Departamento de Estatística e Informática (DEINFO)



Introdução à Programação II – Prof. Leandro M. Nascimento

Turma: BC3 – 2017.2

**Exercício prático para a 3ª. VA – UFRPE *Flight Systems***

As férias se aproximam e muita gente se prepara para viajar. Uma nova empresa de Pernambuco, responsável por operar voos para praias paradisíacas, resolveu contratar você para implementar um sistema de gerenciamento de voos e passagens aéreas, as quais estão sendo chamadas no sistema de ‘Bilhetes’. Este sistema permite a realização de CRUD de voos e passageiros e a venda de bilhetes. O sistema deve seguir a arquitetura Model-View-Controller (3 camadas) já apresentada em sala de aula algumas vezes. Sua missão é implementar o sistema por completo, desde a sua interface gráfica até os seus repositórios de dados.

O funcionamento básico do sistema acontecerá em três passos simples da seguinte forma:

1. Cadastramento e manutenção de voos com suas devidas informações e validações;
2. Cadastramento e manutenção de passageiros com suas devidas informações e validações;
3. Emissão de bilhetes para um passageiro previamente cadastrado em um voo também já cadastrado;
4. Consulta dos bilhetes emitidos para um determinado voo, ou alguma consulta específica a ser detalhada.

Algumas regras de negócio precisam ser validadas, tais como: cada bilhete deve ser associado a um assento na aeronave e só pode ser emitido para um único passageiro; um voo não pode ter a sua data/hora de saída (decolagem) atualizadas se esta hora já aconteceu, ou seja, se a data/hora atual é posterior a data/hora de saída.

As diversas regras de negócio e as informações a serem manipuladas pelo sistema serão esclarecidas nos requisitos que se seguem:

1. O sistema deve permitir o cadastramento de voos com as seguintes informações: código de identificação único, cidade origem, cidade destino, horário de saída, horário estimado de chegada, tipo de aeronave que realiza o voo. É importante observar que os horários de saída e chegada do voo devem considerar os fusos horários locais da origem e destino (ZonedDateTime, em Java). É necessário armazenar de alguma forma em cada voo, quais assentos já estão ocupados, respeitando a capacidade da aeronave. As cidades e as aeronaves não precisam de nenhum tipo de manutenção (CRUD), ou seja, elas podem ter valores pré-definidos em código (enumerações). A enumeração que representa a Cidade deve ter somente um atributo com o seu nome. A enumeração que representa a Aeronave deve ter os atributos código-tipo (exemplo “Airbus A320”) e a capacidade, referente ao número de assentos da aeronave.
2. O sistema deverá listar todos os voos já cadastrados usando uma tabela que será automaticamente atualizada se um voo for adicionado, atualizado ou removido (vide Figura 1).
3. O sistema deve permitir a edição de voos já cadastrados a partir da seleção do registro na tabela citada no REQ2, com as seguintes restrições: um voo não pode ser alterado se o mesmo já aconteceu, ou seja, suas datas de saída e chegada já aconteceram. Se a data/hora de saída de um voo já aconteceu, mas a sua data/hora estimada de chegada ainda não, então somente a data/hora estimada de chegada poderá ser atualizada. A aeronave de um voo não poderá ser alterada se já houver bilhetes vendidos para o mesmo.
4. O sistema deve permitir a remoção de voos já cadastrados a partir da seleção do registro na tabela citada no REQ2, com as seguintes restrições: um voo não poderá ser removido se o mesmo já decolou; um voo não poderá ser removido se houver bilhetes emitidos para o mesmo.
5. O sistema deve permitir o cadastramento de passageiros com as seguintes informações: nome, último nome, cpf e passaporte. O sistema não pode permitir a inserção de passageiros com o mesmo cpf ou passaporte de outro passageiro já previamente cadastrado.
6. O sistema deve listar todos os passageiros cadastrados, independente se os mesmos adquiriram ou não passagens em voos (vide Figura 2).
7. O sistema deve permitir a edição das informações de passageiros. Somente o nome e o último nome podem ser editados. CPF e passaporte são imutáveis.
8. O sistema deve permitir a remoção de passageiros. Na tentativa da remoção, um passageiro não poderá ser removido caso tenha adquirido algum bilhete e o voo ainda não tenha acontecido. Se houver algum foi bilhete emitido em seu nome, mas o voo já aconteceu, então a remoção poderá ser realizada sem problemas.
9. O sistema deve permitir a emissão de bilhetes para passageiros em voos pré-cadastrados. A emissão do bilhete se dará em três passos: 1) a partir de uma lista de voos, o usuário irá selecionar o voo desejado; 2) em seguida, a partir de uma lista de passageiros, o usuário irá selecionar o passageiro desejado; 3) depois disso, o usuário irá selecionar qual assento será alocado para este passageiro no voo. A emissão somente poderá ocorrer se, e somente se, aquele assento não tenha sido ocupado por nenhum outro passageiro. Seja criativo no desenvolvimento da tela que permitirá a seleção de assentos de tal a não permitir já na GUI que um assento ocupado possa ser selecionado.
10. O sistema deve permitir a listagem de passageiros por voo. A partir da lista de voos e com algum voo selecionado, o usuário poderá acionar a funcionalidade de listagem de passageiros e verá todas as informações de bilhetes daquele voo em uma tabela, tais como: primeiros e últimos nomes dos passageiros, número de seu passaporte e CPF e o assento escolhido.
11. Todas as exceções do sistema deverão ser tratadas de tal forma a mostrar para o usuário alguma informação que o deixe ciente do que está acontecendo (javaFX Alerts podem ser usados). O tratamento de exceções também dispara a criação de várias entidades que irão representar os diversos tipos de exceção descritos nos vários requisitos.
12. O sistema deve obedecer ao padrão arquitetural MVC, ou seja, deve ser estruturado em camadas, assim como apresentado em sala de aula.

