Students Rest API – With Spring Framework.

By Matheus Ferreira Santos.

Introdução

Abaixo irei apresentar os testes de cada end-point da API, ela tem por finalidade o cadastro, a alteração, a pesquisa (por um ou por vários) e a exclusão de alunos.

Para esse “exercicio” foi utilizado a linguagem de programação Java, utilizando o Spring, como framework principal, H2, como banco de dados em memória, Open Feign, para requisição em APIs externas (VIACEP), JPA/Hibernate e etc.

Também disponibilizei essa API no GitHub e no Docker, caso queira testar em sua máquina é possível executar a api utilizando algum simulador do Docker, como: … Também é possível ver o código em merepositório remoto, segue o link de ambos:

GitHub: https://github.com/Matheus-FSantos/4log-students
Docker Hub: https://hub.docker.com/repository/docker/matheusfsantos/4log-students/general (na última página segue um tutorial)

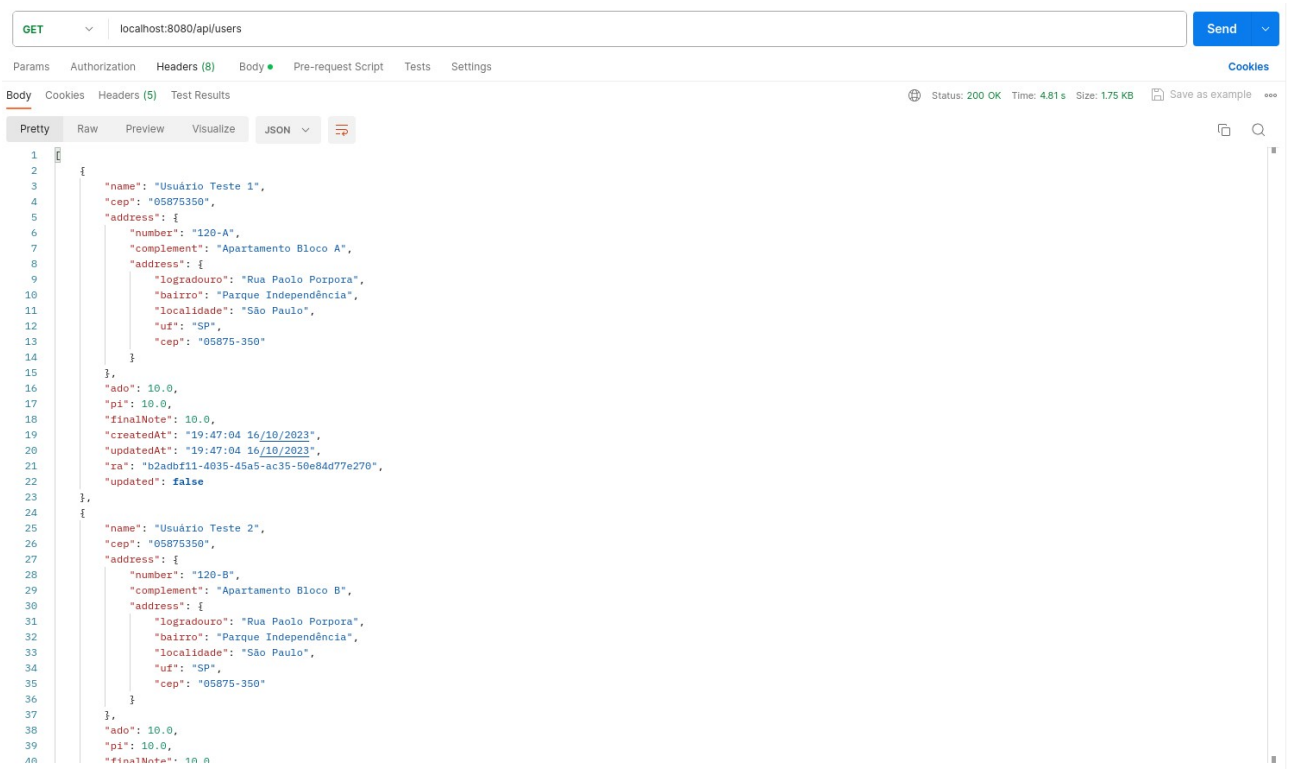
OBS.: Será possível perceber que para esse exercicio não utilizei nenhum serviço de criptografia, afinal, não se fez necessário.

