FullStack [Trindade]

Módulo 1 - Projeto Avaliativo

SUMÁRIO

1 INTRODUÇÃO

A **365 Medical Inc**, empresa líder no segmento tecnológico para gestão hospitalar, está tomando algumas ações para testar e automatizar determinados processos nos atendimentos de pacientes em âmbito hospitalar. O seu perfil chamou a atenção dos gestores, para criar o Produto Viável Mínimo (Inglês: MVP) da API Rest, que deverá ser construída utilizando **JavaScript**, **ExpressJS** e **PostgreSQL.**

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 1 - Projeto Avaliativo**, presente na semana 11 do AVA até o dia **23/04/2023** às **23h55**.

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

- Henrique Douglas Cavalcante - **douglas-cavalcante**
- 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 **JavaScript** com uso do **ExpressJS**;
2. Utilizar o banco de dados **PostgreSQL**;
1. Ser versionado no **GitHub**, possuindo uma documentação **readme.md** sobre o projeto e como utilizar;
3. Possuir um **vídeo** explicativo sobre o projeto.
4. Seguir o **Roteiro da Aplicação**;

4 ROTEIRO DA APLICAÇÃO

A **LABMedicine**, deseja automatizar algumas ações de atendimento, criando um sistema para armazenamento de informações referente aos pacientes, enfermeiros e médicos.

4.1 FORMATO DO SISTEMA

O sistema deve conter os tipos de cadastros abaixo, cada um com suas características.

Carregamento de Dados Iniciais

- Deve ser utilizado como Sistema Gerenciador de Banco de Dados o **PostgreSQL**, e a aplicação deve usar como nome do banco de dados ***labmedicinebd***.

S01 - Cadastro de Paciente

- Serviço de cadastro de **Paciente**, cuja deve possuir os seguintes atributos:
  - 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.
  - CPF: Deve ser texto
  - Telefone: Deve ser texto
  - Contato de Emergência: Obrigatório, Deve ser texto
  - Lista de Alergias: Não obrigatório para a criação da classe
  - Lista de Cuidados Específicos: Não obrigatório para a criação da classe
  - Convênio: Não obrigatório para a criação da classe
  - Status de Atendimento: Um paciente pode estar com as seguintes situações:
    - Aguardando Atendimento
    - Em Atendimento
    - Atendido
    - Não Atendido
  - Total de atendimentos realizados.
    - Este item é um contador que inicia em zero. Sempre que um médico realiza um atendimento este valor deve ser incrementado. Por padrão inicializar com o valor 0.

- 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 "atendimentos", 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.

S02 - 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 , exceto o status do atendimento e total_de_atendimentos .
        - 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.

S03 - Atualização do Status de Atendimento

- Serviço para alterar/atualizar o status de atendimento de determinado paciente.
- O usuário do sistema poderá alterar esta situação sempre que necessário.

- Definição do Endpoint:
  - Request:
    - **HTTP PUT no path /api/pacientes/{identificador}/status**
    - No corpo da request, informar objeto json com os campos.
    - O campo deve ser validado como sendo obrigatório e pertencente aos valores possíveis para este campo.
  - 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 - 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 prever um *query param* opcional para filtrar o resultado da consulta pelo status de atendimento.
      - *query param* = "status" (não obrigatório ser informado na request)
      - Valores possíveis para serem informados na requisição = AGUARDANDO_ATENDIMENTO, EM_ATENDIMENTO, ATENDIDO e NAO_ATENDIDO
      - Exemplo de path com o *query param* informado:
        - */api/pacientes?status=ATENDIDO*
    - 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.

S05 - 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.

S06 - 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.
  - 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.

S07 - Cadastro de Médico

- Serviço de cadastro de **Médico** e possuir os seguintes atributos:
  - 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.
  - CPF: Deve ser texto
  - Telefone: Deve ser texto
  - Instituição de Ensino da Formação: Obrigatório, deve ser texto.
  - Cadastro do CRM/UF: Obrigatório, deve ser texto.
  - Especialização Clínica: Obrigatório com as seguintes opções
    - Clínico Geral
    - Anestesista
    - Dermatologia

        - Ginecologia
        - Neurologia
        - Pediatria
        - Psiquiatria
        - Ortopedia
    - Estado no Sistema
        - Ativo (Por padrão, caso não informado, cadastra o médico como ativo)
        - Inativo
    - Total de atendimentos realizados:
        - Este item é um contador que inicia em zero. Sempre que um médico realiza um atendimento este valor deve ser incrementado
        - O sistema deve perguntar qual foi o médico e paciente que participaram do atendimento. O atendimento médico deve ter o Identificador do paciente.

