

Douglas Santos Silva

Documentação do projeto finalizado em Spring Boot e java para controle de patrimônios da empresa, com recursos de segurança e melhores práticas de programação com o objetivo de aprendizagem.

E-mail: dssilva21@latam.stefanini.com

**São Paulo
2023**

Introdução

Esta documentação tem como objetivo apresentar o projeto desenvolvido em Spring Boot e Java para o controle de patrimônios da empresa. O sistema foi criado para gerenciar e controlar os bens da empresa, desde a sua aquisição até a baixa do patrimônio. Através do sistema é possível registrar informações como a descrição, valor e data de aquisição

O sistema possui recursos que garantem a segurança das informações. O projeto foi desenvolvido seguindo as melhores práticas de programação, com foco em aprendizado e evolução na empresa.

Nesta documentação serão apresentados os requisitos, a arquitetura do sistema, as tecnologias utilizadas, as funcionalidades implementadas, as instruções para a instalação e execução do sistema, além de informações visuais como modelagem e técnicas utilizadas para chegada final do projeto.

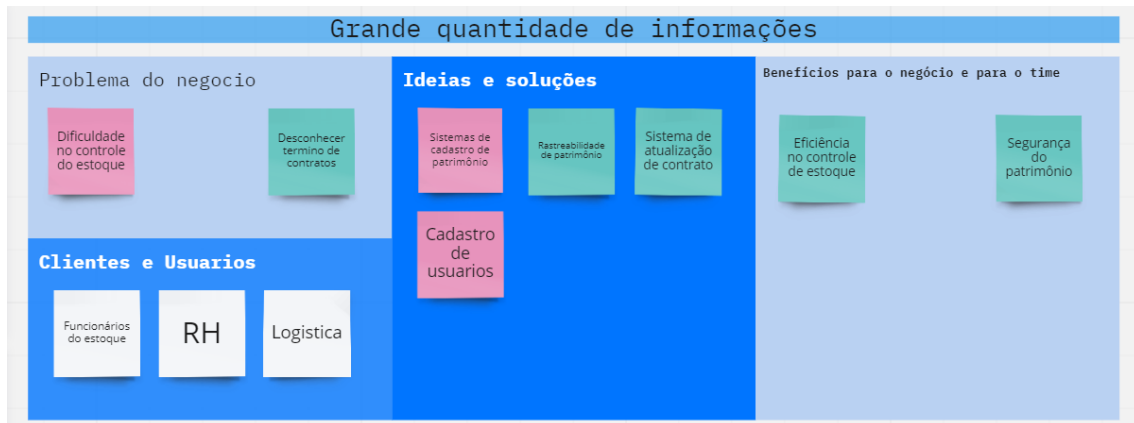
Sumário

1. Ferramentas Utilizadas.....	4
2. Instalação	5
3. Cadastrar pelo insomnia	8

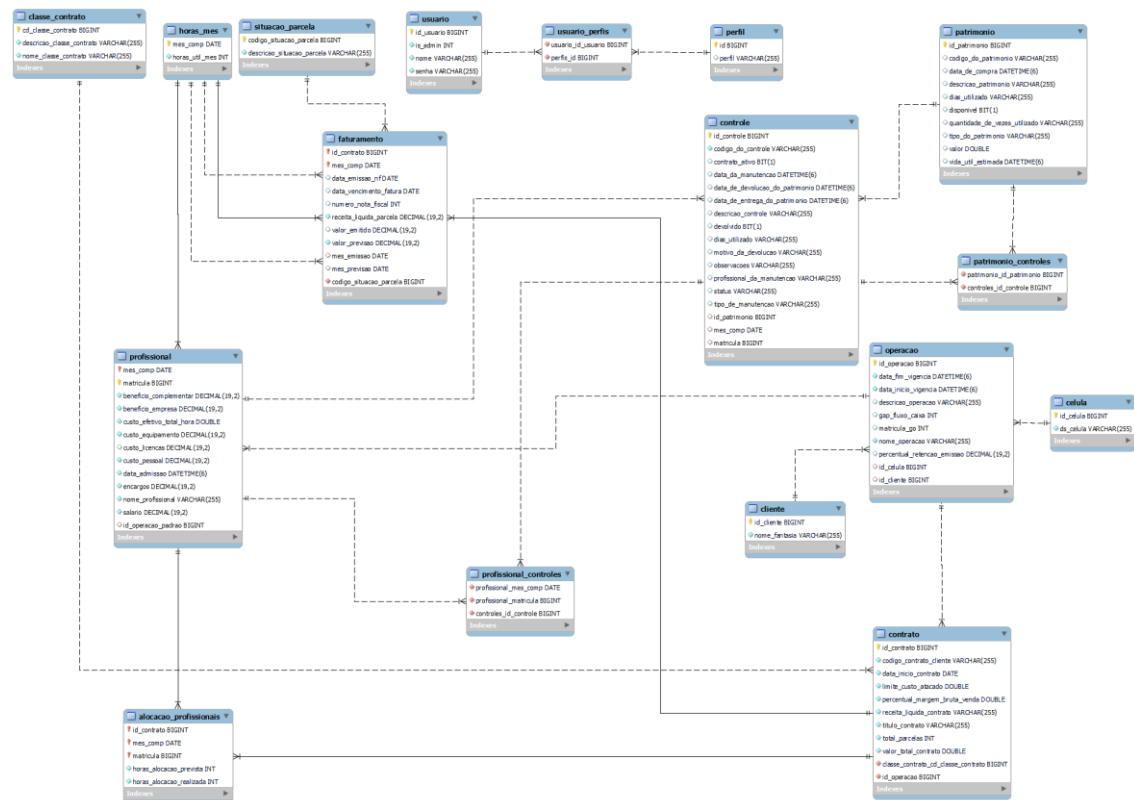
Ferramentas utilizadas:

Learn ux canvas:

Utilizado o método de Learn Ux Canvas (alterado) para conseguir descobrir as principais funcionalidades do sistema



Modelagem do projeto:



Instalação

O Arquivo final se encontra no GitHub na pasta “stfinancial-backend-master”

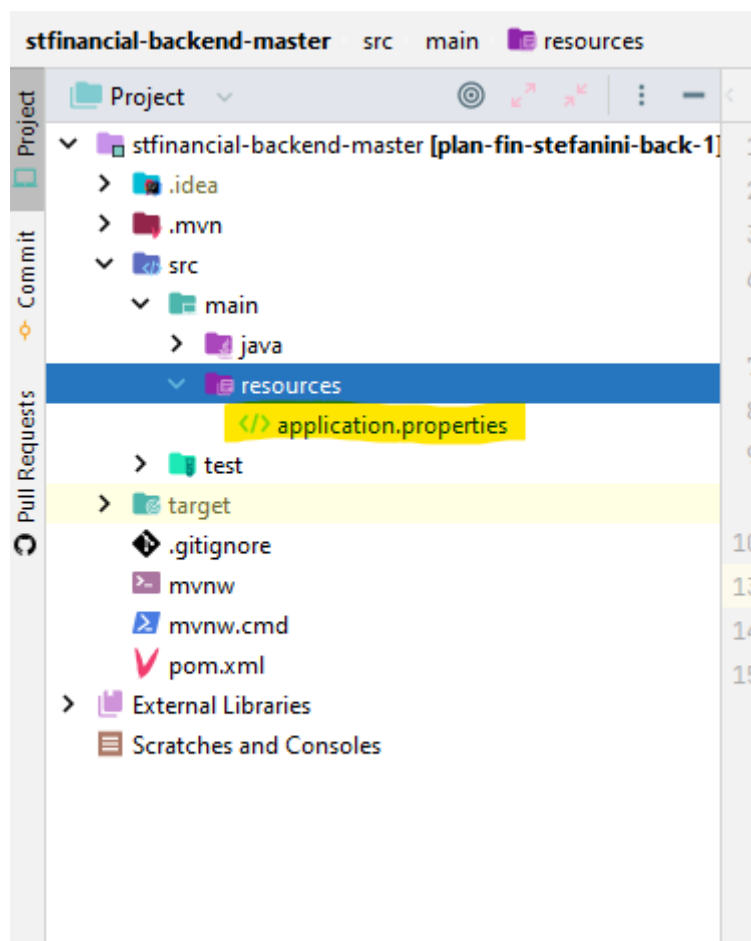
Link para o Git Clone: <https://github.com/Shadowbmo/Backend.git>

Após dar o git clone você terá acesso ao projeto, recomendado utilizar a IDE IntelliJ e INSOMNIA.