Método GET – http://localhost:{PORT}/api/users

Esse método busca todos os alunos que estão cadastrados dentro da API e retorna uma DTO, onde, dentro dessa DTO é contido informações como:

- O endereço completo;
- O número de RA do aluno;
- O seu nome;
- A sua média das notas (Pi e ADO) somadas e divididas por 2 (quantidade de avaliações);
- Quando o aluno teve o primeiro acesso (cadastro) na plataforma;
- Se o aluno foi alterado em algum momento (deixando salvo somente a ultima alteração).

Abaixo segue uma Screenshoot da saida experada:

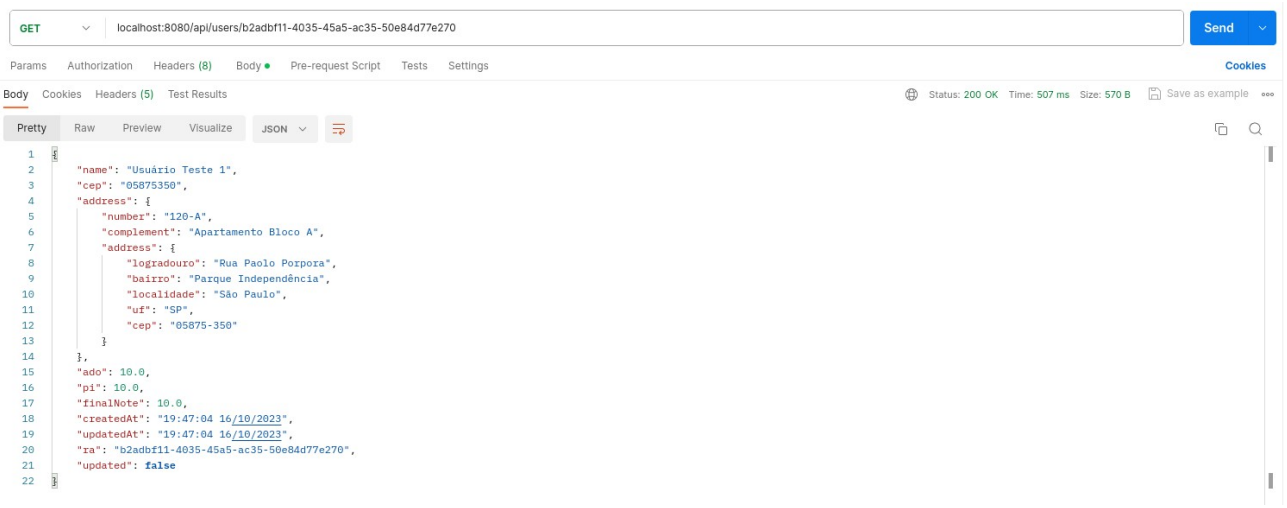


Método GET – http://localhost:{PORT}/api/users/{RA}

Esse método ESPERA encontrar um aluno cadastrado dentro da API e retorna uma DTO, onde, dentro dessa DTO é contido informações como:

- O endereço completo;
- O número de RA do aluno;
- O seu nome;
- A sua média das notas (Pi e ADO) somadas e divididas por 2 (quantidade de avaliações);
- Quando o aluno teve o primeiro acesso (cadastro) na plataforma;
- Se o aluno foi alterado em algum momento (deixando salvo somente a ultima alteração).

Abaixo segue uma Screenshoot da saida experada:



Método POST – http://localhost:{PORT}/api/users

Esse método espera receber, por JSON, os dados de um novo aluno no sistema, para salvar-lo no banco de dados, as informações esperadas, no arquivo JSON são:

```
{
  "name": "Um Nome Qualquer",
  "cep": "99999999",
  "number": "999",
  "complement": "",
  "ado": 10,
  "pi": 10
}
```

sendo que, cada campo deve serguir o seguinte padrão:

1. Nome: Deve sempre começar com uma *string*, maiuscula ou minuscula, e pode ser seguido por espaços em branco, que por sua vez devem ser seguidos por novas *strings* maisculas e minusculas;

