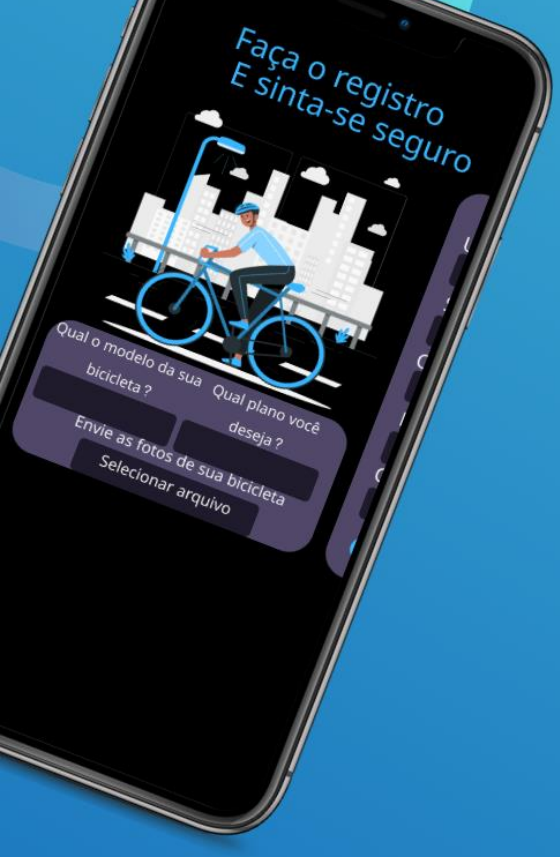
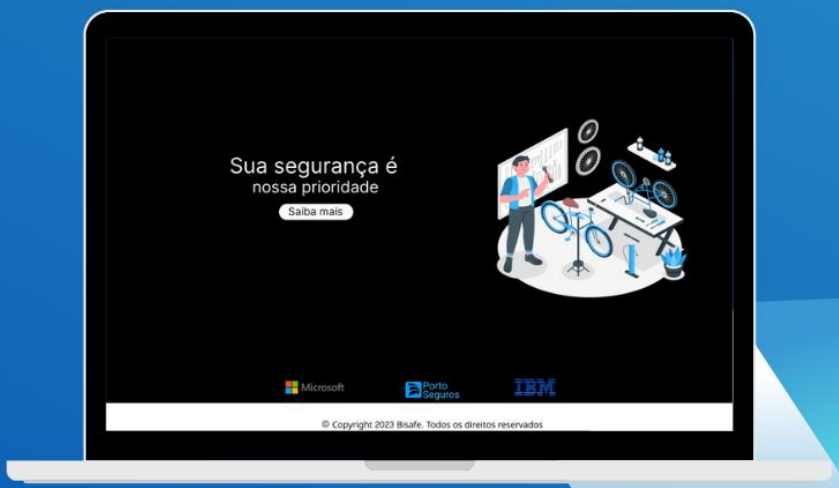


BISAFE



Alysson Pinheiro - 550837
Arthur Koga - 99503
Gabriel Henrique - 98633
Gabriel Benjamim - 552254
Henry Ribeiro - 550684
Murilo José - 99538

Sumário

BISAFE

Introdução.

A BiSafe é uma empresa dedicada a oferecer soluções de forma totalmente digital aos nossos clientes. Desenvolvemos um site que disponibiliza diversos planos de seguro para bicicletas, permitindo que os clientes escolham a opção mais adequada às suas preferências e orçamento. O processo de solicitação do seguro é simples e eficiente. Após a seleção do plano desejado, os clientes são guiados por uma série de etapas essenciais. Isso inclui o cadastro do cliente, a inserção dos detalhes da bicicleta e um minucioso processo de vistoria, composto por fotografias de diferentes partes do veículo. Essas medidas garantem uma maior segurança e certeza de que podemos contar com o cliente, assim como ele pode contar conosco.
Para esclarecer dúvidas e proporcionar uma experiência ainda mais interativa, disponibilizamos um ChatBot que está acessível 24 horas por dia. Este não apenas responde às perguntas dos clientes, mas também desempenha um papel crucial na verificação das imagens enviadas durante o processo de vistoria. Este ChatBot representa nosso diferencial, contribuindo significativamente para o sucesso deste projeto inovador.