1 – o primeiro passo de muitos deve acessar a “application.properties” na qual se encontra na pasta “resources” que está dentro da “main” que está dentro de “src”

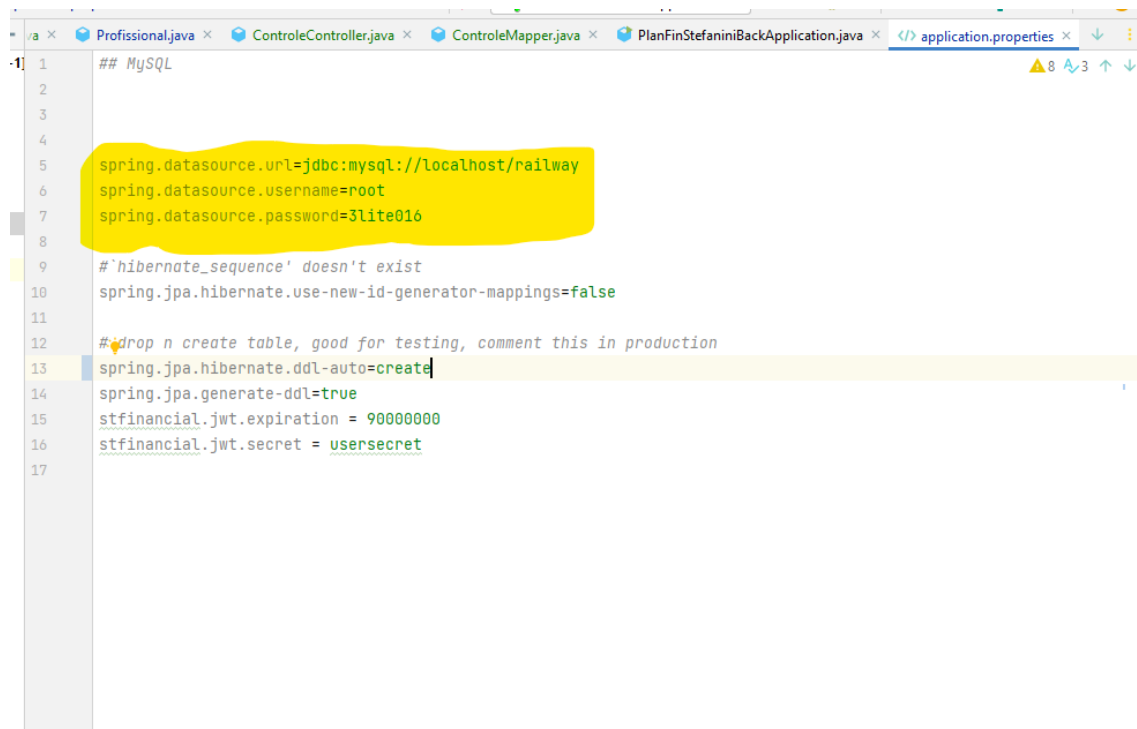
Seguindo o seguinte caminho:

src > main > resources > application.properties



2 – Ao acessar o arquivo deve configurar de acordo com o banco que sera utilizado e com suas credenciais correta, como por exemplo USERNAME , URL e senha

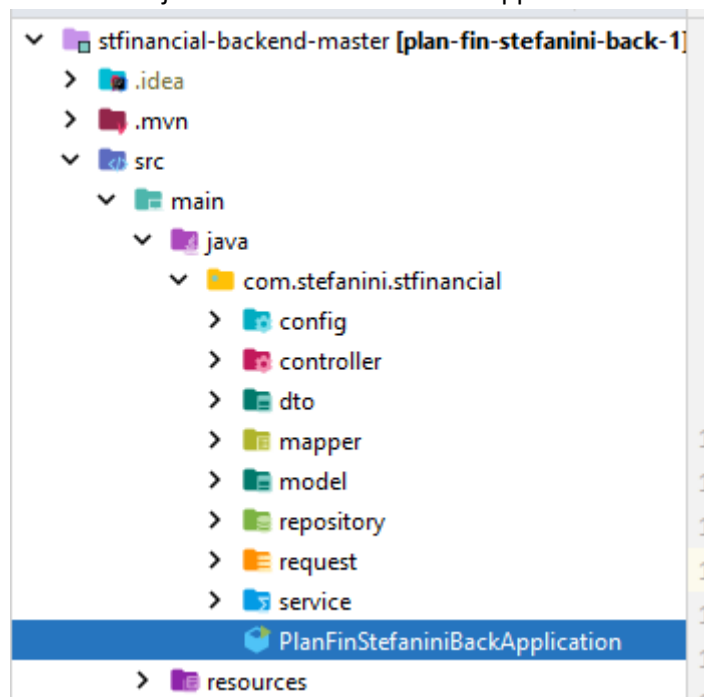
Presentes na linha 5,6 e 7 na imagem a seguir:



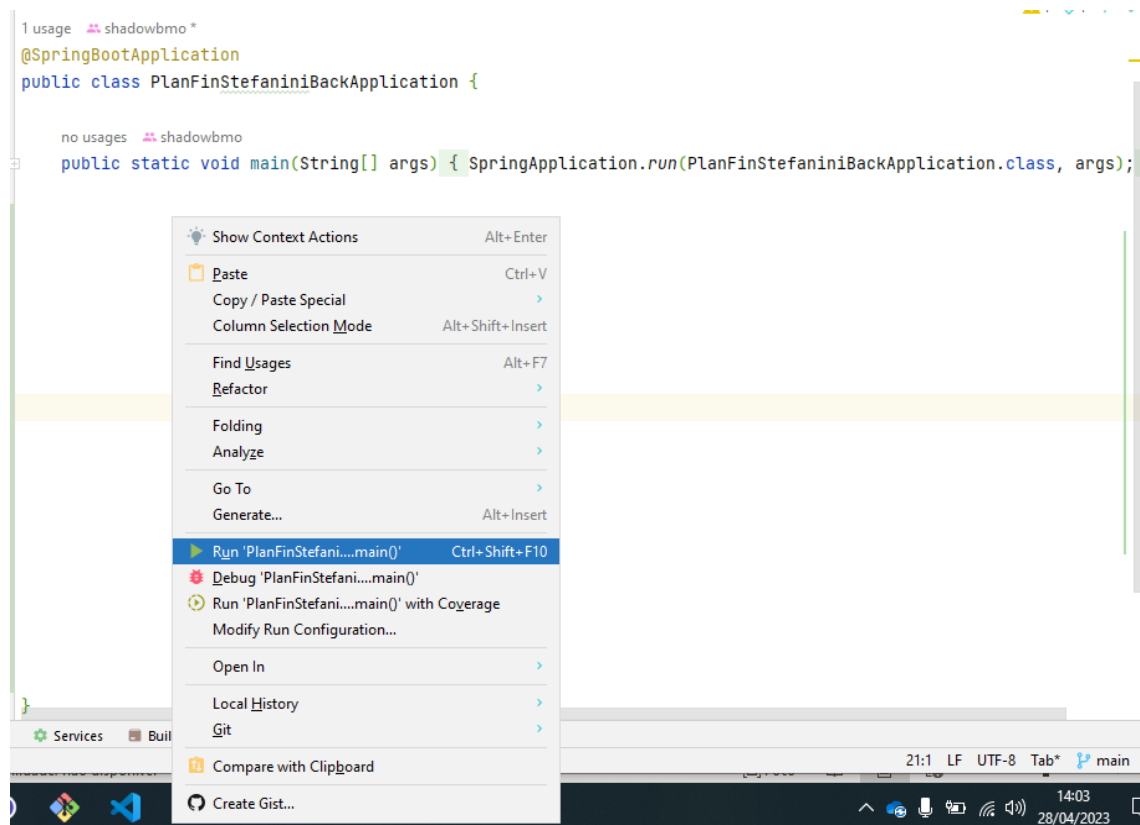
```
1  ## MySQL
2
3
4
5  spring.datasource.url=jdbc:mysql://localhost/railway
6  spring.datasource.username=root
7  spring.datasource.password=3lite016
8
9  #'hibernate_sequence' doesn't exist
10 spring.jpa.hibernate.use-new-id-generator-mappings=false
11
12 #drop n create table, good for testing, comment this in production
13 spring.jpa.hibernate.ddl-auto=create
14 spring.jpa.generate-ddl=true
15 stfinancial.jwt.expiration = 90000000
16 stfinancial.jwt.secret = usersecret
17
```