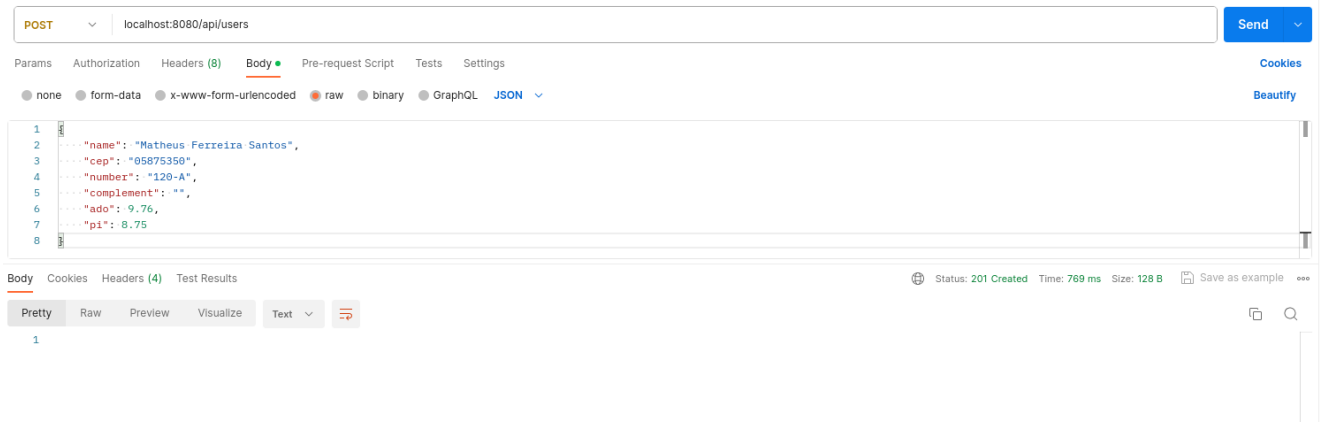
2. CEP: Deve ser informado um CEP existente, caso contrário será IMPOSSIVEL cadastrar um usuário no sistema;

3. Número: Não pode ser nulo, não tem nenhum outro critério de avaliação já que não se pode definir um padrão nesse campo;

4. Complemento: Sem ressalvas;

5. ADO e PI: Deve ser informado valores NUMERICOS de 0 a 10.

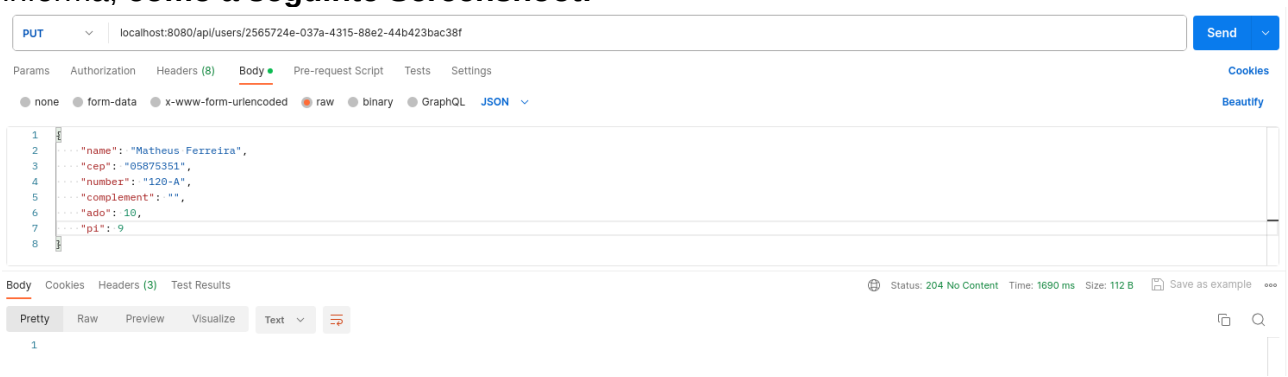
No envio de dados válidos para essa requisição, a api deve retornar somente um HTTP Status de 201 (criado) sem nada em seu corpo, **como a seguinte Screenshoot:**



