**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**

**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**

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 endpoints que possibilita operações de CRUD com a entidade as entidades Cliente e Produto 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 com 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:

**Cliente**

|  |  |  |
| --- | --- | --- |
| Método | URL | Sobre |
| POST | /clientes | Insere um novo cliente |
| GET | /clientes | Recupera todos os clientes. |
| GET | /clientes/{id} | Recupera um cliente pelo ID |
| PUT | /clientes/{id} | Atualiza um cliente pelo ID |
| DELETE | /clientes/{id} | Deleta um cliente pelo ID |
| POST | /clientes/login | Valida um Log-in |

**Produto**

|  |  |  |
| --- | --- | --- |
| Método | URL | Sobre |
| POST | /produtos | Insere um novo produto |
| GET | /produtos | Recupera todos os produtos. |
| GET | /produtos/{id} | Recupera um produto pelo ID |
| PUT | /produtos/{id} | Atualiza um produto pelo ID |

**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/clientes?size=5&page=0

Retorno:

[

    {

        "id": 1,

        "cpf": "35678356705",

        "nome": "João Silva",

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

        "dataDeCadastro": "2024-05-09",

        "dataDeExclusao": **null**

    },

    {

        "id": 2,

        "cpf": "25743887703",

        "nome": "Maria Santos",

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

        "dataDeCadastro": "2024-05-09",

        "dataDeExclusao": **null**

    },

...

]

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/clientes/1

Retorno:

{

    "id": 1,

    "cpf": "35678356705",

    "nome": "João Silva",

    "email": "joao.silva@email.com",

    "senha": "avlis@3457",

    "dataDeNascimento": "2004-09-12",

    "estadoCivil": "SOLTEIRO",

    "dataDeCadastro": "2024-05-09",

    "dataDeExclusao": **null**,

    "nomeGenero": "Masculino",

    "ddiTelefone": "+55",

    "dddTelefone": 11,

    "numeroTelefone": "934762778"

}

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/clientes

Exemplo de objeto que deve ser enviado:

{

    "cpf": "65384563208",

    "nome": "Gustavo Oliveira",

    "email": "gustavo.oliveira@email.com",

    "senha": "arievilo@5753",

    "dataDeNascimento": "2009-02-03",

    "estadoCivil": "CASADO",

    "idGenero": 1,

    "ddiTelefone": "+55",

    "dddTelefone": 11,

    "numeroTelefone": "998735562"

}

Status HTTP:

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

**4.4 ATUALIZANDO UM DADO**

URL para a chamada:

http://localhost:8080/clientes/3

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

{

    "cpf": "65384563208",

    "nome": "Gustavo Oliveira",

    "email": "gustavo.oliveira@email.com",

    "senha": "arievilo@8742",

    "dataDeNascimento": "2009-02-03",

    "estadoCivil": "VIUVO",

    "dataDeExclusão": **null**,

    "idGenero": 1,

    "ddiTelefone": "+55",

    "dddTelefone": 11,

    "numeroTelefone": "987346528"

}

Status HTTP:

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

**4.5 DELETANDO UM DADO**

URL para chamada:

http://localhost:8080/clientes/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/clientes/login

Exemplo de objeto que deve ser enviado:

{

    "email": "gustavo.oliveira@email.com",

    "senha": "arievilo@5753"

}

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**

**Texto

Descrição gerada automaticamente**