DEVinHouse

Módulo 2 - Projeto Avaliativo 2

SUMÁRIO

1 INTRODUÇÃO

A **LABMedicine LTDA**, empresa líder no segmento tecnológico para gestão hospitalar, está atualizando o seu sistema **LABMedical**, um software para gestão de inventário médico.
Você participou da criação do Front-End da aplicação, e por esse motivo o seu perfil chamou a atenção dos gestores, para agora criar a aplicação Back-End, que deverá ser construída utilizando **Spring** com **Java**.

Está animado para iniciar o desenvolvimento?
Leia atentamente os itens abaixo para ficar por dentro das regras de negócio e endpoints que devem ser criados nesta aplicação. Lembre-se também de ler atentamente as regras de entrega do projeto.

2 ENTREGA

O código deverá ser inserido e versionado no **GitHub** em modo privado, e o vídeo deverá ser inserido no **Google Drive** em modo leitor para qualquer pessoa com o link. Ambos os links deverão ser disponibilizados na tarefa **Módulo 2 - Projeto Avaliativo**, presente na semana 12 do AVA até o dia **30/04/2023** às **23h55**.

O repositório **privado** deverá ter as seguintes pessoas adicionadas:

- André Santana Nunes - **andresantnunes**
- Operação DEVinHouse - **devinhouse-operacao**

Importante:

1. Não serão aceitos projetos submetidos **após a data limite da atividade**, e, ou **alterados** depois de entregues. Lembre-se de **não modificar** o código no GitHub até receber sua nota e feedback.
2. Não esqueça de **submeter os links no AVA**. Não serão aceitos projetos em que os links não tenham sido submetidos.

3 REQUISITOS DA APLICAÇÃO

A aplicação que deverá ser realizada **individualmente** deve contemplar os seguintes requisitos:

1. Ser uma **API Rest** desenvolvida em **Java** com uso de **Spring**;
2. Utilizar o banco de dados **SQL Oracle**;
3. Ser versionado no **GitHub**, possuindo uma documentação **readme.md** sobre o projeto e como utilizar;
4. Possuir um **vídeo** explicativo sobre o projeto.
5. Seguir o **Roteiro da Aplicação**;

4 ROTEIRO DA APLICAÇÃO

A **LABMedicine** deseja criar a API Rest da aplicação **LABMedical**, que foi desenvolvida anteriormente por você. Este software será utilizado em todos os atendimentos médicos dos hospitais da rede.

4.1 FORMATO DO SISTEMA

O back-end deve conter os tipos de cadastros abaixo, cada um com suas características.
Todos os cadastros de pessoas devem ser derivados da classe **Pessoa**, que possui os seguintes atributos, todos são obrigatórios:

- Identificador: Um número que deve ser incrementado automaticamente
- Nome Completo: Deve ser um texto
- Gênero: Deve ser um texto
- Data de Nascimento: Obrigatório, data válida (dd/MM/yyyy).
- CPF: Deve ser texto
- RG: Deve ser texto
- Estado Civil:Obrigatório com as seguintes opções, pode ser ou string ou numérico
    - Solteiro
    - Casado
    - Separado
    - Divorciado
    - Viúvo
- Telefone: Deve ser texto
- E-Mail: Deve ser texto
- Naturalidade: Deve ser texto

Carregamento de Dados Iniciais

- Deve ser utilizado como Sistema Gerenciador de Banco de Dados o **SQL Oracle**, e a aplicação deve usar como nome do banco de dados ***labmedicalbd***. Na inicialização do sistema, devem ser carregados os dados iniciais de alguns usuários fictícios (seeders).
- Implementar estratégia para não haver problemas de inserção duplicada de dados ou falha de carregamento devido a dados anteriormente inseridos.
- Dados iniciais para carregamento:
    - **Usuários (Médicos)**: Crie 2 registros com dados fictícios.
    - **Pacientes**: Crie 2 registros com dados fictícios.
    - **Consultas**: Crie 4 registros com dados fictícios.
    - **Exames**: Crie 4 registros com dados fictícios.
    - **Endereco**: Crie 4 registros com dados fictícios.

- Dica: Utilize o **4Devs** para ajudar na criação dos dados.

