**DOCUMENTAÇÃO DA API MEDICAR**

* **Descrição**

Trata-se de uma aplicação BackEnd desenvolvida utilizando a linguagem C# (C Sharp) no framework .Net Core 6.

* **Instalação**

A aplicação necessita apenas de dois requisitos: O .Net Framework 6 instalado e o docker.

A criação do ambiente para o correto funcionamento da aplicação consiste em duas etapas: A criação dos containers Docker que hospedam os bancos utilizados e após isso, a aplicação das migrations para construção da estrutura do banco de dados.

1. **Criação dos Containers:** Dois containers foram utilizados para a construção da aplicação, um container Microsoft SQL Server e o container MongoDB. A seguir, estão os comandos utilizados para a instalação dos containers.

**Criando o Container MSSQL**

docker run --name SqlServer -e "ACCEPT\_EULA=Y" -e "MSSQL\_SA\_PASSWORD=abcd.1234" -p 1433:1433 -d mcr.microsoft.com/mssql/server:2022-latest

**Criando Container Mongo**

docker run --name MongoDb -d -p 27017:27017 -p 28017:28017 -e MONGO\_INITDB\_ROOT\_USERNAME=admin -e MONGO\_INITDB\_ROOT\_PASSWORD=abcd.1234 mongo:latest

1. **Criação e aplicação das Migrations:** O fonte desenvolvido utilizou o método de Code First com o Entity Framework, que dispensa a criação manual do banco a partir do Manager, os comandos a seguir devem ser executados como terminal aberto dentro do projeto de repositório da API (Pasta: MedicarApi.Repositories):   
   dotnet ef migrations add FirstMigration  
   dotnet ef database update

* **Tecnologias e métodos utilizados**

Para o desenvolvimento da aplicação, seguem algumas das tecnologias e métodos utilizados, suas descrições e funções para a aplicação.

* **Imagens hospedadas em Docker:** Apesar de não estarem diretamente inclusas no código, imagens de servidores Microsoft SQL Server e MongoDB hospedadas localmente utilizando o Docker foram utilizadas durante 100% do processo de desenvolvimento da aplicação. A utilização das imagens viabilizou e facilitou o uso de recursos de infraestrutura mais complexa que em um ambiente de produção seriam facilmente vinculados a aplicação.
* **Bancos de Dados:** Para o desenvolvimento da aplicação foram utilizados dois bancos de dados distintos, o banco de dados relacional **Microsoft SQL Server** e o banco de dados NoSQL (Não Relacional) **MongoDB**. O motivo das escolhas se deve ao objetivo de a aplicação envolver o uso de dados de diferentes volatilidades. Utilizando o banco NoSQL para o armazenamento e manipulação de dados mais voláteis e o banco de dados Relacional para o armazenamento e manipulação de dados mais permanentes.
* **ORM:** Como escolha para o uso de técnicas de mapeamento de objeto relacional, o ORM utilizado foi o **Entity Framework.** Pela possibilidade do uso das funções do LINQ como construtor de Queries que aumenta a produtividade e agilidade na hora do desenvolvimento.
* **Autenticação por Token:** Um método de autenticação por token bearer também foi implementado. O a autenticação utilizada foi o **jwt.io**. A autenticação foi utilizada para controle de acessos, principalmente por roles. Para a agilização do processo de desenvolvimento, os usuários autenticados foram colocados de forma fixa no código, porém, o modelo possui total possibilidade de aplicação de métodos de refresh token e consulta dinâmica de usuários.
* **Arquitetura:** O modelo de arquitetura utilizado foi o **CQRS** com o uso da biblioteca **MediatR**. O modelo de segregação de responsabilidade faz o desacoplamento do domínio e separa as funções em manipuladores de comando e consulta.
* **Validações:** Para realizar algumas validações nos dados inseridos a partir das requests, a biblioteca **Fluent Validation** foi aplicada, realizando as validações mais simples como por variáveis nulas ou vazias e convergências de datas. O objetivo no uso das validações era abranger o maior alcance de casos possíveis e tornar a aplicação mais preparada contra erros e inserções incorretas.
* **Controladores e Ações**

1. **AuthenticationController:** Tem o propósito de fornecer a Action que gera o token de autenticação.
   1. POST /Login/Authenticate  
      **Descrição:** A partir de um objeto contendo usuário e senha, gera um token com validade de 8 horas a partir do momento da requisição. **Requisição:**

|  |  |
| --- | --- |
| 1  2  3  4 | {  "username": "Hashirama Senju",  "password": "doakcqtvcabyz"  } |

