Trabalho 1 – WhatsApl (Parte 2)

Complemente o sistema já implementado com as seguintes funcionalidades:

- Pretende-se saber quais são os utilizadores mais fervorosos do WhatsApl. Para tal, guarde referências aos utilizadores numa **árvore binária de pesquisa**. A árvore deve estar ordenada pelo número de mensagens enviadas durante os últimos 3 dias, e, em caso de empate, pelo número de grupos em que o utilizador é membro. Devem ser permitidas listagens várias tirando partido da ordenação da árvore.
- Para auxiliar um moderador na gestão do grupo, pretende-se acrescentar a funcionalidade de validar as mensagens que lhe são remetidas. Para tal, as mensagens por validar são guardadas numa **fila de prioridade**, em que os principais critérios de prioridade são a data da mensagem e a antiguidade do remetente no grupo (por ordem decrescente da data de registo, de modo a fomentar a participação de membros mais recentes). Deve ser permitido aprovar ou censurar mensagens. Aprovar uma mensagem significa permitir que ela chegue ao grupo de membros não bloqueados. A censura de uma mensagem deve permitir de imediato bloquear o seu remetente, por opção do moderador.
- Para lidar com o grande volume de utilizadores do WhatsApl, mas muitos dos quais se tornam inativos por longos períodos, decidiu-se guardar exclusivamente numa **tabela de dispersão** os utilizadores que não utilizem a aplicação durante mais de um mês. Estes continuam a poder usufruir do sistema, sendo que quando voltarem a emitir mensagens o seu registo volta a constar na estrutura de dados desenhada para o efeito na parte 1 deste trabalho. Deve portanto ser permitido adicionar utilizadores inativos (por transição da estrutura de dados anterior) e removê-los (voltando à estrutura original). As operações de manutenção de utilizadores desenvolvidas na parte 1 devem estender-se à tabela de dispersão. Deve ainda ser possível listar os utilizadores inativos.

Nota: O trabalho deve ilustrar a realização das operações básicas CRUD (*Create*, *Read*, *Update*, *Delete*) sobre as estruturas de dados: árvore binária de pesquisa, fila de prioridade e tabela de dispersão.

Trabalho 2 – Faturação de IVA (Parte 2)

Complemente o sistema já implementado com as seguintes funcionalidades:

- Pretende-se saber quais são os sujeitos passivos de IVA com maior volume faturado em cada mês. Para tal, deve ser mantida uma **árvore binária de pesquisa** que permita manter os sujeitos passivos de IVA ordenados por volume faturado. Devem ser permitidas listagens várias tirando partido da ordenação da árvore.
- Sempre que há declarações de IVA em falta, deve ser criado um registo de incumprimento, contendo informação relativa à identificação do sujeito passivo de IVA, volume de IVA entregue ao estado no mês homólogo do ano anterior, e indicação do mês/ano da declaração em falta. Esta informação deve ser guardada numa **fila de prioridade**, ordenada por mês/ano, volume de IVA previsto e volume de IVA entregue no mês homólogo do ano anterior (arredondando estas quantias a milhares de euros). Esta fila de prioridade serve o propósito de notificar os sujeitos em falta. No caso de sujeitos já notificados e cujo atraso na falta da declaração ultrapasse um mês (contado a partir da data da notificação), o sujeito deverá ser impedido de registar novas faturas.
- As faturas correspondentes a declarações de IVA aceites são arquivadas numa **tabela de dispersão**. Estas faturas poderão ser consultadas listando, por NIF do sujeito passivo de IVA e mês/ano, as faturas existentes. Deve ser possível eliminar definitivamente estas faturas, em grupo, mas apenas passados 4 anos.

Nota: O trabalho deve ilustrar a realização das operações básicas CRUD (*Create, Read, Update, Delete*) sobre as estruturas de dados: árvore binária de pesquisa, fila de prioridade e tabela de dispersão.