- Definição do Endpoint:
    - Request:
        - **HTTP POST no path /api/medicos**
        - No corpo da request, informar objeto json com os campos
        - Todos os campos obrigatórios devem ser validados. O CPF deve ser único por 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 médico cadastrado, além dos demais campos. No response, retornar os campos adicionais “identificador” e “atendimentos”, 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.

S08 - Atualização dos dados de Médicos

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

- Definição do Endpoint:

- Request:
  - **HTTP PUT no path /api/medicos/{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 médico.
  - **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.

S09 - Atualização do Estado do Médico no sistema

- Serviço para alterar/atualizar o estado do médico no sistema.
- O usuário do sistema poderá alterar este estado sempre que necessário.

- Definição do Endpoint:
  - Request:
    - **HTTP PUT no path /api/medicos/{identificador}/status**
    - No corpo da request, informar objeto json com os campos
    - O campo deve ser validado como sendo obrigatório e pertencente aos valores possíveis para este campo.
  - Response:
    - **HTTP Status Code 200 (OK)** em caso de sucesso, constando no corpo da resposta os dados atualizados do médico.
    - **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.

S10 - Listagem de Médicos

- Serviço de listagem de médicos cadastrados.

- Definição do Endpoint:

- Request:
  - **HTTP GET no path /api/medicos**
  - Não é necessário request body
  - Deve prever um *query param* opcional para filtrar o resultado da consulta pelo estado no sistema.
    - *query param* = "status" (não obrigatório ser informado na request)
    - Valores possíveis para serem informados na requisição = ATIVO e INATIVO
    - Exemplo de path com o *query param* informado:
      - */api/pacientes?status=ATIVO*
  - 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.

S11 - Listagem de Médico pelo identificador

- Serviço de consulta de médicos pelo seu código identificador.

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

S12 - Exclusão de Médico

- Serviço para excluir um médico pelo código identificador.

- Definição do Endpoint:
  - Request:
    - **HTTP DELETE no path /api/medicos/{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 Enfermeiro

- Serviço de cadastro de **Enfermeiro**, cuja entidade deve herdar de **Pessoa** e possuir os seguintes atributos:
  - 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.
  - CPF: Deve ser texto
  - Telefone: Deve ser texto
  - Instituição de Ensino da Formação: Obrigatório, deve ser texto.
  - Cadastro do COFEN/UF: Obrigatório, deve ser texto.

- Definição do Endpoint:
  - Request:
    - **HTTP POST no path /api/enfermeiros**
    - No corpo da request, informar objeto json com os campos
    - Todos os campos obrigatórios devem ser validados. O CPF deve ser único por enfermeiro. 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 enfermeiro cadastrado, além dos demais campos. No response, retornar o campo adicional "identificador", usando obrigatoriamente este nome para o campo.
      - **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.

S14 - Atualização dos dados de Enfermeiros

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

- Definição do Endpoint:
  - Request:

        - **HTTP PUT no path /api/enfermeiros/{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 enfermeiro.
        - **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.

S15 - Listagem de Enfermeiros

- Serviço de listagem de enfermeiros cadastrados.

- Definição do Endpoint:
    - Request:
        - **HTTP GET no path /api/enfermeiros**
        - Não é necessário request body
        - Deve retornar todos os registros da base de dados.
    - Response:
        - **HTTP Status Code 200 (OK)**, com a lista de enfermeiros.

S16 - Listagem de Enfermeiro pelo identificador

- Serviço de consulta de enfermeiro pelo seu código identificador.

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

S17 - Exclusão de Enfermeiro

- Serviço para excluir um enfermeiro pelo código identificador.

- Definição do Endpoint:
  - Request:
    - **HTTP DELETE no path /api/enfermeiros/{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.

S18 - Realização de Atendimento Médico

- Serviço de atendimento médico, onde deve ser informado o código(id) do paciente e código(id) do médico que participou do atendimento.
- Sempre que um atendimento é realizado, devem ser incrementados os atributos de atendimento do paciente e médico envolvidos.
- Sempre que um atendimento é realizado, o status de atendimento do paciente deve ser alterado para “Atendido” (valor = “ATENDIDO”).

- Definição do Endpoint:
  - Request:
    - **HTTP POST no path /api/atendimentos**
    - No corpo da request, informar objeto json com os campos de identificador do paciente e identificador do médico.
    - Ambos os campos devem ser validados como sendo de preenchimento obrigatório.
  - Response:
    - **HTTP Status Code 200 (OK)** em caso de sucesso, constando no corpo da resposta todos os campos previstos para paciente e médico, conforme.
    - HTTP Status Code 400 (Bad Request) em caso de requisição com dados inválidos/faltantes, informando mensagem de erro explicativa no corpo do response.
    - HTTP Status Code 404 (Not Found) em caso de Paciente ou Médico não encontrado com o código informado, com mensagem de erro explicativa no corpo do response.

