Atividade
Testes de integração da camada Web com MockMVC

GitHub: https://github.com/brunoqp78/projeto-client-mockmvc-template
(projeto atualizado com novos endpoints)

Parte1 (70%): Utilizando o projeto base que estamos fazendo em aula, implementar no mínimo (devem criar no mínimo um endpoint diferente e testá-lo) os seguintes testes na classe ClientResourcesTestsIT

- **findAll –** feito em sala de aula, modifique os testes para inserir novos **.andExpect**

- **findById –** Testar busca por um id que existe e um id que não existe.
  - Similar ao FindAll, entretanto retorna apenas o json de um client, exemplo:

```
{
"id": 3,
"name": "Clarice Lispector",
"cpf": "10919444522",
"income": 3800.0,
"birthDate": "1960-04-13T07:50:00Z",
"children": 2
}
```

  - criar .andExpect para verificar os dados retornados
    result.andExpect(jsonPath("$.id").value(existingId));
  - Testar a busca por um id que não existe, que retorna o json de erro:

```
{
"timestamp": "2024-08-20T00:36:05.891918Z",
"status": 404,
"error": "Resource not found",
"message": "Entity not found",
"path": "/clients/id/33"
}
```

  - Verificar se retorna status notfound
  - Verificar dados da mensagens também:
    result.andExpect(jsonPath("$.error").value("Resource not found"));

- **findByIncome** (teste que retonar clientes e teste que não retorna clientes)
  - retornar OK (código 200), bem como os clientes que tenham o Income informado. Verificar se o Json Paginado tem a quantidade de clientes correta e se os clientes retornados são aqueles esperados. (similar ao exemplo feito em sala de aula – findall).
  - Necessário passar o parametro do income:

```
mockMvc.perform(get("/clients/income/")
        .param("income", String.valueOf(salarioResultado))
        .accept(MediaType.APPLICATION_JSON));
```

  - Cuidado com os valores para teste, pois o delete pode apagar algum registro. Para evitar isso, poderiam utilizar uma base de dados vazia e inserir registros em um método com a

notação @BeforeEach na classe de teste. Ou deixe o teste do delete para o fim, fazer os find primeiro.

- **findByIncomeGreaterThan** : similar ao anterior, mas com casos de teste diferentes.

- **findByCPFLike** : similar ao anterior, mas com casos de teste diferentes.

- **insert** deveria retornar “created” (código 201), bem como o produto criado, verifique no mínimo dois atributos. Lembrando que esse método retorna um Json contendo o registro criado. Para implementar esse teste é necessário definir um objeto a ser inserido, esse objeto é um clientDTO que você irá criar no teste.

  - Como criar um Json no java (clientDTO é um objeto já instanciado):

```
// Junto com os atributos da classe de teste, faça a injeção de dependência abaixo
@Autowired
private ObjectMapper objectMapper;

// no método de teste insira os códigos de criação de JSON
String json = objectMapper.writeValueAsString(clientDTO);
ResultActions result =
                    mockMvc.perform(post("/clients/")
                    .content(json)
                    .contentType(MediaType.APPLICATION_JSON)
                    .accept(MediaType.APPLICATION_JSON));
```

  - A ação retorna um Json contendo o objeto cadastrado, faça comparações de modo similar ao findbyid.

```
result.andExpect(jsonPath("$.name").value(nomeEsperado));
```

- **delete** deveria
  - retornar status “no content” (código 204) quando o id existir
  - retornar status “not found” (código 404) quando o id não existir

- **update** deveria
  - retornar “ok” (código 200), bem como o json do produto atualizado para um id existente, verifique no mínimo dois atributos. (similar ao insert, precisa passar o json modificado).

```
result.andExpect(jsonPath("$.name").value(nomeEsperado));
```

  - retornar “not found” (código 204) quando o id não existir. Fazer uma assertion para verificar no json de retorno se o campo “error” contém a string “Resource not found”.

Parte2 (20%): criar a simulação (mockbean) da classe **service.**

Observações:

- Para a entrega me enviem o repositório no github.
- Vou deixar o link do projeto base que estamos usando em aula, porém nada impede que vocês usem outro projeto. Neste caso eu peço que vocês disponibilizem uma descrição básica do sistema e os endpoints daquele sistema.