UGHEATS
FOOD DELIVERY

Realizado por:
Mariana Ramos (up201806869)
Raquel Sepúlveda (up201806664)
Rita Silva (up201806527)

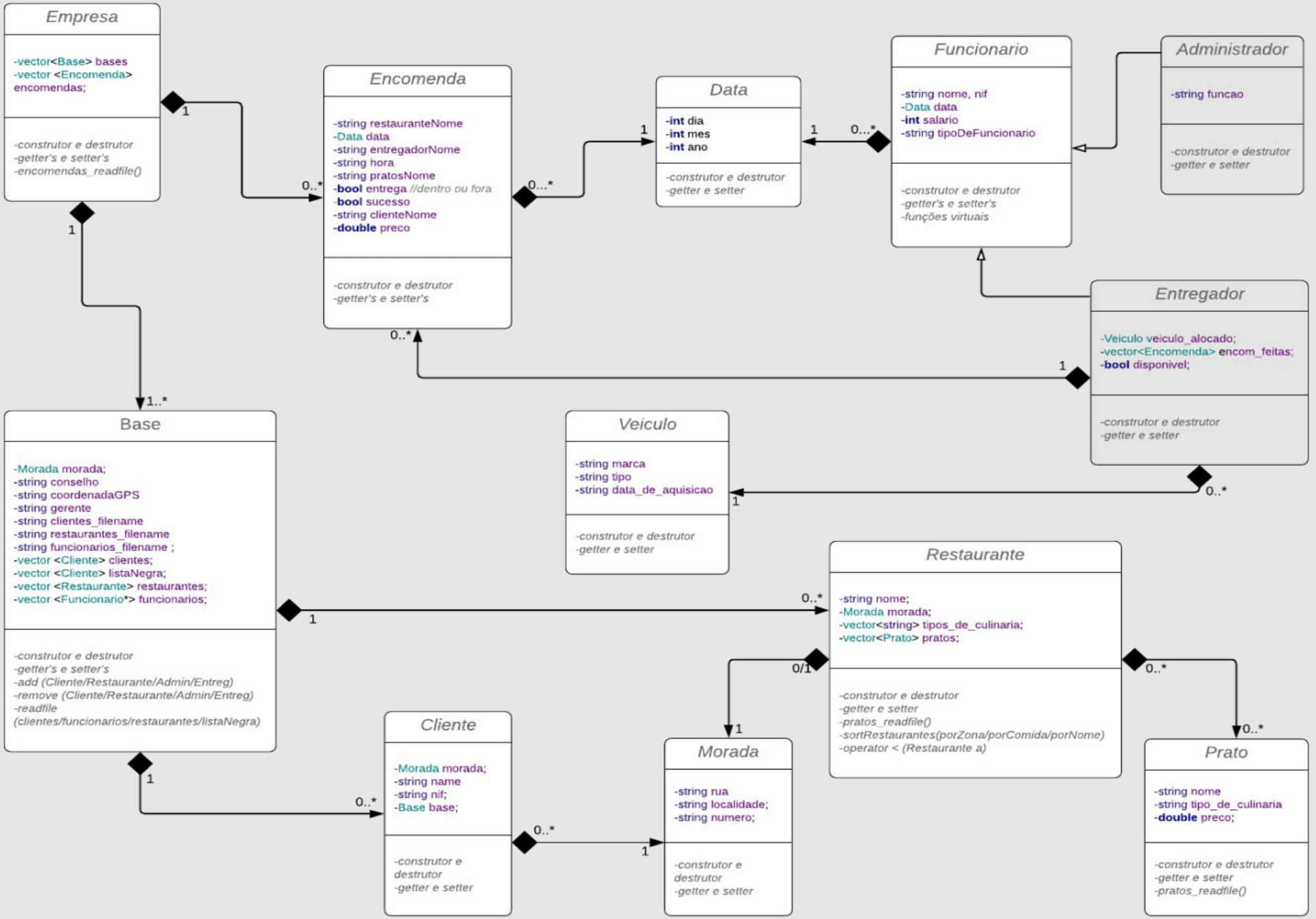
Sumário

Descrição do problema

- Pretende-se construir uma aplicação para gerir a atividade da “UghEats”
  - *Gerir a atividade dos funcionários, clientes, fornecedores e encomendas/entregas;*
  - *Monitorar os clientes e fornecedores e consultar as respetivas encomendas/entregas;*
  - *Conhecer os valores que a empresa retira dos seus serviços, por base, por fornecedor, por cliente e no geral;*
  - *Manter um registo de todos os veículos da empresa usados na entrega de encomendas – através da utilização de* ***árvores binárias de pesquisa****.*
  - *Gerir serviços de técnicos especializados em manutenção de veículos que são guardados numa* ***fila de prioridade.***
  - *Manter registo de todos os funcionários numa* ***tabela de dispersão.*** *Fazer distinção entre funcionários antigos e atuais.*

