**FACULDADE DE INFORMÁTICA E ADMINISTRAÇÃO PAULISTA**

CARLOS ALBERTO MACHARELLI JUNIOR

CARLOS EDUARDO MENDONÇA DA SILVA

EDUARDO TOSHIO ROCHA OKUBO

KAUÊ ALEXANDRE DE OLIVEIRA

VITOR MACHADO MIRANDA

**DOCUMENTAÇÃO DO PROJETO OCEANDEX**

CARLOS ALBERTO MACHARELLI JUNIOR

CARLOS EDUARDO MENDONÇA DA SILVA

EDUARDO TOSHIO ROCHA OKUBO

KAUÊ ALEXANDRE DE OLIVEIRA

VITOR MACHADO MIRANDA

**DOCUMENTAÇÃO DO PROJETO OCEANDEX**

Trabalho acadêmico apresentado à disciplina de Java Advanced do Curso Análise e Desenvolvimento de Sistemas da Faculdade de Informática e Administração Paulista como requisito de nota da Global Solution do primeiro semestre de 2024. Requerido pelo professor Thiago Toshiyuki Izumi Yamamoto.

**SUMÁRIO**

**1 INTRODUÇÃO .....................................................................................................................4**

**3 COMO FAZER CHAMADAS A API..........................................................................................5**

**4 EXEMPLO DAS CHAMADAS PARA A API................................................................................6**

**4.1 RECUPERANDO TODOS OS DADOS........................................................................6**

**4.2 RECUPERANDO UM DADO ESPECÍFICO..................................................................7**

**4.3 ADICIONANDO UMA NOVA INFORMAÇÃO............................................................8**

**4.4 ATUALIZANDO UMA INFORMAÇÃO.......................................................................9**

**4.5 DELETANDO UMA INFORMAÇÃO..........................................................................9**

**5 MODELO DO BANCO DE DADOS.........................................................................................10**

**6 UML DAS CLASSES .............................................................................................................11**

**1 INTRODUÇÃO**

A conservação do ecossistema marinho é um desafio global que exige a conscientização e a participação de todas as pessoas. Nossa solução surge com o objetivo de facilitar a identificação e o monitoramento de espécies marinhas, apoiando pesquisas e esforços de conservação. Por meio de uma aplicação inovadora, os usuários poderão reconhecer uma espécie a partir de uma imagem, registrá-la em sua conta e acessar informações detalhadas sobre o animal avistado.

Um exemplo prático dessa tecnologia é durante passeios turísticos, onde visitantes podem avistar diversas espécies marinhas. Utilizando o aplicativo, poderão aprender mais sobre esses animais e serem incentivados a contribuir para a preservação do meio ambiente. A falta de compreensão sobre o ecossistema marinho tem levado a uma preocupante falta de conscientização sobre a necessidade de proteger esse ambiente delicado e vital. Nosso aplicativo, com sua abordagem gamificada e integração de inteligência artificial, visa preencher essa lacuna, educando e engajando os usuários de maneira divertida e interativa.

Com um enfoque educacional robusto, nossa plataforma não apenas informa, mas também inspira ações de cuidado e preservação, permitindo que todos possam conhecer e ajudar a cuidar do ecossistema marinho.

**3 COMO FAZER CHAMADAS A API**

Como requisição da matéria de Java Advanced, foi desenvolvido uma API RESTful na linguagem Java usando framework do Spring Boot. A aplicação foi pensada para que possa integrar o Front-End com o banco de dados, tornando possível a interação com os dados armazenados.

Até o presente momento, a API desenvolvida possui endpoints que possibilita algumas operações de CRUD com as entidades do banco de dados, ou seja, esses endpoints possuem os métodos GET (Um para trazer todos os dados da respectiva tabela e outro para trazer um específico por meio do seu ID), POST, PUT e DELETE, além de outras funções, como a de validar um login.