S01 - Cadastro de Usuários (Médicos)

- Serviço de cadastro de médicos **usuários** que irão utilizar o sistema.
- A entidade **Usuarios** deve herdar de **Pessoa** e possuir os seguintes atributos:
  - Cadastro do CRM/UF: Obrigatório, deve ser texto.
  - Especialização Clínica: Obrigatório com as seguintes opções, pode ser string ou um valor numérico
    - Clínico Geral
    - Anestesista
    - Dermatologia
    - Ginecologia
    - Neurologia
    - Pediatria
    - Psiquiatria
    - Ortopedia
  - Senha:
    - Senha alfanumérica de no mínimo 8 caracteres.

- Definição do Endpoint:
  - Request:
    - **HTTP POST no path /api/usuarios**
    - No corpo da request, informar objeto json com os campos
    - Todos os campos obrigatórios devem ser validados. O CPF deve ser único por usuário (médico). Validar se o CPF informado já foi cadastrado no sistema.
  - Response:
    - **HTTP Status Code 201 (CREATED)** em caso de sucesso, constando no corpo da resposta o código atribuído ao novo usuário cadastrado, além dos demais campos. No response, retornar os campos cadastrados.
      - **HTTP Status Code 400 (Bad Request)** em caso de requisição com dados inválidos, informando mensagem de erro explicativa no corpo do response.
      - **HTTP Status Code 409 (Conflict)** em caso de CPF já cadastrado, informando mensagem de erro explicativa no corpo do response.

S02 - Atualização dos dados de Usuários (Médicos)

- Serviço para alterar/atualizar os dados de determinado usuário.
- O usuário do sistema poderá alterar sempre que necessário.

- A senha não deve ser atualizada

- Definição do Endpoint:
  - Request:
    - **HTTP PUT no path /api/usuarios/{identificador}**
    - No corpo da request, informar objeto json com os campos.
    - Os campos validados como sendo obrigatórios devem possuir os valores possíveis para estes campos.
  - Response:
    - **HTTP Status Code 200 (OK)** em caso de sucesso, constando no corpo da resposta os dados atualizados do paciente.
    - **HTTP Status Code 400 (Bad Request)** em caso de requisição com dados inválidos, informando mensagem de erro explicativa no corpo do response.
    - **HTTP Status Code 404 (Not Found)** em caso de não ser encontrado registro com o código informado, retornando mensagem de erro explicativa no corpo do response.
    - ADICIONAR REGRA PARA NÃO MODIFICAR CPF E RG

S03 - Atualização da senha de Usuários (Médicos)

- Serviço para alterar/atualizar a senha de determinado usuário.
- O usuário do sistema poderá alterar sempre que necessário.

- Definição do Endpoint:
  - Request:
    - **HTTP PUT no path /api/usuarios/{identificador}/senha**
    - No corpo da request, informar objeto json com os campos.
    - Validar campo senha.
  - Response:
    - **HTTP Status Code 200 (OK)** em caso de sucesso, constando no corpo da resposta os dados atualizados do paciente.
    - **HTTP Status Code 400 (Bad Request)** em caso de requisição com dados inválidos, informando mensagem de erro explicativa no corpo do response.
    - **HTTP Status Code 404 (Not Found)** em caso de não ser encontrado registro com o código informado, retornando mensagem de erro explicativa no corpo do response.

S04 - Cadastro de Paciente

- Serviço de cadastro de **Paciente**, cuja entidade deve herdar de **Pessoa** e possuir os seguintes atributos:
  - Lista de Alergias: Não obrigatório. Deve ser texto
  - Lista de Cuidados Específicos: Não obrigatório. Deve ser texto
  - Contato de Emergência: Obrigatório. Deve ser texto
  - Convênio: Não obrigatório. Deve ser texto
  - Número da Carteira do Convênio: Não obrigatório. Deve ser texto
  - Validade da Carteira do Convênio: Não Obrigatório, data válida.
  - Endereço: Este deve ser uma referência a tabela endereço, numa requisição o endereço deve ser identificado pelo Identificador do Endereço no Banco de Dados (identificador endereço)

