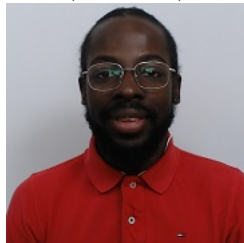


Programação Orientada aos Objetos
Trabalho Prático - DriveIt
Relatório de Desenvolvimento - Grupo 14

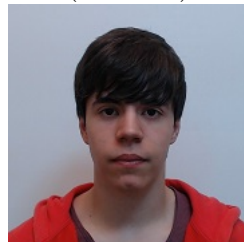
Áureo Joel Costa dos Santos Benedito
(A76068)



Pedro Simão Lemos Silva
(A85625)



Rui Miguel Pimenta e Cunha
(A85877)



11 de Junho de 2020

Conteúdo

1	Introdução	2
2	Resolução	3
2.1	Funcionamento	3
2.2	Classes	3
3	Algumas Funções	14
4	Diagrama	19
5	Conclusão	20

Capítulo 1

Introdução

No âmbito da unidade curricular de Programação Orientada aos Objetos foi-nos proposto a realização de um trabalho prático de forma a podermos aplicar o conhecimento adquirido nas aulas. Neste trabalho iremos aplicar os conhecimentos para resolvermos a tarefa que nos foi dada.

O projeto solicitado passa por construir uma aplicação em Java, em criar uma aplicação que permita conjugar as actuais necessidades de entregar encomendas a pessoas que estão confinadas nas suas habitações.

Capítulo 2

Resolução

Aqui falaremos das implementações das classes e do funcionamento das mesmas.

2.1 Funcionamento

A app permite que um ou mais utilizadores façam encomendas de produtos onde:

Os utilizadores da TrazAqui! - Serviço de Entregas Encomendas em Casa poderão:

- solicitar a entrega de uma encomenda que foi pedida a uma loja;
- aceitar, ou não, o serviço de entrega proposto por uma empresa transportadora (os serviços feitos por voluntários são automaticamente aceites pelo sistema);
- aceder à informação das entregas efectuadas num determinado período e por voluntário ou transportador;
- classificar o voluntário ou a empresa de transportes mediante o grau de satisfação com o serviço;

Os voluntários poderão:

- sinalizar que estão dispostos para recolher encomendas;
- escolher ir buscar uma encomenda de um utilizador que é disponibilizada por uma loja;
- fazer o transporte da encomenda e registar quanto tempo demorou;

As empresas transportadoras poderão:

- sinalizar que estão dispostos para recolher encomendas;
- determinar o preço de transporte de uma encomenda em função da distância e do tempo de espera na loja (estimado ou fornecido pela loja);
- fazer o transporte da encomenda e registar quanto tempo demorou e o custo associado;

2.2 Classes

Para a construção da app usamos varias classes que listaremos agora: user, utilizador, transportador, empresa, loja, loja com fila, loja sem fila, encomenda, linha de encomenda, voluntario, encomendaBD

Temos o Utilizador, Transportador e Loja que herdam de User suas variaveis de estancia.

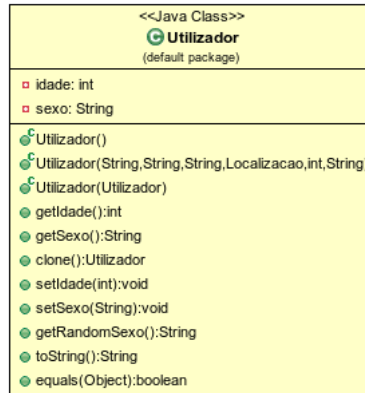


Figura 2.1: Utilizador



Figura 2.2: Transportador

Temos *Empresa* e *voluntario* que herda de *Transportador* suas variáveis de instância. A classe *Loja*, também com dois "herdeiros", *LojaSemFila* e *LojaComFila*, é a classe que trata das Lojas que produzem as encomendas.


