Desenvolvimento WEB

-

Full Stack Completo: Java + React

Prof. Roberto Alencar

Implementando a Alteração no projeto do Back-end

```java
@RestController
@RequestMapping("/api/cliente")
public class ClienteController extends GenericController {

    @Autowired
    private ClienteService clienteService;

    @PostMapping
    public ResponseEntity<Cliente> save(@RequestBody @Valid ClienteRequest request) {

        Cliente cliente = clienteService.save(request.build());
        return new ResponseEntity<Cliente>(cliente, HttpStatus.CREATED);
    }

    @GetMapping
    public List<Cliente> listarTodos() {

        return clienteService.listarTodos();
    }

    @GetMapping("/{id}")
    public Cliente obterPorID(@PathVariable Long id) {

        return clienteService.obterPorID(id);
    }

    @PutMapping("/{id}")
    public ResponseEntity<Cliente> update(@PathVariable("id") Long id, @RequestBody ClienteRequest request) {

        clienteService.update(id, request.build());
        return ResponseEntity.ok().build();
    }
}
```

Implementando a Alteração

1) Acrescente na classe `ClienteController` o método abaixo:

```
...
public class ClienteController {

 ...

 @PutMapping("/{id}")
 public ResponseEntity<Cliente> update(@PathVariable("id") Long id, @RequestBody ClienteRequest request) {

        clienteService.update(id, request.build());
        return ResponseEntity.ok().build();
 }

 ...

}
```

Esse método recebe o `ID` do cliente que será alterado e um objeto do tipo `ClienteRequest` contendo os dados alterados do cliente. No corpo do método, um objeto `Cliente` é criado com método `build()` a partir da classe do Request e esse objeto criado é enviado para o `service`.

Implementando a Alteração

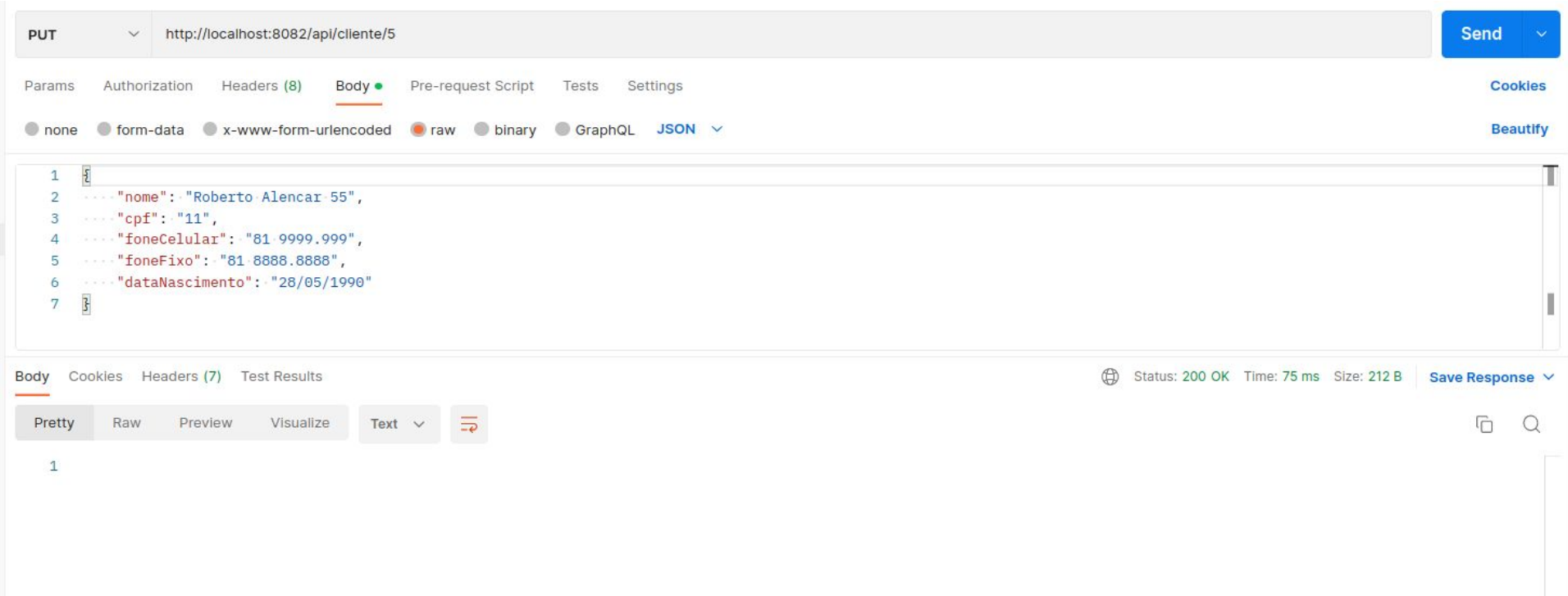
2) Acrescente na classe `ClienteService` o método abaixo:

```
...
@Service
public class ClienteService {
    ...
    @Transactional
    public void update(Long id, Cliente clienteAlterado) {

        Cliente cliente = repository.findById(id).get();
        cliente.setNome(clienteAlterado.getNome());
        cliente.setDataNascimento(clienteAlterado.getDataNascimento());
        cliente.setCpf(clienteAlterado.getCpf());
        cliente.setFoneCelular(clienteAlterado.getFoneCelular());
        cliente.setFoneFixo(clienteAlterado.getFoneFixo());

        cliente.setVersao(cliente.getVersao() + 1);
        repository.save(cliente);
    }
    ...
}
```

Esse método recebe o `ID` do cliente que será alterado e um objeto do tipo `Cliente` contendo os dados alterados do cliente. No corpo do método, o cliente que será alterado é consultado no banco de dados, os dados do cliente são passados para esse objeto consultado e ele é gravado posteriormente no banco através do método save do repositório..

Implementando a Alteração

3) Teste a alteração do cliente na ferramenta Postman



Dúvidas

Exercícios

Implemente a alteração no projeto do back-end (para o teste no Postman) de:

- `Cliente`
- `Produto`
- `Entregador`

Obrigado !