- Definição do Endpoint:
  - Request:
    - **HTTP POST no path /api/pacientes**
    - No corpo da request, informar objeto json com os campos
    - Todos os campos obrigatórios devem ser validados. O CPF deve ser único por paciente. Validar se o CPF informado já foi cadastrado no sistema.
  - Response:
    - **HTTP Status Code 201 (CREATED)** em caso de sucesso, constando no corpo da resposta o código atribuído ao novo paciente cadastrado, além dos demais campos. No response, retornar os campos adicionais "identificador" e "exames", usando obrigatoriamente estes nomes para os campos.
      - **HTTP Status Code 400 (Bad Request)** em caso de requisição com dados inválidos, informando mensagem de erro explicativa no corpo do response.
      - **HTTP Status Code 409 (Conflict)** em caso de CPF já cadastrado, informando mensagem de erro explicativa no corpo do response.

S05 - Atualização dos dados de Pacientes

- Serviço para alterar/atualizar os dados de determinado paciente.
- O usuário do sistema poderá alterar sempre que necessário.

- Definição do Endpoint:
  - Request:
    - **HTTP PUT no path /api/pacientes/{identificador}**

        - No corpo da request, informar objeto json com os campos.
        - Os campos validados como sendo obrigatórios devem possuir os valores possíveis para estes campos.
        - Os campos RG e CPF não devem ser enviados, pois são imutáveis
    - Response:
        - **HTTP Status Code 200 (OK)** em caso de sucesso, constando no corpo da resposta os dados atualizados do paciente.
        - **HTTP Status Code 400 (Bad Request)** em caso de requisição com dados inválidos, informando mensagem de erro explicativa no corpo do response.
        - **HTTP Status Code 404 (Not Found)** em caso de não ser encontrado registro com o código informado, retornando mensagem de erro explicativa no corpo do response.

S06 - Listagem de Pacientes

- Serviço de listagem de pacientes cadastrados.

- Definição do Endpoint:
    - Request:
        - **HTTP GET no path /api/pacientes**
        - Não é necessário request body
        - Deve-se prever um *query param* opcional para filtrar o resultado da consulta pelo nome do paciente.
        - Caso não seja informado o parâmetro de pesquisa, deve retornar todos os registros da base de dados.
    - Response:
        - **HTTP Status Code 200 (OK)**, com a lista de pacientes.

S07 - Listagem de Paciente pelo identificador

- Serviço de consulta de paciente pelo seu código identificador.

- Definição do Endpoint:
    - Request:
        - **HTTP GET no path /api/pacientes/{identificador}**
        - Não é necessário request body.
    - Response:
        - **HTTP Status Code 200 (OK)**, com os dados do paciente.

        - **HTTP Status Code 404 (Not Found)** em caso de não ser encontrado registro com o código informado, retornando mensagem de erro explicativa no corpo do response.

S08 - Exclusão de Paciente

- Serviço para excluir um paciente pelo código identificador.

- Definição do Endpoint:
    - Request:
        - **HTTP DELETE no path /api/pacientes/{identificador}**
        - Não é necessário request body.
        - Adicione validação em código para que pacientes com exames ou consultas não possam ser deletados
    - Response:
        - **HTTP Status Code 204 (No Content)** em caso de sucesso, sem necessidade de response body.
        - **HTTP Status Code 404 (Not Found)** em caso de requisição com código não existente na base de dados.
        - **HTTP Status Code 400 (Bad Request)** em caso de o paciente ter um exame ou consulta

S09 - Cadastro de Consulta

- Serviço de cadastro de **Consulta**, cuja entidade possuir os seguintes atributos:
    - Identificador: Um número que deve ser incrementado automaticamente
    - Motivo da Consulta: Obrigatório. Deve ser texto
    - Data e Hora da Consulta: Obrigatório. Data e hora válida pelo timestamp durante a inserção
    - Descrição do Problema: Obrigatório. Deve ser texto
    - Medicação Receitada: Obrigatório. Deve ser texto
    - Dosagens e Precauções: Obrigatório. Deve ser texto
    - Identificador Paciente: Obrigatório. Deve ser Paciente válido
    - Identificador Medico: Obrigatório. Deve ser Paciente válido

- Definição do Endpoint:
    - Request:
        - **HTTP POST no path /api/consultas**
        - No corpo da request, informar objeto json com os campos
        - Todos os campos obrigatórios devem ser validados. O identificador deve ser único por consulta.

