ESINF REPORT

Group 118

NUNO BARBOSA – 1210822

KEVIN SOUSA-1180853

RODRIGO FERREIRA - 1191447

JOÃO MARTINS- 1201539

Subject: Estruturas de Informação (ESINF)

Teacher: Nuno Filipe Teixeira Malheiro (NFM)

DECEMBER 2021

Content

Introduction..3

Algorithm Analysis 4

US301 – Import data from countries, ports, borders and seadists from the database to build a freight network 5

US302 – Colour the map using as few colours as possible...6

US303 – Know which places (cities or ports) are closest to all other places (closeness places) …….7

Domain Model Diagram.. 8

Class Diagram …….9

Introduction

In the following report, we will present our User Story development as well as the respective Domain Model for the whole project.

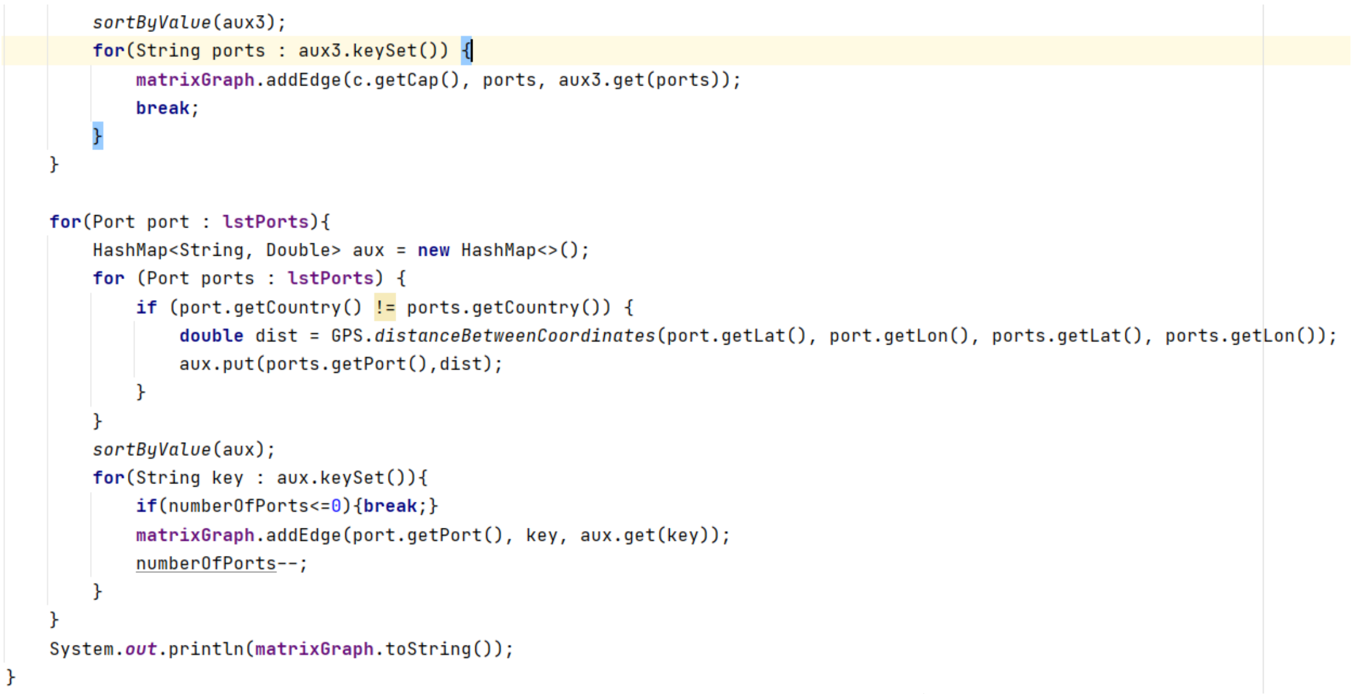
First, we decided to use a graph, as told on the US301,

