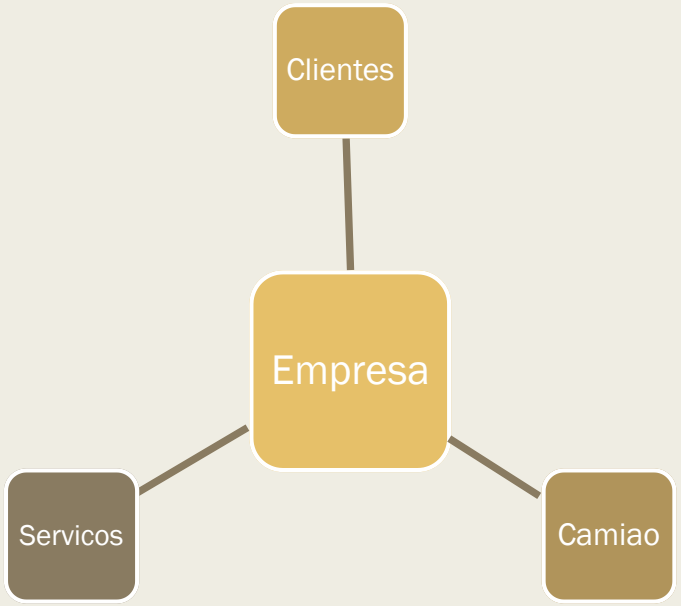
TRABALHO DE AEDA
GRUPO T4_G7

Allan Borges de Sousa	up201800149
Amanda de Oliveira Silva	up201800698
Juliane de Lima Marubayashi	up201800175

Descrição

- O Trabalho consiste no gerenciamento de uma empresa de transportes, a qual desloca mercadoria dos tipos: congelada, animal, perigosa e sem requisitos especiais.
- Os clientes da empresa são identificados pelo nome e NIF e, além disso, clientes podem requisitar serviços, os quais o preço varia de acordo com: as características de cada categoria, tamanho da carga, tipo de transporte e a distância do deslocamento traduzido em horas.
- Ainda, o gerenciamento da empresa inclui a visualização de dados em ordem a escolha do usuário e podendo ser relativo aos serviços, lucro e clientes.



Soluções

- Para o alocamento de camiões, optamos por alocá-los de forma que todos tenham lucros totais parecidos. Para isto, criamos uma variável totalProfit, que é incrementada com o preço de serviço toda vez que o camião é adicionado a um serviço (na inicialização do programa e no request do cliente). Usamos o sort da STL para organizar uma cópia do vetor de camiões em ordem crescente de totalProfit e **percorremos linearmente (pesquisa linear)** o vetor buscando os primeiros camiões que satisfaçam os requisitos de forma que a carga total do serviço seja menor ou igual a soma das cargas totais dos camiões.
- Camiões podem ser **removidos** e adicionados. A remoção não é de fato realizada. A carga é posta como negativo, assim a empresa manterá o registro do camião, mas este não será usado na alocação.
- Clientes podem ser adicionados, modificados e **removidos**. A remoção é feita de forma a por o NIF como negativo. Assim a empresa mantém o registro do ex-cliente, mas este não pode efetuar mais requests.
- A visualização de dados é realizada da seguinte maneira: como a ordem de como as classes estão dispostas no vetor importa, criamos uma cópia x da classe que queremos visualizar. Ordenamos x de acordo com o critério que o cliente deseja utilizando **sort da stl** e depois é feito o display das informações percorrendo o vetor x linearmente.
- O cálculo do tempo de viagem em horas foi realizado utilizando uma fórmula a qual considera a latitude e longitude do local de partida e chegada.