- Response:
  - **HTTP Status Code 201 (CREATED)** em caso de sucesso, constando no corpo da resposta o código atribuído a nova consulta cadastrada, além dos demais campos. No response, retornar os campos adicionais "identificador" e "indicador_paciente".
    - **HTTP Status Code 400 (Bad Request)** em caso de requisição com dados inválidos, informando mensagem de erro explicativa no corpo do response.

S10 - Atualização dos dados de Consultas

- Serviço para alterar/atualizar os dados de determinada consulta.
- O usuário do sistema poderá alterar sempre que necessário.

- Definição do Endpoint:
  - Request:
    - **HTTP PUT no path /api/consultas/{identificador}**
    - No corpo da request, informar objeto json com os campos.
    - Os campos validados como sendo obrigatórios devem possuir os valores possíveis para estes campos. A Data e Hora do exame devem ser imutáveis, sendo assim não são enviadas
  - Response:
    - **HTTP Status Code 200 (OK)** em caso de sucesso, constando no corpo da resposta os dados atualizados do paciente.
    - **HTTP Status Code 400 (Bad Request)** em caso de requisição com dados inválidos, informando mensagem de erro explicativa no corpo do response. Também deve ser retornado caso a data e a hora sejam enviadas.
    - **HTTP Status Code 404 (Not Found)** em caso de não ser encontrado registro com o código informado, retornando mensagem de erro explicativa no corpo do response.

S11 - Listagem de Consulta por Identificador

Serviço de listagem de consulta pelo código identificador.

- Definição do Endpoint:
  - Request:
    - **HTTP GET no path /api/consultas/{identificador}**
    - Não é necessário request body
  - Response:

        - **HTTP Status Code 200 (OK)**, retorna a consulta correta.

S12 - Exclusão de Consulta

- Serviço para excluir uma consulta pelo código identificador.

- Definição do Endpoint:
    - Request:
        - **HTTP DELETE no path /api/consultas/{identificador}**
        - Não é necessário request body.
    - Response:
        - **HTTP Status Code 204 (No Content)** em caso de sucesso, sem necessidade de response body.
        - **HTTP Status Code 404 (Not Found)** em caso de requisição com código não existente na base de dados.

S13 - Cadastro de Exame

- Serviço de cadastro de **Exame**, cuja entidade possuir os seguintes atributos:
    - Identificador: Um número que deve ser incrementado automaticamente
    - Nome do Exame: Obrigatório. Deve ser texto
    - Data e Hora do Exame: Obrigatório. Data e hora válida pelo timestamp durante a inserção
    - Tipo de Exame: Obrigatório. Deve ser texto
    - Laboratório: Obrigatório. Deve ser texto
    - Arquivo de Exame: Opcional. Deve ser a url do Arquivo PDF do Exame
    - Resultados do Exame: Obrigatório. Deve ser texto
    - Identificador Paciente: Obrigatório. Deve ser Paciente válido
    - Identificador Medico: Obrigatório. Deve ser Paciente válido

- Definição do Endpoint:
    - Request:
        - **HTTP POST no path /api/exames**
        - No corpo da request, informar objeto json com os campos
        - Todos os campos obrigatórios devem ser validados. O identificador deve ser único por consulta.
    - Response:
        - **HTTP Status Code 201 (CREATED)** em caso de sucesso, constando no corpo da resposta o código atribuído ao novo exame cadastrado, além dos demais campos. No response, retornar os campos adicionais “identificador” e “indicador_paciente”.

- **HTTP Status Code 400 (Bad Request)** em caso de requisição com dados inválidos, informando mensagem de erro explicativa no corpo do response.

S14 - Atualização dos dados de Exames

- Serviço para alterar/atualizar os dados de determinado exame.
- O usuário do sistema poderá alterar sempre que necessário.

- Definição do Endpoint:
  - Request:
    - **HTTP PUT no path /api/exames/{identificador}**
    - No corpo da request, informar objeto json com os campos.
    - Os campos validados como sendo obrigatórios devem possuir os valores possíveis para estes campos.
    - A Data e Hora do exame devem ser imutáveis
  - Response:
    - **HTTP Status Code 200 (OK)** em caso de sucesso, constando no corpo da resposta os dados atualizados do paciente.
    - **HTTP Status Code 400 (Bad Request)** em caso de requisição com dados inválidos, informando mensagem de erro explicativa no corpo do response. Também deve ser retornado caso a data e a hora sejam enviadas.
    - **HTTP Status Code 404 (Not Found)** em caso de não ser encontrado registro com o código informado, retornando mensagem de erro explicativa no corpo do response.

