

Modelo de Transferência de Dados

Modelo de Transferência de Dados

API REST

Modelo de Transferência de Dados

Histórico de versão

Versão	Autor	Data	Ação
2.0.0.0	Rosfran Borges	03/02/2020	Novo modelo XSD e parâmetros
2.0.0.1	Rosfran Borges	28/02/2020	Inclusão de exemplo XML com novos campos
2.0.0.1	Rosfran Borges	09/03/2020	Novo aplicativo Validador

Modelo de Transferência de Dados

Sumário

1. Autenticação	1
2. Configuração de acesso com certificado – Caso de Ocorrência de Erro de HandShake SSL1	
3. Processos API	6
Validar Serviço.....	7
Enviar Processos.....	7
Pesquisa de todos os protocolos.....	9
Pesquisa protocolos por filtro	9
4. Acompanhamento do processamento dos arquivos enviados.....	11
5. Aplicativo Validador de XMLs.....	15
1.1. Configuração	15
1.2. Uso do Validador	17
1.3. Fazendo uma Requisição – Simulando um Envio de XML.....	17
1.4. Operação de Validação	20

1. Autenticação

O tipo de autenticação utilizada é **Basic Auth**. Quando a API é chamada, é necessário incluir no cabeçalho da requisição (request HTTP) o usuário e a senha do tribunal. O **login e a senha de acesso** a API são informados pelo **CNJ**.

Basicamente, o funcionamento desse método de autenticação é suportado pelo protocolo HTTP através da inclusão do campo **Authorization** no cabeçalho. Cada usuário (Tribunal) tem um valor único gerado para esse campo. É MUITO importante que o Tribunal tenha cuidado ao preencher esse campo, pois as informações sobre o usuário são obtidas desse campo **Authorization**, e caso o campo **Authorization** contenha as credenciais de outro Tribunal, o envio será creditado ao Tribunal especificado no campo **Authorization**.

Exemplo abaixo de uma requisição para confirmação do status da conexão com o serviço (método /v1/processos):

```
GET      https://wwwh.cnj.jus.br/modelo-de-transferencia-de-dados/v1/processos/G1
HTTP/1.1
Accept-Encoding: gzip,deflate
Authorization: Basic UXXXXXQ1NmRiOGJjOGI0YTXXXXZIYzQ0NMDZIZTI0
Host: wwwh.cnj.jus.br
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)
```

2. Configuração de acesso com certificado – Caso de Ocorrência de Erro de HandShake SSL

Se, após o teste do passo 1 acima, ocorrer um erro de handshake SSL, aparecendo mensagens como a abaixo:

```
com.sun.jersey.api.client.ClientHandlerException:  
javax.net.ssl.SSLHandshakeException:  
sun.security.validator.ValidatorException: PKIX path building failed:
```

sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target

...

***Caused by: javax.net.ssl.SSLHandshakeException:
sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find
valid certification path to requested target***

...

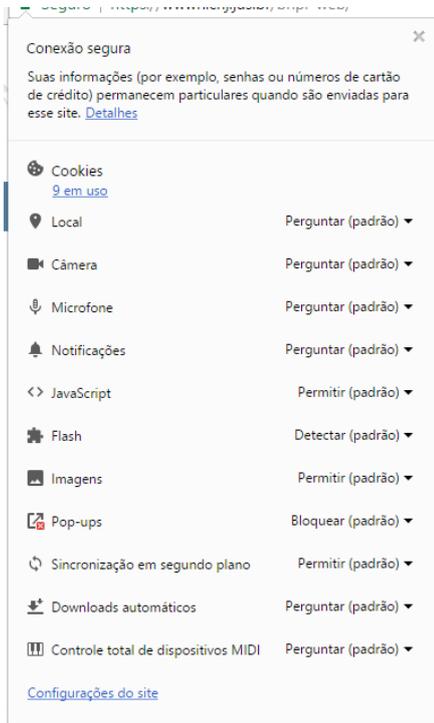
Será necessário importar o certificado da página do CNJ, incluir esse certificado numa key store e executar o seu cliente. O procedimento para gravar e importar o certificado do CNJ é o seguinte:

1. Vá para a página do CNJ, com suporte a HTTPS, do Selo (<https://www.cnj.jus.br/modelo-de-transferencia-de-dados/v1/processos>):