Alocação

- Aloca serviço de 600€, onde camiõess 4 e 3 satisfazem as condições com carga 20:

Inicio [Camiões sem sort]:

Id:1 Total Profit: 1000€ Carga: 40	id:2 Total Profit:1200€ Carga: 5	id:3 TotalProfit: 400€ Carga: 30	id:-4 TotalProfit: 200€ Carga: 30

Meio [Camiões com sort]:

Id:-4 200€ 30	id:3 400€ 30	id:1 1000€ 40	id:2 1200€ 5

Fim [Camiões alocados]:

Id:-4 200€ 30	id:3 400€ 30	id:1 1000€ 40	id:2 1200€ 5

Como o camião 4 foi removido, não o usamos para a alocação e, já que o camião 3 tem carga o suficiente para o transporte, será utilizado e TotalProfit será incrementado.

Diagrama de classes

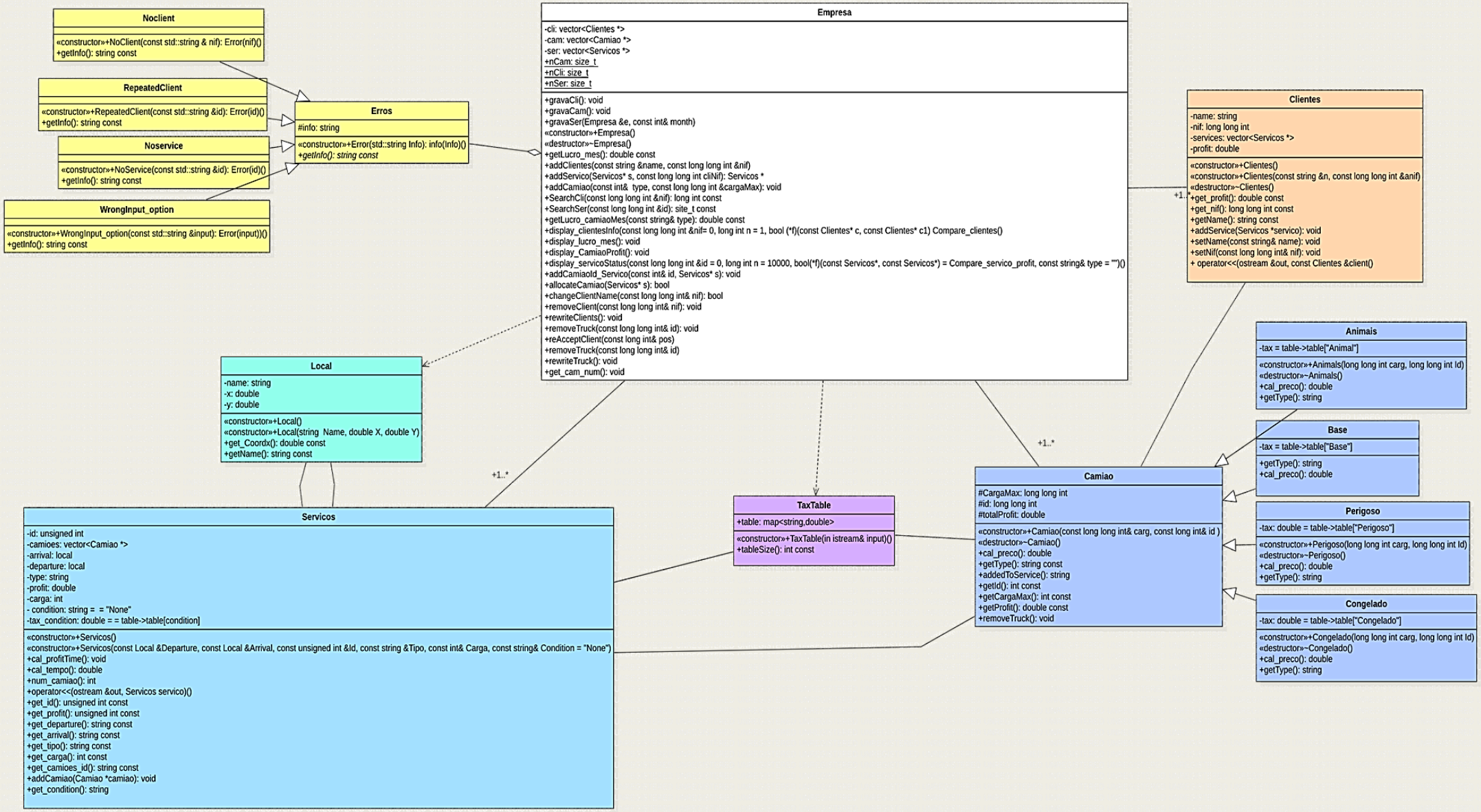


Diagrama – Empresa e Clientes

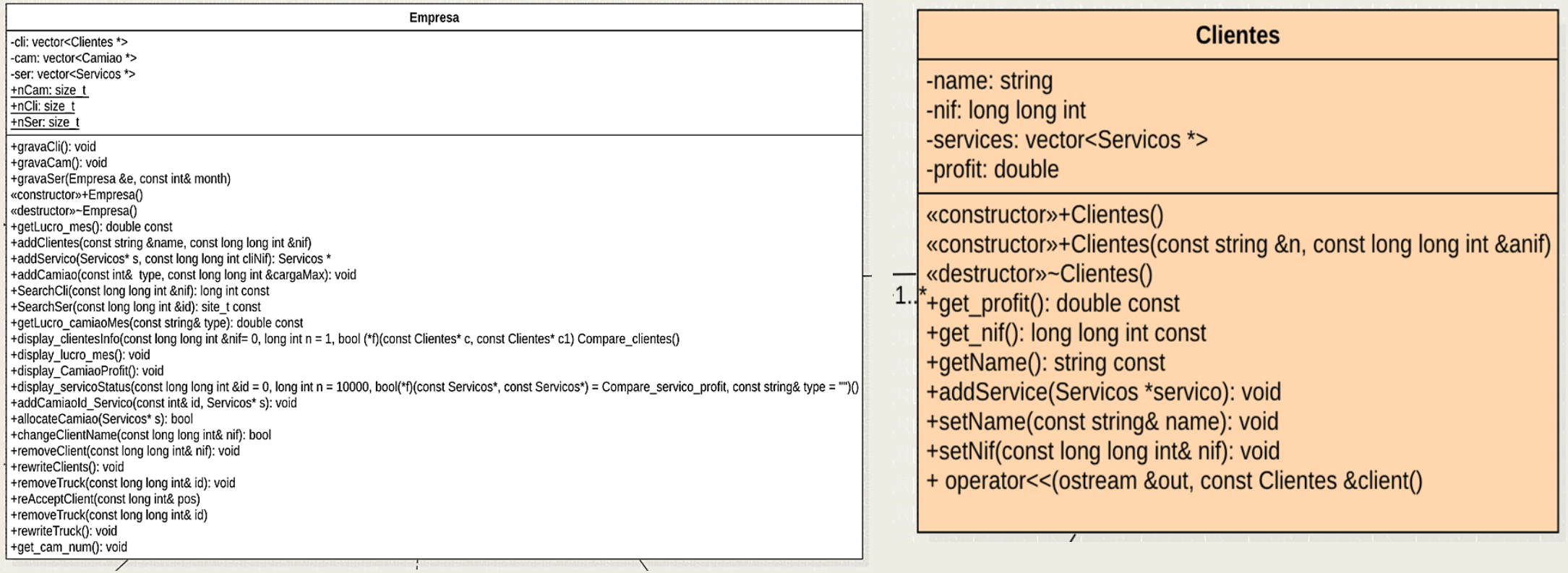
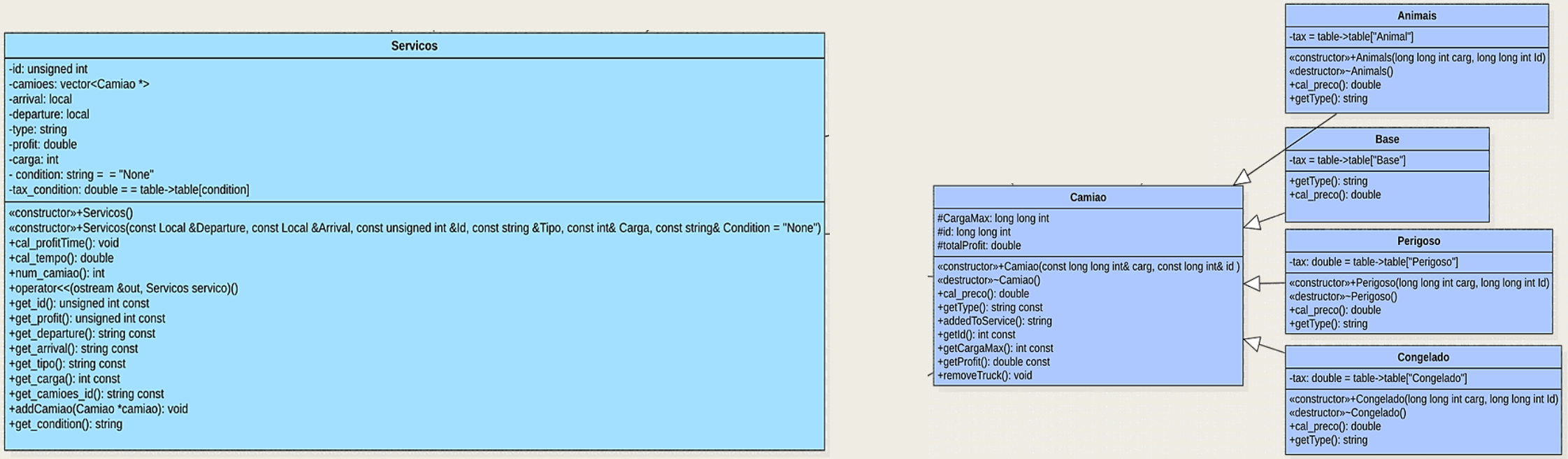