Soluções e algoritmos relevantes

- Algoritmo para remover/adicionar;
- Algoritmo de leitura/escrita de ficheiros;
- Algoritmo para calcular os lucros da empresa;
- Implementação do operador < (usado para fazer sort());
- Ordenação de vetores usando sort();
- Implementação de árvores binárias de pesquisa contendo os veículos da empresa;
- Implementação de uma fila de prioridade que contém os técnicos de manutenção de veículos ordenados pela sua disponibilidade;
- Implementação de uma tabela de dispersão que contém todos os funcionários da empresa incluindo os que já foram despedidos;


Empresa
-vector<Base> bases
-vector <Encomenda> encomendas;
-construtor e destrutor
-getter's e setter's
-encomendas_readfile()
Encomenda
-string restauranteNome
-Data data
-string entregadorNome
-string hora
-string pratosNome
-bool entrega //dentro ou fora
-bool sucesso
-string clienteNome
-double preco
-construtor e destrutor
-getter's e setter's
Data
-int dia
-int mes
-int ano
-construtor e destrutor
-getter e setter
Funcionario
-string nome, nif
-Data data
-int salario
-string tipoDeFuncionario
-construtor e destrutor
-getter's e setter's
-funções virtuais
Administrador
-string funcao
-construtor e destrutor
-getter e setter
Entregador
-Veiculo veiculo_alocado;
-vector<Encomenda> encom_feitas;
-bool disponivel;
-construtor e destrutor
-getter e setter
Base
-Morada morada;
-string conselho
-string coordenadaGPS
-string gerente
-string clientes_filename
-string restaurantes_filename
-string funcionarios_filename ;
-vector <Cliente> clientes;
-vector <Cliente> listaNegra;
-vector <Restaurante> restaurantes;
-vector <Funcionario*> funcionarios;
-construtor e destrutor
-getter's e setter's
-add (Cliente/Restaurante/Admin/Entreg)
-remove (Cliente/Restaurante/Admin/Entreg)
-readfile (clientes/funcionarios/restaurantes/listaNegra)
Veiculo
-string marca
-string tipo
-string data_de_aquisicao
-construtor e destrutor
-getter e setter
Restaurante
-string nome;
-Morada morada;
-vector<string> tipos_de_culinaria;
-vector<Prato> pratos;
-construtor e destrutor
-getter e setter
-pratos_readfile()
-sortRestaurantes(porZona/porComida/porNome)
-operator < (Restaurante a)
Cliente
-Morada morada;
-string name
-string nif;
-Base base;
-construtor e destrutor
-getter e setter
Morada
-string rua
-string localidade;
-string numero;
-construtor e destrutor
-getter e setter
Prato
-string nome
-string tipo_de_culinaria
-double preco;
-construtor e destrutor
-getter e setter
-pratos_readfile()
1
0..*
0...*
1..*
0/1

Estrutura de ficheiros

- Bases:
  1. *Rua*
  2. *Concelho*
  3. *Número da porta*
  4. *Base*
  5. *Coordenadas GPS*
  6. *Nome do gerente*
  7. *Ficheiro dos clientes da base respetiva*
  8. *Ficheiro dos restaurantes da base respetiva*
  9. *Ficheiro dos funcionários da base respetiva*
- Clientes:
  1. *Nome*
  2. *NIF*
  3. *Morada (rua, concelho, numero)*
  4. *Base*
- Restaurantes:
  1. *Nome*
  2. *Morada*
  3. *Tipo de culinária*
  4. *Ficheiro dos pratos*

```
Rua de Berlin
Se
13
Faro
37.0188205 -7.9189382
Diana Almeida
clientesFaro.txt
restaurantesFaro.txt
funcionariosFaro.txt
```

```
Paulo Ribeiro
162657389
Rua Dr. Antonio Macedo, Gondomar, 17
Porto
:::::
```

```
Limoncello
Rua Dr. Nunes da Ponte, Matosinhos, 54
Italiana
pratos.txt
:::::
```

Estrutura de ficheiros

- Funcionários:
  - *Entregador*
    1. Nome
    2. NIF
    3. Data de nascimento (dia/mês/ano)
    4. Salário
    5. Veiculo alocado (Marca, Modelo, Data de alocação)
    6. Disponibilidade
  - *Administrador*
    1. Nome
    2. NIF
    3. Data de nascimento (dia/mês/ano)
    4. Salário
    5. Cargo

```
Entregador
Armando Rocha
3574936538
17/10/1976
600
Suzuki, Katana, 09/10/2019
1
:::::
Administrador
Alvaro Amaral
124892353
1/10/1973
1200
CFO
:::::
```