3 – Com essas alterações feitas devemos iniciar o projeto em sua main que apresenta no seguinte caminho

Src > main > java > PlanFinStefaniniBackApplication



4 – Ao abrir devemos dar o “run” no arquivo para iniciar o projeto



Cadastrar pelo Insomnia

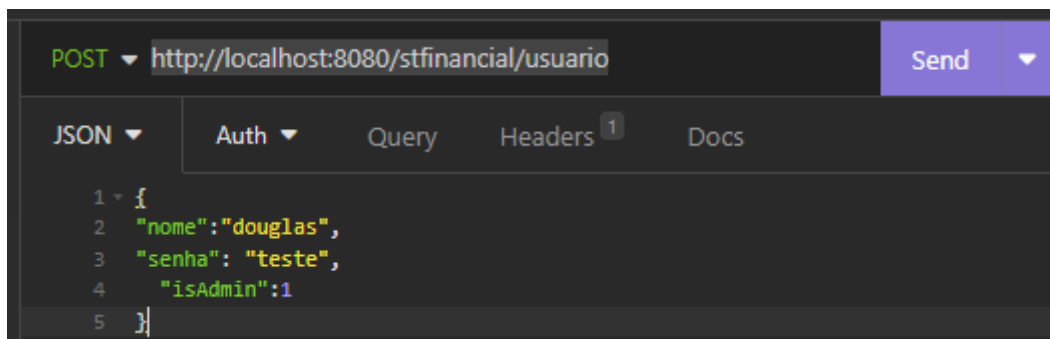
1 – Utilizando o json a seguir com o método POST e com a url informada devemos cadastrar um usuário para ter acesso as demais funcionalidades. (nota-se que a senha vai criptografada ao banco)

```
{"nome": "Qualquer Nome",
```

```
"senha": "Qualquer Senha",
```

```
"isAdmin": 1}
```

URL utilizada: <http://localhost:8080/stfinancial/usuario>

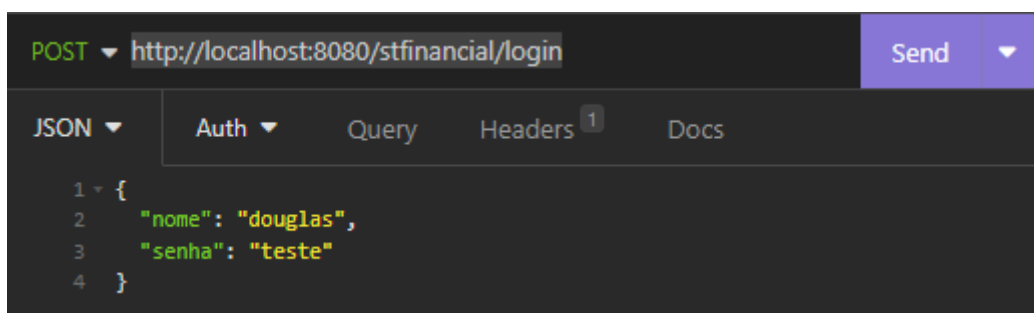


2 – ao realizar o cadastro devemos passar as mesma informações no próximo json e próxima URL utilizando o método POST novamente

```
{"nome": "nome Informado Acima",
```

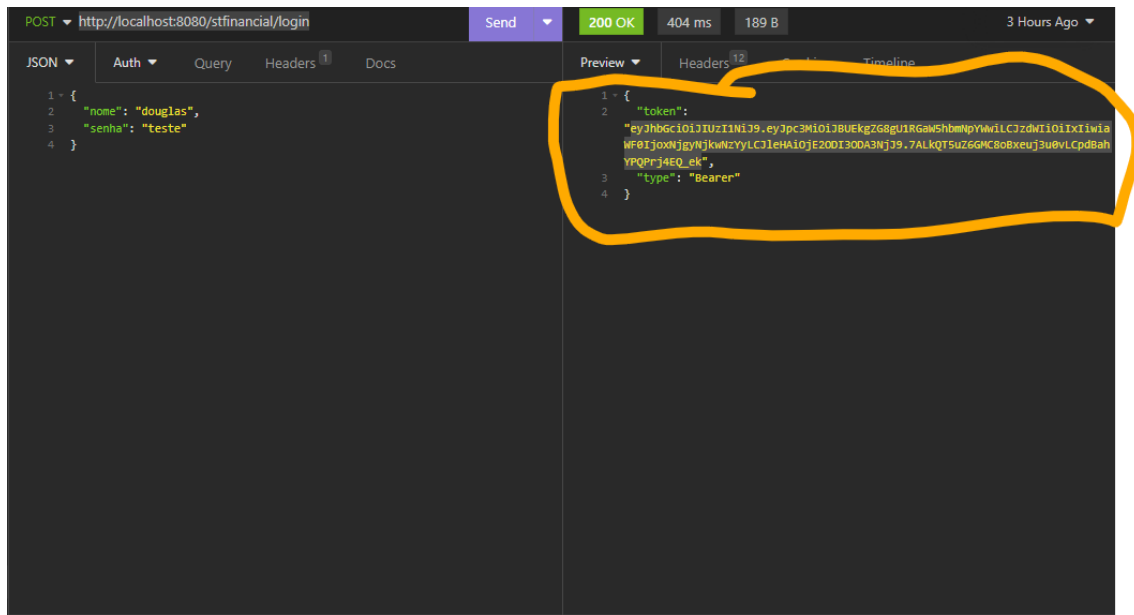
```
"senha": "senha Informada Acima"}
```