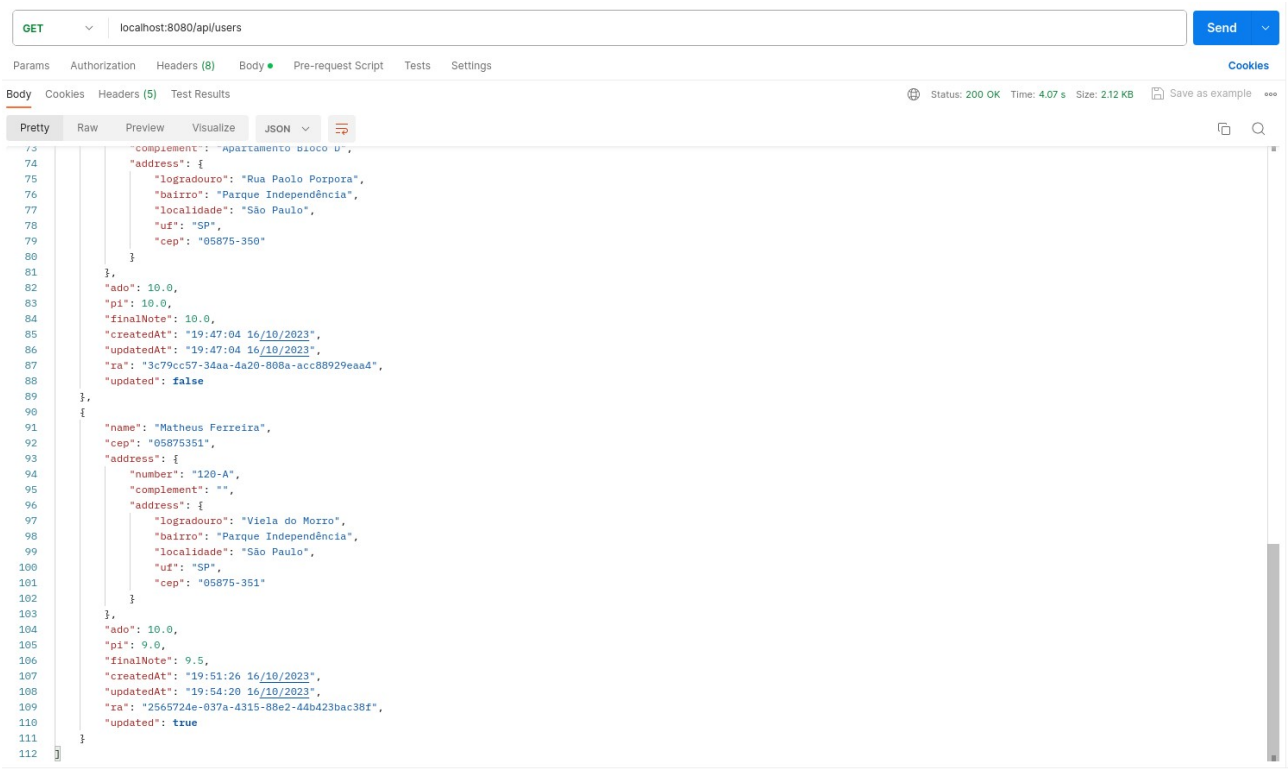
Método PUT – http://localhost:{PORT}/api/users/{ra}

Este método é exatamente igual ao método anterior, o do tipo POST, a única diferença é que ele espera receber um RA como *route params* para saber qual usuário deverá receber os dados que serão informados pelo JSON, que segue exatamente o mesmo padrão do exemplo anterior.

No envio de dados válidos para essa requisição, a api deve retornar somente um HTTP Status de 204 (sem conteúdo) sem nada em seu corpo, como o próprio status code já informa, **como a seguinte Screenshoot:**



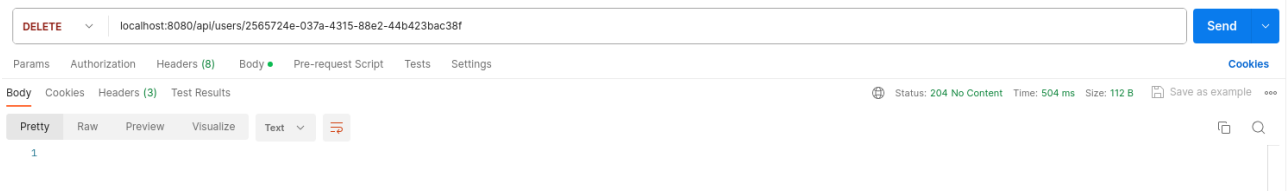
Se formos buscar esse usuário utilizando o primeiro método (o último objeto):



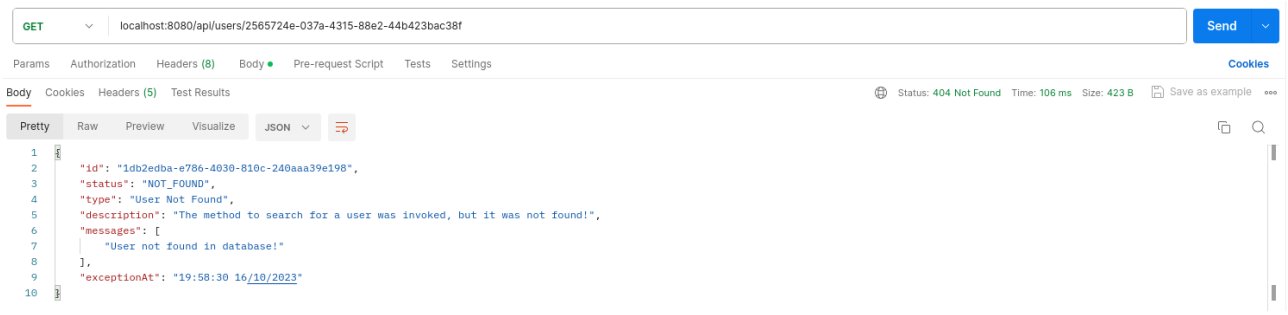
Se formos dentro do método que busca somente um usuário e compararmos as datas de criação e atualização é possível notar que este método faz a alteração da data de atualização e já muda o status do usuário para “atualizado”, **como é possível ver na foto acima**