Algorithm Analysis

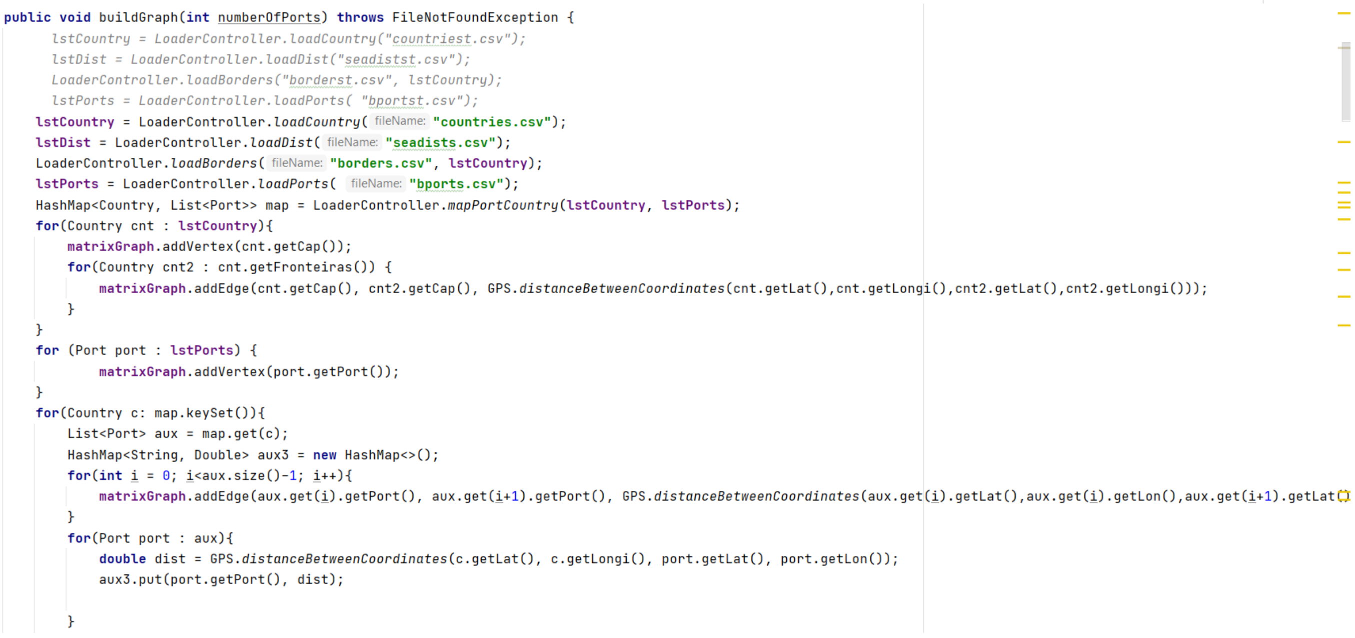
For each User story, we will explain the code written, as well as the difficulties had while developing.

We also determine the time complexity for each method, according to the Big-O notation

US301 – Import data from countries, ports, borders and seadists from the database to build a freight network



