POO Avançado

Trabalho G2

Daniel Lins Maciel – Israel Tasca da Luz

Novembro – 2020

Sumário

Requisitos

Microsoft Visual Studio Community 2019 ou superior

https://visualstudio.microsoft.com/pt-br/vs/community/

.Net Core SDK 2.1.1

https://dotnet.microsoft.com/download/dotnet-core/2.1

Extensão Keyoti.Conveyor (para poder testar a API com o Postman)

https://marketplace.visualstudio.com/items?itemName=vs-publisher-1448185.ConveyorbyKeyoti

Postman versão Desktop (para fazer as requisições HTTP)

https://www.postman.com/downloads/

Heidi SQL (para consultar no banco de dados)

https://www.heidisql.com/download.php

Tecnologias usadas

Microsoft .Net Core SDK 2.1.1

Linguagem de programação: C#

Banco de dados: MySQL v. 10.2

Certificado SSL: Let’s Encrypt

Bibliotecas:

Json 4.6.0

MySQL Data 6.10.9

Active Directory 4.5.0

Arquitetura

Repositório GitHub:

A API é REST. Trabalha com requisições POST baseados em endereços. Não é RESTFull, pois as requisições trabalham somente com o método POST. No caso de uma API RESTFull, cada tipo de requisição do CRUD deveria um método HTTP distinto: GET, POST, PUT, DELETE

Optamos por utilizar somente o método POST por questões de escolha pessoal.

Estrutura do projeto:

Informações sobre funcionamento de funções, consulte os comentários no código.

Classes.cs = Onde estão as estruturas das classes.

Controladores.cs = A camada dos controladores, onde estão as chamadas de funções a serem usadas nos métodos POST.

DB.cs = Classe que trata da conversão dos comandos recebidos para chamadas MySQL. Também controla a conexão com o banco.

Utilz.cs = Funções genéricas que são usadas por todo o aplicativo.

Vars.cs = Arquivo de configuração. Por questões de segurança, o arquivo está em branco. As configurações devem ser as seguintes:

a) Chave_Criptografia = Chave usada para criptografar as senhas
b) BancoUsers = Nome do banco de dados MySQL
c) BancoActiveDirectory = Banco MySQL para gravar dados se for usado Active Directory
d) TabelaUsers = "api_usuarios"
e) Servidor = Nome do servidor MySQL
f) Servidor2 = Nome do servidor MySQL para Active Directory
g) Porta = Porta MySQL (Padrão: 3306)
h) Senha = Senha MySQL
i) Usuario = Usuário MySQL
j) LDAP = Servidor LDAP usado pelo Active Directory.

Para ativar a funcionalidade que utiliza o Active Directory, basta alterar a propriedade ‘Tipo’ na classe banco para ‘ActiveDirectory’ e setar o LDAP na propriedade no arquivo ‘Vars.cs’

O servidor ISS precisa estar dentro do domínio.

Tabelas

SGBD Utilizado: MySQL

api_usuarios = tabela onde são gravados os registros de usuários

api_usuarios_denunciar = tabela onde são gravados os registros de denúncia

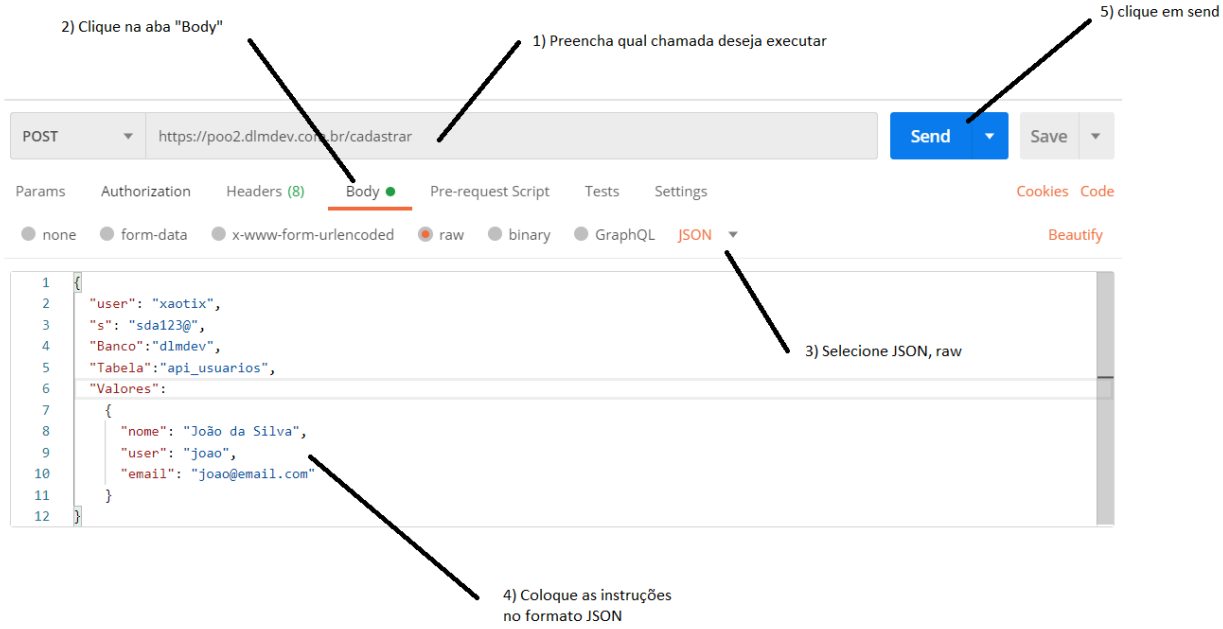
Scripts para criação das tabelas:

```
CREATE TABLE IF NOT EXISTS `api_usuarios` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `nome` varchar(150) DEFAULT NULL,
  `user` varchar(50) DEFAULT NULL,
  `email` varchar(50) DEFAULT NULL,
  `s` varchar(500) DEFAULT NULL,
  `ultima_edicao` timestamp NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),
  `criado` timestamp NULL DEFAULT current_timestamp(),
  PRIMARY KEY (`id`),
  KEY `nome` (`nome`),
  KEY `ma` (`user`) USING BTREE
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;


CREATE TABLE IF NOT EXISTS `api_usuarios_denuncia` (
  `nome` varchar(150) DEFAULT NULL,
  `email` varchar(150) DEFAULT NULL,
  `denunciado_login` varchar(150) DEFAULT NULL,
  `denunciado_id` int(11) DEFAULT NULL,
  `denunciado_nome` varchar(150) DEFAULT NULL,
  `denunciado_descricao` varchar(5000) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Utilizando o Postman

Baixe o aplicativo: https://www.postman.com/downloads/




