

32. Atualizando Categoria com PUT, outros pequenos ajustes

Vamos implementar a operação de atualização de uma categoria usando o método HTTP PUT.

Para atualizar uma categoria no postman é preciso alterar o método para PUT e manter o URI com o ID da categoria atual.

No body deve ser passado o novo nome da categoria { “nome” : “Computadores” }.

Como este tipo de requisição irá usar o código da categoria, precisamos fazer algo parecido com o método GET.

Em CategoriaResource criar um método update(). Este método é uma mistura dos métodos GET e POST porque ele recebe um objeto JSON e então ele terá um @RequestBody Categoria obj como parâmetro igualmente ao método insert(). Ele também recebe o parametro ID da URI para identificar a categoria, então temos @PathVariable Integer id como segundo parâmetro igualmente ao método find().

Inserir a anotação @RequestMapping(value = “/{id}”, method = RequestMethod.PUT) sob o método update().

Para garantir que a categoria a ser atualizada é a que foi obtida através do código da URI fazemos obj.setId(id), apenas por desincargo de consciência.

Retornamos uma ResponseEntity sem conteúdo.

```
@RequestMapping( value = “/{id}”, method = RequestMethod.PUT)
public ResponseEntity<Void> update(@RequestBody Categoria obj, @PathVariable Integer id) {
    obj.setId(id);
    obj = categoriaService.update(obj);
    return ResponseEntity.noContent().build();
}
```

Em CategoriaService implementar um método também chamado de update()

```
update(Categoria obj) {
    find(obj.getId());
    return categoriaRepository.save(obj);
}
```

categoriaRepository.save(obj) é a mesma operação utilizada no método insert(Categoria obj) porém existe um detalhe. O método save() do repository do spring data serve para inserir e para atualizar. A diferença é que quando o ID é null ele insere o objeto, caso contrário ele atualiza o objeto.

Vamos aproveitar para verificar se o ID passado em update() existe, para fazer isso vamos utilizar o método find() que já possui uma estrutura de exceção. Vamos chama-lo antes do return.

Testando o EndPoint antes de atualizar, utilizando o método HTTP GET:

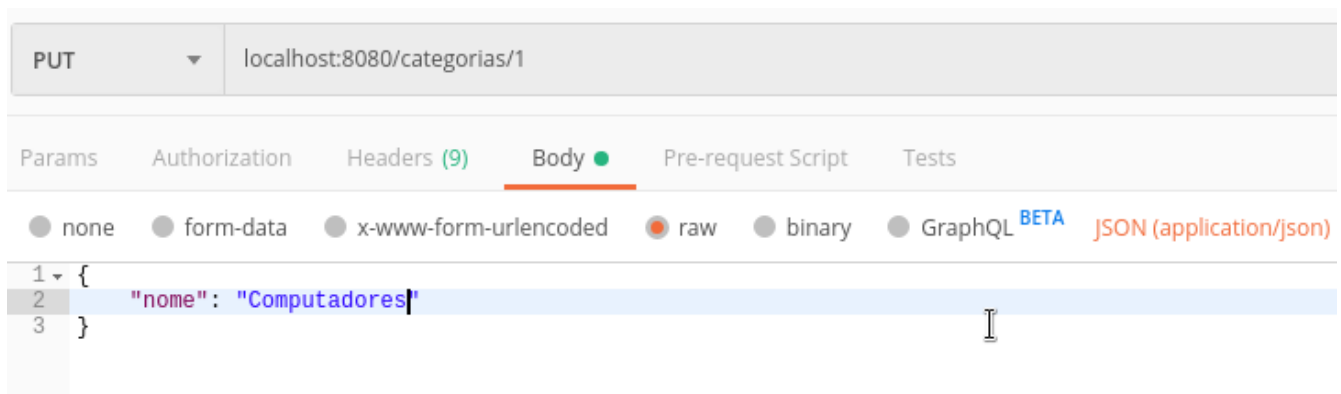


```
GET localhost:8080/categorias/1

Pretty Raw Preview JSON ↺

1 {
2   "id": 1,
3   "nome": "informática",
```

Método HTTP PUT e nome da categoria, perceba que o URI com ID é mantido:



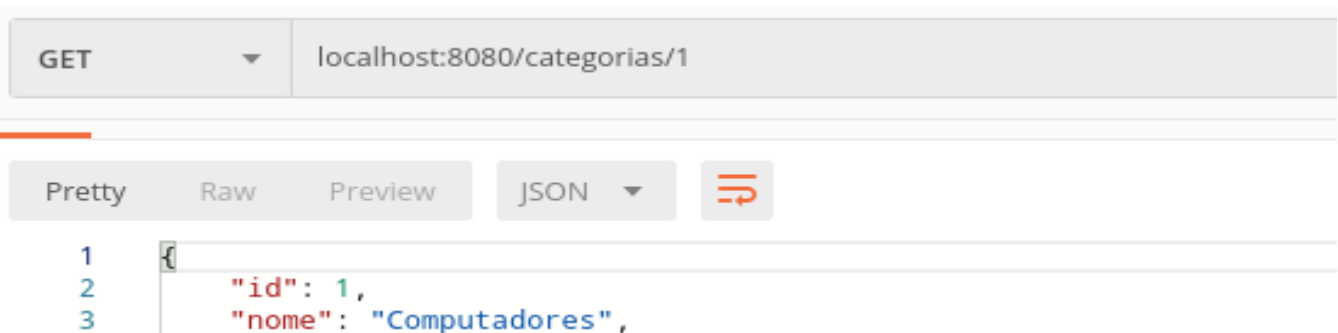
```
PUT localhost:8080/categorias/1

Params Authorization Headers (9) Body ● Pre-request Script Tests

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL BETA JSON (application/json)

1 {
2   "nome": "Computadores"
3 }
```

Categoria informática alterada para Computadores:



```
GET localhost:8080/categorias/1

Pretty Raw Preview JSON ↺

1 {
2   "id": 1,
3   "nome": "Computadores",
```