<<Java Class>>  Empresa (default package)	
<ul style="list-style-type: none"> ▢ custo_km: double ▢ custo_peso: double ▢ nif: String 	
<ul style="list-style-type: none"> ☞ Empresa() ☞ Empresa(String,String,String,Localizacao,double,boolean,boolean,Map<String,Integer>,HashMap<String,Encomenda>,double,double,String) ☞ Empresa(Empresa) ☞ getNif():String ☞ getCusto_Km():double ☞ getCusto_Peso():double ☞ setNif(String):void ☞ setCusto_Km(double):void ☞ setCusto_peso(double):void ☞ toString():String ☞ equals(Object):boolean ☞ clone():Empresa 	

Figura 2.3: Empresa


<<Java Class>>  Voluntario (default package)	
<ul style="list-style-type: none"> ▢ idade: int ▢ sexo: String 	
<ul style="list-style-type: none"> ☞ Voluntario() ☞ Voluntario(String,String,String,Localizacao,double,boolean,boolean,Map<String,Integer>,HashMap<String,Encomenda>,int,String) ☞ Voluntario(Voluntario) ☞ getIdade():int ☞ getSexo():String ☞ clone():Voluntario ☞ setIdade(int):void ☞ setSexo(String):void ☞ getRandomSexo():String ☞ toString():String ☞ equals(Object):boolean 	

Figura 2.4: Loja Sem Fila


<<Java Class>>  Loja (default package)	
<ul style="list-style-type: none"> ▢ encomendas: Map<String,Encomenda> ▢ classificacao: Map<String,Integer> 	
<ul style="list-style-type: none"> ☞ Loja() ☞ Loja(String,String,String,Localizacao,HashMap<String,Encomenda>,Map<String,Integer>) ☞ Loja(Loja) ☞ getEncomendas():HashMap<String,Encomenda> ☞ getClassificacao():HashMap<String,Integer> ☞ setEncomendas(HashMap<String,Encomenda>):void ☞ setClassificacao(Map<String,Integer>):void ☞ addEncomenda(Encomenda):void ☞ removeEncomenda(String):void ☞ addClassificacao(String,int):void ☞ classMedia():double ☞ clone():Loja ☞ toString():String ☞ equals(Object):boolean 	

Figura 2.5: Loja

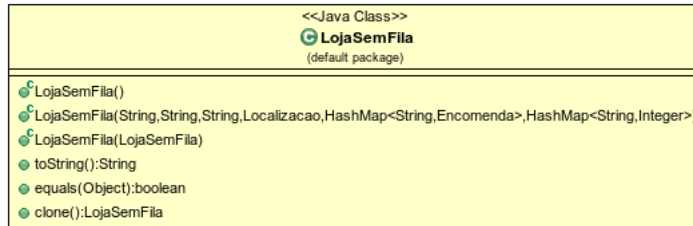


Figura 2.6: Loja Sem Fila

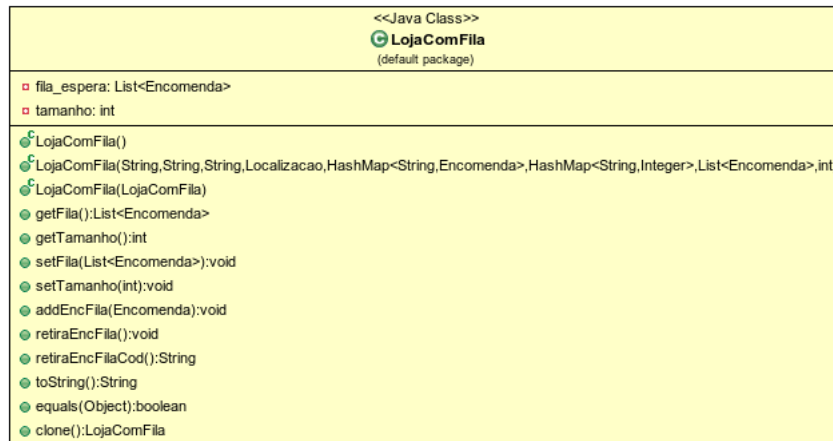


Figura 2.7: Loja Com Fila

Na classe Localização encontramos a informação necessária para saber da localização de uma loja, empresa, transportador, voluntario, utilizador, user, loja, lojaSemFila ou LojaComFila, ou seja, ela é "chamada" em cada uma destas classes para este efeito, saber da localização.

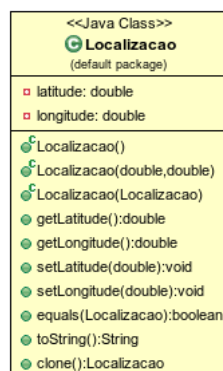


Figura 2.8: Localização

A classe Encomenda como nome diz trata das encomendas, tem como métodos (alguns). `getPreco`, que retorna o preço de uma certa encomenda, `setTransportador`, que associa o transportador de uma certa encomenda, ou seja, a classe encomenda lida com as informações de uma certa encomenda.

