

DESAFIO: CRUD de clientes

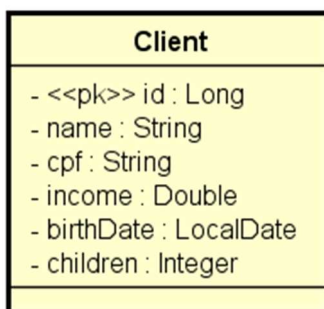
Forma de entrega: [link do projeto no Github](#)

Você deverá entregar um projeto Spring Boot contendo um CRUD completo de web services REST para acessar um recurso de clientes, contendo as cinco operações básicas aprendidas no capítulo:

- Busca paginada de recursos
- Busca de recurso por id
- Inserir novo recurso
- Atualizar recurso
- Deletar recurso

O projeto deverá estar com um ambiente de testes configurado acessando o banco de dados H2, deverá usar Maven como gerenciador de dependência, e Java como linguagem.

Um cliente possui nome, CPF, renda, data de nascimento, e quantidade de filhos. A especificação da entidade Client é mostrada a seguir (você deve seguir à risca os nomes de classe e atributos mostrados no diagrama):



powered by Astah

Seu projeto deverá fazer um seed de pelo menos 10 clientes com dados SIGNIFICATIVOS (não é para usar dados sem significado como "Nome 1", "Nome 2", etc.).

Seu projeto deverá tratar as seguintes exceções:

- Id não encontrado (para GET por id, PUT e DELETE), retornando código 404.
- Erro de validação, retornando código 422 e mensagens customizada para cada campo inválido. As regras de validação são:
 - Nome: não pode ser vazio
 - Data de nascimento: não pode ser data futura (dica: use @PastOrPresent)

Atenção: crie um **novo projeto** para este trabalho. Não é para simplesmente acrescentar a classe Client no projeto feitos nas aulas.

Atenção: lembre-se de que por padrão a JPA transforma nomes de atributos em camelCase para snake_case, como foi o caso do campo imgUrl das aulas, que no banco de dados tinha o nome img_url. Assim, o campo birthDate acima será criado no banco de dados como birth_date, então seu script SQL deverá seguir este padrão.

Atenção: cuidado para não salvar no seu projeto arquivos e pastas que não devem ser salvas no Git, tais como a pasta .metadata do Eclipse ou .idea do IntelliJ.

Como o trabalho será corrigido?

1) Importação do projeto

O professor deverá ser capaz de fazer um simples clone do projeto Github, e importar e executar o mesmo na IDE sem necessidade de qualquer configuração especial diferente daquelas das aulas.

2) Testes manuais no Postman

O professor já terá preparado em seu computador as requisições Postman abaixo. Todas elas deverão funcionar corretamente:

Busca de cliente por id

GET /clients/1

Busca paginada de clientes

GET /clients?page=0&size=6&sort=name

Inserção de novo cliente

POST /clients

```
{
  "name": "Maria Silva",
  "cpf": "12345678901",
  "income": 6500.0,
  "birthDate": "1994-07-20",
  "children": 2
}
```

Atualização de cliente

PUT /clients/1

```
{
  "name": "Maria Silvaaa",
  "cpf": "12345678901",
  "income": 6500.0,
  "birthDate": "1994-07-20",
  "children": 2
}
```

Deleção de cliente

DELETE /clients/1

CHECKLIST:

1. Busca por id retorna cliente existente
2. Busca por id retorna 404 para cliente inexistente
3. Busca paginada retorna listagem paginada corretamente
4. Inserção de cliente insere cliente com dados válidos
5. Inserção de cliente retorna 422 e mensagens customizadas com dados inválidos
6. Atualização de cliente atualiza cliente com dados válidos
7. Atualização de cliente retorna 404 para cliente inexistente
8. Atualização de cliente retorna 422 e mensagens customizadas com dados inválidos
9. Deleção de cliente deleta cliente existente
10. Deleção de cliente retorna 404 para cliente inexistente