Descrição

Temos como principal funcionalidade realizar a contratação do seguro de bicicletas e o processo de vistoria em nosso site sem a necessidade de um ser humano ativo na plataforma, apenas com o auxilio da inteligência artificial para sanar dúvidas (ChatBot) e armazenar as imagens.

3.

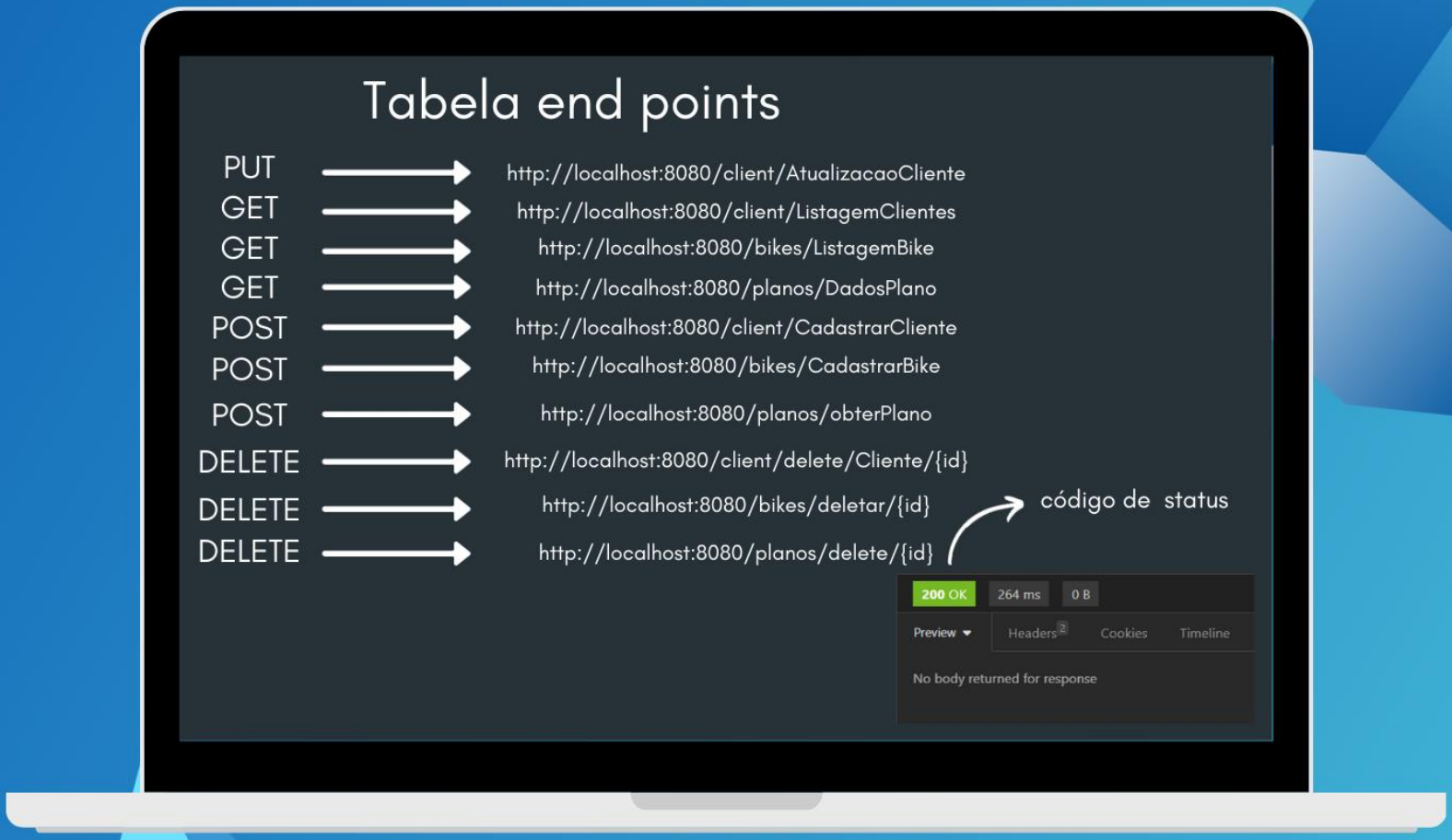
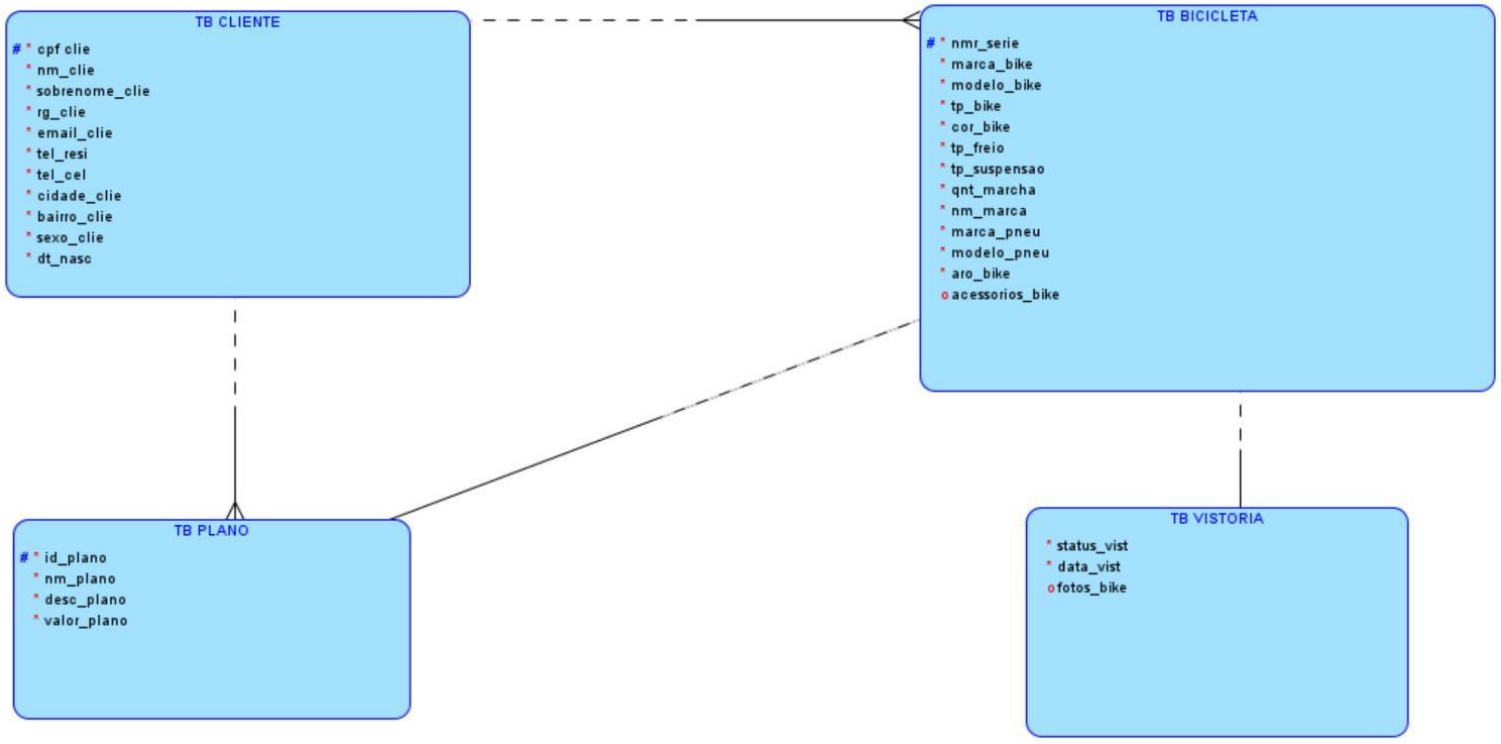