URL utilizada: <http://localhost:8080/stfinancial/login>



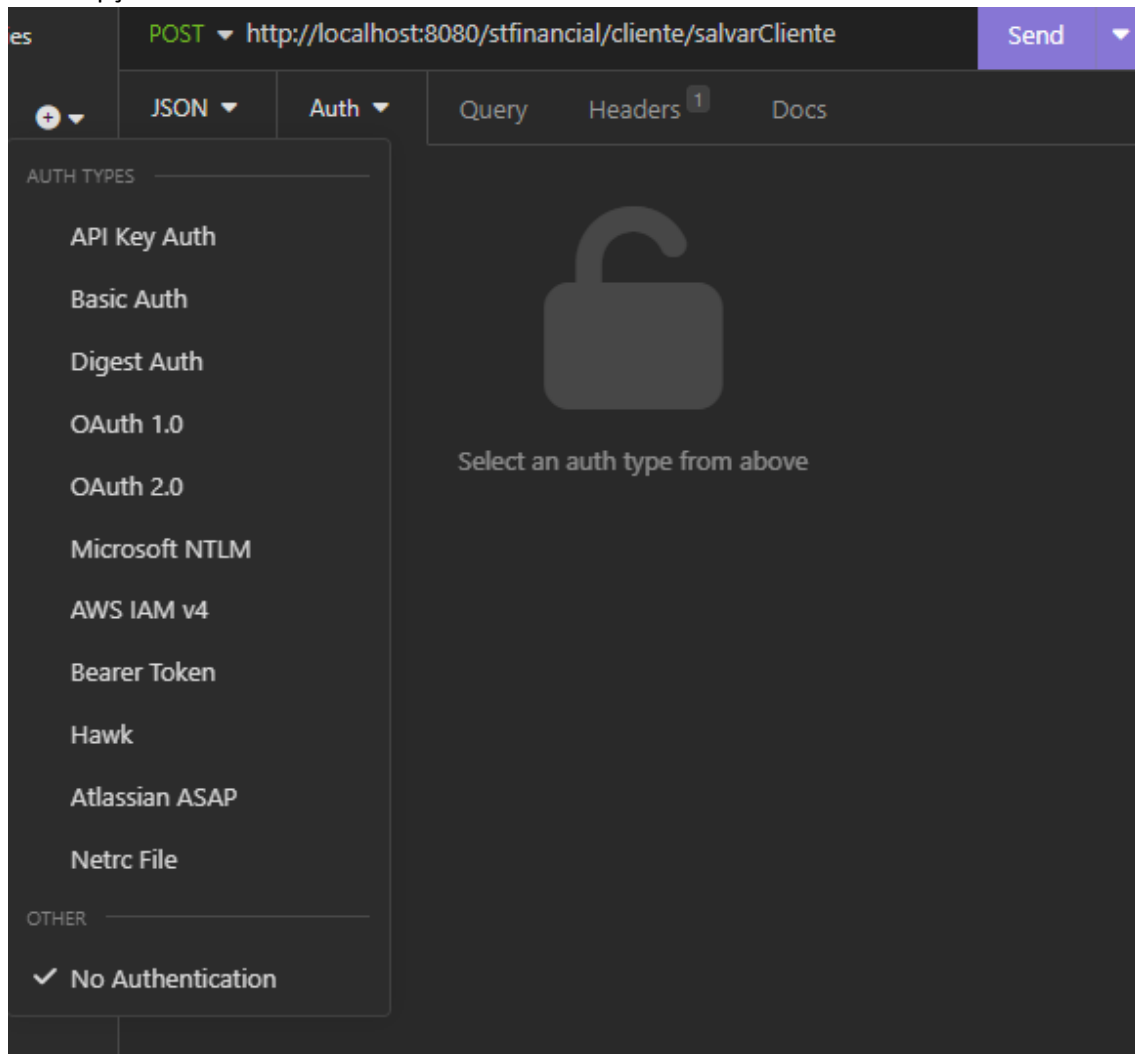
3 – ao realizar o cadastro devemos pegar o token que foi passado (somente o que a dentro das aspas)

Exemplo: se o token vim “abc” você só deve pegar o: abc

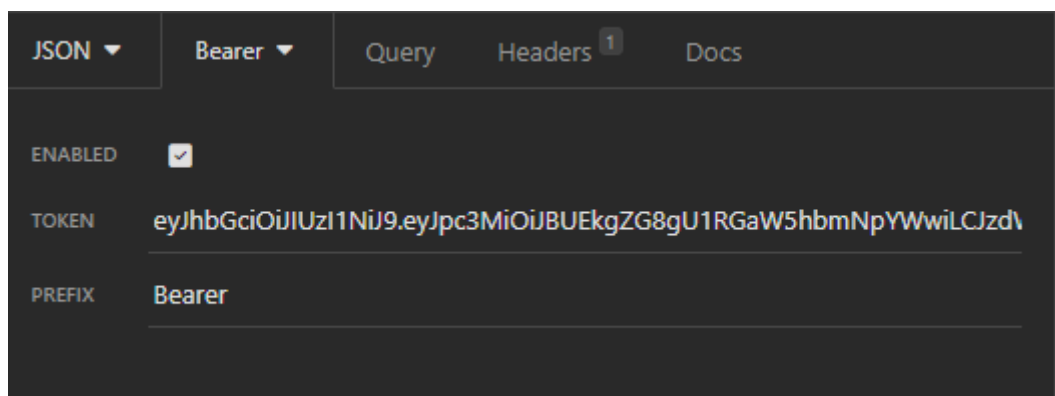


4 – Ao clicar em “Auth” no header do insomnia deve abrir esse menu de opções.

Vá na opção “Bearer Token”



5 – Deve ser inserido o Token que foi copiado no passo “3” no campo de “Token” e no “prefix” deve colocar “Bearer” com o B maiúsculo para conseguir ter acesso ao cadastro



10 – Agora somente seguir com o cadastro das seguintes classes para que comece a parte do back-end da monitoração de patrimônio (LEMBRANDO DE COLOCAR O TOKEN SE NÃO APITA O ERRO 403)

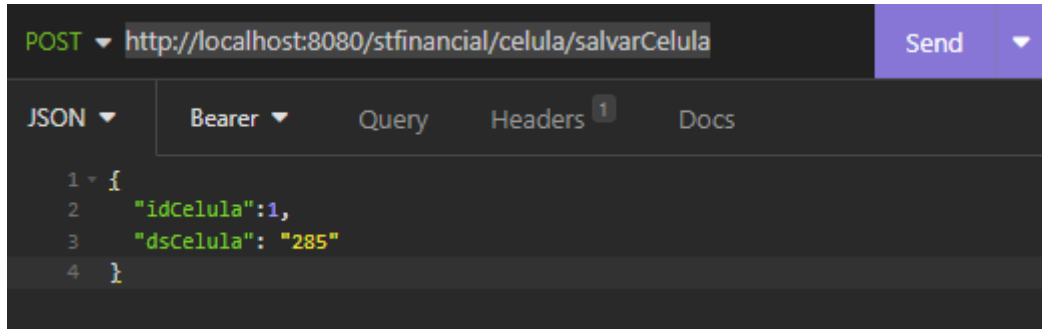
- Celula:

Deve ser utilizado a URL: <http://localhost:8080/stfinancial/celula/salvarCelula>

o Json que deve ser colocado no insomnia:

```
{"idCelula":1,
```

```
"dsCelula": "285"}
```



- Classe Contrato:

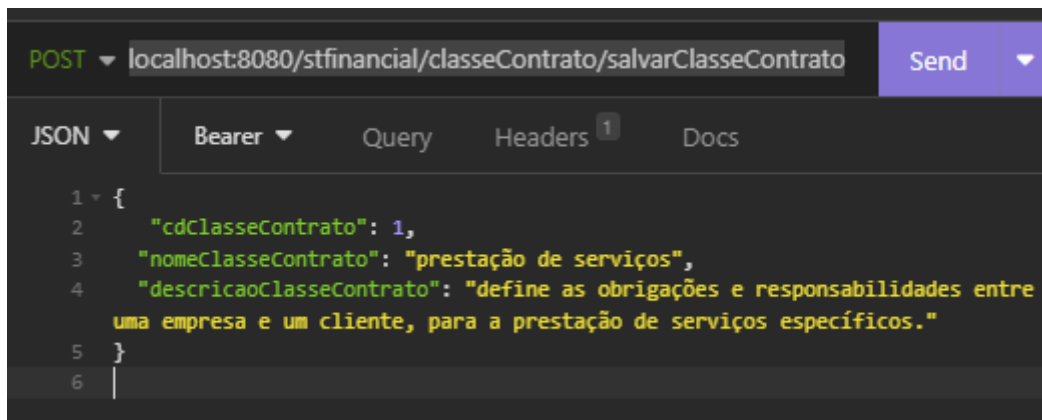
Deve ser utilizado a URL: <http://localhost:8080/stfinancial/classeContrato/salvarClasseContrato>

o Json que deve ser colocado no insomnia

```
{"cdClasseContrato": 1,
```

```
"nomeClasseContrato": "prestação de serviços",
```