Diagrama – Serviços e Camiao



Estrutura de ficheiros

- camioes.txt

«blank»

Carga [int]

Tipo [string]

- Tipos possiveis:

-Congelado
-Perigoso
-Animal
-Base

- servicos.txt

«blank»

Local de partida [string]

Latitude de partida [double]

Longitude de partida [double]

Local de destino [string]

Latitude de destino [double]

Longitude de destino [double]

Tipo de transporte [string]

Nif do cliente [long int]

Caracterisitca [string]

Ids camioes separados por espaço

- Caracteristicas
  - Se congelado:

Frio

Medio

Quente

  - Se Perigoso:

Toxico

Inflamavel

Quebravel

  - Se Animal

Pequeno

Medio

Grande

  - Se base:

None

- clientes.txt

Nome [string]

Nif [long int]

- tax.txt

Tipo/Caracterisitca [string]

Taxa [double]

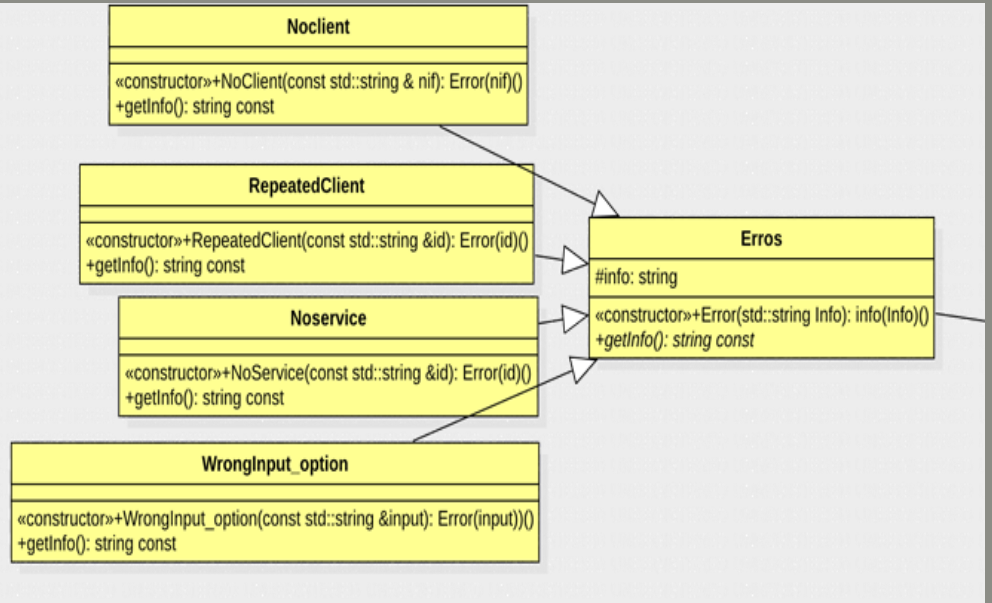
- Observações:

Não podem haver linhas extras no fim dos arquivos. Todas os tipos e características devem ter uma taxa.

Excessões

Erros tratados:

- Cliente Inexistente [NoClient]
- Cliente Repetido [Repeated Client]
- Serviço Inexistente [NoService]
- Input Errado [WrongInput_option]



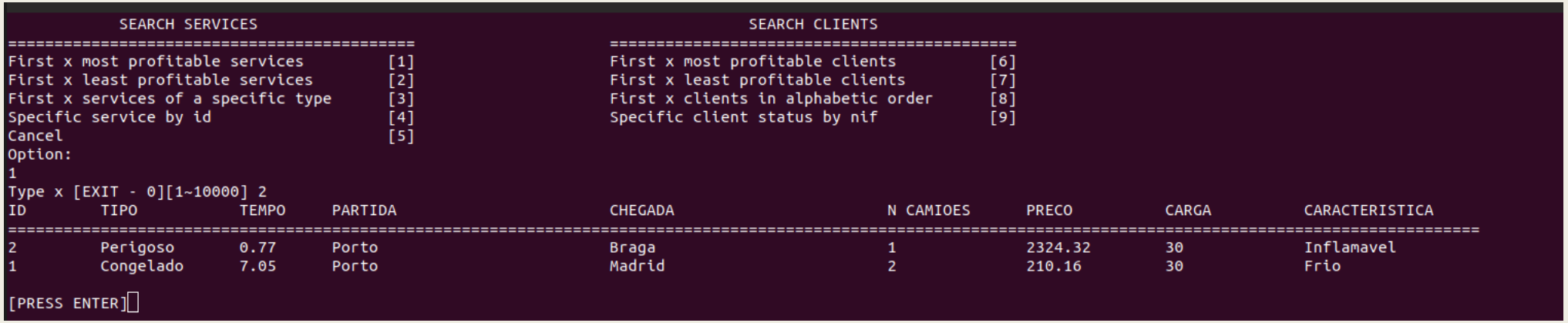
Exemplo de excessão:

```
int checkOption(int min, int max) {
    int input;

    while (true) {
        try {
            cin >> input;
            if (cin.fail()) {
                cin.clear();
                cin.ignore(1000, '\n');
                throw WrongInput_option("Invalid Input. Please enter an integer");

            }
                //if it's not in the interval
            else if (input > max || input < min) {
                cin.clear();
                cin.ignore(1000, '\n');
                throw WrongInput_option("Given input is not an option. Try again");
            } else
                return input;
        }
        catch (WrongInput_option &error) {
            cout << error.getInfo() << endl;
            continue;
        }
    }

}
```

Funcionalidades - Listagem

- Primeiros x servicos mais rentavais [OK]
- Primeiros x servicos menos rentaveis [OK]
- Primeiros x servicos mais rentaveis de um tipo especifico [OK]
- Primeiros x clientes mais rentaveis [OK]
- Primeiros x clientes menos rentaveis [OK]
- X clientes em ordem alfabética [OK]
- Informação sobre lucro de tipos de camiões [OK]

Exemplo de pesquisa de serviços mais rentáveis;



Funcionalidades - Pesquisa

- Serviço com id específico [OK]
- Cliente com nif específico [OK]