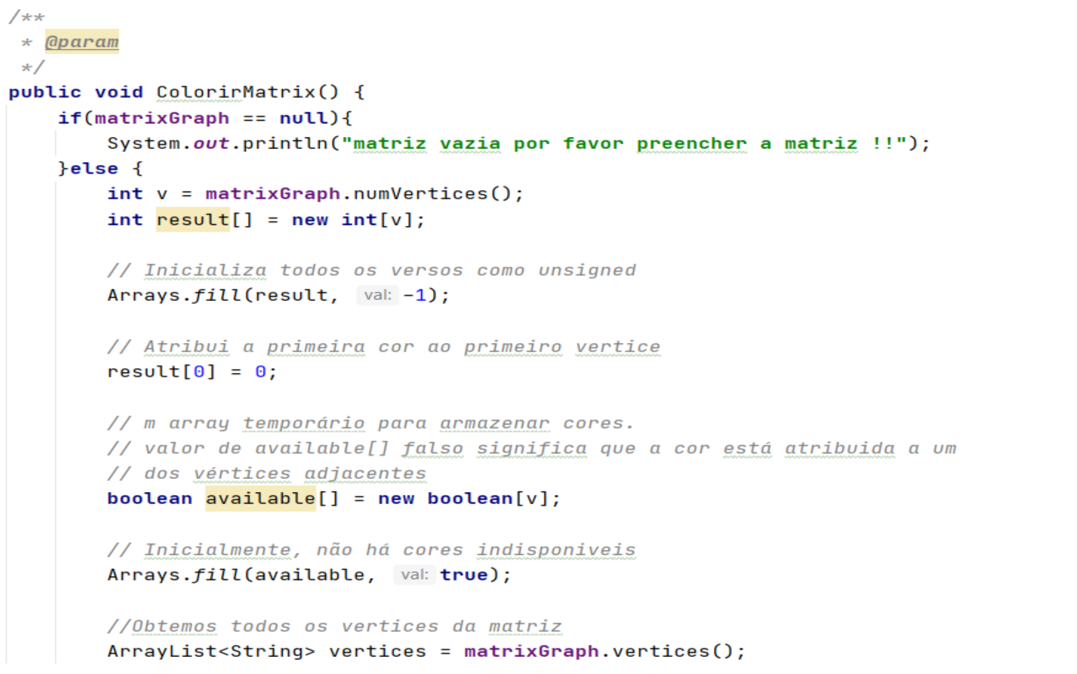
```java
        sortByValue(aux3);
        for(String ports : aux3.keySet()) {
            matrixGraph.addEdge(c.getCap(), ports, aux3.get(ports));
            break;
        }
    }

    for(Port port : lstPorts){
        HashMap<String, Double> aux = new HashMap<>();
        for (Port ports : lstPorts) {
            if (port.getCountry() != ports.getCountry()) {
                double dist = GPS.distanceBetweenCoordinates(port.getLat(), port.getLon(), ports.getLat(), ports.getLon());
                aux.put(ports.getPort(),dist);
            }
        }
        sortByValue(aux);
        for(String key : aux.keySet()){
            if(numberOfPorts<=0){break;}
            matrixGraph.addEdge(port.getPort(), key, aux.get(key));
            numberOfPorts--;
        }
    }
    System.out.println(matrixGraph.toString());
}
```