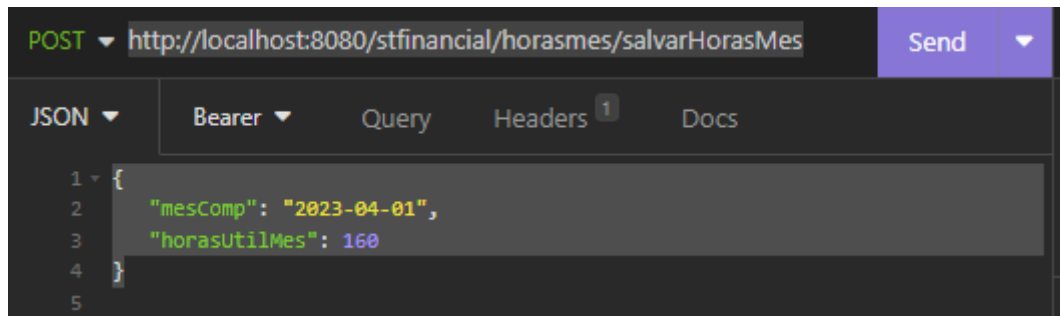
```
"descricaoClasseContrato": "define as obrigações e responsabilidades entre uma empresa e um cliente, para a prestação de serviços específicos."}
```



- Classe Mês e horas

Deve ser utilizado a URL: <http://localhost:8080/stfinancal/horasmes/salvarHorasMes>
o Json que deve ser colocado no insomnia:

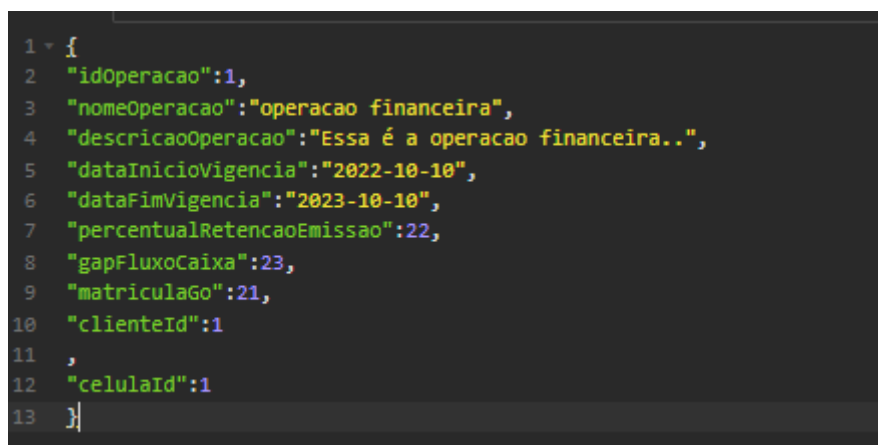
```
{ "mesComp": "2023-04-01",  
  "horasUtilMes": 160 }
```



- Classe Operação:

Deve ser utilizado a URL: <http://localhost:8080/stfinancal/operacao/salvarOperacao>

```
{ "idOperacao":1,  
  "nomeOperacao":"operacao financeira",  
  "descricaoOperacao":"Essa é a operacao financeira..",  
  "dataInicioVigencia":"2022-10-10",  
  "dataFimVigencia":"2023-10-10",  
  "percentualRetencaoEmissao":22,  
  "gapFluxoCaixa":23,  
  "matriculaGo":21,  
  "clienteId":1,  
  "celulaId":1 }
```



- Classe Profissional:

Deve ser utilizado a URL: <http://localhost:8080/stfinancial/profissional/salvarProfissional>
o Json que deve ser colocado no insomnia:

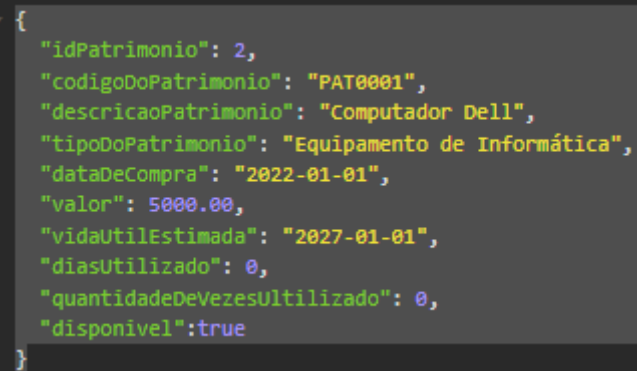
```
{"matricula": 2,  
"nomeProfissional": "thiago henrique",  
"dataAdmissao": "2021-07-10",  
"salario": 1800,  
"encargos": 400,  
"beneficioComplementar": 150,  
"beneficioEmpresa": 160,  
"custoEquipamento": 200.00,  
"custoLicencas": 100.00,  
"custoPessoal": 250,  
"custoEfetivoTotalHora": 40,  
"operacao":{"operacaoPadrao":2},  
"idOperacao": 1,  
"mesComp":"2023-04-01"}
```

```
1 {  
2   "matricula": 2,  
3   "nomeProfissional": "thiago henrique",  
4   "dataAdmissao": "2021-07-10",  
5   "salario": 1800,  
6   "encargos": 400,  
7   "beneficioComplementar": 150,  
8   "beneficioEmpresa": 160,  
9   "custoEquipamento": 200.00,  
0   "custoLicencas": 100.00,  
1   "custoPessoal": 250,  
2   "custoEfetivoTotalHora": 40,  
3   "operacao":{  
4     "operacaoPadrao":2  
5   },  
6   "idOperacao": 1,  
7   "mesComp":"2023-04-01"  
8 }
```

- Classe Patrimonio:

Deve ser utilizado a URL: <http://localhost:8080/stfiscal/patrimonio/salvarPatrimonio>
o Json que deve ser colocado no insomnia:

```
{ "idPatrimonio": 2,  
  "codigoDoPatrimonio": "PAT0001",  
  "descricaoPatrimonio": "Computador Dell",  
  "tipoDoPatrimonio": "Equipamento de Informática",  
  "dataDeCompra": "2022-01-01",  
  "valor": 5000.00,  
  "vidaUtilEstimada": "2027-01-01",  
  "diasUtilizado": 0,  
  "quantidadeDeVezesUtilizado": 0,  
  "disponivel": true }
```



```
{  
  "idPatrimonio": 2,  
  "codigoDoPatrimonio": "PAT0001",  
  "descricaoPatrimonio": "Computador Dell",  
  "tipoDoPatrimonio": "Equipamento de Informática",  
  "dataDeCompra": "2022-01-01",  
  "valor": 5000.00,  
  "vidaUtilEstimada": "2027-01-01",  
  "diasUtilizado": 0,  
  "quantidadeDeVezesUtilizado": 0,  
  "disponivel": true  
}
```

- Classe Controle:

Deve ser utilizado a URL: <http://localhost:8080/stfiscal/control/salvarControle>
o Json que deve ser colocado no insomnia:

```
{
  "idControle": 1,
  "fkPatrimonio": 1,
  "matricula": 1,
  "idPatrimonio": 1,
  "codigoDoControle": "1234",
  "descricaoControle": "Descrição do controle",
  "status": "Ativo",
  "dataDeEntregaDoPatrimonio": "2023-04-25",
  "dataDeDevolucaoDoPatrimonio": "2023-04-30",
  "observacoes": "Observações do controle",
  "dataDaManutencao": "2023-05-01",
  "tipoDeManutencao": "Manutenção preventiva",
  "profissionalDaManutencao": "Nome do profissional",
  "quantidadeDeVezesUtilizado": "10",
  "diasUtilizado": "20",
  "motivoDaDevolucao": "Motivo da devolução",
  "devolvido": true, "mesComp": "2023-04-01",
  "idProfissional": {
    "mesComp": "2023-04-01",
    "matricula": 1 },
  "patrimonio": {"idPatrimonio": 1}
}
```

```

{
  "idControle": 1,
  "fkPatrimonio":1,
  "matricula":1,
  "idPatrimonio":1,
  "codigoDoControle": "1234",
  "descricaoControle": "Descrição do controle",
  "status": "Ativo",
  "dataDeEntregaDoPatrimonio": "2023-04-25",
  "dataDeDevolucaoDoPatrimonio": "2023-04-30",
  "observacoes": "Observações do controle",
  "dataDaManutencao": "2023-05-01",
  "tipoDeManutencao": "Manutenção preventiva",
  "profissionalDaManutencao": "Nome do profissional",
  "quantidadeDeVezesUtilizado": "10",
  "diasUtilizado": "20",
  "motivoDaDevolucao": "Motivo da devolução",
  "devolvido": true,
    "mesComp": "2023-04-01",
  "idProfissional": {

    "mesComp": "2023-04-01",
    "matricula":1
  },
  "patrimonio": {
    "idPatrimonio": 1
  }
}

```


Regras de negócio e validações

Porque Primary Key composta no cadastro de controle?