```
{
  "user": "xaotix",
  "s": "sda123@",
  "Banco":"dlmdev",
  "Tabela":"api_usuarios",
  "Valores":
    {
      "nome": "João da Silva",
      "user": "joao",
      "email": "joao@email.com"
    }
}
```

Utilizando a API

Uma versão compilada da API está hospedada no link: https://poo2.dlmdev.com.br/

As chamadas são feitas em POST. É obrigatório o uso dos dados de autenticação para poder rodar.

Os métodos são:

https://poo2.dlmdev.com.br/consultar

No mínimo 1 coluna da tabela deve ser especificada.

JSON Exemplo:

```
{
  "user": "xaotix",
  "s": "sda123@",
  "Banco":"dlmdev",
  "Tabela": "api_usuarios",
   "Filtros":
    {
      "id": "%%"
    }
}
```

https://poo2.dlmdev.com.br/cadastrar

JSON Exemplo:

```
{
  "user": "xaotix",
  "s": "sda123@",
  "Banco":"dlmdev",
  "Tabela":"api_usuarios",
  "Valores":
    {
      "nome": "João da Silva",
      "user": "joao",
      "email": "joao@email.com"
    }
}
```

https://poo2.dlmdev.com.br/apagar

Por questões de segurança não é possível enviar comandos de apagar vários itens de uma vez.

JSON Exemplo:

```
{
  "user": "xaotix",
  "s": "sda123@",
  "Banco":"dlmdev",
  "Tabela":"api_usuarios",
  "Filtros":
    {
      "user": "joao"
    }
}
```

https://poo2.dlmdev.com.br/atualizar

Chave filtros obrigatória: Determina na busca qual registro será editado.

```
{
  "user": "xaotix",
  "s": "sda123@",
  "Banco":"dlmdev",
  "Tabela":"api_usuarios",
  "Valores":
    {
      "user": "joao2"
    },
      "Filtros":
    {
      "user": "joao"
    }
}
```

Como fazer a denúncia:

Utilizando o Postman

1) Faça uma consulta com o método ‘consultar’ para listar todos os usuários:

```
{
  "user": "xaotix",
  "s": "sda123@",
  "Banco":"dlmdev",
  "Tabela": "api_usuarios",
   "Filtros":
    {
      "id": "%%"
    }
}
```

2) Com o id do usuário desejado, utilize o método ‘cadastrar’, gravando na tabela as seguintes colunas:

```
{
  "user": "xaotix",
  "s": "sda123@",
  "Banco":"dlmdev",
  "Tabela":"api_usuarios_denuncia",
  "Valores":
    {
      "nome": "Seu Nome",
      "email": "seuemail@email.com",
      "denunciado_login": "xaotix",
      "denunciado_id": "1",
      "denunciado_descricao": "Descreva o motivo da denúncia. No máximo 5 mil
caracteres."
    }
}
```

3) Consulte se foi cadastrado:

```
{
  "user": "xaotix",
  "s": "sda123@",
  "Banco":"dlmdev",
  "Tabela": "api_usuarios_denuncia",
   "Filtros":
    {
      "id": "%%"
    }
}
```