- Pratos:
  1. *Nome*
  2. *Tipo de culinária*
  3. *Preço*

```
Massa a bolonhesa
Italiana
8
:::::
```

- Lista negra de clientes:
  - *Mesma estrutura dos ficheiros de clientes*

Estrutura de ficheiros

- Encomendas:
  - *Nome do restaurante*
  - *Data*
  - *Nome do entregador*
  - *Hora da realização do pedido*
  - *Pratos encomendados*
  - *Encomenda realizada dentro ou fora do concelho do cliente*
  - *Encomenda realizada com sucesso*
  - *Nome do cliente*
  - *Preço total*

```
Le Moustache
15/11/2019
Mariana Ramos
11:18
Bacalhau a Bras, Francesinha
1
1
Raquel Sepulveda
34
:::::
```

Tratamento de exceções

- Exceções lançadas quando:
  - *É tentado registar um cliente que se encontra na lista negra*
  - *É tentado remover um funcionário que não existe*
  - *É tentado remover um restaurante que não existe*

Lista de funcionalidades implementadas

- Clientes:
  - *Login*
    - Realizar encomendas:
      - *Procurar restaurantes por:*
        1. Zona geográfica (OK)
        2. Ordem alfabética (OK)
        3. Tipo de culinária (OK)
        4. Intervalo de preço (OK)
    - Ver sua informação (OK)
    - Eliminar o seu registo (OK)
  - *Registo (OK)*

- Funcionários :
  - *Administrador:*
    - Adicionar / remover funcionário (OK)

    Com nova política de contratar funcionários já conhecidos (OK)
    - Adicionar / remover restaurante (OK)
    - Ver lista de funcionários:
      - *Por base (OK)*
      - *De todas as bases (OK)*
  - *Entregador:*
    - Ver suas entregas (OK)
    - Ver técnico mais próximo de estar disponível (OK)

    - Técnico

Lista de funcionalidades implementadas

- Gestão:
  - *Economia (OK)*
    - Ver lucro da empresa por:
      1. *Base (OK)*
      2. *Cliente (OK)*
      3. *Restaurante (OK)*
      4. *Mês (OK)*
      5. *Geral (OK)*
  - *Ver lista de :*
    - Funcionários: ----- Antigos e Atuais (OK)
      1. *Por base (OK)*
      2. *De todas as bases (OK)*
    - Restaurantes:
      1. *Por base (OK)*
      2. *De todas as bases (OK)*
    - Clientes :
      1. *Por base (OK)*
      2. *De todas as bases (OK)*
    - Veículos ordenados (OK)
    - Encomendas:
      - *Por base (OK)*
      1. *Por restaurante (OK)*
      2. *Por entregador (OK)*
      3. *Todas (OK)*

Destaque de funcionalidade

- Realizar encomendas:
  - *Permite realizar encomendas baseando-se em vários fatores:*
    - Zona geográfica , intervalo de preço, tipo de culinária
  - *Permite realizar encomendas de um ou mais pratos*
  - *Permite que o cliente que realize a encomenda verifique o seu pagamento através de um código enviado*
    - Se este não confirmar o mesmo, será removido da lista de clientes e adicionado à lista negra da empresa
    - Se este confirmar o mesmo, será informado do tempo de chegada da sua encomenda, o seu entregador e o veículo do mesmo

Principais dificuldades encontradas

- Voltar de uma função para a anterior
- Validação dos inputs
- Organização de classes

Exemplos de execução:

Árvores binárias de pesquisa

```
void Empresa::addVeiculo(const Veiculo& v){
    veiculos.insert(v);
}
vector<Veiculo> Empresa::verVeiculo(){
    BSTItrIn<Veiculo> it( bt: veiculos);
    vector<Veiculo> vs;
    string m;
    Veiculo v;
    int i=0;
    while (!it.isAtEnd()){

        v=it.retrieve();
        vs.push_back(v);
        it.advance();
    }

    return vs;
}
```

Exemplos de execução:

Fila de prioridade

```
HEAP_TECNICO Base::getTecnicoFila() const{
    return tecnicos_fila;
}
void Base::addTecnicoToFila(Funcionario *r){
    tecnicos_fila.push(r);
}
void Base::removeTecnicoFromFila(string nif){
    HEAP_TECNICO aux;
    bool existe = false;
    if(tecnicos_fila.empty()){
        existe=false;
    }
    while(!existe || !tecnicos_fila.empty()){
        if(tecnicos_fila.top()->getNif()==nif){
            existe=true;
        }
        else{
            aux.push( x: tecnicos_fila.top());
        }
        tecnicos_fila.pop();
    }
    while(!aux.empty()) {
        tecnicos_fila.push(aux.top());
        aux.pop();
    }

}
```

Exemplos de execução:
Tabela de dispersão