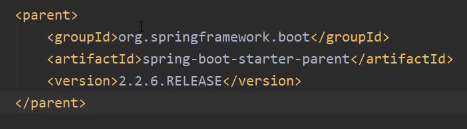
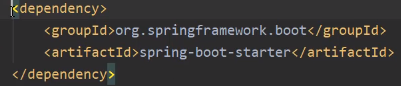
Posso criar coom o passo a passo d inicio ou fazer por sute

Start.sprint.io

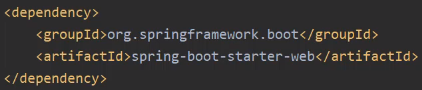
Para transformar minha aplicação em spring



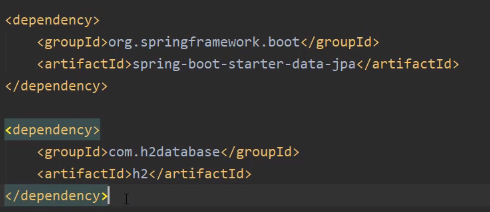
Ela tem os container do spring, anotation, etc



Para fazer nosso REst API



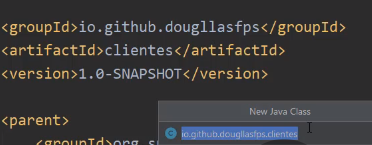
JPA pra conexão com o banco de dados.





Criação da classe que vai iniciar a minha aplicação

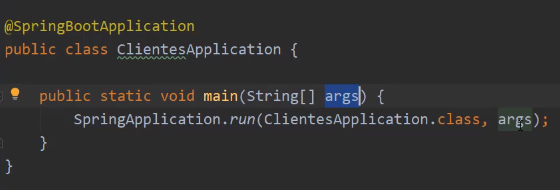
Pra iniciar o pacote estou usando o mesmo que esta em groupId



Padrão do spring.



Final:



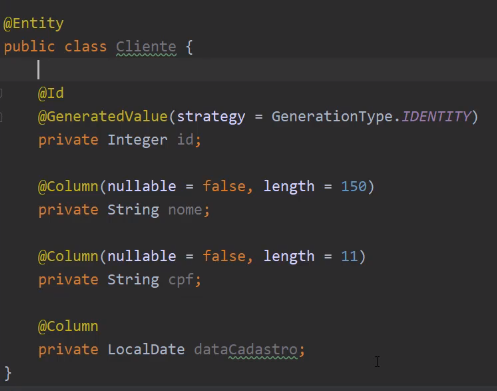
18. Mapeando a Entidade Cliente

Criar uma entidade entity

Para deixar alguma variável obrigatória + tamanho do nome em caracteres.

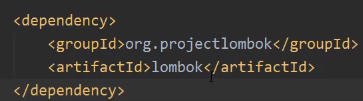


Final:



19. Adicionando o Lombok e configurando o Plugin

Lombok, faz a criação de código, construtores, geters e seters. Hashs



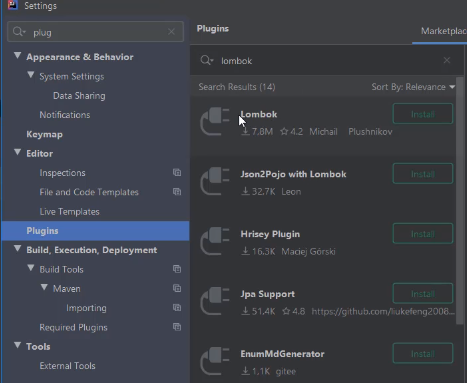
Agora posso colocar essa anotação:



Ela faz com que em tempo de compilação, o projeto seja gerado e essa classe seja gerada os gets e sets automaticamente,

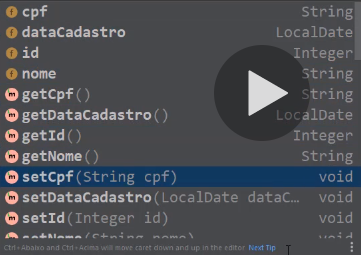
Instalando um plugin pra ajudar.

Settings/plugins



Reiniciar a IDE.

