Mestrado Integrado

Engenharia Informática e Computação

Algoritmos e Estruturas de Dados

Tema 5 - Empresa de Mudanças (Parte 2)

Turma 2 – Grupo 4

Gonçalo Regueiras dos Santos | 201603265 | up201603265@fe.up.pt

João Lourenço Teixeira Vieira | 201603190 | up201603190@fe.up.pt

João Miguel Mendes Ribeiro Agulha | 201607930 | up201607930@fe.up.pt

5 de janeiro de 2018

Universidade do Porto

FEUP Faculdade de Engenharia

Índice

Descrição do Tema

O presente trabalho permite a uma empresa de mudanças gerir informaticamente a sua rede de clientes e os serviços que a mesma presta.

A gestão engloba tanto serviços prestados a empresas como a clientes particulares, oferecendo, para o mesmo tipo de serviços, modalidades de pagamento diferente, consoante o tipo de cliente.

Todo o processo de tratamento do(s) objeto(s) a transportar são geridos pela empresa, sendo possível uma localização precisa e constante do(s) mesmo(s).

Descrição da solução implementada

Parte 1

A empresa (*Company* - classe principal do programa) tem como atributos vetores para todos os clientes (classe *Client*), para todos os Serviços (class *Services*) e para todos os pagamentos (classe *Payment*), o NIB, a entidade e a referência para os pagamentos. Guarda ainda a data (classe Data) e a hora (classe *Hour*).

Cada serviço contém o endereço de origem e destino, o volume da encomenda (em m^3), o preço e a data e hora de inicio e fim de todas as etapas do serviço (classes *Packaging*, *Shipping* e *Delivery*). Contem ainda um indicativo de se é ou não pagamento no fim do mês (class *EOMPayment*)

Cada cliente tem um nome, um endereço (classe *Address*), o seu NIF e dois vetores, um para os serviços requisitados e outro para os pagamentos correspondentes.

A classe *Address* guarda a morada, o número da porta, a cidade, o distrito e o país de um endereço.

Os clientes estão divididos em 3 tipos, particular (classe *Personal*), empresarial (classe *Business*) e não registados (classe *Unregistered*). Os clientes particulares e não registados têm acesso aos mesmos meios de pagamentos (classe *Payment*), pagamento por Multibanco (classe *DebitCard*) ou transferência bancária (classe *BankTransfer*). Os clientes empresariais, além destes meios de pagamento, podem ainda pagar por cartão de crédito (classe *CreditCard*).

Os clientes empresariais usufruem de dois serviços exclusivos, o armazenamento das encomendas nos armazéns da empresa e o pagamento no final do mês. O serviço de armazenamento é gratuito pelos primeiros 5 dias, após este prazo, passam a ter um custo diário. Se o cliente desejar, pode no final do mês pagar todos os serviços que ficaram pendentes ao longo do mês.

Por motivos de simplificação, quando o cliente (empresarial) requisita o armazenamento da encomenda nos armazéns da empresa, esse tempo é incluído no tempo de transporte, isto é, o número de dias entre o fim do shipping e o início do delivery representa o número de dias de armazenamento. Também por simplificação, quando a cidade de um endereço indicado pelo Cliente é uma capital de distrito de Portugal, considera-se as coordenadas geográficas do centro da cidade como sendo as do endereço (*Address*) introduzido pelo cliente.

A distância entre a origem e o destino do serviço é calculada através de um algoritmo de *Haversine* **[1]**, utilizando as coordenadas GPS dos dois pontos.

O número de dias entre duas datas é calculado convertendo essas datas para o formato *Rata Die* **[2],** que representa uma data pelo número (inteiro) de dias desde o dia 1 de janeiro do ano 1. Assim, a diferença de dias entre duas datas torna-se uma operação trivial, permitindo-nos abstrair do número de dias num mês e se o ano é bissexto ou não.

Quanto a organização geral do programa, baseamo-nos no padrão de arquitetura e desenho de software, mais precisamente, MVC (Model-View-Controller), que implementados para cada caso de uso. O *modelo* (*model*) consiste nos dados da aplicação. O *view* pode ser qualquer saída de representação dos dados, como uma tabela ou um diagrama bem como a interação direta com o utilizador. E, finalmente, o controller é aquele que faz a integração, ou por outras palavras, a ponte entre o model e o view. Por esta razão, é o controller o responsável por se adaptar aos outros dois anteriormente referidos.

Parte 2

Para a parte 2 do projeto, implementamos uma Árvore Binária de Pesquisa (*BST*) para guardar as faturas dos clientes, uma *priority_queue* para guardar os veículos, uma *queue* para guardar os serviços que ainda não têm um veículo disponível e uma Tabela de Dispersão para guardar os clientes inativos. Para fazer alterações à empresa e realizar operações que não estão diretamente ligadas a nenhum cliente em específico, criamos um administrador (id = 1 e password = admin).

Para implementar a BST usamos o header file disponibilizado no moodle. Aqui as faturas são armazenadas por nome do cliente, da mais antiga à mais recente.

Os veículos têm diversas informações que os permitem identificar, tal como matrícula, marca e modelo. Têm duas *flags* que indicam o estado do veículo, se está disponível ou se está em manutenção. Guardam ainda a data e hora de quando voltarão a estar disponíveis e a data da próxima manutenção agendada. Apenas o administrador do sistema pode fazer alterações aos veículos, os clientes não podem adicionar novos veículos ou mandar veículos para manutenção, por exemplo. Os veículos são armazenados numa *priority_queue* e são ordenados pelo estado, isto é, veículos disponíveis e não em manutenção estão no topo da fila e, por ordem crescente em relação ao momento em que estarão disponíveis de novo.

