GESTÃO DE CRUZEIROS DO RIO DOURO

Um projeto para a unidade curricular de aeda, no 2º ano do MIEIC

Realizado pelo grupo 8 da turma 3:

- Carlos Vieira; Nº de Estudante: up201606868
  E-mail: up201606868@fe.up.pt
- João Álvaro Ferreira; Nº de Estudante: up201605592;
  E-Mail: up201605592@fe.up.pt
- João Carlos Maduro; Nº de Estudante: up201605219;
  E-Mail: up201605219@fe.up.pt

Data: 15 de Novembro de 2017

GESTÃO DE CRUZEIROS DO RIO DOURO

ÍNDICE

Tema do trabalho 3
Implementação da solução para a proposta dada 4
Relatório UML 7
Casos de Utilização 8
    Classes 8
    Ficheiros de texto 10
Dificuldades Sentidas 11
Contribuição de cada membro 12

GESTÃO DE CRUZEIROS DO RIO DOURO

TEMA DO TRABALHO

A proposta de trabalho que escolhemos implica o desenvolvimento de software para a gestão de uma empresa que funciona como agente de reservas. A empresa, "Porto Rivers", faz isto criando estruturas de dados tanto para os clientes como para os fornecedores, facilitando a interação entre ambos o máximo possível.

Esta proposta implica, para além de criar estruturas para guardar informação de clientes e de fornecedores, fazer o mesmo para as ofertas dos fornecedores (que a estes estão associadas) e para as reservas efetuadas pelos clientes. Fazemos isto atendendo aos pormenores indicados para cada fornecedor, diferenciando entre clientes registados ou não (registo este que implica acumulação de pontos que podem ser gastos para receber descontos).

Foi nos exigido que desenvolvêssemos métodos para gerir a informação descrevida utilizando classes, servindo-nos de polimorfismo e herança para estas mesmas classes, estruturas lineares (arrays, vetores, etc.), que passássemos a informação guardada para ficheiros e que qualquer erro do utilizador fosse detetado com uma exceção – algo que fizemos, como será descrito mais a fundo na próxima secção.

GESTÃO DE CRUZEIROS DO RIO DOURO

IMPLEMENTAÇÃO DA SOLUÇÃO PARA A PROPOSTA DADA

O primeiro passo no desenvolvimento do programa foi implementar as classes dos fornecedores, das ofertas, das reservas e dos clientes, sendo que esta última tem como subclasse os clientes reservados. Estas classes interagem constantemente, sendo, por exemplo, um dos membros privados da classe fornecedores um vetor com todas as ofertas que tem disponíveis.

Geramos uma classe Time, e fizemos overload aos operadors "<" e "==", de modo a facilitarmos e tornarmos uniforme o uso de datas e horas nas funções que estas requerem (principalmente no que toca a reservas). Para obter o tempo atual, criamos também uma classe RealTime, derivada da classe Time.

De modo tornar mais direta a manipulação de dados, implementamos também uma class Empresa com usos mais generalistas – tem vetores dos fornecedores, dos clientes e das reservas, e métodos que manipulam estes vetores de acordo com o input do utilizador (que é feito em funções dos menus, que também são métodos desta classe).

Estes vetores são ordenados com algoritmos sort e é feita a pesquisa de elementos através de vários tipos de pesquisa binária ou sequencial, dependendo do tipo de estrutura dentro da qual estamos a pesquisar por um objeto.

Para além disto, criamos os métodos que passam toda a informação guardada nestas classes para ficheiros de texto individuais (clientes.txt, clientesregistados.txt, fornecedores.txt e reservas.txt), de modo a não perder a informação após o fecho do programa.

Após a estruturação inicial, foram feitos os menus e toda a interface com que o utilizador irá interagir. Foi criado um sub-menu para os métodos de cada classe previamente referida, desde o registo, modificação e eliminação de clientes ou fornecedores, à criação de novas reservas ou adição de ofertas para um fornecedor.

Nesta fase, tratamos de alguns pormenores particulares ao nosso projeto, como a implementação do sistema de acumulação e consumo de pontos e o sistema de cancelamento de reservas.