Trabalho 3 – OLZ (Parte 2)

Complemente o sistema já implementado com as seguintes funcionalidades:

- Pretende-se saber quais são os utilizadores com mais negócios concretizados no OLZ. Para tal, deve ser mantida uma **árvore binária de pesquisa** que permita manter os utilizadores ordenados por número de negócios e, em caso de empate, pela data do último negócio. Devem ser permitidas listagens várias tirando partido da ordenação da árvore.
- Pretende-se acrescentar a funcionalidade de permitir aos anunciantes pagar para que o seu anúncio (novo ou já existente) apareça nos primeiros lugares dos resultados de pesquisas. Para tal, o anunciante paga para ter o anúncio em destaque durante um determinado número de dias. Os anúncios passam a estar guardados numa **fila de prioridade**, ordenada por data decrescente de publicação (mais recentes primeiro), sendo que os anúncios destacados devem estar no topo da fila (ordenados por data decrescente de destaque).
- Os negócios concretizados são arquivados numa **tabela de dispersão**. Podem depois ser consultados listando os negócios para um determinado anunciante, tipo ou categoria.

Nota: O trabalho deve ilustrar a realização das operações básicas CRUD (*Create*, *Read*, *Update*, *Delete*) sobre as estruturas de dados: árvore binária de pesquisa, fila de prioridade e tabela de dispersão.

Trabalho 4 – Empresa de Transporte de Mercadorias (Parte 2)

Complemente o sistema já implementado com as seguintes funcionalidades:

- Os motoristas da empresa são caracterizados pelo nome, NIF e número de horas de serviço diárias (já efetuadas). Registe esta informação numa **árvore binária de pesquisa** que permite manter os motoristas ordenados por número de horas de serviço diárias. Quando é efetuado qualquer transporte, é escolhido um motorista para conduzir o camião e as horas de serviço são atualizadas. Devem ser permitidas listagens várias, tirando partido da ordenação da árvore.
- Considere uma **fila de prioridade** que guarda informação sobre oficinas. Uma oficina é caracterizada pela sua denominação, marcas de automóveis em que é especializada e disponibilidade (nº dias até que possa efetuar o serviço). Um camião da empresa pode necessitar da oficina para serviços específicos (substituição de uma peça,…) ou serviços usuais (mudança de óleo,…). O sistema deve indicar a oficina disponível mais cedo: i) que é especializada na marca do veículo em causa, se se tratar de um serviço específico, ou ii) qualquer oficina, no caso de se tratar de um serviço normal. Sempre que uma oficina aceita um serviço, a sua disponibilidade diminui. Considere também a ocorrência de término de um serviço, caso em que a disponibilidade da oficina aumenta.
- Os clientes inativos (que deixaram de requisitar os serviços da empresa há mais de 1 ano) devem ser guardados numa **tabela de dispersão**. Para efeitos de marketing, interessa manter a informação atualizada de clientes inativos. Devem ser permitidas listagens ou pesquisas de clientes a especificar.

Nota: O trabalho deve ilustrar a realização das operações básicas CRUD (*Create*, *Read*, *Update*, *Delete*) sobre as estruturas de dados: árvore binária de pesquisa, fila de prioridade e tabela de dispersão.

Trabalho 5 – Condomínio (Parte 2)

Complemente o sistema já implementado com as seguintes funcionalidades:

- Considere que um condomínio é caracterizado, para além da informação já considerada na primeira parte do trabalho, por designação e localização. Uma empresa de gestão de condomínios guarda os condomínios que administra numa **árvore binária de pesquisa**, sendo a ordenação efetuada por número de habitações e, em caso de empate, por número de vivendas. Devem ser permitidas listagens várias, tirando partido da ordenação da árvore.

- O condomínio dispõe de um sistema de informação sobre transportes públicos, para apoio aos seus condôminos. Um transporte público é caracterizado por tipo (metro, autocarro, comboio, …), destino e localização do ponto de paragem mais próximo do condomínio. Os transportes públicos são guardados numa **fila de prioridade**. O sistema deve indicar ao condômino a localização do ponto de paragem mais próximo do transporte público que: i) tem como destino o especificado pelo condômino, ou ii) qualquer, no caso do condômino não especificar local de destino. Deve ser possível desativar pontos de paragem, criar novos pontos de paragem ou alterar o destino de um transporte.

- Os prestadores de serviços que o condomínio possui são guardados numa **tabela de dispersão**. Devem ser permitidas listagens ou pesquisas de prestadores de serviço a especificar.