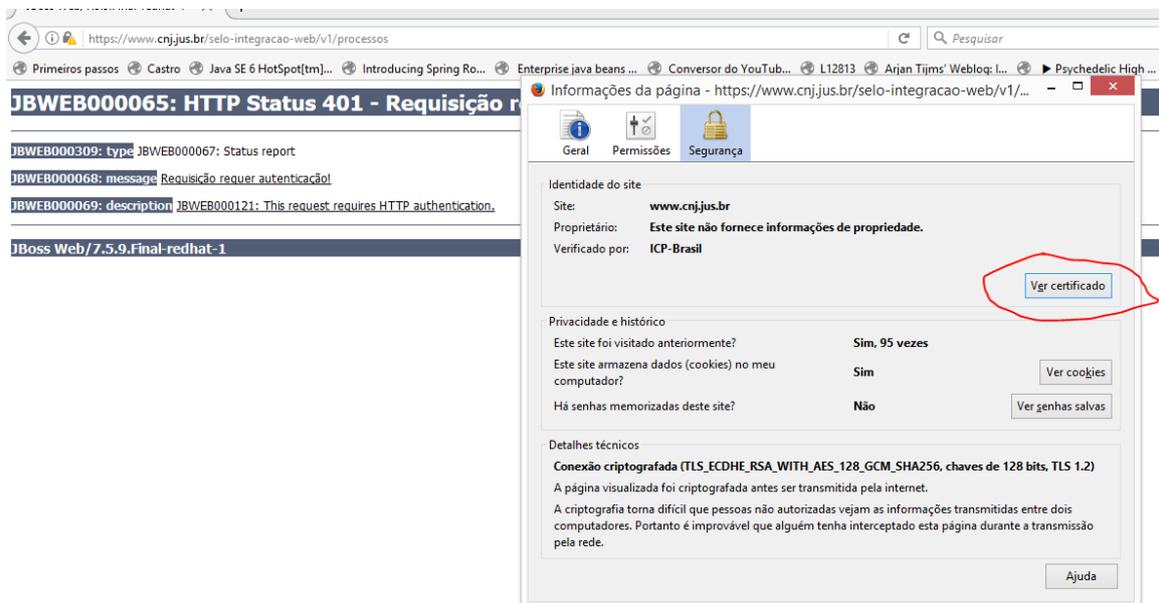


2. Aparecerá a tela pedindo autenticação, mas não tem problema. Clique no ícone com um cadeado, na parte esquerda da caixa de entrada da URL de conexão:

Modelo de Transferência de Dados

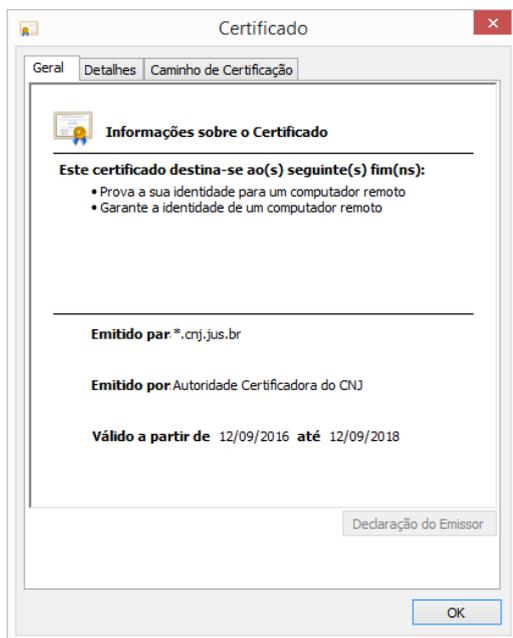


3. Clique no link 'Detalhes'. Aparecerá uma tela parecida com a abaixo. Perceba que tem um botão chamado "View Certificate" ou "Ver Certificado" (destacado em vermelho):