Implementamos deteção de exceções (com try, throw e catch) nos inputs do utilizador que levariam a uma procura por um objeto com um

nome específico, de forma a lidar com a exceção caso este não fosse encontrado por qualquer pesquisa. Isto foi implementado via classes template.

Na segunda fase do trabalho, criamos uma Binary Search Tree para guardar as faturas por ordem alfabética (sendo que clientes com várias faturas têm-nas organizadas das mais antigas para as mais recentes. Implementamos também uma hash table para guardar os clientes que não efetuam uma reserva há mais de 180 dias e uma priority queue, que organiza as ofertas com base na ultima reserva feita de ofertas com o mesmo tipo de barco. Por fim, expandimos o trabalho feito na primeira fase, tendo agora bastantes novas funcionalidades (como, por exemplo, ver todas as reservas efetuadas a um fornecedor, ou todos os clientes que reservaram uma determinada oferta).

Abrindo o programa pela primeira vez, assumindo que todos os registos estão limpos, é necessário começar pela adição de novos clientes (que podem ser considerados registados, permitindo acumulação de pontos), fornecedores e respetivas ofertas. Existindo fornecedores com ofertas disponíveis e clientes para as reservas, torna-se possível efetuar uma reserva.

O utilizador, sendo parte da Empresa Porto Rivers, poderá alterar ou remover qualquer propriedade de qualquer um destes objetos. Os clientes poderão cancelar reservas, tendo ainda que pagar dependendo da proximidade da data da cancelação à data da viagem. Poderão também ganhar pontos (se tiverem zero), recebendo pontos iguais a 1/5 do preço da reserva, ou gastar os acumulados abatendo diretamente ao preço da reserva feita.

É possível ver a informação guardada de variadíssimas maneiras, desde ver todas as reservas de um cliente, todos os clientes que reservaram uma particular oferta, todos os fornecedores, todas as ofertas disponíveis, entre outras;