**Resposta:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | {  "user": {  "id": "a41dab94-d09a-46c9-8344-bf4e9d5f2ff4",  "username": "Hashirama Senju",  "password": "",  "role": "Hokage"  },  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1bmlxdWVfbmFtZSI6Ikhhc2hpcmFtYSBTZW5qdSIsInJvbGUiOiJIb2thZ2UiLCJuYmYiOjE2ODcyMDU0MTMsImV4cCI6MTY4NzIzNDIxMywiaWF0IjoxNjg3MjA1NDEzfQ.iIchAK2m3oqLOztt2oeShvawqOxcFJps2i4hGf7E3XM"  } |

1. **MedicoController:** O propósito da controlleré conter todas as ações referentes a médicos, que se entende como o cadastro de médicos e a criação de suas disponibilidades.
   1. POST /Medico/InserirMedico  
      **Descrição:** Action que realiza o cadastro de novos médicos e os armazena na base.  
      **Requisição:**

|  |  |
| --- | --- |
| 1  2  3  4  5 | {  "nome": "Pamela Isley",  "crm": "CRM 1010BR/CE",  "email": "dra.ivy@gmail.com"  } |

**Resposta:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | {  "status": "Medico cadastrado com sucesso!",  "validation": {  "isValid": **true**,  "errors": [],  "ruleSetsExecuted": [  "default"  ]  }  } |

* 1. POST /Medico/CriarAgenda  
     **Descrição:** Action que realiza o cadastro de agenda para os médicos já inseridos na base.  
     **Requisição:**

|  |  |
| --- | --- |
| 1  2  3  4  5 | {  "crm": "CRM 1007BR",  "dia": "2023-06-19",  "horarios": ["09:30", "10:30", "11:30"]  } |

**Resposta:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | {  "medico": {  "nome": "Pamela Isley",  "crm": "CRM 1010BR/CE"  },  "status": "Agenda criada com sucesso!",  "validation": {  "isValid": **true**,  "errors": [],  "ruleSetsExecuted": [  "default"  ]  }  } |

1. **AgendaController:** Contém as funções que giram em torno das agendas dos médicos já cadastrados, desde a marcação de horário até consultas de horários disponíveis.
   1. GET /consultas  
      **Descrição:** Quando acessada, traz as informações de todas as consultas marcadas que ainda irão ocorrer. **Resposta:**
   2. POST /consultas  
      **Descrição:** Realiza a marcação de uma consulta e marca na agenda do médico. Recebe um objeto contendo o dia, a hora e o id do médico. **Requisição:**

|  |  |
| --- | --- |
| 1  2  3  4  5 | {  "medico\_id": **4**,  "dia": "2023-06-21",  "horario": "09:30"  } |

**Resposta:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | {  "id": "6490bd3aef56f83faefa0217",  "medico": {  "id": **4**,  "nome": "Pamela Isley",  "crm": "CRM 1010BR/CE",  "email": "dra.ivy@gmail.com"  },  "data\_agendamento": "2023-06-19T17:40:26.2076484-03:00",  "dia": "2023-06-21T00:00:00",  "horario": "09:30"  } |

* 1. GET /agendas  
     **Descrição:** Realiza a consulta pelas agendas disponíveis dos médicos, podendo ser filtrada por um período contendo data de início e data de fim de consulta e por uma listagem de médicos que pode ser pela identificação interna ou pelo crm. Todas as informações devem ser passadas na query da URL como parâmetros. **Requisição:**

|  |
| --- |
| /agendas?medico=1&medico=2&medico=3&medico=4&data\_inicio=2023-06-18&data\_final=2023-06-21  Ou  /agendas?crm=CRM 1007BR&crm=CRM 1000BR&crm=CRM 1010BR/CE&data\_inicio=2023-06-20&data\_final=2023-06-25 |

**Resposta:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | [  {  "medico": {  "id": **2**,  "crm": "CRM 1007BR",  "nome": "Sakura Haruno",  "email": "sakura.haruno@gmail.com"  },  "agenda": [  {  "dia": "2023-06-20T00:00:00",  "horarios": [  "13:00",  "15:00"  ]  }  ]  },  {  "medico": {  "id": **4**,  "crm": "CRM 1010BR/CE",  "nome": "Pamela Isley",  "email": "dra.ivy@gmail.com"  },  "agenda": [  {  "dia": "2023-06-21T00:00:00",  "horarios": [  "10:30",  "11:30"  ]  }  ]  }  ] |

* 1. DELETE /consultas/{id}  
     **Descrição:** Desmarca a consulta indicada pelo ID da consulta, como o registro de consultas é colocado em um banco não relacional, a identificação passa a ser uma string contendo o GUID que é informado na resposta da requisição de marcação de consulta. A Action não tem retorno. **Requisição:**

|  |
| --- |
| /consultas/6490bd3aef56f83faefa0217 |