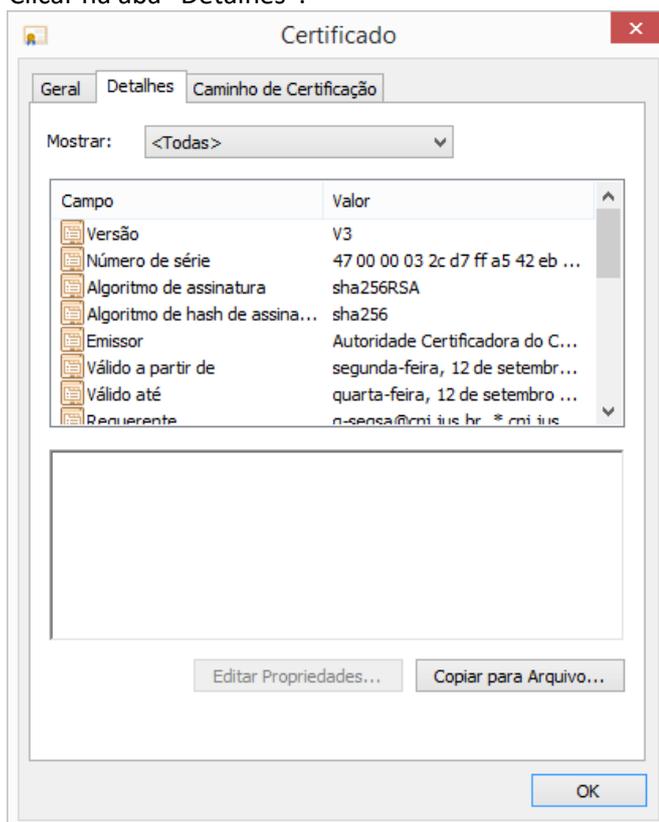


4. Abrirá uma janela para ver o certificado:

Modelo de Transferência de Dados



5. Clicar na aba “Detalhes”:



6. Clicar no botão “Copiar para Arquivo...”, e clique no botão “Avançar”:

Modelo de Transferência de Dados

5



Assistente para Exportação de Certificados

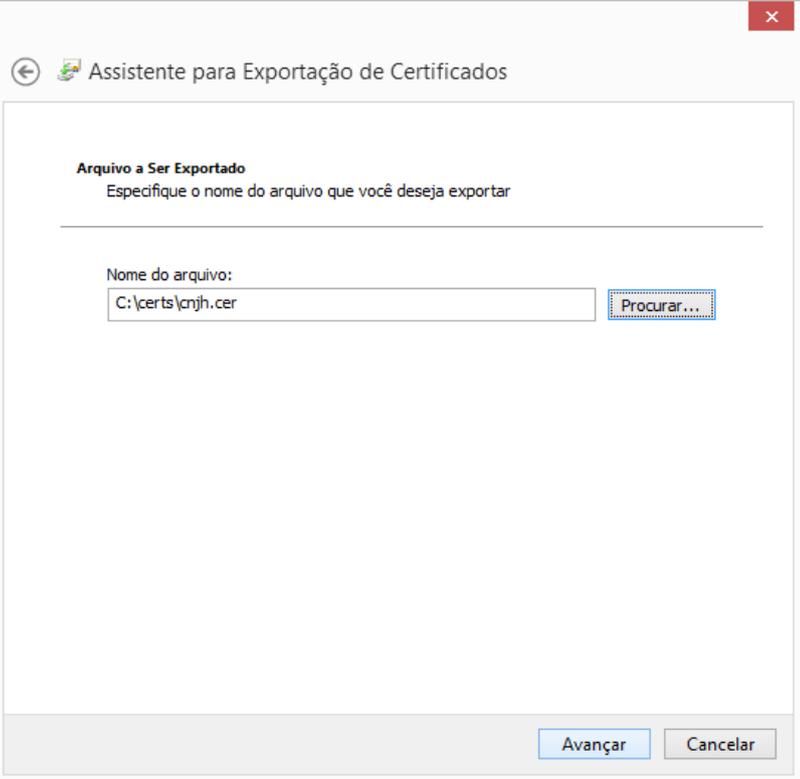
Formato do Arquivo de Exportação
Os certificados podem ser exportados em diversos formatos de arquivo.

Selecione o formato a ser usado:

- X.509 binário codificado por DER (*.cer)
- X.509 codificado na base 64 (*.cer)
- Padrão de Sintaxe de Mensagens Criptografadas - Certificados PKCS n°7 (*.p7b)
 - Incluir todos os certificados no caminho de certificação, se possível
- Troca de Informações Pessoais - PKCS n° 12 (.PFX, .P12)
 - Incluir todos os certificados no caminho de certificação, se possível
 - Excluir a chave privada se a exportação tiver êxito
 - Exportar todas as propriedades estendidas
- Repositório de certificados serializado da Microsoft (.SST)