Seguindo minha regra de negócio o mesmo patrimônio pode ter acesso a diversos profissionais então isso só é possível se for composta, e o mesmo profissional pode ter acesso a mais de 1 patrimônio seja ele notebook, ou qualquer outro equipamento que a empresa pode oferecer (seja um cadeado, ou até mesmo um teclado).

```
4 usages  shadowbmo
@Embeddable
@Data
@AllArgsConstructor
@NoArgsConstructor
public class ControlePK implements Serializable {

    no usages
    @Column(name = "idControle")
    private Long idControle;

    no usages
    @Columns(columns = {@Column(name = "mesComp"), @Column(name = "matricula")})
    private ProfissionalId profissional;

    no usages
    @MapsId("patrimonio")
    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "idPatrimonio")
    private Patrimonio patrimonio;
}

no usages
@ManyToOne(fetch = FetchType.EAGER)
@JoinColumns({
    @JoinColumn(name = "mesComp", insertable = false, updatable = false),
    @JoinColumn(name = "matricula", insertable = false, updatable = false)
})
private Profissional profissional;
```

Retiramos o trace com o seguinte código no “application.propies”

```
server.error.include-stacktrace = never
```

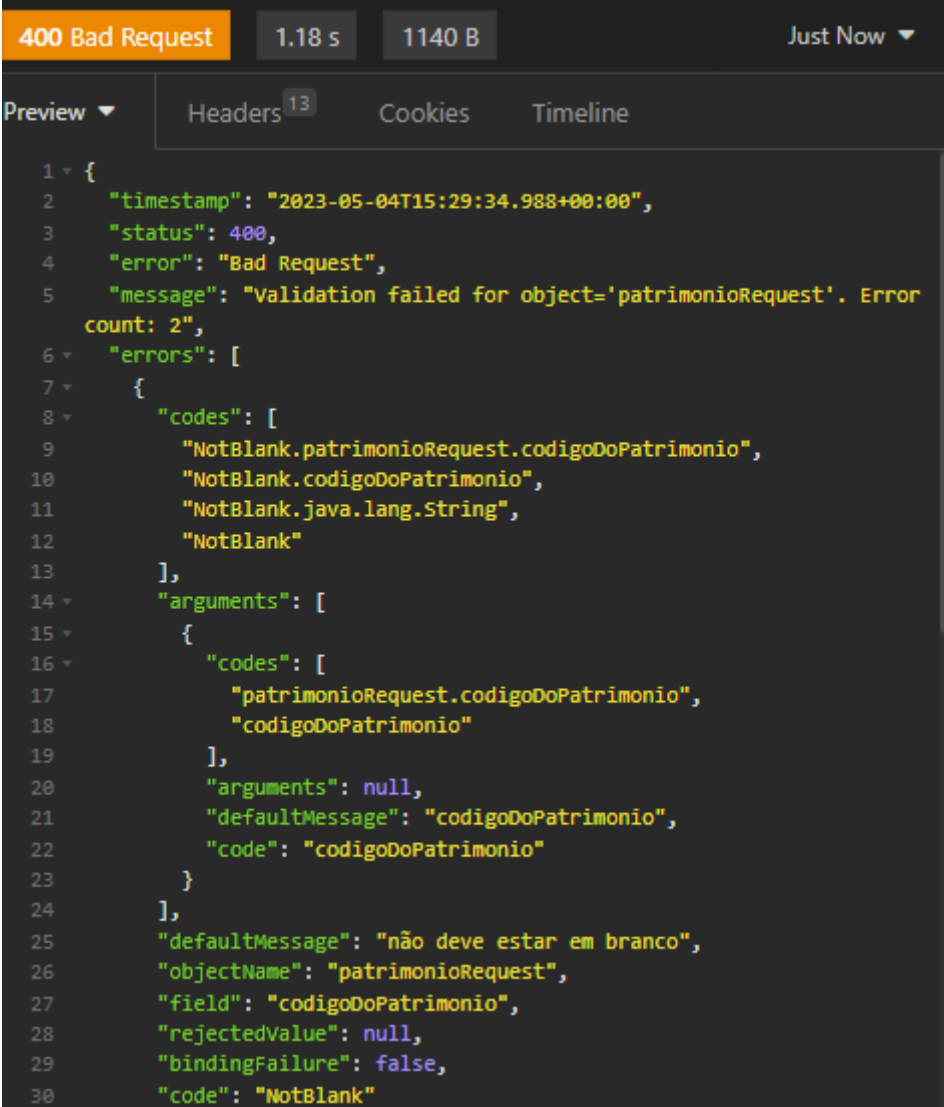
O que é o trace?

Trace é aquele código que mostra o erro muito detalhado, porém numa aplicação o cliente não vai ver o erro como o código a baixo:

```
"trace": "org.springframework.orm.jpa.JpaObjectRetrievalFailureException:
Unable to find com.stefanini.stfinanciamodel.Profissional with id
ProfissionalId(matricula=13, horasMes=2023-04-01); nested exception is
javax.persistence.EntityNotFoundException: Unable to find
com.stefanini.stfinanciamodel.Profissional with id
ProfissionalId(matricula=13, horasMes=2023-04-01)\r\n\tat
org.springframework.orm.jpa.EntityManagerFactoryUtils.convertJpaAccessExcep
tionIfPossible(EntityManagerFactoryUtils.java:379)\r\n\tat
org.springframework.orm.jpa.vendor.HibernateJpaDialect.translateExceptionIf
Possible(HibernateJpaDialect.java:235)\r\n\tat
org.springframework.orm.jpa.AbstractEntityManagerFactoryBean.translateExcep
tionIfPossible(AbstractEntityManagerFactoryBean.java:551)\r\n\tat
org.springframework.dao.support.ChainedPersistenceExceptionTranslator.trans
lateExceptionIfPossible(ChainedPersistenceExceptionTranslator.java:61)\r\n\tat
org.springframework.dao.support.DataAccessUtils.translateIfNecessary(DataAc
cessUtils.java:242)\r\n\tat
org.springframework.dao.support.PersistenceExceptionTranslationInterceptor.
invoke(PersistenceExceptionTranslationInterceptor.java:152)\r\n\tat
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(Reflec
tiveMethodInvocation.java:186)\r\n\tat
org.springframework.data.jpa.repository.support.CrudMethodMetadataPostProce
ssor$CrudMethodMetadataPopulatingMethodInterceptor.invoke(CrudMethodMetadat
aPostProcessor.java:174)\r\n\tat
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(Reflec
tiveMethodInvocation.java:186)\r\n\tat
org.springframework.aop.interceptor.ExposeInvocationInterceptor.invoke(Expo
```

E por isso foi feito algumas validações na quais devolvem uma resposta personalizado para dizer exatamente onde está o erro para que seja mais objetivo.