**Escopo negativo:**

1. O sistema NÃO precisa ter controle de acesso para usuários diferentes, mas, uma vez executando o sistema, a tela com informações sobre qual amigo secreto cada pessoa tirou deve ser controlado pela senha pessoal.
2. O sistema NÃO precisa realizar a edição de bilhetes já emitidos para, por exemplo, alterar o assento selecionado.
3. O sistema NÃO precisa realizar nenhum tipo de ordenação dos registros a partir dos repositórios, pois toda a ordenação pode ser feita diretamente no componente de tabela do JavaFX (ou Swing) na GUI.

Para melhorar o entendimento de como o sistema será executado, seguem as telas e suas respectivas funcionalidades: (**OBSERVAÇÃO**: essa é somente uma **SUGESTÃO** de como as telas poderiam ser. Elas podem ser implementadas de formas distintas das apresentadas a seguir, desde que mantenham suas funcionalidades)

As telas de Alerts com mensagens diversas para o usuário final não são apresentadas.



Figura 1. Tela principal do sistema. Gerenciamento de voos e emissão de bilhetes. Com um voo selecionado, é possível emitir bilhetes para ele

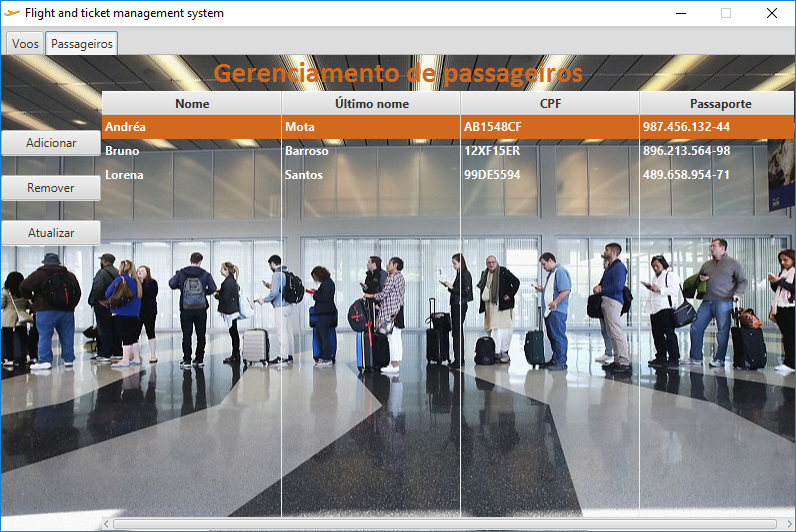


Figura 2. Gerenciamento de passageiros. CRUD completo de passageiros. Com um registro selecionado é possível editá-lo.

# Pontuação

Parte-se do princípio que a pontuação somente será creditada se a implementação estiver COMPLETA e CORRETA. **TOTAL DE PONTOS POSSÍVEIS**: 12

* (1,5) Estruturação do sistema em camadas - implementação de classes básicas, repositórios e controladores usando MVC (REQ12). As classes que precisam estar presentes no mínimo no sistema são (atenção para seus repositórios):
  + Voo
  + Cidade (Enum)
  + Aeronave (Enum)
  + Passageiro
  + Bilhete
* (2,5) Funcionalidade de gerenciamento de voos (REQ1, REQ2, REQ3, REQ4).
* (2,0) Funcionalidade de gerenciamento passageiros (REQ5, REQ6, REQ7, REQ8).
* (3,0) Funcionalidade de emissão de bilhetes para um voo, com escolha de passageiro e seleção de assento (REQ9).
* (1,5) Funcionalidade de consulta de passageiros por voo (REQ10)
* (1,5) Criação, manipulação e tratamento de exceções para as diversas regras de negócio do sistema, apresentando mensagens significativas para o usuário final (REQ11).

# Dicas de implementação:

* Não implemente nenhum tipo de CRUD de cidades ou aeronaves. Você pode utilizar enumerações para essas duas entidades.
* Crie um controlador separado somente para checar as regras de negócio da emissão de bilhetes.
* Apresente mensagens de confirmação de ações na tela como cadastro de passageiros, voos e emissão de bilhetes usando Alert de confirmação.