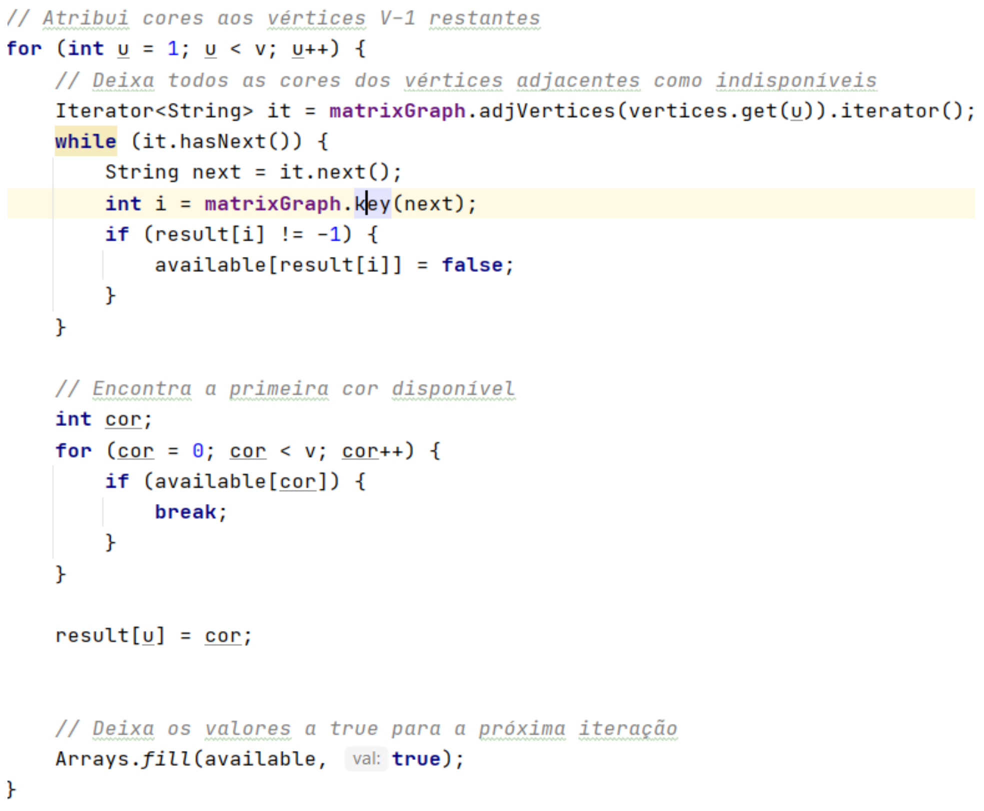
```java
public void buildGraph(int numberOfPorts) throws FileNotFoundException {
      lstCountry = LoaderController.loadCountry("countriest.csv");
      lstDist = LoaderController.loadDist("seadistst.csv");
      LoaderController.loadBorders("borderst.csv", lstCountry);
      lstPorts = LoaderController.loadPorts( "bportst.csv");
    lstCountry = LoaderController.loadCountry( fileName: "countries.csv");
    lstDist = LoaderController.loadDist( fileName: "seadists.csv");
    LoaderController.loadBorders( fileName: "borders.csv", lstCountry);
    lstPorts = LoaderController.loadPorts(  fileName: "bports.csv");
    HashMap<Country, List<Port>> map = LoaderController.mapPortCountry(lstCountry, lstPorts);
    for(Country cnt : lstCountry){
        matrixGraph.addVertex(cnt.getCap());
        for(Country cnt2 : cnt.getFronteiras()) {
            matrixGraph.addEdge(cnt.getCap(), cnt2.getCap(), GPS.distanceBetweenCoordinates(cnt.getLat(),cnt.getLongi(),cnt2.getLat(),cnt2.getLongi()));
        }
    }
    for (Port port : lstPorts) {
            matrixGraph.addVertex(port.getPort());
    }
    for(Country c: map.keySet()){
        List<Port> aux = map.get(c);
        HashMap<String, Double> aux3 = new HashMap<>();
        for(int i = 0; i<aux.size()-1; i++){
            matrixGraph.addEdge(aux.get(i).getPort(), aux.get(i+1).getPort(), GPS.distanceBetweenCoordinates(aux.get(i).getLat(),aux.get(i).getLon(),aux.get(i+1).getLat(
        }
        for(Port port : aux){
            double dist = GPS.distanceBetweenCoordinates(c.getLat(), c.getLongi(), port.getLat(), port.getLon());
            aux3.put(port.getPort(), dist);

        }
```

We imported the list in one different method according to following User Stories where each one of them is useful. This method has a O (n^2) complexity because we are importing data from the database, and we use 2 loops.

US302 – Colour the map using as few colors as possible



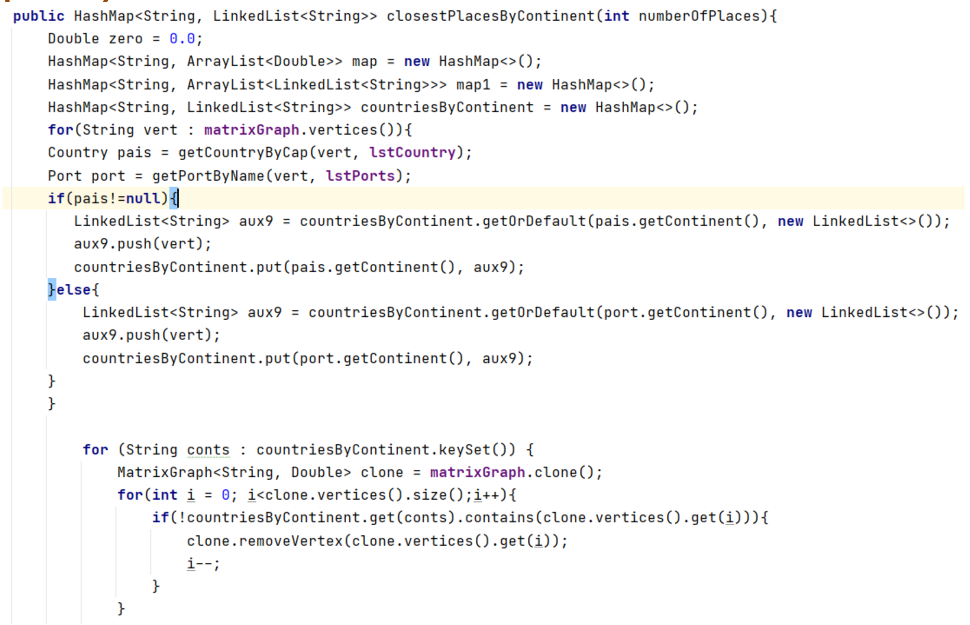
```
/**
 * @param
 */
public void ColorirMatrix() {
    if(matrixGraph == null){
        System.out.println("matriz vazia por favor preencher a matriz !!");
    }else {
        int v = matrixGraph.numVertices();
        int result[] = new int[v];

        // Inicializa todos os versos como unsigned
        Arrays.fill(result,  val: -1);

        // Atribui a primeira cor ao primeiro vertice
        result[0] = 0;

        // m array temporário para armazenar cores.
        // valor de available[] falso significa que a cor está atribuida a um
        // dos vértices adjacentes
        boolean available[] = new boolean[v];

        // Inicialmente, não há cores indisponiveis
        Arrays.fill(available,  val: true);

        //Obtemos todos os vertices da matriz
        ArrayList<String> vertices = matrixGraph.vertices();
```