Nota: O trabalho deve ilustrar a realização das operações básicas CRUD (*Create*, *Read*, *Update*, *Delete*) sobre as estruturas de dados: árvore binária de pesquisa, fila de prioridade e tabela de dispersão.

Trabalho 6 – Gestão de Correspondência (Parte 2)

Complemente o sistema já implementado com as seguintes funcionalidades:

- Os funcionários da estação de correios são caracterizados pelo nome, morada, NIF e data de início do último contrato e são responsáveis por um ou mais carteiros (entidade implementada na parte 1). Os funcionários são guardados numa **árvore binária de pesquisa**, ordenada por data de início de contratação e, em caso de empate, por nome. O contrato de um funcionário pode ser revalidado ou cancelado e podem ser efetuadas novas contratações. Se o contrato de um funcionário é cancelado, os carteiros à sua responsabilidade, são distribuídos pelos restantes funcionários. Devem ser permitidas listagens várias, tirando partido da ordenação da árvore.

- A recolha de correspondência é realizada por uma ou mais carrinhas em horário pré-determinado. Uma carrinha é caracterizada por tipo (carta ou encomenda) e capacidade. As carrinhas disponíveis na estação de correios para recolha de correspondência estão guardadas numa **fila de prioridade**. A correspondência deve ser colocada nas carrinhas apenas no horário de recolha e a correspondência relativa a correio verde deve ser carregada em primeiro lugar. A carrinha a usar na recolha de uma certa quantidade de correspondência deve ser a carrinha do tipo específico (carta ou encomenda) que ainda tenha capacidade disponível para essa encomenda.

- Os clientes inativos (que não enviam correspondência há mais de 1 ano) devem ser guardados numa **tabela de dispersão**. Para efeitos de marketing, interessa manter a informação atualizada de clientes inativos. Devem ser permitidas listagens ou pesquisas de clientes a especificar.

Nota: O trabalho deve ilustrar a realização das operações básicas CRUD (*Create*, *Read*, *Update*, *Delete*) sobre as estruturas de dados: árvore binária de pesquisa, fila de prioridade e tabela de dispersão.

Trabalho 7 – Campeonatos Polidesportivos (Parte 2)

Complemente o sistema já implementado com as seguintes funcionalidades:

- O planeamento do calendário de um campeonato, organizando as datas previstas das realizações das provas, deve ser guardado numa **árvore binária de pesquisa**. As provas são ordenadas pela sua dada e hora de início, e têm uma previsão da sua duração. No caso de haver provas realizadas no mesmo dia e início na mesma hora, estas deverão ser ordenadas alfabeticamente, pela sua designação. Durante o planeamento do campeonato, pode ser necessário alterar as datas e horas de início de uma prova, caso haja muitas sobreposições ou seja necessário adiar, antecipar ou cancelar uma prova. Devem ser permitidas várias consultas ao calendário de provas, tirando partido da ordenação da árvore.

- Simule a realização de um campeonato, e crie um ranking das equipas participantes, ordenando-as numa **fila de prioridade** a partir do seu número de medalhas ganhas: primeiro as que têm maior número de medalhas de ouro, depois de prata, e por fim de bronze. À medida que o campeonato avança e as equipas ganham novas medalhas, o ranking das equipas deve ser atualizado. Considerando haver controlo anti-doping, é possível que algumas equipas sejam desclassificadas de um prova para a qual ganharam medalhas, caso em que a respectiva medalha lhes é retirada, devendo isso refletir-se no ranking. Apenas devem figurar no ranking as equipas que tenham ganho pelo menos uma medalha, devendo as demais permanecer fora do ranking.

- Considere agora que a aplicação deverá também gerir a audiência das provas, pela venda de bilhetes eletrónicos aos adeptos das equipas. Ao comprar um bilhete, este é associado ao comprador a partir do seu endereço eletrónico; outros dados devem também ser associados ao bilhete, como as provas a que dá acesso, nome do adepto e morada. A informação dos bilhetes é guardada numa **tabela de dispersão**. Os adeptos poderão desejar associar aos seus bilhetes novas provas, ou trocar provas anteriormente associadas. É possível também que um adepto coloque o seu bilhete à venda, e caso seja comprado por outro um adepto que já tenha bilhete, as provas são somadas às do comprador; caso seja um novo comprador, este deverá ser adicionado à tabela. O adepto que vendeu o seu bilhete será retirado da tabela. Devem ser permitidas listagens ou pesquisa de adeptos com bilhetes válidos.