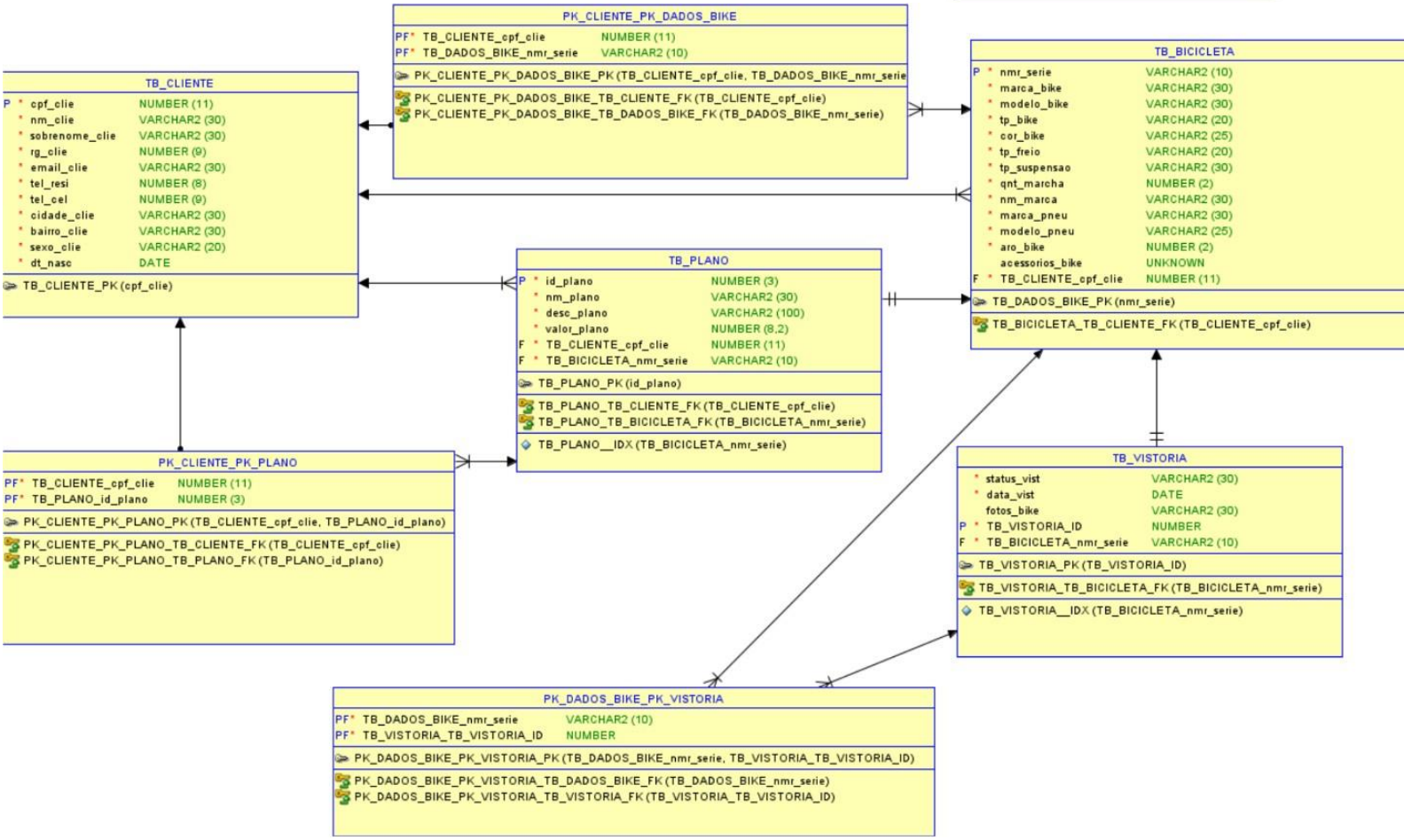
Tabela end points
PUT
http://localhost:8080/client/AtualizacaoCliente
GET
http://localhost:8080/client/ListagemClientes
GET
http://localhost:8080/bikes/ListagemBike
GET
http://localhost:8080/planos/DadosPlano
POST
http://localhost:8080/client/CadastrarCliente
POST
http://localhost:8080/bikes/CadastrarBike
POST
http://localhost:8080/planos/obterPlano
DELETE
http://localhost:8080/client/delete/Cliente/{id}
DELETE
http://localhost:8080/bikes/deletar/{id}
DELETE
http://localhost:8080/planos/delete/{id}
código de status
200 OK
264 ms
0 B
Preview
Headers
Cookies
Timeline
No body returned for response