Agora tenho acesso a todos os gets e sets.

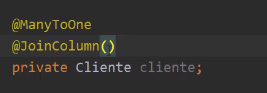


Essa anotção, tamto cria os getes e seters como também cria o toString, hashCode, cria um construtor sem parâmetros, outro com os construtores obrigatoris.



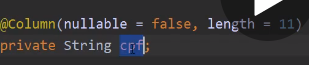
20. Mapeando a Entidade Servico Prestado

Quando eu faço isso:



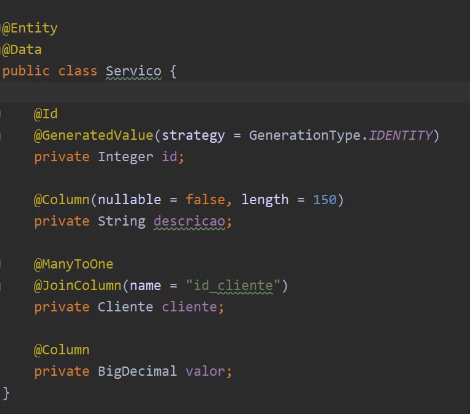
O JoinClumn define uma chave estrangeria que a tabela de serviço vai ter a chave estrangeira para a tabela de clientes.

Quando eu faço assim:



O nome da coluna no banco de dados vai ser igual ao que esta na propriedade.

Final:

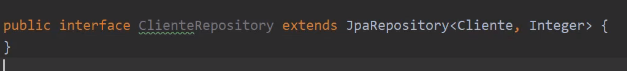


21. Criando os Repositorios JPA

Para criar a classe e o pacote juntos, seria como coriar a classe mais na hora de colocar o nome dele=a:

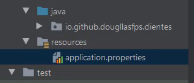
model.repository.ClienteRepository

repositório:



22. Configurando e obtendo conexões com base de dados

Criando o arquivo application.properties.



Esse é o dialeto do banco:

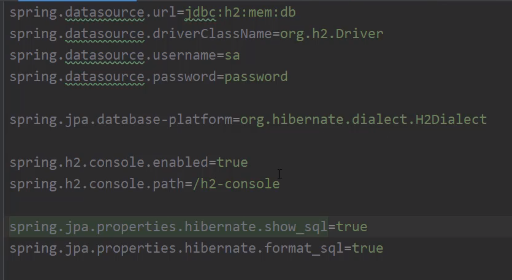


Dependendo do banco ele muda.

Para eu poder acessar ele pelo navegador, e definir qual a url pra eu poder usar pelo console.

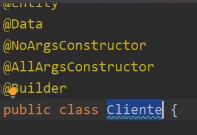


Para ele mostrar o sql gra pelo hibernet no console. Formatar o sqlS



23. Testando a conexão com o banco de dados

Todo comando com @Bean e linerunner, vais ser inicializado assim que eu rodar a aplicação.



Essa anotação serve pra ele formar um construtor sem argumentos.



Gera um construtor com todos os argumentos:

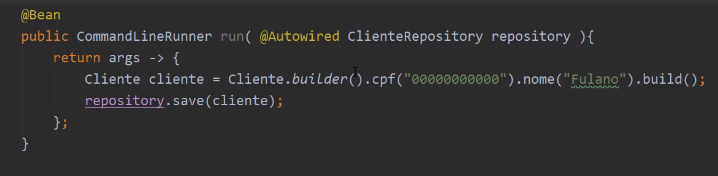


Gera um builder dentro de clientes que permite que eu crie clientes dessa forma:



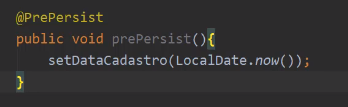


Dentro da classe principal:



O que essa anotação faz?

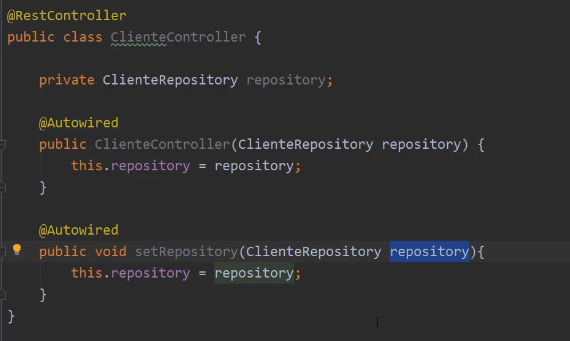
Quando minha entidade Cliente for persistir as informações no banco, antes disso ela é iniciada e pega a data e hora no momento



24. Iniciando a API de Clientes

Criação dos controllers.