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.

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 **60% 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	1,50	
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**	**1,25 a 1,50**
2	Criou uma documentação com readme.md?	Não criou a documentação.	Criou uma documentação, porém de forma muito simplificada.	Criou uma documentação completa com ao menos todos os tópicos sugeridos no item 4.2
Nº	**Critério de Avaliação**	**0**	**0,25 a 0,45**	**0,50 a 0,75**
3	Criou a rota de cadastro do paciente ?	Não criou a rota de cadastro do paciente.	Criou a rota de cadastro de paciente, porém não implementou todas as validações ou possíveis respostas da requisição corretamente.	Criou a rota de cadastro com todas as validações e respostas conforme o documento. Além disso, aplicou as boas práticas de programações e aplicou uma estrutura de pastas corretas para o contexto da aplicação.
4	Criou a rota de cadastro do médico?	Não criou a rota de cadastro do médico.	Criou a rota de cadastro de médico, porém não implementou todas as validações ou possíveis respostas da requisição corretamente.	Criou a rota de cadastro com todas as validações e respostas conforme o documento. Além disso, aplicou as boas práticas de programações e aplicou uma estrutura de pastas corretas para o contexto da aplicação.
5	Criou a rota de cadastro do enfermeiro?	Não criou a rota de cadastro do enfermeiro.	Criou a rota de cadastro de enfermeiro, porém não implementou todas as validações ou	Criou a rota de cadastro com todas as validações e respostas conforme o documento. Além disso, aplicou as boas práticas de

			possíveis respostas da requisição corretamente.	programações e aplicou uma estrutura de pastas corretas para o contexto da aplicação.
Nº	**Critério de Avaliação**	**0**	**0,15**	**0,25 a 0,50**
6	Criou a rota de atualização do paciente ?	Não criou a rota de atualização do paciente.	Criou a atualização de cadastro de paciente, porém não implementou todas as validações ou possíveis respostas da requisição corretamente.	Criou a rota de atualização com todas as validações e respostas conforme o documento. Além disso, aplicou as boas práticas de programações e aplicou uma estrutura de pastas corretas para o contexto da aplicação.
7	Criou a rota de atualização do médico ?	Não criou a rota de atualização do médico.	Criou a rota de atualização do médico, porém não implementou todas as validações ou possíveis respostas da requisição corretamente.	Criou a rota de atualização com todas as validações e respostas conforme o documento. Além disso, aplicou as boas práticas de programações e aplicou uma estrutura de pastas corretas para o contexto da aplicação.
8	Criou a rota de atualização do enfermeiro?	Não criou a rota de atualização do médico.	Criou a rota de atualização de enfermeiro, porém não implementou todas as validações ou possíveis respostas da requisição corretamente.	Criou a rota de atualização com todas as validações e respostas conforme o documento. Além disso, aplicou as boas práticas de programações e aplicou uma estrutura de pastas corretas para o contexto da aplicação.

9	Criou a rota de cadastro de atendimento médico ?	Não criou a rota de cadastro de atendimento médico.	Criou a rota de cadastro de atendimento médico, porém não implementou todas as validações ou possíveis respostas da requisição corretamente.	Criou a rota de cadastro com todas as validações e respostas conforme o documento. Além disso, aplicou as boas práticas de programações e aplicou uma estrutura de pastas corretas para o contexto da aplicação.
Nº	**Critério de Avaliação**	**0**	**0,10**	**0,15 a 0,25**
10	Criou a rota de deleção do paciente ?	Não criou a rota de deleção paciente.	Criou a rota de delação do paciente , porém não implementou todas as validações ou possíveis respostas da requisição corretamente.	Criou a rota de deleção com todas as validações e respostas conforme o documento. Além disso, aplicou as boas práticas de programações e aplicou uma estrutura de pastas corretas para o contexto da aplicação.
11	Criou a rota de deleção do médico?	Não criou a rota de deleção de médico.	Criou a rota de delação do médico, porém não implementou todas as validações ou possíveis respostas da requisição corretamente.	Criou a rota de deleção com todas as validações e respostas conforme o documento. Além disso, aplicou as boas práticas de programações e aplicou uma estrutura de pastas corretas para o contexto da aplicação.
12	Criou a rota de deleção do enfermeiro?	Não criou a rota de deleção de enfermeiro.	Criou a rota de delação do enfermeiro, porém não implementou todas as validações ou	Criou a rota de deleção com todas as validações e respostas conforme o documento. Além disso, aplicou as boas práticas de

