

Explicações dos itens A-D

- A) Na requisição GET é esperado que o tempo de resposta seja pelo menos igual ao sistema sem cache do exercício 01, e possivelmente até um pouco maior. Isso ocorre porque, quando o cache está vazio, a aplicação precisa buscar os dados originais do banco de dados Postgres.

GET <http://localhost:3030/books>

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (7) Test Results Status: 200 OK Time: 202 ms Size: 532 B

Pretty Raw Preview Visualize JSON ↕

```
1 {
2   "message": "Success Postgres",
3   "status": 200,
4   "data": [
5     {
6       "id": 1,
7       "title": "titulo 1"
```

GET <http://localhost:3030/books>

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (7) Test Results Status: 200 OK Time: 8 ms Size: 529 B

Pretty Raw Preview Visualize JSON ↕

```
1 {
2   "message": "Success Redis",
3   "status": 200,
4   "data": [
5     {
6       "id": 1,
7       "title": "titulo 1"
```

- B) Isso ocorre porque o cache permite que a aplicação atenda às requisições diretamente, sem a necessidade de consultar o banco de dados.

GET http://localhost:3030/books

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (7) Test Results

Status: 200 OK Time: 6 ms Size: 529 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Success Redis",
3   "status": 200,
4   "data": [
5     {
6       "id": 1,
7       "title": "titulo 1"
8     },
9     {
10      "id": 9,
11      "title": "titulo 1"
12    }
13  ]
14 }
```

- C) Sim, isso ocorre porque, após a expiração do cache, a aplicação precisa buscar novamente os dados no Postgres.

Resultado da busca depois da expiração dos dados no Redis:

GET http://localhost:3030/books

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (7) Test Results

Status: 200 OK Time: 57 ms Size: 532 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Success Postgres",
3   "status": 200,
4   "data": [
5     {
6       "id": 1,
7       "title": "titulo 1"
8     },
9     {
10      "id": 9,
11      "title": "titulo 1"
12    }
13  ]
14 }
```

- D) Sim, para manter os dados que estão no Redis atualizados, sempre que é feito um post o cache será invalidado, assim, quando for feita uma requisição Get logo em seguida do Post o cache não existe mais, e o código volta a consultar o Postgres para pegar os dados atualizados.
- Resultado das consultas:

Primeiro Get:

GET ▼ http://localhost:3030/books

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

```
1 [{"title": "titulo 10"}]
```

Body Cookies Headers (7) Test Results 🌐 Status: 200 OK Time: 257 ms Size: 652 B 📄

Pretty Raw Preview Visualize **JSON** ▼ 🔄

```
1 [{"title": "titulo 10"}]
2
3 "message": "Success Postgres",
4 "status": 200,
5 "data": [
6   {
7     "id": 1,
8     "title": "titulo 1"
9   },
10  {
11    "id": 9,
```

Post:

POST ▼ http://localhost:3030/books

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

```
1 [{"title": "titulo 14"}]
```

Body Cookies Headers (7) Test Results 🌐 Status: 200 OK Time: 15 ms Size: 319 B 📄 Save

Pretty Raw Preview Visualize **JSON** ▼ 🔄

```
1 [{"title": "titulo 14"}]
2
3 "message": "Success",
4 "status": 200,
5 "data": {
6   "id": 27,
7   "title": "titulo 14"
8 },
```

Segundo Get:

The screenshot displays a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:3030/books
- Body:**

```
{ "title": "titulo 11" }
```
- Response:** Status: 200 OK, Time: 55 ms, Size: 742 B
- Response Body (Pretty):**

```
{  "message": "Success Postgres",  "status": 200,  "data": [    {      "id": 1,      "title": "titulo 1"    }  ]}
```