Projeto Prático de Desenvolvimento de Software

Gestão de Reservas em Alojamento Local

Usando Estruturas Dinâmicas de Dados em C

Linguagens de Programação 1
Rui Silva Moreira
rmoreira@ufp.edu.pt

Bruno Gomes
bagomes@ufp.edu.pt

Algoritmos e Estruturas de Dados 1
José Torres
jtorres@ufp.edu.pt

Bruno Gomes
bagomes@ufp.edu.pt

(versão 1.0)
Outubro de 2020
Universidade Fernando Pessoa
Faculdade de Ciência e Tecnologia

Descrição do problema

Uma empresa de alojamento local do Porto, a PortoFlats, possui S estúdios (e.g. S = 50+) para alojamento local, distribuídos por E edifícios distintos (e.g. E = 3+), situados em diferentes pontos da cidade e identificados por uma morada e respectivas coordenadas GPS (cf. latitude, longitude). Cada estúdio tem uma configuração, e.g., 1 quarto (T1), 2 quartos (T2), 3 quartos duplex (T3D), etc.

A empresa que administra o aluguer destes alojamentos pretende ter uma aplicação própria de gestão de reservas e simultaneamente trabalhar com P plataformas de aluguer distintas (e.g. AirBnC, AirBnD, AirBnE, AirPlaces, etc.) para gerir as reservas. Neste sentido, a PortoFlats deverá trabalhar com várias agendas/calendários de reserva, um por cada plataforma de aluguer.

Cada estúdio terá uma agenda principal (Master Calendar) na qual serão registados todos os eventos relevantes (e.g. dias ocupados, limpezas, manutenção, facturação, geração de relatórios de utilização/estatísticas, etc.). Cada estúdio terá ainda uma agenda por cada plataforma de aluguer (Branch Calendar) onde são registadas as reservas (que podem, ocasionalmente, ser canceladas) efectuadas através das respectivas plataformas.

Periodicamente deverá haver uma sincronização das várias agendas das plataformas com as respectivas agendas master dos estúdios. Na agenda principal terá que ficar registado o hóspede, o preço praticado (por dia ou por mês) e a plataforma usada na reserva. Não está excluída a possibilidade de haver situações de overbooking (mais do que uma reserva para os mesmos dias). Quando isso acontecer o sistema de reserva deve detectar essa situação e sugerir resoluções para o overbooking (e.g., sugerir outro estúdio para uma das reservas ou outras datas de reserva, etc.).

Para além do registo das reservas na agenda, é importante manter um histórico do registo (tabela) dos hóspedes para efeitos de envio de publicidade e promoções e outros.

Relativamente aos preços praticados, deverá haver preços base para cada estúdio (por m2) em função das zonas dos edifícios. A cada estúdio aplicar-se-á um desses preços base específicos que permitirá definir o preço diário base. O preço diário efetivamente praticado poderá no entanto depender de várias regras a aplicar. Assim, o preço da estadia poderá variar em função de um conjunto de regras:

- A configuração do estúdio poderá afectar o preço (e.g. ser duplex ou ter terraço pode aplicar uma taxa extra);
- A duração da estadia: i) menos de uma semana; ii) uma semana e menos de um mês; iii) um mês ou mais (NB: a faturação para durações superiores a um mês pode ser feita ao mês ou em frações de ½ mês em vez de ser feita ao dia);

- A altura ou época do ano, como por exemplo, através de uma divisão entre época baixa e época alta (e.g. ao fim de semana, a estadia é normalmente mais cara);
- A plataforma onde é efectuada a reserva poderá aplicar preços e promoções específicas;
- A modalidade de reserva: i) ao estúdio, em que o hóspede reserva o alojamento para um estúdio em particular; ii) à estadia, em que o hóspede apenas escolhe o período de alojamento e a tipologia do estúdio e a PortoFlats escolhe qual o estúdio em que o hóspede ficará alojado. Nesta segunda modalidade, a PortoFlats tem uma oportunidade de otimização de ocupação dos estúdios, por esse motivo, a tarifa poderá ser mais atrativa;
- Antecedência da reserva (cf. normal, antecipada, última hora).
- Taxa de ocupação, à data da reserva, para o período reservado.

A política de preços, como se subentende, pode ser complexa. Poderão haver diferentes políticas de preços em diferentes plataformas. Uma política de preços pode incorporar os aspectos mencionados acima e, eventualmente, outros, através de um sistema de regras inteligente e dinâmico. Poderão criar-se modelos (templates) de políticas de preços. Cada template terá um conjunto de regras que poderão ser aplicadas em simultâneo ou em exclusivo (por exemplo, utilizando prioridades).

Pretende-se desenvolver um sistema para gerir dinamicamente toda a informação necessária e que permita gerir as reservas da empresa PortoFlats de acordo com os requisitos abaixo elencados.

Requisitos

Pretende-se que os alunos proponham um conjunto de estruturas de dados dinâmicas (cf. arrays dinâmicos e listas ligadas) e respectivas funções de manipulação e interface com essas estruturas de modo a obterem uma solução que cumpra os requisitos funcionais abaixo indicados. As estruturas e funções associadas devem utilizar apontadores e estruturas dinâmicas para agregar e manipular os dados necessários. Deverão ser desenvolvidos, ainda, algoritmos de processamento, pesquisa e gestão da informação.