Posso fazer uma injeção de @Autowired de 3 formas:



Utilizando uma abordagem REstFull, quando eu criar algo no servidor através do POSTMapping, vou retorna a query de estatus igual a criated, se não colegar ele retorna 200.



Ou 

Método para salvar um cliente:



25. Cadastro de Cliente através do Postman

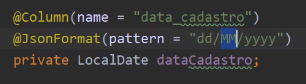
Consertando o erro de cima, dessa forma meu método post vai saber que esse cliente que esta vindo, esta vindo pela requisição. Colocando o @requestBody eu digo que esse objeto esta vindo pelo Json da requisição.



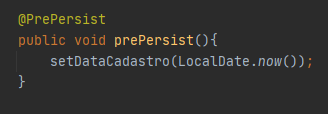
Com o crated ele retorn no postman assim:



Formatando a data para o jeito certo:



Lembrando que, sempre que eu salvar um contato ele inicia o prepersist e pega a hora e slava automaticamente pra mim.



26. Obtendo as informações de um Cliente

A ideia é achar um cliente através de um id.

Quando um objeto retorna um tipo OPTIONAL, quer disser que ele pode retornar um objeto ou não.

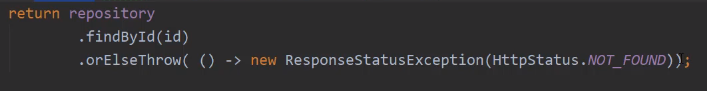
Quando escrevo isso:



Quero disser que, ele foi no repositório achou meu cliente com o id que passei e eu quero que ele retorne esse cliente que ele achou.

IMPORTANTE: uqnado retorna um erro 404 quer disser que entreie em uma pagina que não existe,

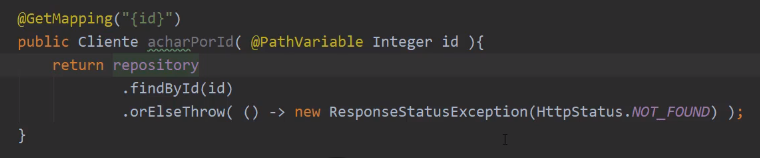
Nesse trecho de código, peço que ele retorne um cliente que pedi pra ele ir buscar no banco pelo ID, caso ele não encontre quero que ele retorno uma exceção de not found.



Esse pacote serve para retornar um erro de estatus:



Final:



Pronto agora só testar.

27. Deletando um Cliente

O cliente que esta dentro de map, é o cliente que achei dentro do banco e dados:





Quando eu faço um mao, ele me retorna um optional e eu posso escolher fazer alguma coisa com o objeto que ele esta retoenando no fidById,

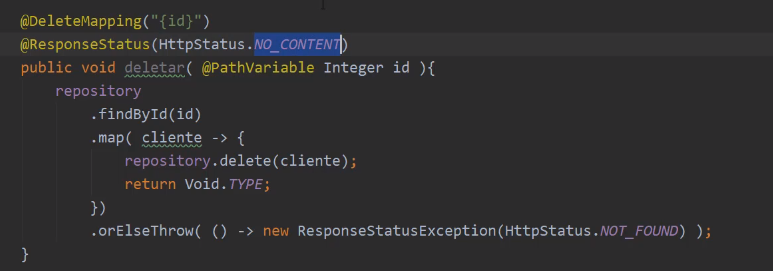
O map serve para fazer o mapeamento de um objeto pra alguma coisa, nesse meu caso eu preciso deletar o cliente.

Eu não posso retornar nulo se não ele entra detro da linha de baoxo no orelsethrow, mas também não posso retornar o cliente pq não vou fazer nada com ele. Pra não retornar vazio::



O no\_content indica que não tem nenhum recurso de retirno.

