FullStack [Coqueiros]

Módulo 1 - Projeto Avaliativo

# SUMÁRIO

[1 INTRODUÇÃO 1](#_p9e2s15k061r)

[2 ENTREGA 2](#_142jktg038o9)

[3 REQUISITOS DA APLICAÇÃO 2](#_dg7uhywn42wb)

[4 ROTEIRO DA APLICAÇÃO 3](#_xeebetpgp6a)

[4.1 FORMATO DO SISTEMA 3](#_a6utvxkrz181)

[4.2 DOCUMENTANÇÃO NO README.MD 13](#_yulmb3p6y1ck)

[4.3 GRAVAÇÃO DE VÍDEO 13](#_x8dp9qnikzbv)

[5 CRITÉRIOS DE AVALIAÇÃO 14](#_o2xebzow95bl)

[6 PLANO DE PROJETO 15](#_r5h1whaoo55l)

# 

# 1 INTRODUÇÃO

A **LABMedicine LTDA**, 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 **C#**, **.Net** e **SQL Server.**

# 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~~** **26/04/2023** às **23h55**.

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

* Jamil Salim Daruix - **jamildaruix**
* Vitor Ranieri Lassen - **Vitor-Lassen**
* 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 **C#** com uso de **.Net**;
2. Utilizar o banco de dados **SQL Server Express**;
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 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.

Todos os cadastros devem ser derivados da classe **Pessoa**, que possui 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

**Carregamento de Dados Iniciais**

* Deve ser utilizado como Sistema Gerenciador de Banco de Dados o **SQL Server**, e a aplicação deve usar como nome do banco de dados ***labmedicinebd***. 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:
  + **Pacientes**: Crie 10 registros com dados fictícios.
  + **Enfermeiros**: Crie 2 registros com dados fictícios.
  + **Médicos**: Crie 2 registros com dados fictícios.
  + Dica: Utilize o [**4Devs**](https://www.4devs.com.br/) para ajudar na criação dos dados.

**S01 - Cadastro de Paciente**

* Serviço de cadastro de **Paciente**, cuja entidade deve herdar de **Pessoa** e possuir os seguintes atributos:
  + 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.
* 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.
    - 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 o campo status, e seu novo valor; EX: { “Status”:”Atendido”}
    - 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**, cuja entidade deve herdar de **Pessoa** e possuir os seguintes atributos:
  + 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
    - 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 o campo status, e seu novo valor; EX: { “Status”:”Ativo”}
    - 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 médico.
    - **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:
  + 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 do paciente e código 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 PUT 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** | **0,50** | **0,75 a 1,00** |
| 3 | Estrutura do projeto | Não organizado | Organizado | Organizado e seguindo os padrões utilizados |
| 4 | CRUD do endpoint conforme documentação | Não efetuou | Fez alguns requisitos conforme e utilizou boas práticas documento | Fez todos os requisitos pedidos conforme os verbo e aplicou as boas práticas |
| 5 | Aplicou o conceito de POO | Não efetuou | Efetuou parcialmente alguns princípios | Utilizou os princípios sobre os 4 pilares de POO |
| 6 | Configuração do Entity Framework e modelagem utilizando o Code First | Não efetuou | Aplicou parcialmente a configuração e modelagem dos modelos conforme documentação | Aplicou corretamente a configuração e fez a modelagem correta conforme documentação |
| 7 | Criou os resources e retorno dos HTTP Status code conforme documentação | Não efetuou | Aplicou parcialmente o resource e Http Status seguindo a documentação | Aplicou corretamente o resource e Http Status seguindo a documentação |
| 8 | Implementação das validações de Negócio para evitar inserção de registros duplicados | Não efetuou | Aplicou parcialmente as regras. | Aplicou corretamente todas as validações para evitar duplicidade. |
| 9 | Aplicação da regra de negócio de atendimento médico. (S18) | Não efetuou | Aplicou Parcialmente as regras | Aplicou corretamente as regras de negócio |
| 10 | Extra: Criar a tabela Atendimento, que armazenará o histórico do atendimento, poderá contar com uma descrição, o médico que atendeu e o Paciente atendido. será aplicada junto ao método (S18) | Não efetuou | Aplicou Parcialmente | Criou a tabela, e aplicou seus relacionamentos corretamente, junto com as informações adicionais. |

# 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.
* **C#**: Sintaxe, Compilação, Execução, DataTypes, Variáveis, Arrays, Listas, POO, Tratamento de Exceções.
* **.Net**: Ambiente, Estrutura de projeto, API Rest, CRUD, Integração com banco de dados e Entity Framework.
* **Banco de Dados**: Modelo Relacional e SQL com SQL Server.
* **Skills**: Organização, criação de documentação e apresentação de solução.