4.

Modelo do Banco de Dados.

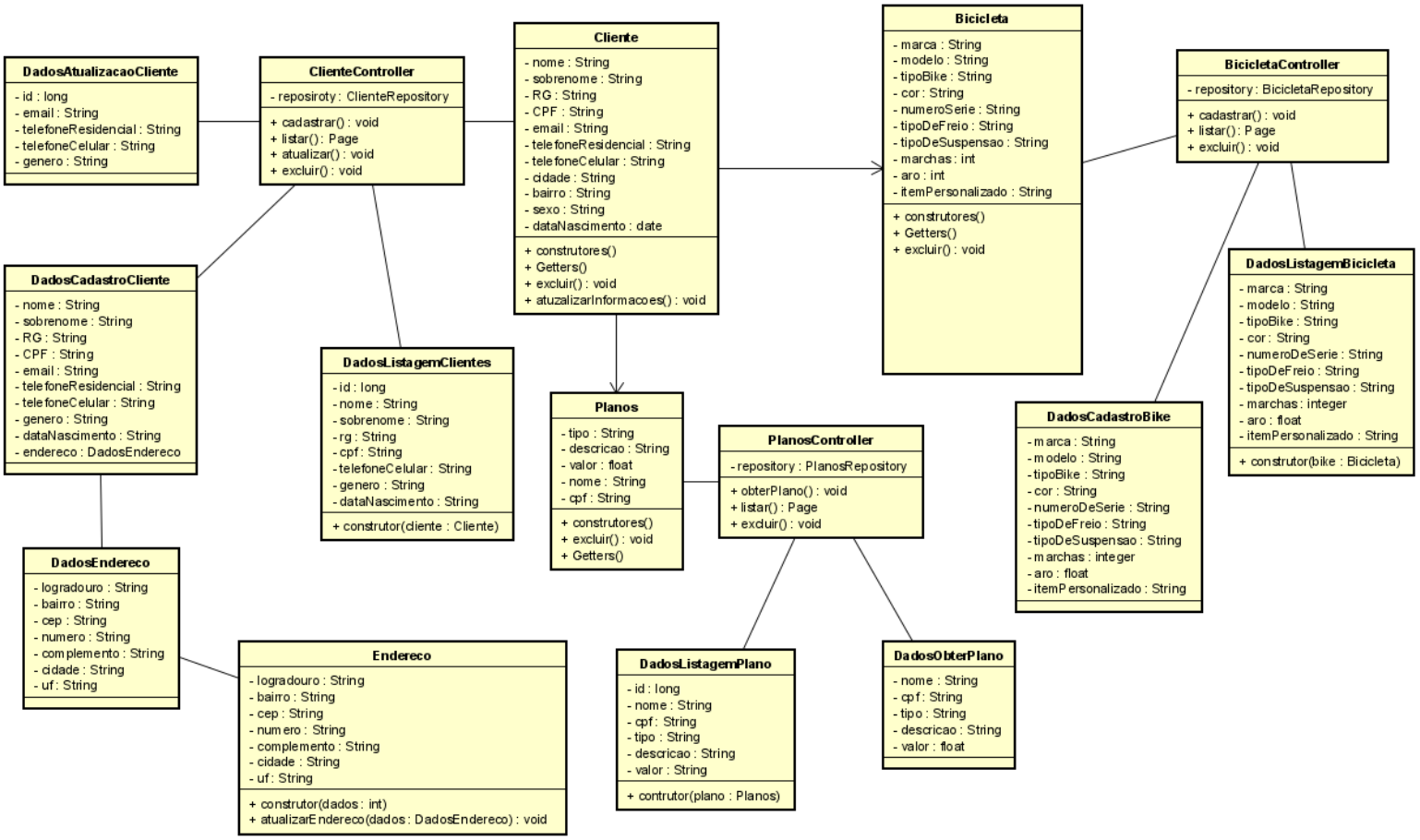
MODELOS LÓGICO E RELACIONAL





5.

Diagrama UML.



Procedimento da Aplicação.

Desenvolvemos uma API por meio da ferramenta Spring Boot, composta por quatro métodos essenciais: Cadastro, Listagem, Atualização e Exclusão de dados no banco Oracle SQL. A singularidade dessa API reside na sua capacidade de realizar essas operações sem a necessidade de acesso direto ao programa, sendo totalmente acessível por meio de protocolos HTTP em uma plataforma de execução de APIs, como o Insomnia, que foi a ferramenta escolhida para nossos testes.

Vamos abordar brevemente cada um dos métodos, começando pelo "Cadastrar". Este método tem como principal objetivo receber e armazenar os dados do cliente, da bicicleta ou do plano na tabela correspondente, sem a necessidade de interação direta com o programa. Para acessar, basta passar a URL “/client/CadastrarCliente”:

```
no usages
@PostMapping("CadastrarCliente")
@Transactional
public void cadastrar(@RequestBody @Valid DadosCadastroCliente dados) { repository.save(new Cliente(dados)); }
```

Posteriormente, implementamos o método de "Listagem", o qual tem como finalidade a apresentação de dados específicos e essenciais para a identificação do cliente, proporcionando uma visão clara e organizada dessas informações. Para executar esse método basta inserir a URL “/client/ListagemClientes”:

```
no usages
@GetMapping("/ListagemClientes")
@Transactional
public Page<DadosListagemClientes> lista (Pageable paginacao){