Avançar Cancelar

7. Informe o caminho onde o arquivo com extensão .CER será gravado – no caso, utilizei o caminho **C:\certs\cnjh.cer**:



Assistente para Exportação de Certificados

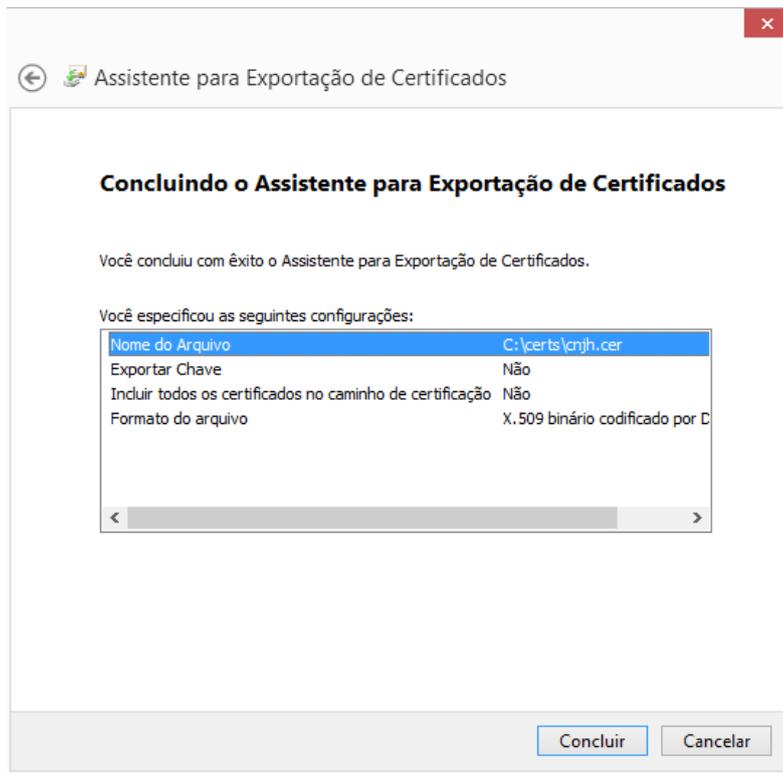
Arquivo a Ser Exportado
Especifique o nome do arquivo que você deseja exportar

Nome do arquivo:
C:\certs\cnjh.cer

Procurar...

Avançar Cancelar

8. Clique em Concluir para finalizar a gravação:



9. Importe o certificado .cer para o repositório cacerts da JDK que está utilizando com o comando: **keytool -import -file "c:\certs\cnjh.cer" -storepass changeit -keystore cacerts -alias cnjh**
ou
keytool -import -trustcacerts -alias cnjh -file "C:\certs\cnjh.cer" -keystore "C:\Program Files\Java\jdk1.8.0_102\jre\lib\security\cacerts"
10. Após isso, pode executar o cliente Java que acessa o web service do Selo.

3. Processos API

O namespace **processos** contém coleções de recursos para envio e pesquisa de dados processuais. O XML para envio dos processos tem de ser desenvolvido utilizando o XSD `replicacao-nacional.xsd` e o `modelo-de-transferencia-de-dados-1.0.xsd`. São schemas complementares – o `replicacao-nacional.xsd` define uma lista de processos, e a estrutura de cada um desses processos é obtida do `modelo-de-transferencia-de-dados-1.0.xsd`. Detalhes nos anexos I e II.

Pode-se enviar arquivos de qualquer tamanho. NÃO há mais tamanho máximo permitido para envio do arquivo.

Todas as mensagens de resposta do serviço são no formato JSON.

URI homologação	https://wwwh.cnj.jus.br/modelo-de-transferencia-de-dados/v1
URI produção	https://www.cnj.jus.br/modelo-de-transferencia-de-dados/v1

Validar Serviço

GET	/v1/processos/
-----	----------------

Valida a conexão com o serviço. A mensagem de resposta é:

Código HTTP	200
Status	SUCESSO
Sucess	Mensagem

Enviar Processos

POST	/v1/processos/{GRAU}
PUT	/v1/processos/{GRAU}
POST	/v1/processos/json/{GRAU}

Envia o arquivo XML contendo um processo ou uma lista de processos. **DEVE ser enviado 1 único arquivo por vez, ou seja, por requisição.**

Como opção em relação aos endpoints acima, existe o endpoint **/v1/processos/json/{GRAU}**, cujo corpo da requisição contém um JSON, ao invés de um XML.

O parâmetro GRAU, limitado a até 3 caracteres, pode assumir os valores seguintes:

- **SUP** para Tribunais Superiores
- **G2** para 2º grau
- **G1** para 1º grau (justiça comum)
- **TR** para Turmas Recursais

Modelo de Transferência de Dados

8

- **JE** para Juizados especiais
- **TRU** para Turmas Regionais de Uniformização
- **TNU** para Turmas Nacionais de Uniformização

O tipo de Media Type para o envio tem de ser `multipart/form-data`, ou seja, enviar o arquivo como **multipart**.