Numa situação real, os veículos seriam atualizados ao longo do tempo, à medida que a sua condição se fosse alterando, para efeitos de teste, no nosso programa, criamos uma opção que permite ao administrador avançar no tempo. Quando isso acontece, a condição dos veículos é reavaliada, e caso seja necessário, atualizada.

A Tabela de Dispersão para clientes inativos guarda clientes sem qualquer serviço registado (caso não seja a primeira utilização de conta após registo) ou clientes que não requisitem um serviço por um prazo superior ao definido pela empresa. Esta verificação é feita a cada inicialização do programa e também quando á uma alteração da data.

Diagramas UML

Figura 2 - Diagrama UML:Company Class

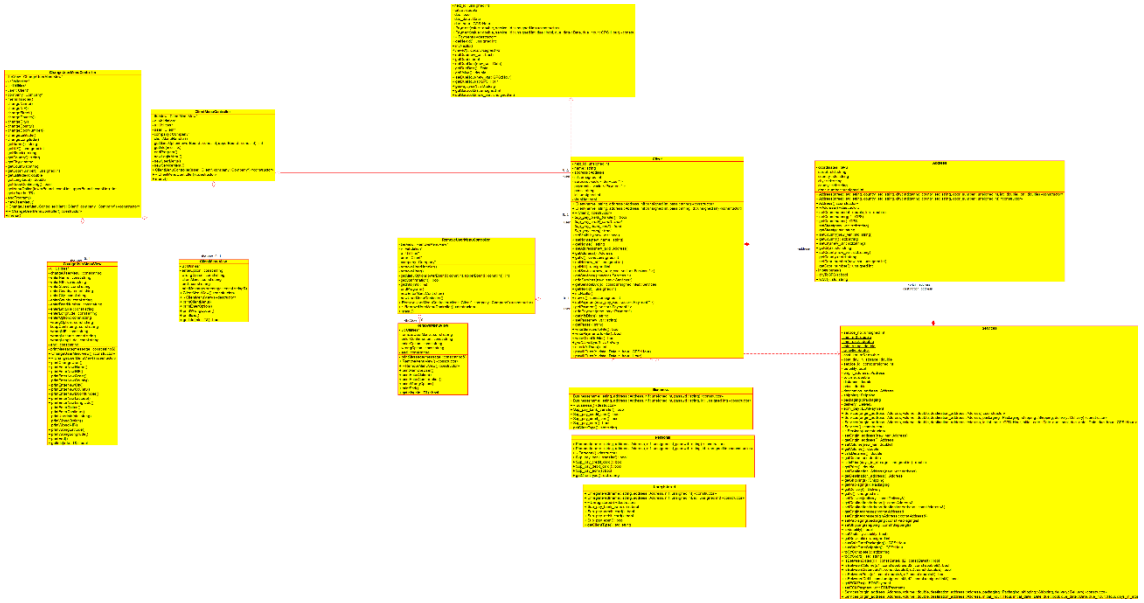

Figura 2 - Diagrama UML:Client Class

Mais imagens do diagrama serão disponibilizadas em anexo, assim como o ficheiro *proj.xmi*, com o mesmo.

Lista de Casos de Utilização

Caso de uso	Ator
Criar novo Perfil de Cliente (Registar-se)	Utilizador
Login de Cliente	Utilizador
Ver Registo de Serviços	Administrador
Definir Campanha Promocional	Administrador
Adicionar Novo Veículo	Administrador
Ver Informações do Veículo	Administrador
Alterar Informações do Veículo	Administrador
Enviar Veículo para Manutenção	Administrador
Remover Veículo	Administrador
Alterar Perfil do Cliente	Cliente
Apagar Perfil do Cliente	Cliente
Requisitar Novo Serviço	Cliente
Ver Serviço	Cliente
Ordenar Listagem de Serviços	Cliente
Filtrar Listagem de Serviços	Cliente
Pagar Serviços	Cliente (Empresarial)

Principais Dificuldades no Desenvolvimento

Gonçalo Santos

Ao longo do desenvolvimento do projeto, foram surgindo alguns desafios, tal como o cálculo da distância entre dois locais, definidos por coordenadas geográficas ou operações aritméticas com datas.

Na segunda parte, o controlo de quando os veículos irão estar disponíveis de novo revelou-se mais complicado do que esperado.

João Vieira

O projeto não se apresenta muito complicado ao nível técnico. Penso que a componente de gestão humana intragrupo se revela o ponto mais desafiante.

Na segunda parte, penso que a nossa gestão foi melhor e o nível de dificuldade do projeto manteve-se mais ao menos o mesmo.

João Agulha

Pelo facto de o projeto ter sido realizado utilizando um padrão de arquitetura de software, mais precisamente, MVC (Model-View-Controller), a integração destes foi se duvida a parte mais exigente e motivadora.

Indicação do Esforço de Cada Elemento

Todos os elementos concordam que o esforço foi igualmente repartido.

Referências bibliográficas

[1]

Baum, Peter. 2017. "Date Algorithms." Abril 28: 42-43.

[2]

Veness, Chris. n.d. *Calculate distance, bearing and more between Latitude/Longitude points.* Accessed Outubro 25, 2017. https://www.movable-type.co.uk/scripts/latlong.html.