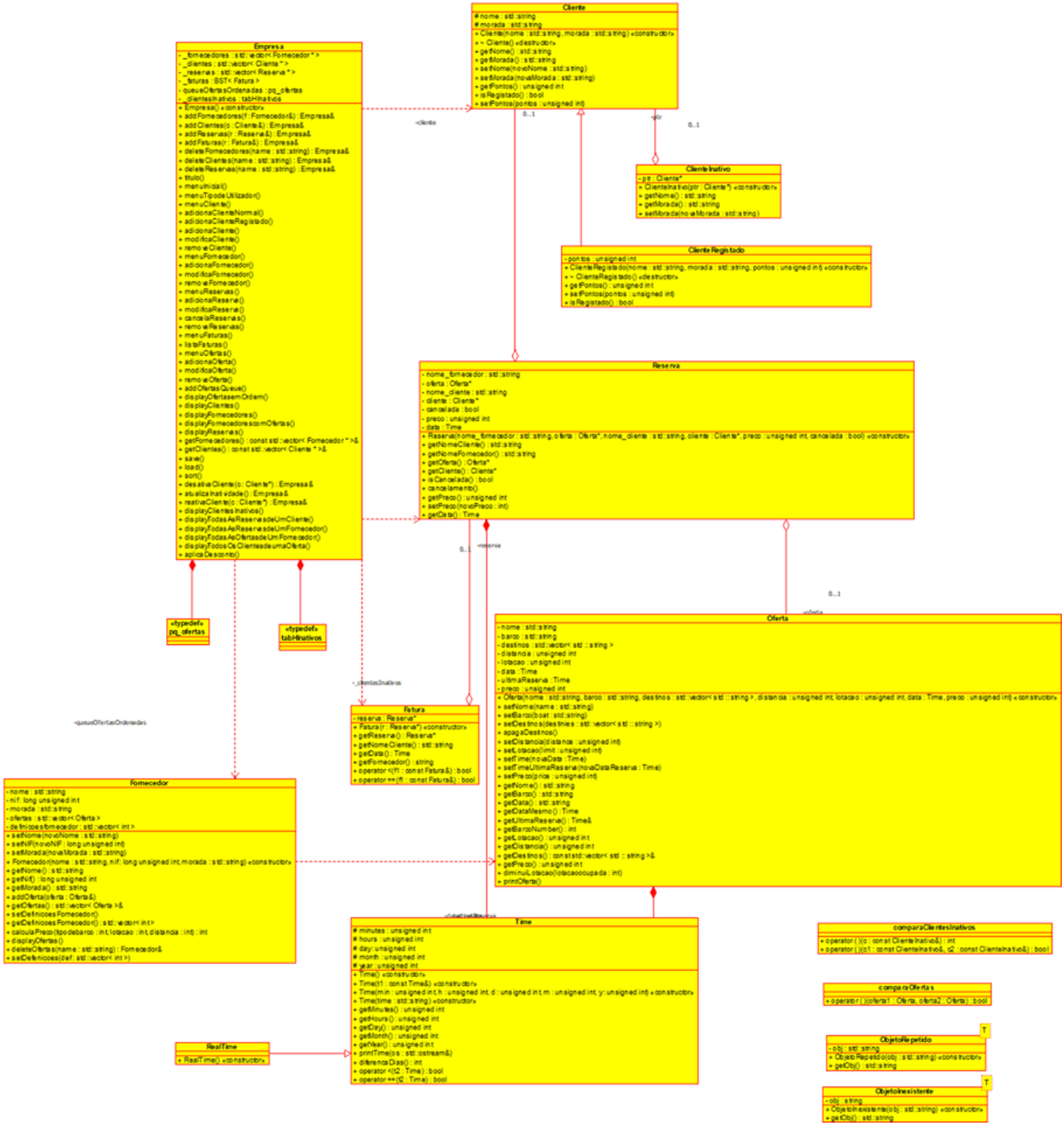
GESTÃO DE CRUZEIROS DO RIO DOURO

Importante e disponível também é: a organização das ofertas pela data mais antiga que alguma com o mesmo barco tenha sido marcada, de modo a dar descontos ao tipo de barco que tem sido marcado menos recentemente; a visualização das faturas das reservas, organizadas por ordem alfabética e, para clientes com histórico de muitas reservas, das mais antigas para as mais recentes e a tabela de dispersão com registos dos clientes inativos, de modo a poder enviar publicidade para as suas moradas (e a possível re-ativação destes mesmos).

GESTÃO DE CRUZEIROS DO RIO DOURO

RELATÓRIO UML

A visualização deste num documento PDF não é de elevada qualidade devido ao grande tamanho, logo será também enviado dentro do ZIP em separado.



GESTÃO DE CRUZEIROS DO RIO DOURO

CASOS DE UTILIZAÇÃO

CLASSES

EMPRESA

A classe Empresa é a nossa classe principal, onde implementamos todas as nossas funções de menu (que utilizamos em menu.cpp) e onde temos os três vetores dos elementos registados Empresa em questão, os Clientes, os Fornecedores e as Reservas. Os clientes incluem tanto ativos como inativos.

Para além disto, esta classe contém também as estruturas que implementamos na segunda parte do trabalho, nomeadamente uma BST para organizar as faturas, uma priority queue das ofertas (ordenadas pela data mais recente que uma oferta com o tipo de barco igual foi reservada, dando prioridade às mais antigas) e uma hash table para os clientes inativos (estando nela clientes que não efetuam uma reserva há mais de 180 dias).

Utilizar esta classe como ponto central do nosso trabalho permite-nos organizarmos e acedermos a toda a nossa informação mais facilmente.

OFERTA

A classe Oferta tem os membros: nome, barco, destinos(vetor de destinos da viagem),distancia, lotacao, data, ultimaReserva (data da ultima reserva feita a uma oferta com o mesmo tipo de barco da oferta em questão) e preco.