A avaliação será baseada na utilização de testes funcionais, i.e., cada requisito proposto será sujeito a um conjunto de testes que avaliarão o cumprimento do mesmo. Na organização de ficheiros do projeto submetido, os testes deverão estar localizados na pasta *test* que será reservada apenas para esse efeito de acordo com as indicações dadas nas aulas. Os dados de entrada para cada teste poderão ser gerados aleatoriamente (dados random) ou com recurso a dados predefinidos (dados determinísticos). Não deverá ser desenvolvida interface interactiva com o utilizador (menus ou interface gráfica).

Para testes determinísticos, e para cada caso de teste, será disponibilizado um ficheiro de dados de entrada e deverá ser produzido um ficheiro de dados de saída (test cases). Este ficheiro de saída será comparado com o output esperado.

A implementação deverá endereçar os seguintes requisitos funcionais:

1. Implementar estruturas de dados para manter a informação a ser processada pela aplicação, nomeadamente: i) uma lista ligada para todos os edifícios; ii) um array dinâmico para os estúdios de cada edifício; iii) um array dinâmico de dias para cada agenda de cada estúdio; iv) uma lista ligada dos eventos de cada dia (e.g. ocupação, limpeza, etc.); v) uma lista ligada dos hóspedes e seu histórico de estadias (reservas); vi) um array dinâmico de regras de custo para cada política de preços; etc.;

2. Implementar uma API base de funções para a criação e inserção ordenada, pesquisa binária, remoção e alteração de informação de cada entidade registada no sistema, nomeadamente: edifício, estúdio, hóspede, política de preços e regras de custo, etc.;

3. Implementar uma API de funções de criação/inserção, pesquisa, remoção e alteração de eventos em cada agenda de cada estúdio (e.g. reservas, limpezas, manutenção, etc.);

4. Implementar uma API de funções de leitura e escrita de ficheiros de texto e binário das diversas entidades: edifícios e respectivos estúdios; hóspedes e histórico de estadias (a tabela poderá ser ordenada por um dos campos (e.g. nome, data crescente ou decrescente); políticas de preços e respectivas regras de custo e prioridades de aplicação;

5. Implementar uma API de funções de leitura e escrita de texto e binário para cada agenda, com os respectivos eventos registados;

6. Implementar a geração de relatórios (para o ecrã e ficheiro) com as estatísticas das taxas de ocupação por estúdio, por edifício ou na totalidade do parque de edifícios. Deverá ser possível escolher o período do relatório a gerar; os dados do relatório devem ser ordenados por campos a determinar (e.g. por ordem alfabética do nome do estúdio e edifício, por taxa de ocupação crescente ou decrescente, etc.).

7. Implementar a geração de relatórios (ecrã e ficheiro) com a faturação por estúdio, por edifício ou na totalidade do parque de edifícios. Deverá ser possível escolher o período do relatório a gerar e a ordem de ordenação (crescente ou decrescente);

8. Implementar a sincronização de uma agenda para a agenda principal (e da principal para as restantes), aplicando estratégias de resolução de conflitos existentes (sincronização e gestão de versões);

9. Implementar um mecanismo alternativo de alocação dinâmica de um conjunto de necessidades de estadias para os vários estúdios disponíveis. Pretende-se criar um mecanismo para otimizar a realização de pré-reservas em função dos estúdios livres (modalidade de reserva à estadia). Por exemplo, tendo conhecimento da existência de um conjunto de necessidades de alojamento para diversos períodos de tempo, associadas a um evento desportivo (cf. diversos hóspedes, de várias equipas de uma dada modalidade desportiva), pretende-se elaborar uma alocação dinâmica dos estúdios que optimize a ocupação (minimizando intervalos livres entre reservas para o mesmo estúdio) e que cumpra as necessidades de alojamento.

Cotação dos Requisitos em AED1 e LP1

Req	1	2	3	4	5	6	7	8	9
AED1	1.5	3	2	3	2	2	2	2	2.5
LP1	1.5	3	2	3	2	2	2	2	2.5

(NB: cotação 0-20)

Processo de Submissão

A aplicação final (código fonte) deve estar depurada de todos os erros de sintaxe e de acordo com os requisitos funcionais pedidos. Só serão considerados os projetos de software que não contenham erros de sintaxe e que implementam as funcionalidades pedidas total ou parcialmente.

A documentação, em html ou pdf, juntamente com todo o código-fonte desenvolvido, deve ser submetida num ficheiro zip/rar (project.zip) na plataforma de elearning canvas (ufp.instructure.com).

Código Fonte e Documentação a entregar

As estruturas de dados e os algoritmos especificados devem ser implementados em linguagem C, juntamente com os comentários apropriados, inseridos no código fonte desenvolvido, de modo a que facilitem a compreensão do mesmo. Todas as funções devem estar anotadas em formato doxygen (www.doxygen.nl) incluindo: uma breve explicação dos algoritmos implementados; uma menção ao desempenho dos algoritmos (quando aplicável) assim como dos testes efetuados/implementados.

As principais estruturas de dados e variáveis devem também estar anotadas neste formato. Deverão usar o software doxygen para gerar a documentação com base nos comentários.

Os alunos deverão entregar também um ficheiro de texto no formato doxygen (.dox), descrevendo explicitamente: as funcionalidades/requisitos implementados, parcialmente implementados e não implementados. Devem mencionar sempre o número do requisito de acordo com a numeração utilizada neste documento de especificação:

- Funcionalidades implementadas: devem identificar todas as funções desenvolvidas para assegurar os requisitos funcionais solicitados.
- Funcionalidades não implementadas: devem identificar as funcionalidades não implementadas e apontar a justificação e/ou dificuldades que impediram o seu desenvolvimento.