Exemplo de pesquisa de Id:

```
           SEARCH SERVICES                                    SEARCH CLIENTS
========================================          ========================================
First x most profitable services         [1]      First x most profitable clients         [6]
First x least profitable services        [2]      First x least profitable clients        [7]
First x services of a specific type      [3]      First x clients in alphabetic order     [8]
Specific service by id                   [4]      Specific client status by nif           [9]
Cancel                                   [5]
Option:
4
Type the id: 1
ID        TIPO          TEMPO     PARTIDA                    CHEGADA                    N CAMIOES      PRECO          CARGA          CARACTERISTICA
=================================================================================================================================================
1         Congelado     7.05      Porto                      Madrid                     2              210.16         30             Frio

[PRESS ENTER]
```

Funcionalidades - CRUD

- Adicionar camiões [OK]
- Remover camião [OK]
- Adicionar cliente [OK]
- Mudar nome do cliente [OK]
- Remover cliente [OK]
- Novo request de serviço [OK]

Exemplo de novo request de serviço:

```
Enter the number of products [EXIT -1] 20
Enter place of departure [EXIT -1] Rua dos Anjos
Enter place of arrival [EXIT -1] Alameda Fonseca
Enter type of package (0 BASE | 1 FROZEN | 2 DANGEROUS | 3 ANIMAL) [EXIT -1] 1
Type the nif [EXIT -1]  201800175
Partida coordenada x (latitude) [EXIT -1] 20.4
Partida coordenada y (Longitude) [EXIT -1] 20.6
Chegada coordenada x (Latitude) [EXIT -1] 14.2
Chegada coordenada y (Longitude) [EXIT -1] 13.2
Enter type temperature of the products (0 COLD | 1 AMBIENT | 2 HOT) [EXIT -1]1
Service added successfully!
[PRESS ENTER]
```

Destaque de funcionalidade

Todas as funcionalidade foram bem implementadas.

No contexto de demonstração, gostariamos demostrar a funcionalidade de adição de novos clientes.

Para adicionar um novo cliente seguimos os passos:

- Pede-se o nif e verifica-se se este é aceito
- O nif do novo cliente é procurado na lista de clientes da empresa verificando se este é um ex-cliente. Caso sim, o cliente é reaceito, caso não passamos para a próxima etapa.
- O nif do novo cliente é procurado na lista de clientes da empresa para evitar repetições. Caso o cliente já exista é lançada uma excessão. O seu tratamento consiste em afirmar o erro e pedir novamente o nif. O usuário pode desistir da ação precionando -1.

```
void handleAddClient(Empresa &e){
    long long int nif;
    string nome;
    cout<<"Number of clients: "<< Empresa::nCli <<endl;


    while (true) {
        cout<<"NIF: ";
        nif = checkNumber();
        if (nif == 0 || nif < -1){
            cout << "Invalid NIF. Try again." << endl;
             continue;                                                  //do not accept negative nifs
        }
        if (nif == -1) return;

        //case it was a ex-client
        long int pos = e.SearchCli((-1) * nif);
        if (pos != -1){
            e.reAcceptClient(pos);
            cout << "Client reaccepted! ";
            wait();
            return;
        }

        cout<<"Nome [EXIT -1]: ";
        cin.ignore();
        getline(cin,nome);
        if (nome == "-1") return;                                       //abort operation


        try {
            e.addClientes(nome, nif);
            ofstream o("../AEDA_Proj1/Ficheiros/clientes", ios_base::app);
            o << "\n" << nome << "\n" << nif;
            o.close();
            cout << "Client added successfully! ";
            wait();
            return;
        }
        catch (RepeatedClient &a) {
            cout << "There is already a client with nif "<< nif << endl;
            cout << "Try again:" << endl;
            continue;
        }
    }
}
```

Dificuldade encontradas

Em geral, não houve grandes dificuldades no projeto, porém algumas funções trabalhosas foram:

- Criação do UML
- Achar solução eficaz para o cálculo do lucro dos camiões
- Achar solução para calculo do tempo da distância em horas

Esforço de cada elemento:

- Allan Borges de Sousa ~ 25%
- Amanda de Oliveira Silva ~20%
- Juliane de Lima Marubayashi ~55%