Método DELETE – http://localhost:{PORT}/api/users/{ra}

Por fim, esse método funciona exatamente como o segundo método, que busca um único usuário, porém, ele espera receber um RA cadastrado para que ele possa ser excluido dentro do sistema, caso o usuário realmente exista, ele será removido e o retorno será um HTTP Status 204, assim como o anterior, **como a seguinte Screenshoot:**



Buscando o usuário criado, que agora está removido:



Rodando a aplicação com Docker Playground:

Para testar a aplicação deve-se entrar no docker playground e realizar o login com alguma conta sua.

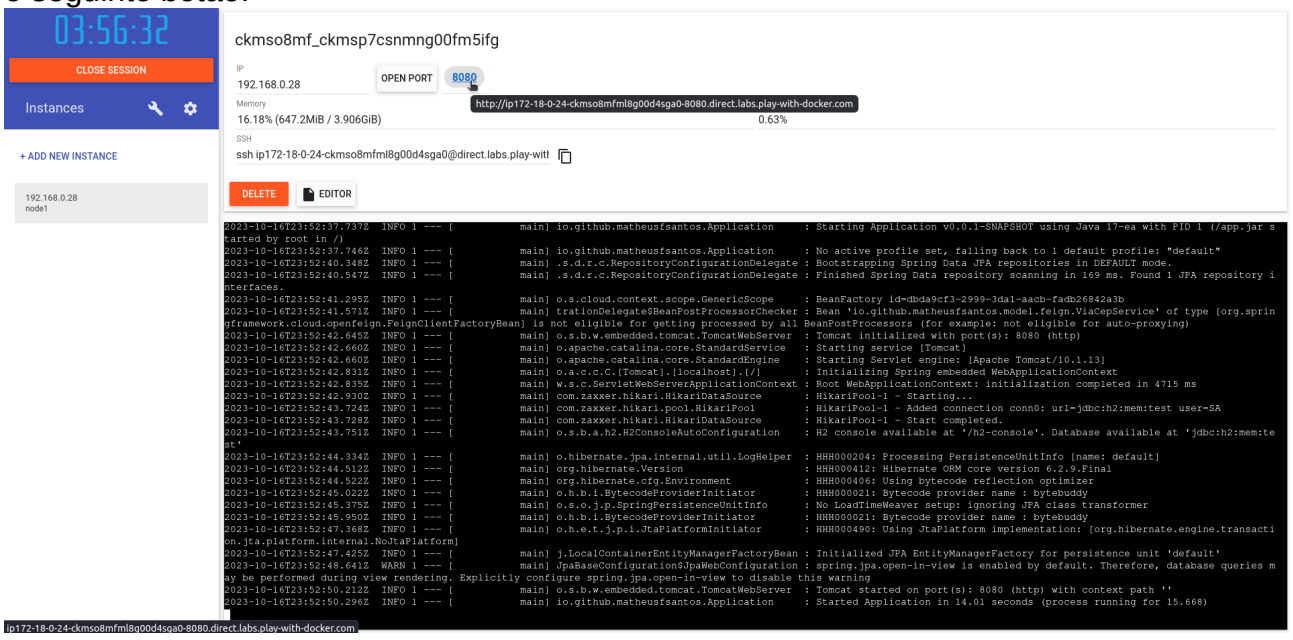
Link do Docker Playground: https://labs.play-with-docker.com/

Quando você realizar o login, irá ser redirecionado para uma página igual a está:

Adicione uma nova instancia e rode os seguintes comandos no terminal que irá ser aberto:

docker pull matheusfsantos/4log-students:1.0.0
docker run -p 8080:8080 matheusfsantos/4log-students:1.0.0

Quando terminar tudo, uma porta 8080 deve ser iniciada em seu computador e aparecerá o seguinte botão:



Clique nele e será redirecionado para uma URL externa, basta adicionar ***/api/users*** e começar a testar a api, *be happy!*