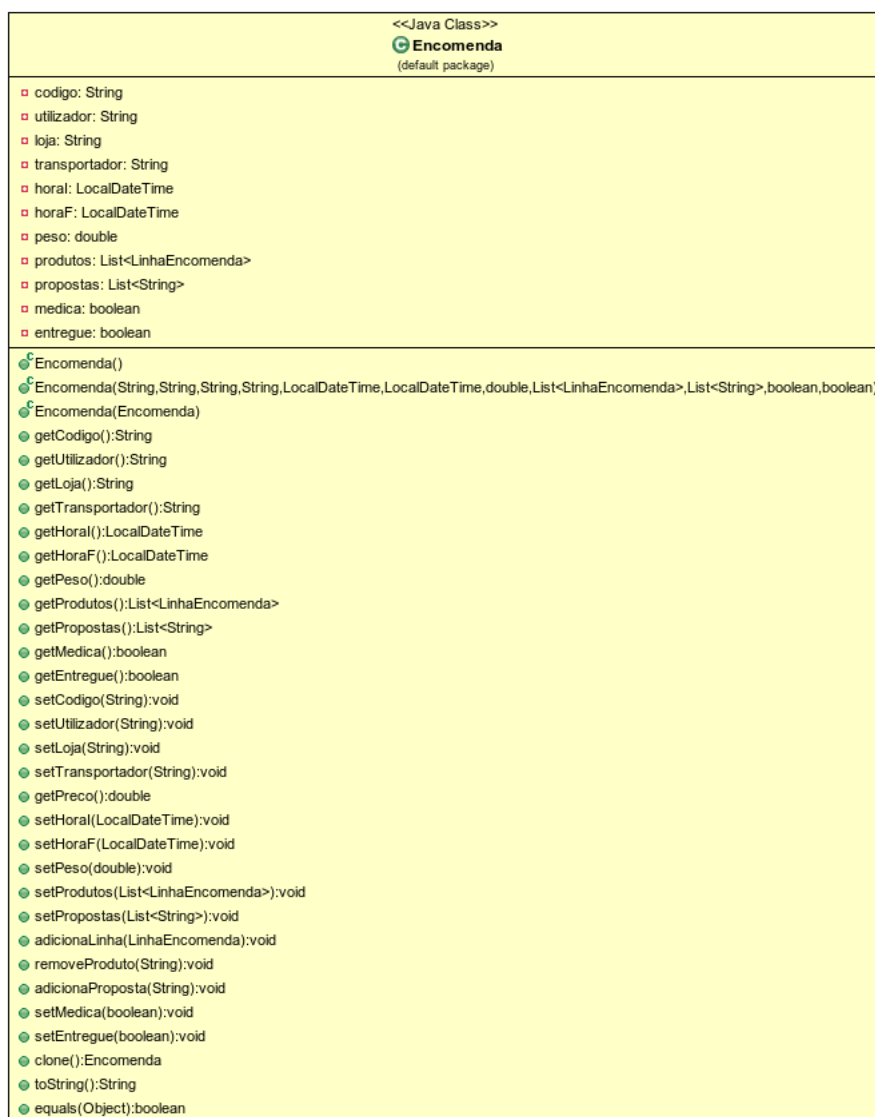


Figura 2.9: Encomenda

A classe EncomendasBD é que guarda todas as encomendas num HashMap, em que a chave é o username do utilizador que pede a encomenda, e o seu conteúdo é a Lista de Encomendas



Figura 2.10: EncomendasBD

A classe LinhaEncomenda guarda os dados de uma dada encomenda.

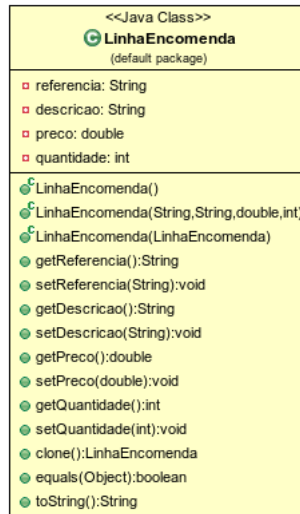


Figura 2.11: Linha de Encomenda

Temos a classe Empresa que fica com as informações de uma dada empresa que faça encomendas, ela trata das empresas de transporte.