Nota: O trabalho deve ilustrar a realização das operações básicas CRUD (*Create*, *Read*, *Update*, *Delete*) sobre as estruturas de dados: árvore binária de pesquisa, fila de prioridade e tabela de dispersão.

Trabalho 8 – Oficina Mecânica (Parte 2)

Complemente o sistema já implementado com as seguintes funcionalidades:

- A oficina mecânica deseja organizar as marcações dos seus serviços, e organiza as datas previstas para um serviço numa **árvore binária de pesquisa**. Os serviços agendados têm um tempo estimado de duração, e são organizados na árvore a partir da dada e hora do seu agendamento. Para agendamentos marcados para o mesmo dia e hora, estes deverão ser organizados pelo nome dos seus clientes. Os agendamentos podem ser alterados, remarcados para outro dia, ou mesmo cancelados. Devem ser possíveis várias consultas ao calendário de agendamentos de serviços da oficina, tirando partido da ordenação da árvore.

- Ao implementar um cartão de fidelização, a oficina passa a dar pontos aos seus clientes pelos serviços realizados. Os pontos são proporcionais ao valor pago pelo cliente, nos diversos serviços que realiza às suas viaturas (por exemplo, a cada X Euros pagos, o cliente receberá Y pontos). A informação dos cartões de pontos é organizada numa **fila de prioridade**; quando há empate, fica à frente o cliente com mais serviços realizados. De tempos em tempos, a oficina oferece serviços gratuitos aos seus melhores N clientes, em troca dos pontos que têm. Caso o cliente utilize os seus pontos, deve ter a sua posição na fila atualizada de acordo. Quando lhe é proposta uma oferta, caso o cliente não a queira utilizar, esta é oferecida ao próximo cliente na fila. Os pontos, entretanto, têm uma validade, e caducam, caso o cliente não os utilize atempadamente. Ao caducarem, a posição do cliente na lista também deve ser atualizada.

- Para efeitos de marketing, a oficina mantém um registo dos seus clientes inativos (aqueles que já não realizam serviços há mais de um determinado tempo, p. ex., um ano). Este registo de clientes inativos é mantido numa **tabela de dispersão**, organizando os clientes pelo seu nome e mantendo informações relativamente à sua morada, contacto de e-mail, e telefones. Estas informações poderão ser atualizadas, entretanto. Caso o cliente realize um novo serviço, deixa de ser inativo, passando a ser ativo e a ter um cartão de pontos. Devem ser permitidas listagens ou pesquisa de clientes inativos.

Nota: O trabalho deve ilustrar a realização das operações básicas CRUD (*Create*, *Read*, *Update*, *Delete*) sobre as estruturas de dados: árvore binária de pesquisa, fila de prioridade e tabela de dispersão.

Trabalho 9 – Agência de Viagens (Parte 2)

Complemente o sistema já implementado com as seguintes funcionalidades:

- A agência de viagem pretende manter um catálogo dos destinos para onde oferece opções de viagem, com ou sem alojamento. Um mesmo destino no catálogo, pode fazer parte de vários pacotes promocionais, por exemplo. Esta informação é guardada numa **árvore de decisão binária**, onde os destinos podem ser consultados pelo nome. No caso de um mesmo destino estar oferecido em vários pacotes, estes são distinguidos do mais barato para o mais caro. Os pacotes dos destinos podem ver os seus valores atualizados, assim como os pacotes promocionais para cada destino. A agência poderá também deixar de promover viagens para um dado destino, ou passar a oferecer viagens a um novo destino. Devem ser possíveis várias consultas ao catálogo da agência, tirando partido da ordenação da árvore.