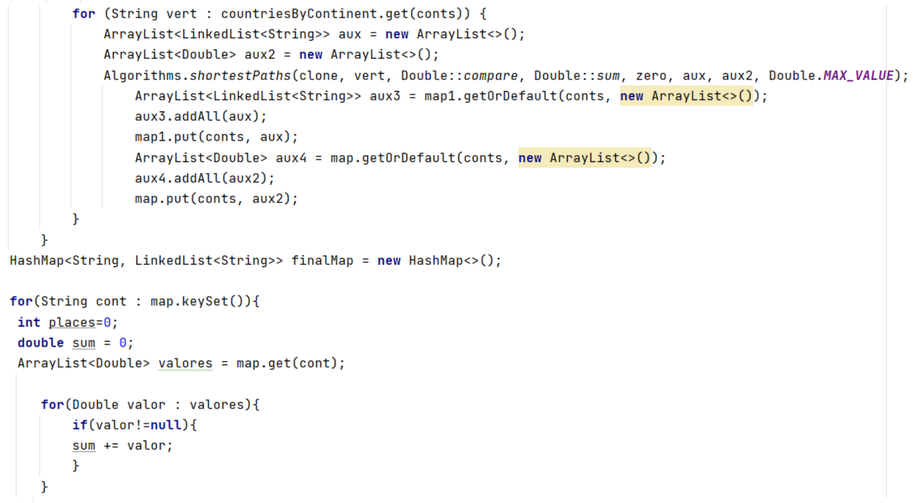
```
        // Atribui cores aos vértices V-1 restantes
        for (int u = 1; u < v; u++) {
            // Deixa todos as cores dos vértices adjacentes como indisponíveis
            Iterator<String> it = matrixGraph.adjVertices(vertices.get(u)).iterator();
            while (it.hasNext()) {
                String next = it.next();
                int i = matrixGraph.key(next);
                if (result[i] != -1) {
                    available[result[i]] = false;
                }
            }

            // Encontra a primeira cor disponível
            int cor;
            for (cor = 0; cor < v; cor++) {
                if (available[cor]) {
                    break;
                }
            }

            result[u] = cor;


            // Deixa os valores a true para a próxima iteração
            Arrays.fill(available,  val: true);
        }
```

In this User Storie, we color our matrix and we using 2 loops, so the complexity is O (n^2)