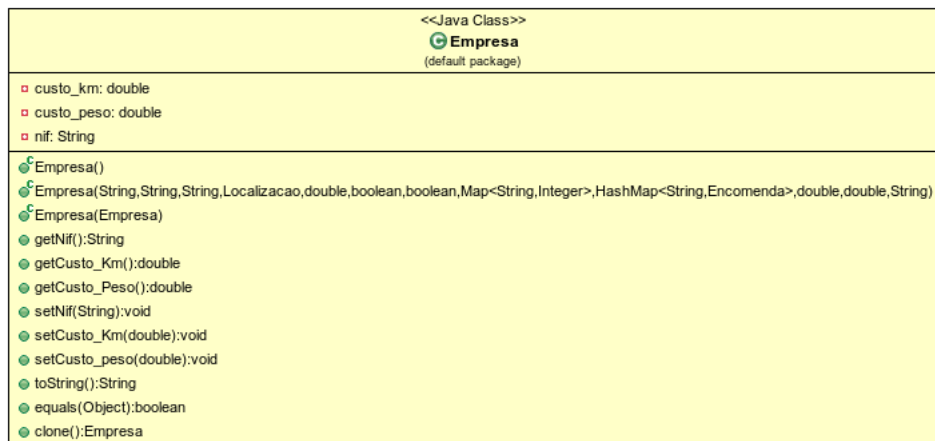


Figura 2.12: Empresa

Também temos a classe Menu que, como o nome diz, trata do Menu da aplicação, ou seja, interage com o usuário.

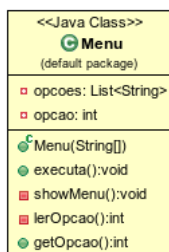


Figura 2.13: Menu

E finalmente temos a Classe TrazAquiApp que trata do sign ups, logins, lida com a várias opções que a aplicação traz.

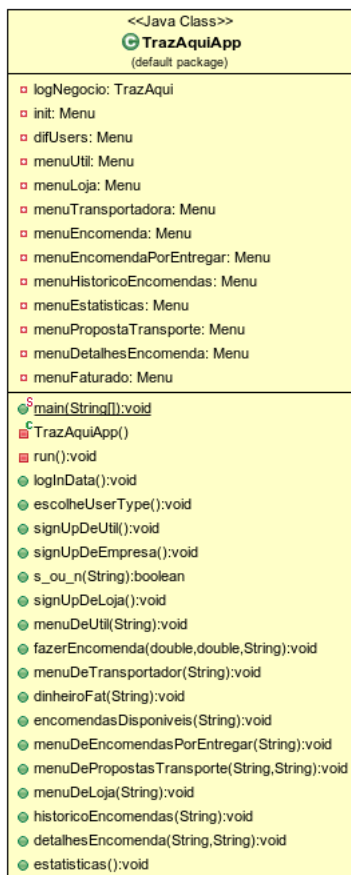


Figura 2.14: TrazAquiApp

[illegible]

13

Capítulo 3

Algumas Funções

A função abaixo ilustrada na figura, trata dos logins dos users, sejam eles empresas, utilizadores ou empresas Transportador, caso, ao fazer o login, o user digite 'u', será "tratado como Utilizador, caso digite "v" ou "t" será tratado como Transportador e caso digite "l", será tratado como Loja.

```
public void logInData() {  
    Scanner scin = new Scanner(System.in);  
  
    System.out.println("Username: ");  
    String username = scin.nextLine();  
    System.out.println("Password: ");  
    String password = scin.nextLine();  
    try {  
        this.logNegocio.existeUser(username,password);  
        if(username.charAt(0) == 'u') {  
            menuDeUtil(username);  
        }  
        if(username.charAt(0) == 'v' || username.charAt(0) == 't') {  
            menuDeTransportador(username);  
        }  
        if(username.charAt(0) == 'l') {  
            menuDeLoja(username);  
        }  
    }  
    catch (UserInexistenteException | IOException | EncomendaInexistenteException e) {  
        System.out.println(e.getMessage());  
    }  
}
```

