

## AT0725-2: Implementar Integração de Sistemas

José Alves de Oliveira Neto - 202201699

Exemplo de execução e verificação da integração de sistemas. O objetivo é demonstrar o funcionamento completo do projeto, desde a criação de dados na camada de persistência ORM (H2) até sua sincronização e validação na camada de persistência ODM (Redis), utilizando o Apache Camel para orquestrar o fluxo de dados.

```
Apache Camel 4.0.0 (camel-1) started in 10ms (build:0ms init:0ms start:10ms)
2025-08-04T00:32:55.720-03:00 INFO 35046 --- [modulo-integrador] [main] c.e.m.ModuloIntegradorApplication
: Started ModuloIntegradorApplication in 11.928 seconds (process running for 12.575)
-> Produto 'Notebook Dell' criado no banco de dados ORM (H2).
-> Produto 'Mouse Razer' criado no banco de dados ORM (H2).
2025-08-04T00:33:56.773-03:00 INFO 35046 --- [modulo-integrador] [timer://myTimer] route1
```

Esta etapa demonstra o ponto de partida da nossa integração. A aplicação é iniciada e a classe DataLoader, configurada para rodar automaticamente, insere dois produtos no banco de dados em memória H2. A saída do console confirma a criação de cada item.

```
2025-08-04T00:35:51.873-03:00 INFO 35046 --- [modulo-integrador] [timer://myTimer] route1
: Verificando produtos para sincronização...
2025-08-04T00:35:51.877-03:00 INFO 35046 --- [modulo-integrador] [timer://myTimer] route1
: Produto encontrado: Notebook Dell
2025-08-04T00:35:51.885-03:00 INFO 35046 --- [modulo-integrador] [timer://myTimer] c.e.modulo_integrador.IntegracaoRoute
: Produto 'Notebook Dell' sincronizado com sucesso no Redis.
2025-08-04T00:35:51.888-03:00 INFO 35046 --- [modulo-integrador] [timer://myTimer] route1
: Produto encontrado: Mouse Razer
2025-08-04T00:35:51.896-03:00 INFO 35046 --- [modulo-integrador] [timer://myTimer] c.e.modulo_integrador.IntegracaoRoute
: Produto 'Mouse Razer' sincronizado com sucesso no Redis.
```

A rota de integração, orquestrada pelo Apache Camel, monitora o banco de dados ORM. A cada ciclo, ela detecta os produtos recém-criados e inicia o processo de sincronização. O log a seguir detalha o fluxo: a rota encontra cada produto individualmente e confirma que a sincronização para o Redis foi bem-sucedida.

```
jntlv@JoseMacBook-Air ~ % docker exec -it my-redis-instance redis-cli
127.0.0.1:6379> KEYS *
1) "Produto:1"
2) "Produto:2"
3) "Produto"
127.0.0.1:6379> █
```

Nesta etapa, confirmamos que os dados foram transferidos para o sistema de destino (Redis). Usando a ferramenta de linha de comando redis-cli, o comando KEYS \* lista todas as chaves existentes. A presença das chaves Produto:1 e Produto:2 prova que a integração criou os registros correspondentes aos dois produtos.

```
[127.0.0.1:6379> HGETALL Produto:1
1) "_class"
2) "com.exemplo.modulo_odm.model.ProdutoRedis"
3) "id"
4) "1"
5) "nome"
6) "Notebook Dell"
7) "preco"
8) "5500.0"
[127.0.0.1:6379> HGETALL Produto:2
1) "_class"
2) "com.exemplo.modulo_odm.model.ProdutoRedis"
3) "id"
4) "2"
5) "nome"
6) "Mouse Razer"
7) "preco"
8) "350.0"
[127.0.0.1:6379>
```

Por fim, inspecionamos os dados para garantir que a integridade da informação foi mantida. O comando HGETALL Produto:1 e HGETALL Produto:2 exibe os atributos completos do respectivo produto, confirmando que todos os dados foram corretamente mapeados e persistidos na camada ODM.