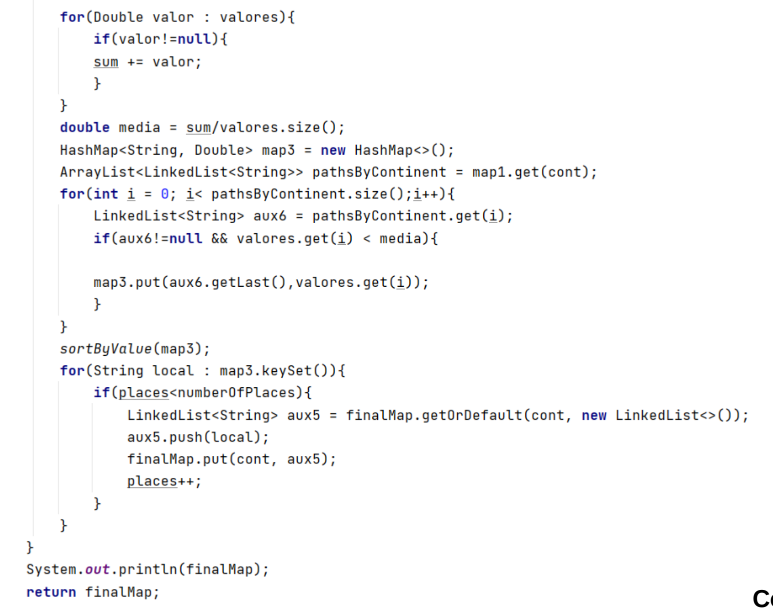
US303 – Know which places (cities or ports) are closest to all other places (closeness places)



```java
public HashMap<String, LinkedList<String>> closestPlacesByContinent(int numberOfPlaces){
    Double zero = 0.0;
    HashMap<String, ArrayList<Double>> map = new HashMap<>();
    HashMap<String, ArrayList<LinkedList<String>>> map1 = new HashMap<>();
    HashMap<String, LinkedList<String>> countriesByContinent = new HashMap<>();
    for(String vert : matrixGraph.vertices()){
    Country pais = getCountryByCap(vert, lstCountry);
    Port port = getPortByName(vert, lstPorts);
    if(pais!=null){
      LinkedList<String> aux9 = countriesByContinent.getOrDefault(pais.getContinent(), new LinkedList<>());
      aux9.push(vert);
      countriesByContinent.put(pais.getContinent(), aux9);
    }else{
       LinkedList<String> aux9 = countriesByContinent.getOrDefault(port.getContinent(), new LinkedList<>());
       aux9.push(vert);
       countriesByContinent.put(port.getContinent(), aux9);
    }
    }

       for (String conts : countriesByContinent.keySet()) {
           MatrixGraph<String, Double> clone = matrixGraph.clone();
           for(int i = 0; i<clone.vertices().size();i++){
               if(!countriesByContinent.get(conts).contains(clone.vertices().get(i))){
                   clone.removeVertex(clone.vertices().get(i));
                   i--;
               }
           }
```