FORNECEDOR

A classe Fornecedor tem os membros: nome, nif, morada, ofertas (vetor com todas as ofertas deste fornecedor) e as definições de fornecedor (um vetor com ints representando os preços base para cada tipo de barco e o multiplicador para a lotação)

GESTÃO DE CRUZEIROS DO RIO DOURO

CLIENTE

A classe Cliente tem os membros: nome e morada.

A classe Cliente tem como derivada a classe Cliente Registado (diferenciando portanto entre clientes pontuais e clientes registados, sendo que estes últimos usufruem de pontos), e tem a classe Cliente Inativo como apontador para ela.

RESERVA

A classe Reserva tem os membros: nome_fornecedor, oferta (um objeto da classe oferta, a oferta que foi reservada), nome_cliente, Cliente (um objeto da classe cliente, que efetuou a reserva), cancelada (um bool que indica se a reserva foi cancelada ou não), preco e data (a data da viagem).

A classe Reserva tem a classe Fatura como apontador para ela.

TIME

A classe Time tem os membros: minutes, hours, day,month, e year (todos ints que representam o respetivo valor da data em questão).

A classe Time tem como derivada a classe RealTime, que regista, dentro dos parâmetros da classe Time, o tempo real.

OBJETO REPETIDO E OBJETO INEXISTENTE

As classes Objeto Repetido e Objeto Inexistente são classes template cuja utilidade é detetar exceções, nomeadamente, quando um objeto que se pretende adicionar já existe (por exemplo, quando se tenta adicionar um Fornecedor cujo nome já está em uso) e quando se procura um objeto numa estrutura que não se encontra presente.

GESTÃO DE CRUZEIROS DO RIO DOURO

FICHEIROS DE TEXTO

CLIENTES.TXT

Lista dos clientes não registados na empresa, guardando os seguintes dados: *nome, morada*

CLIENTES_REGISTADOS.TXT

Lista dos clientes registados na empresa, guardando os seguintes dados: *nome, morada, pontos*

FORNECEDORES.TXT

Lista dos fornecedores registados na empresa, guardando os seguintes dados: *nome, nif, morada, as definições de fornecedor, as ofertas do fornecedor (e respetivos elementos)*

RESERVAS.TXT

Lista das reservas efetuadas na empresa, guardando os seguintes dados: *nome do fornecedor, a oferta reservada (e respetivos elementos), o nome do cliente, o cliente que efetuou a reserva (e respetivos elementos), cancelada, preco e a data da reserva*

DIFICULDADES SENTIDAS

As principais dificuldades sentidas neste trabalho foram a organização e armazenamento eficiente dos dados que nos competia organizar.

Essas dificuldades – principalmente sentidas no início do projeto – levaram mais tarde a algumas complicações desnecessárias em funções que se serviam desses dados. Se tivéssemos feito a estruturação inicial do trabalho de modo mais eficiente, essas complicações não teriam surgido.

No entanto, estas dificuldades foram resolvidas e não são sentidas no fim do trabalho.

GESTÃO DE CRUZEIROS DO RIO DOURO

CONTRIBUIÇÃO DE CADA MEMBRO

Neste trabalho, sentimos que cada membro do grupo investiu aproximadamente o mesmo número de horas de trabalho e de esforço. Apesar de cada um ter tido tarefas mais específicas (que serão enumeradas seguidamente), todos fizeram um pouco de tudo e resolveram erros em todas as partes do trabalho.

A divisão de trabalhos inicial foi:

Carlos Vieira: classes e funções destas, classes Time e RealTime como um todo, métodos que lidam com as faturas e a respetiva Binary Search Tree.

João Carlos Maduro: funções de acessos a ficheiros, desenvolvimento de classes, diversas funções das respetivas classes, exceções e métodos que lidam com atividade e inatividade de clientes e a respetiva hash table.

João Álvaro Ferreira: funções de menu, displays de informação e acesso a dados, funções de modificação de objetos e métodos que lidam com a ordenação de ofertas com base na ultima reserva feita de ofertas com o mesmo tipo de barco, e respetiva priority queue.

No entanto, como já foi dito, após a fase inicial, qualquer um dos membros trabalhou em todo o tipo de funções dentro deste trabalho.