Final:

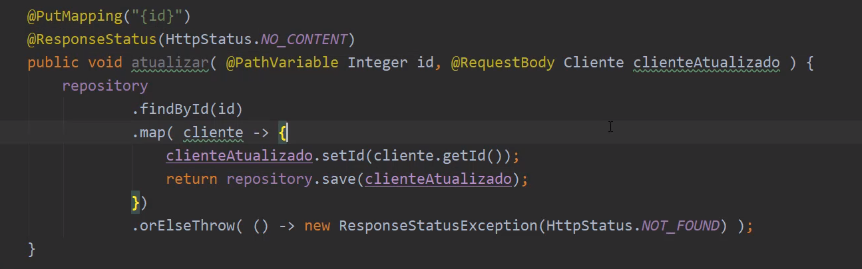


28. Atualizando um cliente

O cliente que esta dentro de map, é o cliente que achei dentro do banco e dados:



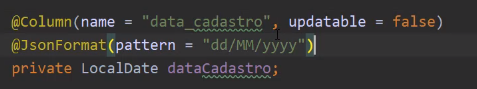
Final:



Testando:

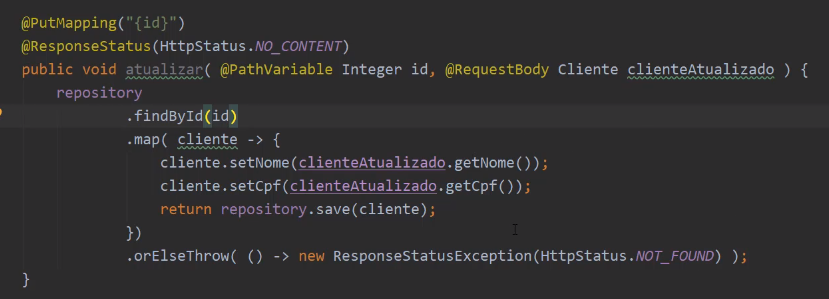
Descobri que a data reorna como null, como resolver isso?

Posso ir na classe cliente e fazer dessa forma:



Com o updatable assim ele nunca ira atualizar essa coluna não importa o valor que coloquei

Segunda forma:



29. Aplicando regras de validação no cliente

Preciso fazer validação no meu código, antes da requisição acontecer com sucesso temos que mandar um nome valido, preenchido e pro cpf a esma coisa.

Pra isso vou usar um recurso do JAVA que é o Bean Validation, que é uma expecificação do java eÈ, então vou usar ele pra fazer a validação dos meus objetos.

Valida se é nulo ou vazia:



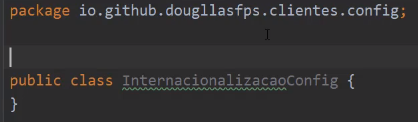
Esse é um validador customizado para o CPF brasileiro. Ele é do hibernate



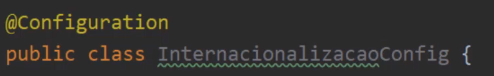
Podemos colocar uma mensagem de validação, que se chama internacionalização.

30. Configurando Internacionalização

Fazer uma nova classe no diretório raiz.



Como ela vai ser uma classe de configuração coloco a anotação



Vou colocar oo @Bean pra quando a classe de configuração for escaneada, ela vai registrar esse bean no contexto, na injeção de dependência.

Minhas mensagens vou colocar em um arquivo resouces:



Dentro dela coloco só isso:



Esse string valor vai ser a mensagem.

Esse é o nome da pasta, como ela esta na raiz posso colocar só assim:



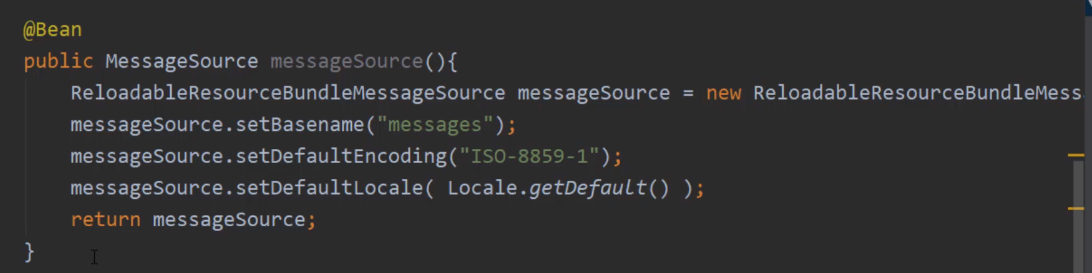
Esse é para reconhecer os caracteres brasileiros:



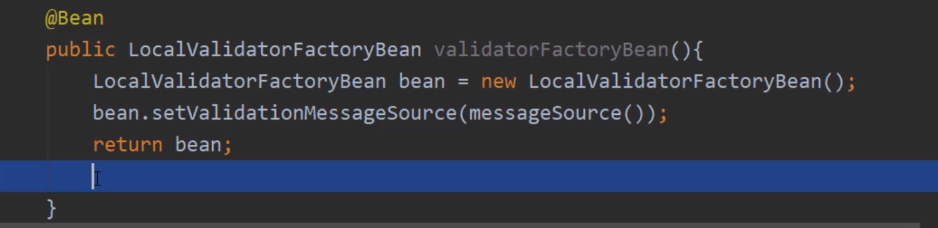
Ele vai reconhecer que meu sistema operacional é do brasil, e coloca aqui:



Final primeiro bean:



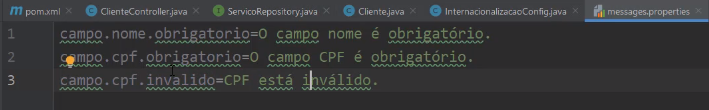
Esse vai fazer a validação dentro do bean validator, e carrega meu primerio método:

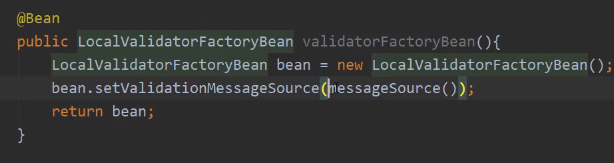


Dessa forma posso criar as mensagens dentro de mensagem.properties e intercalar entre minhas entidades.

31. Concluindo a validação de clientes

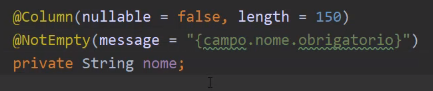
Dentor de mensagem.properties colocao as mensagens que quero retornar e copio as chaves.



Graças a esse cara:  


Que consigo fazer a leitura do arquivo e com isso ele sera capaz de fazer a leitura da chave e interpolar com a classe cliente, onde eu coloco a chave.

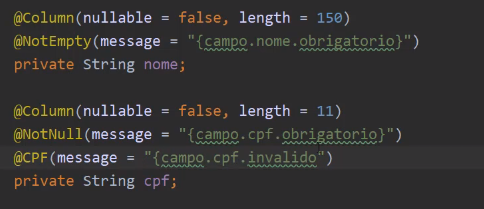
E uso na classe cliente:



Posso criar vários arquivos de mensagens cada um em uma linguagem e dependendo do locale:



Ele vai usar a linguagem, o aq=ruqivo correto.

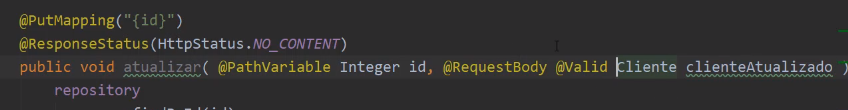


Testando:

Deu erros e para consertar isso tenho que valor pro Spring validar na hora que for fazer a requisição.

Coloquei no post e no atualizar.





E também dizer que ele vai pegar da raiz.  

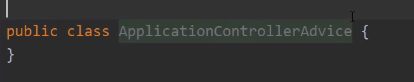

Agora quando eu for cria com algum dos dadis nulos da um dos erros.



32. Tratando erros da API

Vou criar uma classe que sera capaz de interceptar um erro e pode modificar o retorno da minha requisição.