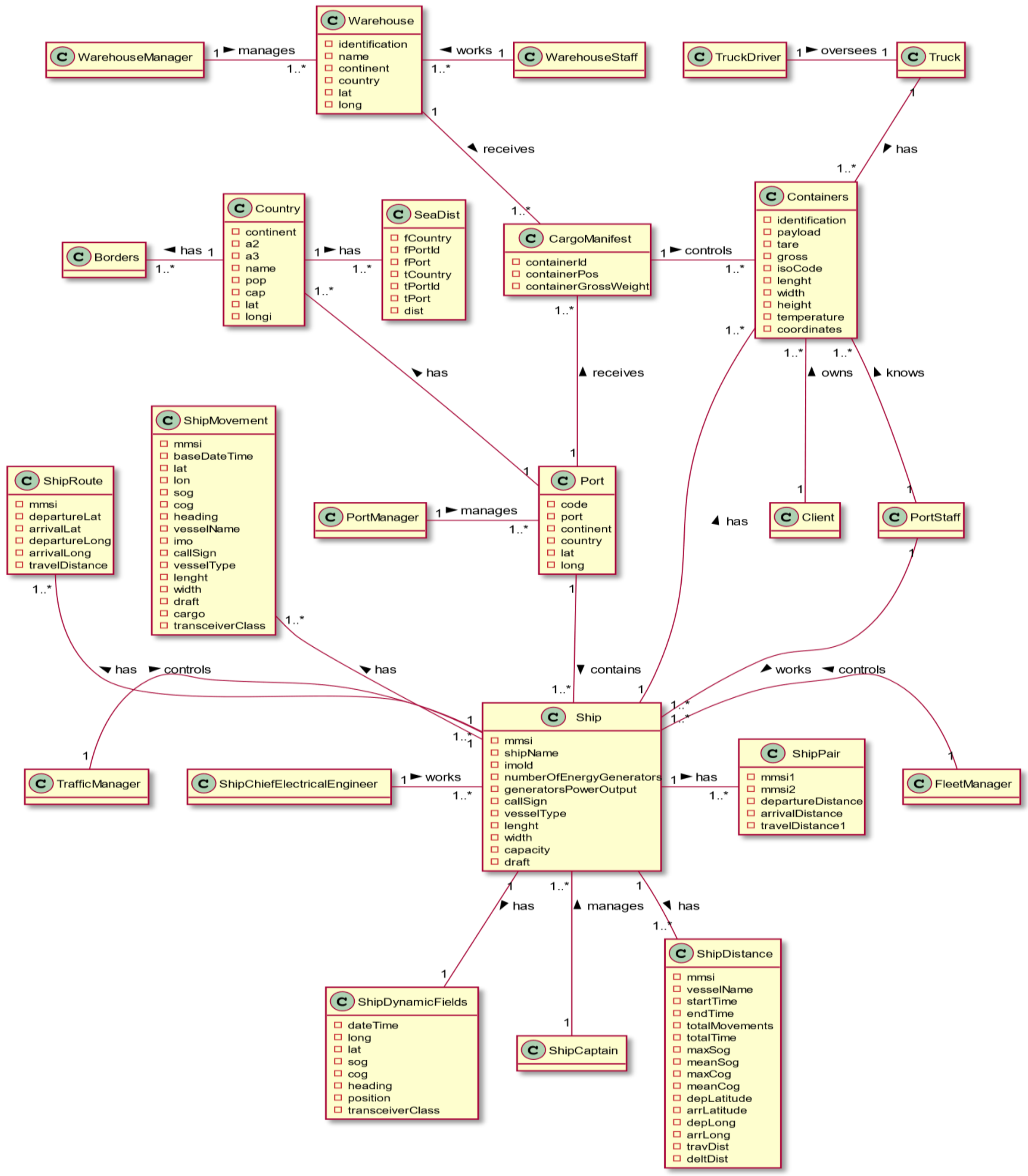
```java
        for (String vert : countriesByContinent.get(conts)) {
            ArrayList<LinkedList<String>> aux = new ArrayList<>();
            ArrayList<Double> aux2 = new ArrayList<>();
            Algorithms.shortestPaths(clone, vert, Double::compare, Double::sum, zero, aux, aux2, Double.MAX_VALUE);
                ArrayList<LinkedList<String>> aux3 = map1.getOrDefault(conts, new ArrayList<>());
                aux3.addAll(aux);
                map1.put(conts, aux);
                ArrayList<Double> aux4 = map.getOrDefault(conts, new ArrayList<>());
                aux4.addAll(aux2);
                map.put(conts, aux2);
        }
    }
HashMap<String, LinkedList<String>> finalMap = new HashMap<>();

for(String cont : map.keySet()){
 int places=0;
 double sum = 0;
 ArrayList<Double> valores = map.get(cont);

    for(Double valor : valores){
        if(valor!=null){
        sum += valor;
        }
    }
```