Validações do próprio Spring boot também trazem exceções e podemos modificá-las para que não fique um erro enorme como um exemplo do @Valid utilizado em “salvarPatrimonio”



```
400 Bad Request 1.18 s 1140 B Just Now ▼
Preview ▾ Headers 13 Cookies Timeline
1 {
2   "timestamp": "2023-05-04T15:29:34.988+00:00",
3   "status": 400,
4   "error": "Bad Request",
5   "message": "Validation failed for object='patrimonioRequest'. Error
count: 2",
6   "errors": [
7     {
8       "codes": [
9         "NotBlank.patrimonioRequest.codigoDoPatrimonio",
10        "NotBlank.codigoDoPatrimonio",
11        "NotBlank.java.lang.String",
12        "NotBlank"
13      ],
14      "arguments": [
15        {
16          "codes": [
17            "patrimonioRequest.codigoDoPatrimonio",
18            "codigoDoPatrimonio"
19          ],
20          "arguments": null,
21          "defaultMessage": "codigoDoPatrimonio",
22          "code": "codigoDoPatrimonio"
23        }
24      ],
25      "defaultMessage": "não deve estar em branco",
26      "objectName": "patrimonioRequest",
27      "field": "codigoDoPatrimonio",
28      "rejectedValue": null,
29      "bindingFailure": false,
30      "code": "NotBlank"
```

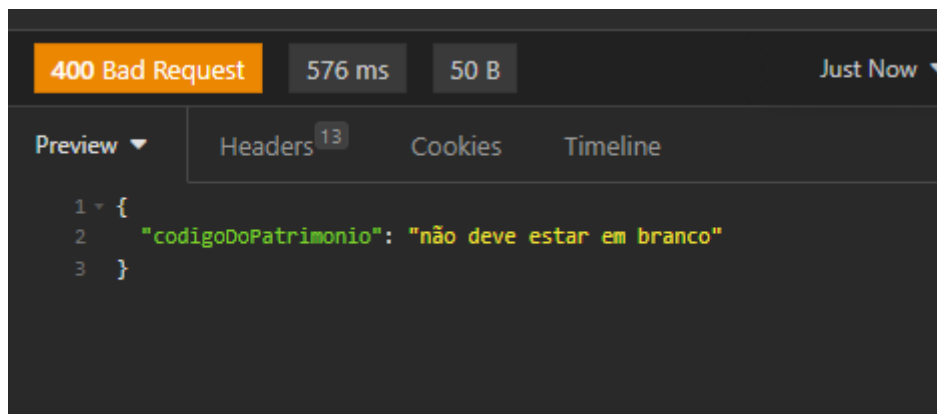
Passando a seguinte linha de código, após estourar a exceção ele vai trocar a mensagem de erro para somente a coluna que deu o erro e qual o erro que foi estourado.

Enviando um retorno do erro para o body da aplicação ficando mais simples

```
no usages shadowbmo
@ExceptionHandler(MethodArgumentNotValidException.class)
public ResponseEntity<Map<String, String>> tratarErro400(MethodArgumentNotValidException ex) {
    Map<String, String> erros = new HashMap<>();
    ex.getBindingResult().getFieldErrors().forEach((error) -> {
        String nomeCampo = error.getField();
        String mensagemErro = error.getDefaultMessage();
        erros.put(nomeCampo, mensagemErro);
    });
    return ResponseEntity.badRequest().body(erros);
}
```

Podemos ver a enorme diferença entre um e outro tornando mais fácil de identificar o erro, e as validações são feitas pelo próprio spring boot utilizando o @Valid e @NotBlank caso queira ver as demais validações que podem ser aplicadas pode acessar no link abaixo

<https://www.baeldung.com/javax-validation>



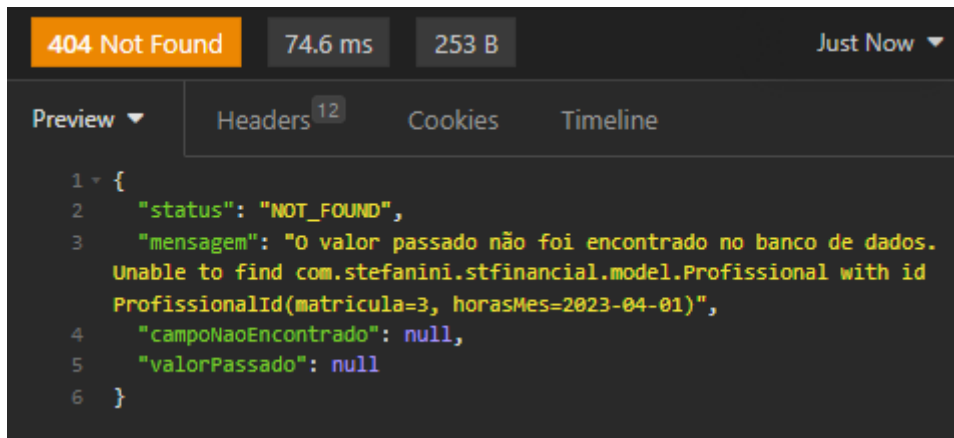
Seguindo o mesmo princípio de o próprio Spring Boot lançar uma exceção sem a necessidade de fazer um código de validação temos o exemplo de caso ele não encontre o ID de algum campo

Foi feito o seguinte código

Na qual ele manda essa mensagem especificando que não foi encontrado e a própria mensagem do Spring boot para entendemos melhor o que aconteceu

```
no usages shadowbmo
@ExceptionHandler(EntityNotFoundException.class)
public ResponseEntity<ErroDTO> tratarErro404(EntityNotFoundException ex) {
    String mensagem = "O valor passado não foi encontrado no banco de dados. " + ex.getMessage();
    ErroDTO erroDTO = new ErroDTO(HttpStatus.NOT_FOUND, mensagem);
    return ResponseEntity.status(HttpStatus.NOT_FOUND).body(erroDTO);
}
```

A sua demonstração fica como a seguir na qual passa a mensagem que não foi encontrado no banco e a mensagem que o próprio Spring Boot demonstraria



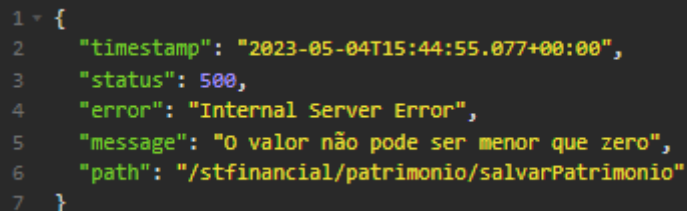
```
1 {
2   "status": "NOT_FOUND",
3   "mensagem": "O valor passado não foi encontrado no banco de dados.
  Unable to find com.stefanini.stfinanciamodel.Profissional with id
  ProfissionalId(matricula=3, horasMes=2023-04-01)",
4   "campoNaoEncontrado": null,
5   "valorPassado": null
6 }
```

Na classe Patrimônio foram feitas as seguintes validações:

Se o valor do patrimônio for menor que 0 ele dispara um erro e nem prossegue com o código (por conta do throw new) jogando a exceção novamente para que o usuário entenda exatamente o que aconteceu

```
if (patrimonio.getValor() < 0) {
    throw new IllegalArgumentException("O valor não pode ser menor que zero");
}
```

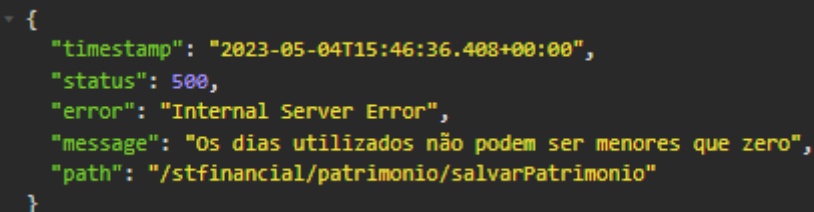
E a visualização aparece da seguinte forma



```
1 {
2   "timestamp": "2023-05-04T15:44:55.077+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "message": "O valor não pode ser menor que zero",
6   "path": "/stfinanciamodel/patrimonio/salvarPatrimonio"
7 }
```

Não pode ter dias utilizados negativos então foi feita uma validação parecida

```
if (patrimonio.getDiasUtilizado() < 0) {
    throw new IllegalArgumentException("Os dias utilizados não podem ser menores que zero");
}
```



```
{
  "timestamp": "2023-05-04T15:46:36.408+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "message": "Os dias utilizados não podem ser menores que zero",
  "path": "/stfinanciamodel/patrimonio/salvarPatrimonio"
}
```

A data inserida que foi comprado o patrimônio não pode ser depois da data de vida útil estimada por isso foi utilizado o comando .after() na qual ela faz essa comparação nessas datas

```
Date dataCompra = patrimonio.getDataDeCompra();
Date dataVidaUtilEstimada = patrimonio.getVidaUtilEstimada();

if (dataCompra.after(dataVidaUtilEstimada)) {
    throw new IllegalArgumentException("A data de compra não pode ser depois à data de vida útil estimada");
}
```

```
1 {
2   "timestamp": "2023-05-04T15:48:47.633+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "message": "A data de compra não pode ser depois à data de vida útil estimada",
6   "path": "/stfinanciam/patrimonio/salvarPatrimonio"
7 }
```