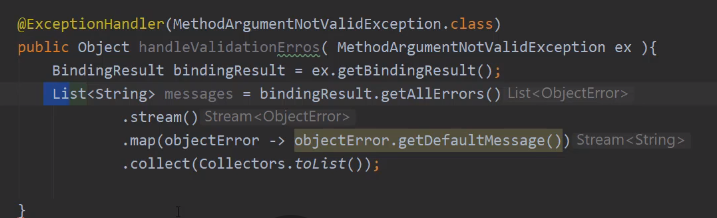
Junto com a classe de controller:



Quando uma exceção com erro de validação é lançada o nome dela é



Final:



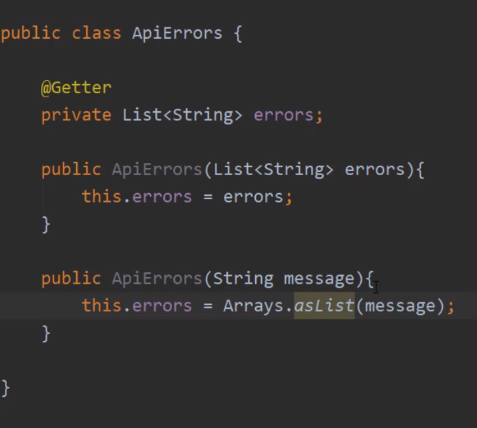
Continuando na próxima aulda.

33. Padronizando o retorno de erros da API

Dentro de controller, criei uma pasta excepion, e dentro uma classe ApiErrors

Dentro dela vou retornar um arrays de strings com meu erros.

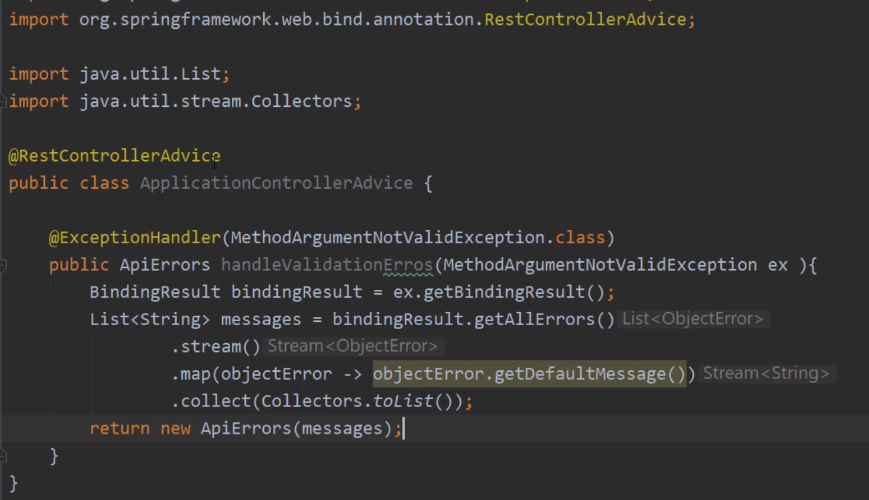
Final:



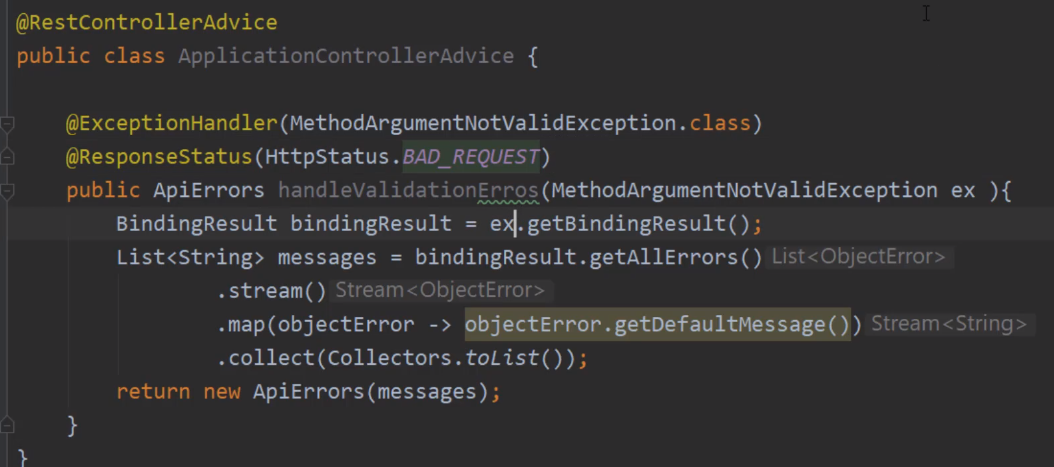
Voltando para minha classe,



Final:



Atualizando:



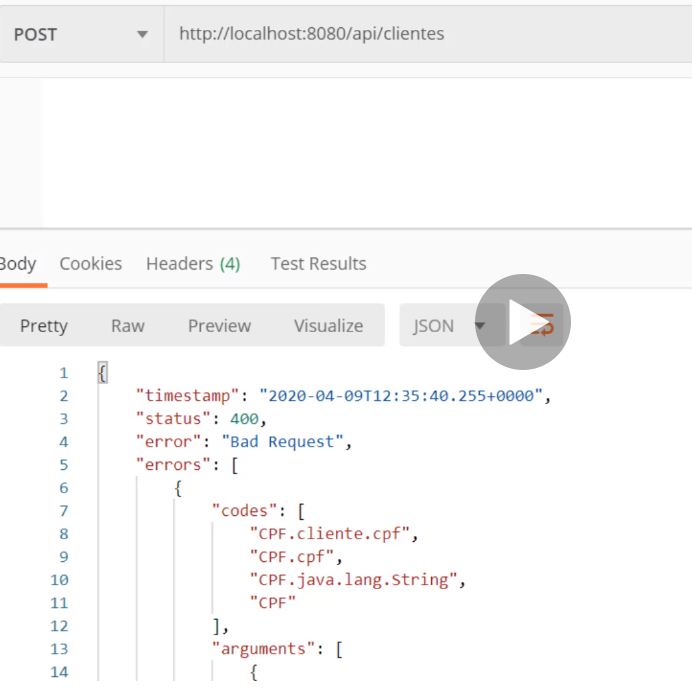
Esse cara contem todos mensagens de erros:



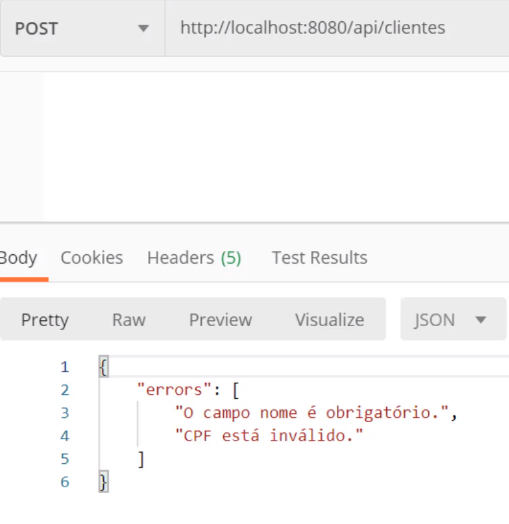
A a ideia principal é retornar as mensagens de erros de forma padronizada.

Testando:

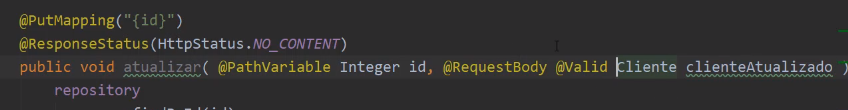
Erro da forma que recebia antigamente:



Erro que recebo agora:



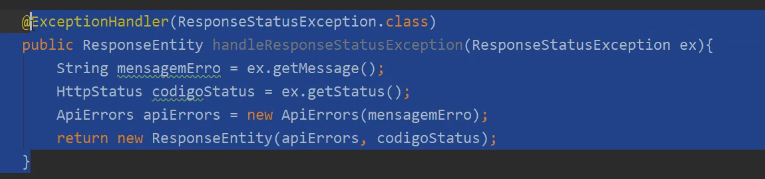
Colocar o valid no update:



34. Exception Handler para ResponseStatusException

Denr=tro de applicationcontrolerAdvaice





Com isso consigo fazer retirnis dinâmicos.

Na classe de controller, onde eu coloquei isso:



Colocar uma frase de erro pra quando o id não for encintrado.