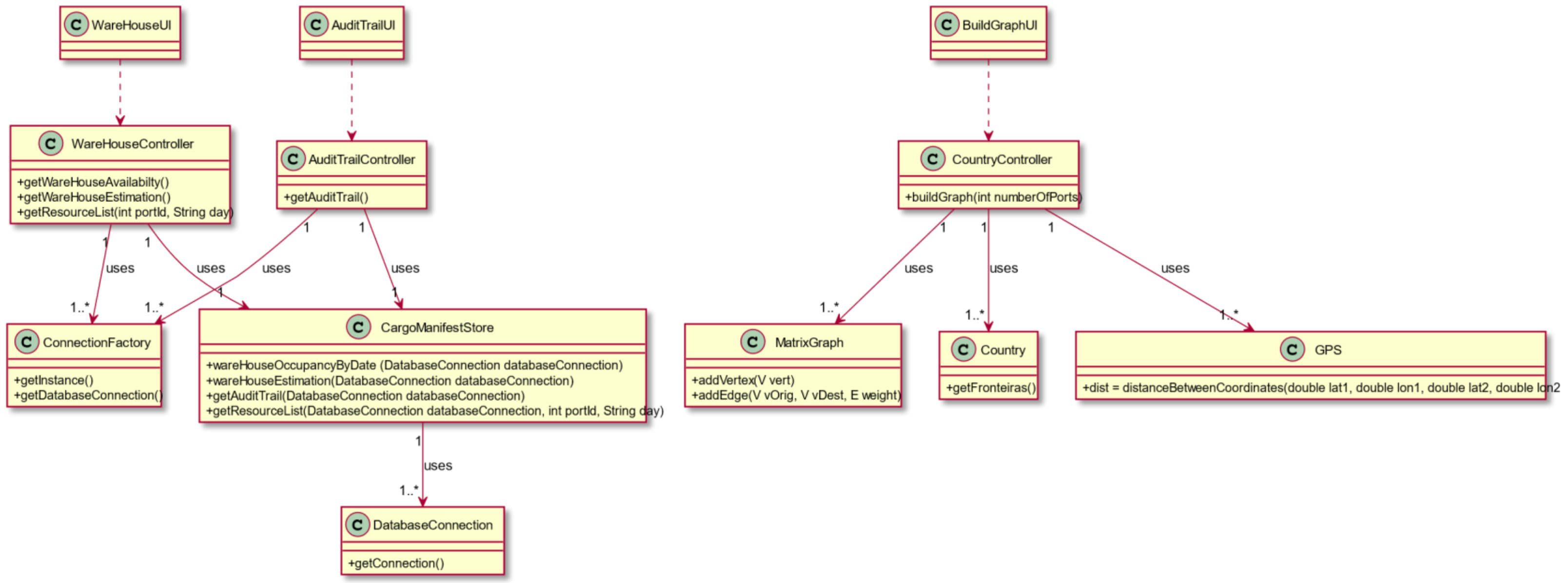
```java
     for(Double valor : valores){
         if(valor!=null){
         sum += valor;
         }
     }
     double media = sum/valores.size();
     HashMap<String, Double> map3 = new HashMap<>();
     ArrayList<LinkedList<String>> pathsByContinent = map1.get(cont);
     for(int i = 0; i< pathsByContinent.size();i++){
         LinkedList<String> aux6 = pathsByContinent.get(i);
         if(aux6!=null && valores.get(i) < media){

         map3.put(aux6.getLast(),valores.get(i));
         }
     }
     sortByValue(map3);
     for(String local : map3.keySet()){
         if(places<numberOfPlaces){
             LinkedList<String> aux5 = finalMap.getOrDefault(cont, new LinkedList<>());
             aux5.push(local);
             finalMap.put(cont, aux5);
             places++;
         }
     }
  }
  System.out.println(finalMap);
  return finalMap;
```

Complexity: O (n^2)

Domain Model Diagram



Class Diagram


WareHouseUI
AuditTrailUI
BuildGraphUI
WareHouseController
+getWareHouseAvailabilty()
+getWareHouseEstimation()
+getResourceList(int portId, String day)
AuditTrailController
+getAuditTrail()
CountryController
+buildGraph(int numberOfPorts)
uses
1
1..*
ConnectionFactory
+getInstance()
+getDatabaseConnection()
CargoManifestStore
+wareHouseOccupancyByDate (DatabaseConnection databaseConnection)
+wareHouseEstimation(DatabaseConnection databaseConnection)
+getAuditTrail(DatabaseConnection databaseConnection)
+getResourceList(DatabaseConnection databaseConnection, int portId, String day)
MatrixGraph
+addVertex(V vert)
+addEdge(V vOrig, V vDest, E weight)
Country
+getFronteiras()
GPS
+dist = distanceBetweenCoordinates(double lat1, double lon1, double lat2, double lon2
DatabaseConnection
+getConnection()