S15 - Listagem de Exame pelo identificador

- Serviço de listagem de exame pelo código identificador.

- Definição do Endpoint:
  - Request:
    - **HTTP GET no path /api/exames/{identificador}**
    - Não é necessário request body
  - Response:
    - **HTTP Status Code 200 (OK)**, com o exame encontrado.

S16 - Exclusão de Exame

- Serviço para excluir um exame pelo código identificador.

- Definição do Endpoint:

- Request:
  - **HTTP DELETE no path /api/exames/{identificador}**
  - Não é necessário request body.
- Response:
  - **HTTP Status Code 204 (No Content)** em caso de sucesso, sem necessidade de response body.
  - **HTTP Status Code 404 (Not Found)** em caso de requisição com código não existente na base de dados.

S17 - Listagem de Estatísticas

- Serviço para listar a quantidade de pacientes, consultas e exames cadastrados no sistema.

- Definição do Endpoint:
  - Request:
    - **HTTP GET no path /api/estatisticas**
    - Não é necessário request body
  - Response:
    - **HTTP Status Code 200 (OK)**, com um objeto listando a contagem total de cada paciente, consulta e exame. A resposta pode ser um objeto contendo os números de cada objeto contado.

S18 - Cadastro de Endereco

- Serviço de cadastro de **Endereco**, cuja entidade possuir os seguintes atributos:
  - Identificador: Um número que deve ser incrementado automaticamente
  - Cep: Obrigatório. Deve ser texto
  - Cidade: Obrigatório. Deve ser texto
  - Estado: Obrigatório. Deve ser texto
  - Logradouro: Obrigatório. Deve ser texto
  - Número: Obrigatório. Deve ser numeral
  - Complemento: Não obrigatório. Deve ser texto
  - Bairro: Obrigatório. Deve ser texto
  - Ponto de Referência: Não obrigatório. Deve ser texto

- Definição do Endpoint:
  - Request:
    - **HTTP POST no path /api/enderecos**
    - No corpo da request, informar objeto json com os campos

        - Todos os campos obrigatórios devem ser validados. O identificador deve ser único por consulta.
    - Response:
        - **HTTP Status Code 201 (CREATED)** em caso de sucesso, constando no corpo da resposta o código atribuído ao novo endereço cadastrado, além dos demais campos.
            - **HTTP Status Code 400 (Bad Request)** em caso de requisição com dados inválidos, informando mensagem de erro explicativa no corpo do response.

S19 - Listagem de Endereco

- Serviço para listar todos os endereços cadastrados no sistema.

- Definição do Endpoint:
    - Request:
        - **HTTP GET no path /api/enderecos**
        - Não é necessário request body
    - Response:
        - **HTTP Status Code 200 (OK)**, com uma listagem de todos os endereços cadastrados no sistema.

4.2 DOCUMENTANÇÃO NO README.MD

Crie um arquivo readme.md no repositório do seu projeto no GitHub, para documentar a sua solução, bem como demonstrar as técnicas e linguagens utilizadas, além do escopo do projeto e como o usuário pode executar o seu sistema.

Algumas dicas interessantes para utilizar na criação do seu portfólio são:

- Criar um nome para o seu software;
- Descrever qual o problema ele resolve;
- Descrever quais técnicas e tecnologias utilizadas. Aqui você também pode inserir alguma imagem ou diagrama para melhor entendimento;
- Descrever como executar;
- Descrever quais melhorias podem ser aplicadas;
- Entre outras coisas.

4.3 GRAVAÇÃO DE VÍDEO

Além do desenvolvimento deste sistema você deverá gravar um vídeo, com tempo **máximo** de 5 minutos, abordando os seguintes questionamentos:

- Qual o objetivo do sistema? E demonstração de funcionamento.
- O que deve ser realizado para executar o sistema?
- Como você organizou as tarefas antes de começar a desenvolver?
- Quais as *branches* você criou e quais os objetivos para cada uma?
- Você acha que faltou algo no seu código que você poderia melhorar?