Figura 3.1: função que trata do log in de um User

Como o próprio nome da função indica, esta função está encarregue de dar a opção a um user, que ao se registrar na app, possa escolher que tipo de user será, Utilizador, Loja ou Empresa

```
public void escolheUserType(){
    do {
        difUsers.executa();
        switch (difUsers.getOpcao()) {
            case 1:
                signUpDeUtil();
                break;
            case 2:
                signUpDeLoja();
                break;
            case 3:
                signUpDeEmpresa();
                break;
        }
    } while (difUsers.getOpcao() != 0);
}
```

Figura 3.2: Escolha do tipo de user (sign up)

Nas figuras abaixo, mostramos a função que trata do Sign Up de um utilizador. Divide-se basicamente em 2 passos:

- Inserção de dados
- Verificação de existência do User

```
public void signUpDeUtil(){
    Scanner scin = new Scanner(System.in);

    System.out.println("Username: ");
    String username = scin.nextLine();
    System.out.println("Nome: ");
    String nome = scin.nextLine();
    System.out.println("Password: ");
    String password = scin.nextLine();
    System.out.println("Idade: ");
    int idade = Integer.parseInt(scin.nextLine());
    System.out.println("Sexo: ");
    String sexo = scin.nextLine();
}
```

Figura 3.3: Recolha dos dados

```
try {
    this.logNegocio.existeUsername(username);
    this.logNegocio.adicionaUser(TrazAqui.criaUtilizador(username,nome,password,TrazAqui.criaLocalizacao(0,0),idade,sexo),sexo);
    this.logNegocio.guardaEstado("/home/simao/Desktop/Universidade/P00/ProjetoP001920/ProjetoP001920/estado.obj");
    menuDeUtil(username);
}
catch (UserInexistenteException e) {
    System.out.println(e.getMessage());
} catch (FileNotFoundException e) {
    System.out.println("Ficheiro objeto não encontrado");
} catch (IOException e) {
    System.out.println("Erro a guardar no ficheiro objeto");
} catch (EncomendaInexistenteException e) {
    e.printStackTrace();
}
```

Figura 3.4: Verificação

Nas figuras que se seguem temos a função que trata do Sign Up de uma Empresa que conta com uma função auxiliar s-ou-n que é usada na escolha caso o a empresa faça transportes de materiais medicos.

```
public void signUpDeEmpresa(){
    Scanner scin = new Scanner(System.in);

    System.out.println("Username: ");
    String username = scin.nextLine();
    System.out.println("Nome: ");
    String nome = scin.nextLine();
    System.out.println("Password: ");
    String password = scin.nextLine();
    System.out.println("NIF: ");
    String nif = scin.nextLine();
    System.out.println("Preço por km: ");
    double preco_km = Double.parseDouble(scin.nextLine());
    System.out.println("Preço por kg: ");
    double preco_kg = Double.parseDouble(scin.nextLine());
    System.out.println("Raio de ação: ");
    double raio = Double.parseDouble(scin.nextLine());
    System.out.println("Latitude: ");
    double latitude = Double.parseDouble(scin.nextLine());
    System.out.println("Longitude: ");
    double longitude = Double.parseDouble(scin.nextLine());
    System.out.println("Encontra-se disponível?[s/n]: ");
    boolean disponivel = s_ou_n(scin.nextLine());
    System.out.println("Faz o transporte de material medicinal?[s/n]: ");
    boolean medica = s_ou_n(scin.nextLine());
}
```

Figura 3.5: Recolha dos dados

```
try {
    this.logNegocio.existeUsername(username);
    this.logNegocio.adicionaUser(TrazAqui.criaEmpresa(username,nome,password,TrazAqui.criaLocalizacao(latitude,longitude,
                                                    new HashMap<>(),new HashMap<>(),preco_km,preco_kg,nif));
    this.logNegocio.guardaEstado("/home/simao/Desktop/Universidade/P00/ProjetoP001920/ProjetoP001920/estado.obj");
    menuDeUtil(username);
}
catch (UserInexistenteException e) {
    System.out.println(e.getMessage());
} catch (FileNotFoundException e) {
    System.out.println("Ficheiro objeto não encontrado");
} catch (IOException e) {
    System.out.println("Erro a guardar no ficheiro objeto");
} catch (EncomendaInexistenteException e) {
    e.printStackTrace();
}
```

Figura 3.6: Verificação

```
public boolean s_ou_n(String b){  
    if(b.equals("s")){  
        return true;  
    }  
    if(b.equals("n")){  
        return false;  
    }  
    return false;  
}
```

Figura 3.7: função s-ou-n

Temos ainda as funções de Menus, que dão opções de escolha ao user, seja ele Utilizador, Empresa ou Loja.

