**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 BEAUTY TECH**

CARLOS ALBERTO MACHARELLI JUNIOR

CARLOS EDUARDO MENDONÇA DA SILVA

EDUARDO TOSHIO ROCHA OKUBO

KAUÊ ALEXANDRE DE OLIVEIRA

VITOR MACHADO MIRANDA

**DOCUMENTAÇÃO DO PROJETO BEAUTY TECH**

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 Sprint 1 da Turma 2TDSPV. Requerido pelo prof. Thiago Toshiyuki Izumi Yamamoto.

**SUMÁRIO**

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

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

**3.1 RECUPERANDO TODOS OS DADOS........................................................................6**

**3.2 RECUPERANDO UM DADO ESPECÍFICO..................................................................7**

**3.3 ADICIONANDO UMA NOVA INFORMAÇÃO............................................................8**

**3.4 ATUALIZANDO UMA INFORMAÇÃO.......................................................................9**

**3.5 DELETANDO UMA INFORMAÇÃO..........................................................................9**

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

**5 UML DAS CLASSES .............................................................................................................11**

**1 INTRODUÇÃO**

O problema que nosso projeto visa solucionar no ramo de cosméticos envolve três desafios principais: O primeiro é a falta de engajamento do cliente em buscar informações e produtos específicos, considerando que o mercado é extremamente grande. O segundo desafio é a gestão de estoque, caracterizada pela ineficiência no controle, evidenciada por um excesso de produtos menos procurados e a escassez daqueles com maior demanda. Por fim, enfrentamos a intensa concorrência nas vendas. Dado que o setor é altamente competitivo, isso nos pressiona a buscar constantemente estratégias melhores para nos destacarmos no mercado.

Nossa solução será um sistema que auxiliará o público-alvo, pessoas a partir dos 13 anos de idade, que já podem ser consumidoras digitais ou híbridas, na busca de produtos cosméticos que melhor se identifiquem com as categorias desejadas por ele (foco, cor, tipo de pele, clima, etc.), listando os produtos desejados de forma precisa e rápida. Essas buscas abrangerão clientes híbridos e digitais, já que o sistema estará disponível tanto no site quanto no aplicativo para vendas online, além de estar integrado em totens nas lojas físicas. Isso ajudará as lojas físicas a oferecerem assistência e venda dos produtos de forma mais eficiente, mostrando quantidade, especificações e detalhes, o que pode reduzir a necessidade de funcionários para assistência e aumentar a precisão das buscas para os clientes.

**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 1 endpoint que possibilita operações de CRUD com a entidade do banco de dados, ou seja, esse endpoint possui 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.

O endpoint pode ser usado para interagir com a seguinte entidade do banco de dados:

|  |  |  |
| --- | --- | --- |
| Entidade | Endpoint | Sobre |
| Cliente | “/cliente” | Recupera e manipula informações das chamadas dos clientes. |

**Como fazer a chamada?**

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)

**Exemplos de Chamadas da API**

**Recuperando todos os dados**

Para recuperar todos os dados presentes na tabela do banco basta informar o endpoint ao qual você deseja acessar as informações e realizar uma requisição GET. Você receberá como retorno um JSON com os dados dos clientes.

URL para a chamada:

http://localhost:8080/clientes

Retorno:

[

    {

        "id": 252,

        "cpf": "12345678901",

        "nome": "Nome do Cliente",

        "email": "cliente@email.com",

        "dataDeCadastro": "2024-04-12",

        "dataDeExclusao": **null**

    },

    {

        "id": 253,

        "cpf": "98765432101",

        "nome": "Nome do Cliente 2",

        "email": "cliente2@email.com",

        "dataDeCadastro": "2024-04-12",

        "dataDeExclusao": **null**

    },

...

]

Status HTTP:

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

**Recuperando um dado específico**

Além do endpoint, adicione também o ID da informação que você deseja recuperar na URL e realize uma requisição GET, assim você terá como retorno um único objeto JSON do dado com respectivo ID.

URL para a chamada:

http://localhost:8080/clientes/252

Retorno:

{

    "id": 252,

    "cpf": "12345678901",

    "nome": "Nome do Cliente",

    "email": "cliente@email.com",

    "dataDeNascimento": "1998-11-02",

    "estadoCivil": "SOLTEIRO",

    "genero": "MASCULINO",

    "dataDeCadastro": "2024-04-12",

    "dataDeExclusao": **null**

}

Status HTTP:

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

**Adicionando uma nova informação**

Para adicionar uma nova informação você precisará enviar, por meio de uma requisição POST, um objeto JSON com os atributos da entidade, respeitando as regras e restrições do banco de dados, no endpoint respectivo a tabela. Não é necessário passar o ID, pois o banco se encarregará de colocá-lo.

URL para a chamada:

http://localhost:8080/clientes

Exemplo de objeto que deve ser enviado:

{

    "cpf": "12345678901",

    "nome": "Nome do Cliente",

    "email": "cliente@email.com",

    "dataDeNascimento": "1998-11-02",

    "estadoCivil": "SOLTEIRO",

    "genero": "MASCULINO"

}

Status HTTP:

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

**Atualizando uma informação**

Caso precise atualizar os dados é possível usar uma requisição PUT e usar a URL com endpoint e o ID da informação que você deseja alterar. Você também precisará enviar um objeto JSON com os atributos da entidade ao quais você deseja realizar as alterações.

URL para a chamada:

http://localhost:8080/clientes /252

Exemplo de objeto que deve ser enviado:

{

    "cpf": "12345678901",

    "nome": "Nome do Cliente Atualizado",

    "email": "cliente@email.com",

    "dataDeNascimento": "2001-06-21",

    "estadoCivil": "VIUVO",

    "dataDeExclusao": "2024-04-11"

}

Status HTTP:

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

**Deletando uma informação**

O método DELETE necessita apenas de informar o endpoint e o ID do dado que se deseja deletar na URL e a informação será apagada no banco de dados.

URL para chamada:

http://localhost:8080/clientes/252

Status HTTP:

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

**4 MODELO DO BANCO DE DADOS**

Diagrama

Descrição gerada automaticamente

**5 UML DAS CLASSES**

**Texto

Descrição gerada automaticamente**