Caso ele passe por essas validações é feita o cadastro do patrimônio

Na classe Controle foram feitas as seguintes validações:

A primeira validação feita é na qual é se a data de devolução que foi inserida está depois da data que o funcionário pegou o patrimônio (e esse campo pode ser nulo em casos onde o mesmo ainda não devolveu)

```
----- Data De Entrega -----
Date dataDeEntrega = controle.getDataDeEntregaDoPatrimonio();
Date dataDeDevolucao = controle.getDataDeDevolucaoDoPatrimonio();

if (dataDeDevolucao != null && dataDeEntrega.after(dataDeDevolucao)) {
    throw new RuntimeException("Não é possível devolver um patrimônio antes de tê-lo entregue");
}
```

```
1 {
2   "timestamp": "2023-05-04T15:48:47.633+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "message": "A data de compra não pode ser depois à data de vida útil estimada",
6   "path": "/stfinanciam/patrimonio/salvarPatrimonio"
7 }
```

Eu verifico se o patrimônio existe e caso não exista já informo que o id inserido está incorreto pois, não foi encontrado no banco

```

----- Patrimonio -----
Long patrimonioId = controle.getControleId().getPatrimonio().getIdPatrimonio();
Patrimonio patrimonio = repoPatri.findById(patrimonioId).orElse( other: null);
if (patrimonio == null) {
    throw new EntityNotFoundException("Patrimônio não encontrado com o ID: " + patrimonioId);
}

```

```

{
  "status": "NOT_FOUND",
  "mensagem": "O valor passado não foi encontrado no banco de dados.
  Patrimônio não encontrado com o ID: 12",
  "campoNaoEncontrado": null,
  "valorPassado": null
}

```

Caso o patrimônio já esteja sendo utilizado por outra pessoa também alertarmos sobre esse erro

```

if(patrimonio.getDisponivel() == false){
    throw new RuntimeException("O patriomonio informado já está sendo utilizado e por isso não foi possível cadastrar");
}

```

```

1 {
2   "timestamp": "2023-05-04T16:00:38.469+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "message": "O patriomonio informado já está sendo utilizado e por
6   isso não foi possível cadastrar",
7   "path": "/stfinancial/controle/salvarControle"
8 }

```

Outra validação é que caso a pessoa não se atente e tente cadastrar um patrimônio com defeito, roubado, vendido ou descartado também alertamos

```

if(patrimonio.getComDefeito() == true){
    throw new RuntimeException("O patriomonio informado está com defeito e por isso não foi possível cadastrar");
}
if(patrimonio.getRoubado() == true){
    throw new RuntimeException("O patriomonio informado está roubado e por isso não foi possível cadastrar");
}
if(patrimonio.getVendido() == true){
    throw new RuntimeException("O patriomonio informado foi vendido e por isso não foi possível cadastrar");
}
if(patrimonio.getDescartado() == true){
    throw new RuntimeException("O patriomonio informado está descartado e por isso não foi possível cadastrar");
}

```

Defeito

```
1 {
2   "timestamp": "2023-05-04T16:02:34.628+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "message": "O patriomonio informado está com defeito e por isso não
6   foi possivel cadastrar",
7   "path": "/stfinanciamal/controle/salvarControle"
8 }
```

Vendido:

```
1 {
2   "timestamp": "2023-05-04T16:02:55.050+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "message": "O patriomonio informado foi vendido e por isso não foi
6   possivel cadastrar",
7   "path": "/stfinanciamal/controle/salvarControle"
8 }
```

Descartado:

```
1 {
2   "timestamp": "2023-05-04T16:04:14.906+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "message": "O patriomonio informado está descartado e por isso não
6   foi possivel cadastrar",
7   "path": "/stfinanciamal/controle/salvarControle"
8 }
```

Roubado:

```
1 {
2   "timestamp": "2023-05-04T16:05:31.705+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "message": "O patriomonio informado está roubado e por isso não foi
6   possivel cadastrar",
7   "path": "/stfinanciamal/controle/salvarControle"
8 }
```

Caso ele passe por todas essas validações ele cadastra normalmente (lembrando que ele também é passo por validações do próprio Spring Boot utilizando @Valid)

Caso ele passe em todas as validações ele muda o atributo do patrimônio de disponível para False pois, ele não se encontra mais disponível e seta a quantidade de vezes que foi utilizado para a quantidade atual +1 e salva essa informação no banco

```
patrimonio.setDisponivel(false);
patrimonio.setQuantidadeDeVezezUtilizado(patrimonio.getQuantidadeDeVezezUtilizado() + 1);
repoPatri.save(patrimonio);

return repo.save(controle);
```

Algumas funcionalidades feitas:

Cada delete é feito de forma lógica e não é realmente apagado do banco de dados pois, o que aconteceu com cada equipamento é importante para o controle dele.

```
no usages  shadowbmo
@RequestMapping(path = "/deletarPatrimonioRoubado/{idPatrimonio}" , method = RequestMethod.DELETE)
public ResponseEntity<?> deletarPatrimonioRoubado(@PathVariable Long idPatrimonio) {
    return ResponseEntity.ok(service.deletarPatrimonioRoubado(idPatrimonio));
}

no usages  shadowbmo
@RequestMapping(path = "/deletarPatrimonioDefeito/{idPatrimonio}" , method = RequestMethod.DELETE)
public ResponseEntity<?> deletarPatrimonioDefeito(@PathVariable Long idPatrimonio) {
    return ResponseEntity.ok(service.deletarPatrimonioDefeito(idPatrimonio));
}

no usages  shadowbmo *
@RequestMapping(path = "/deletarPatrimonioDescartado/{idPatrimonio}" , method = RequestMethod.DELETE)
public ResponseEntity<?> deletarPatrimonioDescartado(@PathVariable Long idPatrimonio) {
    System.out.println(idPatrimonio);
    return ResponseEntity.ok(service.deletarPatrimonioDescartado(idPatrimonio));
}

no usages  shadowbmo
@RequestMapping(path = "/deletarPatrimonioVendido/{idPatrimonio}" , method = RequestMethod.DELETE)
public ResponseEntity<?> deletarPatrimonioVendido(@PathVariable Long idPatrimonio) {
    return ResponseEntity.ok(service.deletarPatrimonioVendido(idPatrimonio));
}
```

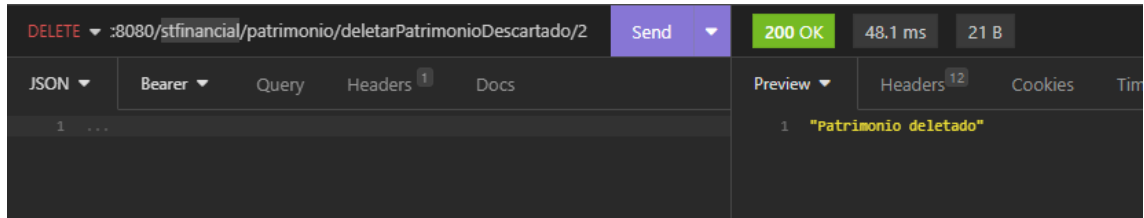
Ele seta o campo informado de false para True e na hora de listar ele não aparece mais, pois não pode ser mais utilizado.

```
public String deletarPatrimonioRoubado(Long id) {
    Optional<Patrimonio> patrimonioOpt = repo.findById(id);
    if (patrimonioOpt.isPresent()) {
        Patrimonio patrimonio = patrimonioOpt.get();
        patrimonio.setRoubado(true);
        repo.save(patrimonio);
    } else {
        throw new RuntimeException("Patrimônio com o ID: " + id + " não foi encontrado");
    }
    return "Patrimonio deletado";
}
```

Este mesmo código é utilizado para os demais atributos o que muda é o que está escrito depois do set;

A mensagem caso positivo

URL: <http://localhost:8080/stfinacial/patrimonio/deletarPatrimonioDescartado/2>



Caso ele não encontre o demonstra a seguinte informação

