

Logo da Orange Talents

# Validando os dados de entrada de uma requisição

## Cenário:

Com o conhecimento sobre Micronaut e Bean Validation, imagine que precisamos cadastrar um aluno(a) em nosso sistema via uma API REST, onde este aluno(a) deve possuir as seguintes informações com suas respectivas restrições:

* todo(a) aluno(a) precisa de um nome, email, idade;
* o nome não pode ter mais do que 60 caracteres;
* o email além de válido deve ter no máximo 42 caracteres;
* o aluno(a) deve ser maior de idade;

Um detalhe importante é que para fazer as validações devemos usar a Bean Validation, porém o arquiteto do time proibiu o uso das anotações @Validated e @Valid por motivos de segurança (existe um bug critico na versão do Micronaut que é utilizada no projeto). Em caso de erro de validação a API REST deve retornar o status HTTP 400 com as mensagens de erro de validação, caso contrário deve-se retornar o status HTTP 200.

Lembrando que não precisamos gravar os dados em um banco de dados.

Com todas as restrições citadas acima, como você implementaria sua API REST para cadastrar um novo aluno(a)?

## O que seria bom ver nessa resposta?

* **Peso 5**: Criação de um controller com um endpoint do tipo POST recebendo uma request com os dados do aluno(a), com o DTO de request anotado com as anotações da Bean Validation, e por fim retornando um status HTTP 200 em caso de sucesso ou status HTTP 400 em caso de erro de validação;
* **Peso 4**: Injeção do bean javax.validation.Validator (ou Validator do Micronaut), invocação do método validator.validate() passando o DTO de request e lançando uma ConstraintViolationException passando as violations retornadas pelo validator;
* **Peso 1**: Definição dos atributos do DTO como nullable (tipos declarados com indicador ? do Kotlin);

## Resposta do Especialista:

* Crio uma nova classe de controller, anoto ela com @Controller, e crio um método adicionar() anotado com @Post para receber os dados da requisição na URI "/api/alunos". Para o payload JSON, eu crio um DTO NovoAlunoRequest que recebe **nome** e **email**, ambos como String, e um atributo **idade** do tipo Int. Todos estes atributos do DTO são nullable (uso o indicador ? nos tipos) para que as validações de obrigatoriedade consigam ser executadas corretamente. Por se tratar de um DTO, eu o crio como data class do Kotlin;
* Ainda na classe NovoAlunoRequest, eu a anoto com @Introspected do Micronaut por causa do AOT (Ahead of Time), e também anoto os atributos da classe com as anotações da Bean Validation:
  + atributo **nome** com @field:NotBlank @field:Size(max = 60);
  + atributo **email** com @field:NotBlank @field:Size(max = 42) @field:Email;
  + atributo **idade** com @field:NotNull @field:Min(18);
* No controller, faço a injeção do validator da Bean Validation (javax.validation.Validator) gerenciado pelo Micronaut, e no método adicionar() eu invoco o método validator.validate() passando a request como parâmetro. Verifico se o retorno do método possui alguma violação para então lançar uma ConstraintViolationException passando as violações como parâmetro. O fato de lançar essa exception é suficiente pois o Micronaut já possui uma exception handler para tratá-la adequadamente: status HTTP 400 juntamente com a lista de erros no corpo da resposta;
* Por fim, caso não exista violações, eu faço o método adicionar() retornar HttpResponse.ok();