logo de inicio chamamos o minDistGraph pois calcula o grafo de distancia mínima usando o Floyd-Warshall.Logo de seguida percorremos a lista de clientes e vemos as distancias entre todas as empresas e guardamos num mapa auxiliar onde depois vai ser mandado para uma Lista de Strings .

# UC305

## Enunciado:

## Determinar a rede que conecte todos os clientes e produtores agrícolas com uma distância

## total mínima.

## Proposta de Solução:

Uma imagem com texto

Descrição gerada automaticamente

## Explicação:

No método findNetworkWithMinimumDistance nos chamamos o minDistGraph para calcular o grafo de distancia mínima e passar para matrixGraph e fazemos o algoritmo Kruskal e retornamos a rede ainda em matriz.

# Funções adicionais:

# VertCycle

Proposta de solução:

Uma imagem com texto

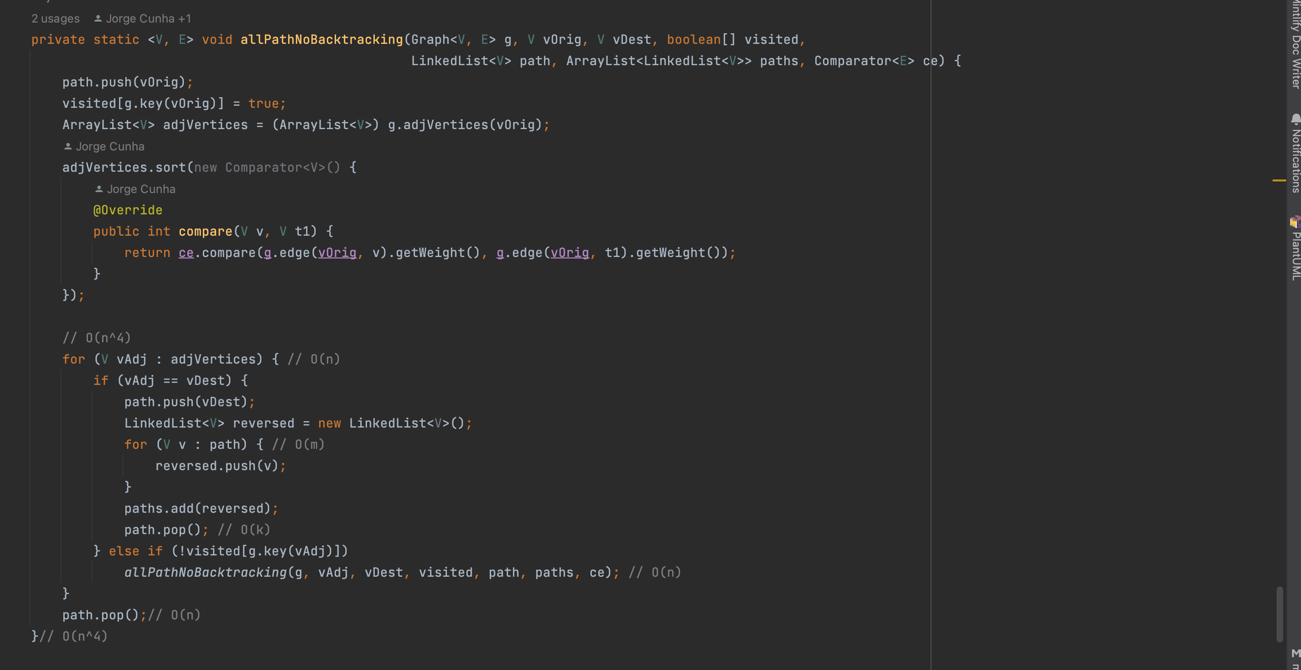
Descrição gerada automaticamente

## Explicação

Este método executa uma versão modificada do algoritmo allPaths onde agora não retrocede, desta forma ele faz todos os ciclos que um simples depthFirstSearch pode encontrar em uma primeira leitura e os adiciona a uma única lista, após a qual o método retorna apenas os ciclos que contenham o vértice fornecido, pois estes podem ser movidos de forma que o vértice esteja no início.

AllPathNoBacktracking:

Proposta de solução:



## Explicação:

O algoritmo inicial do allPath tem uma complexidade que é inutilizável para grafos tão grandes quanto o que estamos usando, portanto, este método não retrocede, mas é compensado pelo fato de ser executado para cada, vértice no gráfico para obter uma melhor certeza.