			possíveis respostas da requisição corretamente.	programações e aplicou uma estrutura de pastas corretas para o contexto da aplicação.
13	Criou a rota de listagens do paciente pelo identificador(:id) ?	Não criou a rota de listagem do paciente por identificador.	Criou a rota de listagem por identificador do paciente , porém não implementou todas as validações ou possíveis respostas da requisição corretamente.	Criou a rota de listagem por identificador com todas as validações e respostas conforme o documento. Além disso, aplicou as boas práticas de programações e aplicou uma estrutura de pastas corretas para o contexto da aplicação.
14	Criou a rota de listagens do médico pelo identificador(:id) ?	Não criou a rota de listagem do médico por identificador.	Criou a rota de listagem por identificador do médico , porém não implementou todas as validações ou possíveis respostas da requisição corretamente.	Criou a rota de listagem por identificador com todas as validações e respostas conforme o documento. Além disso, aplicou as boas práticas de programações e aplicou uma estrutura de pastas corretas para o contexto da aplicação.
15	Criou a rota de listagens do enfermeiro pelo identificador(:id) ?	Não criou a rota de listagem do enfermeiro por identificador.	Criou a rota de listagem por identificador do enfermeiro, porém não implementou todas as validações ou possíveis respostas da requisição corretamente.	Criou a rota de listagem por identificador com todas as validações e respostas conforme o documento. Além disso, aplicou as boas práticas de programações e aplicou uma estrutura de pastas corretas para o contexto da aplicação.

16	Criou a rota de listagens de paciente ?	Não criou a rota de listagem de pacientes.	Criou a rota de listagem, porém não implementou todas as validações ou possíveis respostas da requisição corretamente.	Criou a rota de listagem com todas as validações e respostas conforme o documento. Além disso, aplicou as boas práticas de programações e aplicou uma estrutura de pastas corretas para o contexto da aplicação.
17	Criou a rota de listagens de médico?	Não criou a rota de listagem de médicos.	Criou a rota de listagem, porém não implementou todas as validações ou possíveis respostas da requisição corretamente.	Criou a rota de listagem com todas as validações e respostas conforme o documento. Além disso, aplicou as boas práticas de programações e aplicou uma estrutura de pastas corretas para o contexto da aplicação.
18	Criou a rota de listagens de enfermeiro?	Não criou a rota de listagem de enfermeiro.	Criou a rota de listagem, porém não implementou todas as validações ou possíveis respostas da requisição corretamente.	Criou a rota de listagem com todas as validações e respostas conforme o documento. Além disso, aplicou as boas práticas de programações e aplicou uma estrutura de pastas corretas para o contexto da aplicação.
19	Criou a rota de atualização do status do paciente ?	Não criou a rota de atualização do status do paciente.	Criou a rota de atualização do status, porém não implementou todas as validações ou possíveis respostas da	Criou a rota de atualização de status com todas as validações e respostas conforme o documento. Além disso, aplicou as boas práticas de

			requisição corretamente.	programações e aplicou uma estrutura de pastas corretas para o contexto da aplicação.
20	Criou a rota de atualização do status do médico?	Não criou a rota de atualização do status do médico.	Criou a rota de atualização do status, porém não implementou todas as validações ou possíveis respostas da requisição corretamente.	Criou a rota de atualização de status com todas as validações e respostas conforme o documento. Além disso, aplicou as boas práticas de programações e aplicou uma estrutura de pastas corretas para o contexto da aplicação.
Nº	**Critério de Avaliação**	**0**	**1,00**	
CE	O aluno implementou alguma funcionalidade extra ?	O aluno não implementou nenhuma funcionalidade extra.	O aluno implementou uma funcionalidade extra que agregou valor ao projeto. Além disso, aplicou as boas práticas de programações e aplicou uma estrutura de pastas corretas para o contexto da aplicação.	

Observações importantes:

1. A nota do projeto possui variação de **zero (0)** a **dez (10)**. Caso o aluno tire 10,00 no projeto e alcance o ponto extra, a nota ficará fixada em 10,00.
2. O ponto extra só será atribuído se a funcionalidade for implementada de forma correta, **funcionando perfeitamente**.

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 de Classe, Encapsulamento e Herança.
- **Modelagem**: Criação de Classes e Abstração.
- **Versionamento**: Uso do GitHub para versionamento de código.

- **JavaScript**: Variáveis, Tipos de dados, Operadores, Arrays, Estrutura de Controle de Fluxo, Objetos, JSON, Funções, Arrow Functions, LocalStorage, Interval, Timeout, Operadores Rest e Spread, Módulos, Funções de Arrays e Funções Assíncronas.
- **ExpressJS**: Ambiente, Estrutura de projeto, API Rest, CRUD, Integração com banco de dados e Uso do Framework.
- **Banco de Dados**: Modelo Relacional e SQL com PostgreSQL.
- **Skills**: Organização, criação de documentação e apresentação de solução.