- Num sistema similar ao de milhas, os viajantes frequentes da agência recebem pontos, por cada viagem que realizam. A informação de viajante frequente é guardada numa **fila de prioridade**, permitindo aos clientes acesso privilegiado às ofertas da agência. Ao atingir uma determinada quantidade de pontos, o viajante é promovido à classe de Bronze, Prata, e Ouro, nesta ordem. Assim, viajantes Ouro terão prioridade relativamente aos Prata, que terão prioridade relativamente aos Bronze, e estes últimos aos demais clientes. Dentro de cada classe, a ordem reflete o número de pontos, sendo prioritários aqueles com maior número de pontos, em cada classe. Os pontos são atualizados, sempre que o cliente realiza uma nova viagem. O cliente perde também pontos quando transita de uma classe para outra, por exemplo, quando se passa de Prata para Outro. Os pontos têm uma validade, expirando após dado período de tempo. Ao passarem muito tempo sem pontos, clientes de classes mais altas, como Ouro, perdem o estatuto, sendo despromovidos para as classes imediatamente abaixo. Ao passarem muito tempo sem pontos na classe básica, passarão a clientes antigos.

- Os dados dos clientes antigos, que já não viajam há mais do que um dado período de tempo, são armazenados numa **tabela de dispersão**, para efeitos de marketing. Estes clientes antigos são identificados pelos seus nomes, e guarda-se informação como contactos de e-mail e morada, para envio de publicidade. Quando realizam nova viagem, estes clientes deixam de ser considerados clientes antigos, passando a ser geridos como clientes normais da agência de viagem. É possível, sempre que necessário, atualizar os dados dos clientes antigos. Devem ser permitidas listagens ou pesquisa de clientes antigos.

Nota: O trabalho deve ilustrar a realização das operações básicas CRUD (*Create*, *Read*, *Update*, *Delete*) sobre as estruturas de dados: árvore binária de pesquisa, fila de prioridade e tabela de dispersão.

Trabalho 10 – Hipermercado (Parte 2)

Complemente o sistema já implementado com as seguintes funcionalidades:

- O hipermercado pretende manter um catálogo de produtos que comercializa e respetivos fornecedores, numa árvore binária de pesquisa. Os produtos são organizados em ordem alfabética, podendo o mesmo produto ter vários registos, a depender do tipo de fornecedor (empresas e em nome individual), e preço. As informações do catálogo podem ser alteradas, por exemplo, quando um fornecedor deixa de comercializar o produto, ou quando altera o seu preçário. É também possível adicionar novos produtos ao catálogo, assim como novos fornecedores e preços, para um produto já catalogado. Devem ser possíveis várias consultas ao catálogo, tirando partido da ordenação da árvore.

- O hipermercado também passou a utilizar um sistema de alerta de stock, que lhe permite identificar produtos que estão a esgotar o seu stock. Os alertas de produtos são organizados numa fila de prioridade, tendo os produtos com menor quantidade em stock a prioridade dos alertas. No caso de vários produtos estarem com o mesmo nível de alerta, a prioridade é dada àqueles que podem ser adquiridos por menor preço. Sempre que o hipermercado realiza novas encomendas, os respetivos produtos adquiridos têm as suas quantidades em stock atualizadas, e por conseguinte o seu nível de alerta. Quando o produto esgota, já não tendo disponibilidade em stock, o seu nível de alerta é máximo, sendo gerada automaticamente uma encomenda. Se o produto deixar de ser comercializado no hipermercado, o seu respetivo alerta deixa de existir na fila de prioridade.

- Fornecedores que já não recebem encomendas há mais tempo do que um dado período, por exemplo, um ano, passam a estar guardados numa tabela de dispersão. Os fornecedores inativos são identificados pela sua designação (nome), e contêm os respetivos contactos. Esta informação pode ser sempre atualizada, e sempre que uma encomenda é realizada a um fornecedor inativo, este passa novamente a ativo, deixando de existir o seu registo na tabela de dispersão. Devem ser permitidas listagens ou pesquisa de fornecedores inativos.

Nota: O trabalho deve ilustrar a realização das operações básicas CRUD (*Create*, *Read*, *Update*, *Delete*) sobre as estruturas de dados: árvore binária de pesquisa, fila de prioridade e tabela de dispersão.