    return repository.findAllByAtivoTrue(paginacao).map(DadosListagemClientes::new);
}
```

Adicionalmente, implementamos o método de Atualização de Dados, proporcionando ao cliente a flexibilidade de editar informações específicas, como o endereço de e-mail, número de telefone e gênero, sempre que necessário. Isso permite uma experiência personalizada, dando ao usuário o controle sobre os detalhes relevantes de seu perfil. Esse método, deve-se inserir a URL “/cliente/AtualizacaoCliente:

```
no usages
@Transactional
@PutMapping("/AtualizacaoCliente")
public void atualizar(@RequestBody @Valid DadosAtualizacaoClientes dados){
    var cliente = repository.getReferenceById(dados.id());
    cliente.atualizarInformacoes(dados);
}
```

E, por fim, introduzimos um método de exclusão de dados, com um diferencial. O método "Excluir" não deleta diretamente os dados do cliente e sua bicicleta; em vez disso, ele indica se esses dados estão ativos no banco de dados ou não. Nessa lógica, utilizamos "1" para representar "ativo" e "0" para representar "inativo", oferecendo uma visão clara do status dos dados sem eliminar permanentemente as informações do cliente. E para excluir, basta ver na listagem qual o ID do cliente

(ou da bicicleta) e passa-lo no url “/delete/{id do cliente ou bike}”:

```
no usages
@DeleteMapping("/delete/{id}")
@Transactional
public void excluir(@PathVariable Long id){
    var cliente = repository.getReferenceById(id);
    cliente.excluir();
}
```

```
1 usage
public void excluir() {
    this.ativo = false;
}
```