Você poderá gravar na vertical ou na horizontal. É importante que apareça seu rosto e esteja em um local com boa iluminação. Para realizar a entrega do vídeo, coloque em uma pasta do **Google Drive** em modo leitor para qualquer pessoa com o link, e compartilhe o mesmo na submissão do projeto no AVA. Uma dica interessante é você inserir o vídeo no readme.md do seu projeto no repositório do GitHub.

4.4 COLLECTION POSTMAN

Além do desenvolvimento deste sistema você deverá adicionar uma Collection do Postman contendo todos os endpoints desenvolvidos e com dados válidos para teste do sistema.

O arquivo da Collection deve estar em uma pasta **postman** na **raiz** do projeto.

5 CRITÉRIOS DE AVALIAÇÃO

A tabela abaixo apresenta os critérios que serão avaliados durante a correção do projeto. O mesmo possui variação de nota de 0 (zero) a 10 (dez) como nota mínima e máxima, e possui peso de **40% sobre a avaliação do módulo**.
Serão **desconsiderados e atribuída a nota 0 (zero)** os projetos que apresentarem plágio de soluções encontradas na internet ou de outros colegas. Lembre-se: Você está livre para utilizar outras soluções como base, mas **não é permitida** a cópia.

Nº	Critério de Avaliação	0	1,00	
1	Realizou a gravação de um vídeo?	Não foi realizada a gravação do vídeo.	Gravou o vídeo e abordou todos os tópicos listados no item 4.3.	
Nº	**Critério de Avaliação**	**0**	**0,75**	**1,00**
2	Criou uma documentação com readme.md?	Não criou a documentação.	Criou uma documentação,	Criou uma documentação completa com ao

			porém de forma muito simplificada.	menos todos os tópicos sugeridos no item 4.2
Nº	**Critério de Avaliação**	**0**	**0,25 a 0,50**	**0,75 a 1,00**
3	Adição da Collection do Postman	Não adicionou a Collection do Postman	Adicionou a Collection do Postman, porém com falta de endpoints	Adicionou a Collection do Postman com todos os Endpoints
Nº	**Critério de Avaliação**	**0**	**0,50 a 1,00**	**1,25 a 1,50**
4	A criação de todos os scripts Oracle	Não realizou a criação de todos os scripts Oracle	Realizou a criação parcial de todos os scripts Oracle	Realizou a criação de todos os scripts Oracle
5	Integração do Programa com o Banco de Dados Oracle	Não realizou a Integração do Programa com o Banco de Dados Oracle	Realizou a Integração com um banco de dados H2	Realizou a Integração do Programa com o Banco de Dados Oracle
Nº	**Critério de Avaliação**	**0**	**0,50 a 1,00**	**1,25 a 2,00**
6	Criação dos Endpoints	Não criou todos os Endpoints	Criou parcialmente os Endpoints	Criou todos os Endpoints
7	Criação das resposta dos Endpoints de acordo com as especificações do projeto	Não criou as resposta dos Endpoints de acordo com as especificações do projeto	Criou as resposta dos Endpoints de forma parcial, incompleta ou aquém das especificações do projeto	Criou as resposta dos Endpoints de acordo com as especificações do projeto

6 PLANO DE PROJETO

Ao construir a aplicação proposta, o aluno estará colocando em prática os aprendizados em:

- **Programação Orientada a Objetos**: Conceitos de POO, Classes, Objetos, Métodos, Atributos, Construtores, Modificadores, Encapsulamento, Sobrecarga, Herança e Polimorfismo.
- **Modelagem**: Criação de Classes e Abstração.
- **Versionamento**: Uso do GitHub para versionamento de código.
- **Java**: Java básico, Estruturas de Controle de Fluxo, Arrays e ArrayList, Documentação de Código e Tratamento de Exceções.
- **Spring**: Java Servlets, Spring Core, Spring Boot, Maven, Spring Data, CRUD Rest API.
- **Banco de Dados**: Modelo Relacional e SQL com SQL Oracle.
- **Skills**: Organização, criação de documentação e apresentação de solução.