Diagrama

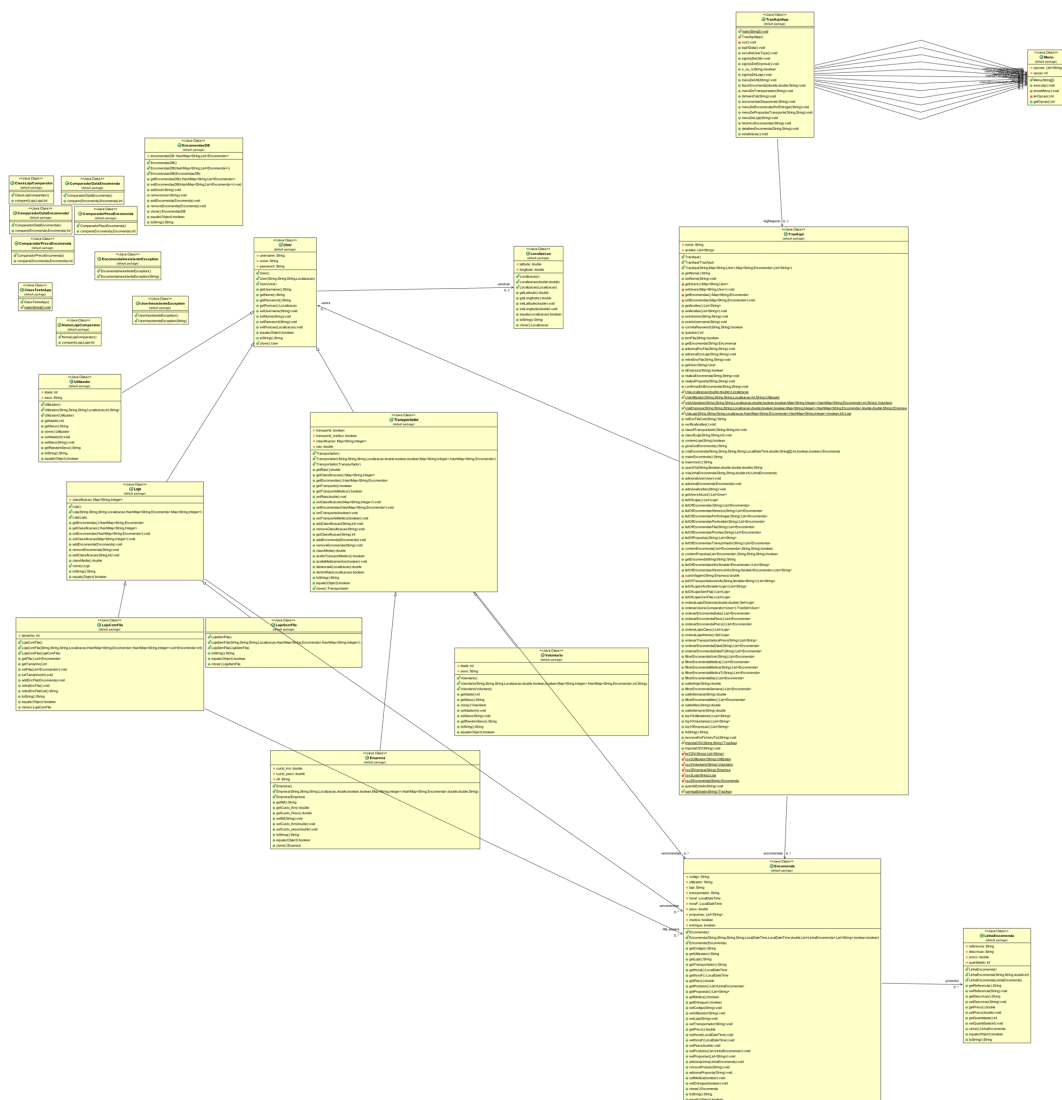


Figura 4.1: Diagrama

Capítulo 5

Conclusão

Para concluir, primeiro queremos agradecer aos professores, pois sabemos que este sementre foi "diferente" para todos e também expressar que á custa disso a realização deste trabalho tornou-se um pouco mais difícil mas o nosso grupo espera ter satisfeito todos os requisitos propostos para a correta realização desta tarefa.