Com o programa rodando, basta usar a seguinte o URL para acessar o endpoint desejado:

[http://localhost:8080 /{endpoint}](http://localhost:8080%20/%7bendpoint%7d)

E com base nessa URL é possível acessar os seguintes endpoints:

**Usuário**

|  |  |  |
| --- | --- | --- |
| Método | URL | Sobre |
| POST | /usuarios | Insere um novo usuário |
| GET | /usuarios | Recupera todos os usuários |
| GET | /usuarios/{id} | Recupera um usuário pelo ID |
| PUT | /usuarios/{id} | Atualiza um usuário pelo ID |
| DELETE | /usuarios/{id} | Deleta um usuário pelo ID |
| GET | /usuarios/email | Recupera um usuário pelo email |
| GET | /usuarios/identificacoes | Recupera as identificações dos animais feitas pelo usuário |
| POST | /usuarios/login | Valida um Log-in |

**Animal**

|  |  |  |
| --- | --- | --- |
| Método | URL | Sobre |
| POST | /animais | Insere um novo animal |
| GET | /animais | Recupera todos os animais |
| GET | /animais/{id} | Recupera um animal pelo ID |
| PUT | /animais/{id} | Atualiza um animal pelo ID |
| DELETE | /animais/{id} | Deleta um animal pelo ID |
| GET | /animais/especie/nome | Recupera todos os animais pelo nome da espécie |
| GET | /animais/dieta/nome | Recupera todos os animais pelo nome da sua dieta |
| GET | /animais/habitat/nome | Recupera todos os animais pelo nome do seu habitat |

**4 EXEMPLOS DAS CHAMADAS PARA A API**

**4.1 RECUPERANDO TODOS OS DADOS**

URL para a chamada (Em métodos que retornam mais de um objeto também há parâmetros para a paginação):

http://localhost:8080/usuarios?size=5&page=0

Retorno:

[

    {

        "id": 1,

        "nome": "João Silva",

        "email": "joao.silva@email.com",

        "senha": "oaoj@5830"

    },

    {

        "id": 2,

        "nome": "Maria Santos",

        "email": "maria.santos@email.com",

        "senha": "airam@9325"

    },

...

]

Status HTTP:

|  |  |
| --- | --- |
| 200 OK | Retornado com sucesso. |
| 404 NOT FOUND | Nenhum dado encontrado. |

**4.2 RECUPERANDO DADOS ESPECÍFICOS**

URL para a chamada:

http://localhost:8080/usuarios/1

Retorno:

{

    "id": 1,

    "nome": "João Silva",

    "email": "joao.silva@email.com",

    "senha": "oaoj@5830",

    "fotoDePerfil": "caminho/para/a/imagem.png",

    "dataDeCadastro": "2024-05-09"

}

Status HTTP:

|  |  |
| --- | --- |
| 200 OK | Retornado com sucesso. |
| 404 NOT FOUND | Nenhum dado encontrado. |

**4.3 INSERINDO NOVOS DADOS**

URL para a chamada:

http://localhost:8080/usuarios

Exemplo de objeto que deve ser enviado:

{

    "nome": "João Silva",

    "email": "joao.silva@email.com",

    "senha": "oaoj@5830",

    "fotoDePerfil": "caminho/para/a/imagem.png"

}

Status HTTP:

|  |  |
| --- | --- |
| 201 CREATED | Criado com sucesso. |
| 400 BAD REQUEST | Envio incorreto. |

**4.4 ATUALIZANDO UM DADO**

URL para a chamada:

http://localhost:8080/usuarios/1

Exemplo de objeto que pode ser enviado (nenhum dos campos para atualização é obrigatório):

{

    "nome": "João Vieira",

    "email": "joao.vieira@email.com",

    "senha": "oaoj@5830",

    "fotoDePerfil": "caminho/para/a/imagem.png"

}

Status HTTP:

|  |  |
| --- | --- |
| 200 OK | Atualizado com sucesso. |
| 404 NOT FOUND | Nenhum dado encontrado. |

**4.5 DELETANDO UM DADO**

URL para chamada:

http://localhost:8080/usuarios/1

Status HTTP:

|  |  |
| --- | --- |
| 200 OK | Deletado com sucesso. |
| 404 NOT FOUND | Nenhum dado encontrado. |

**4.5 AUTENTICANDO UM USUÁRIO**

URL para chamada:

http://localhost:8080/usuarios/login

Exemplo de objeto que deve ser enviado:

{

    "email": "joao.vieira@email.com",

    "senha": "oaoj@5830"

}

E o retorno será os dados do usuário

Status HTTP:

|  |  |
| --- | --- |
| 200 OK | Login feito com sucesso. |
| 404 NOT FOUND | Nenhum dado encontrado. |

**5 MODELO DO BANCO DE DADOS**

Diagrama

Descrição gerada automaticamente

**6 UML DAS CLASSES**

**Diagrama

Descrição gerada automaticamente**