Caso seja executado com sucesso e sem erros, a resposta virá com HTTP status code 201.

A mensagem de resposta do envio com sucesso é uma mensagem JSON no formato abaixo:

```
{  
  "status": "SUCESSO",  
  "protocolo": "SIGLATRIBUNALXXXXXXXXXXXXXXXXXXXXXXXXXXXX"  
}
```

Onde o atributo status aparecerá o valor SUCESSO, e o protocolo será uma composição da sigla do Tribunal, com uma sequência de 26 dígitos. Essa sequência de 26 dígitos é composto pelo seguinte: número de 5 dígitos aleatórios, ano (4 dígitos), mês (2 dígitos), dia (2 dígitos) e o restante é o intervalo de tempo em milissegundos entre a data 01/01/1970 e o dia e hora atuais de envio.

É importante que os tribunais, nas suas soluções de integração, mantenham registro desses números de protocolo. Eles são importantes para pesquisas posteriores na aplicação dedicada a acompanhar o processamento.

Abaixo, uma representação gráfica dos elementos que compõem o protocolo:

```
{
  "status": "SUCESSO",
  "protocolo": "TJAP31845202002121581533417830"
}
```

No exemplo acima, em vermelho, a sigla do Tribunal. Em verde, a sequência aleatória de 5 dígitos, em preto o ano, em laranja o mês, em azul o dia, e em amarelo a marcação atual de tempo em milissegundos.

Pesquisa de todos os protocolos

GET	/v1/protocolos/all
------------	--------------------

Retorna o quantitativo de protocolos enviados pelo órgão que está autenticado. A mensagem de retorno é:

Código HTTP	200
Status	SUCESSO
Resposta	JSON com os protocolos enviados

Pesquisa protocolos por filtro

GET	/v1/protocolos
------------	----------------

Retorna o total de protocolos de acordo com o grau informado. Filtrado por parâmetros:

- protocolo**
- dataInicio**
- dataFim**
- status**
- page**

A mensagem de retorno é:

Modelo de Transferência de Dados

10

Código HTTP	200
Status	SUCESSO
Resposta	JSON com os protocolos filtrados

4. Acompanhamento do processamento dos arquivos enviados

O arquivo XML com os processos a serem enviados serão validados de acordo com os schemas XML do Replicação Nacional, conforme Anexos 1 e 2. No entanto, essa resposta sobre a adequação ou não ao modelo XSD não é respondido de imediato ao Tribunal, pois dependerá de um processamento por parte de alguns sistemas automatizados do CNJ, que irão validar tanto o formato do dado, quanto a consistência das informações.

Caso exista divergência entre o modelo e o arquivo enviado, essa informação estará disponível para consulta através da aplicação web de consulta de status de protocolos do Selo Justiça em Números: <https://replicacao.cnj.jus.br>

Ao acessar o endereço acima, será aberto uma página de login:



As credenciais de acesso (usuário e senha) são os mesmos já utilizados pelos Tribunais para envio dos dados através da interface REST. Não houve alteração nem no nome do usuário, nem nas senhas.

Após fazer o login, será apresentada uma tela de pesquisa por protocolos de envio, como a abaixo (os dados são do ambiente de homologação, não representa casos reais):

Modelo de Transferência de Dados

Acompanhar Envios - TJRO

Nº do Protocolo	Status	Data	Qtde. de Processos	Ação
TJRO96457202002101581377668513	Processado com erro	Enviado em 10/02/2020 20:34:28	1	
TJRO96122202002101581377668286	Processado com erro	Enviado em 10/02/2020 20:34:28	1	
TJRO56511202002101581377668054	Processado com sucesso	Enviado em 10/02/2020 20:34:28	1	
TJRO21132202002101581377494804	Duplicado	Enviado em 10/02/2020 20:31:34	--	
TJRO55132202002101581377491485	Processado com erro	Enviado em 10/02/2020 20:31:31	1	
TJRO86726202002101581377475463	Processado com erro	Enviado em 10/02/2020 20:31:15	1	
TJRO99195202002101581377474574	Processado com erro	Enviado em 10/02/2020 20:31:14	1	
TJRO46479202002101581377473911	Processado com erro	Enviado em 10/02/2020 20:31:13	1	
TJRO63801202002101581377472609	Processado com erro	Enviado em 10/02/2020 20:31:12	1	
TJRO24442202002101581377465804	Processado com erro	Enviado em 10/02/2020 20:31:05	1	

Total: 160999

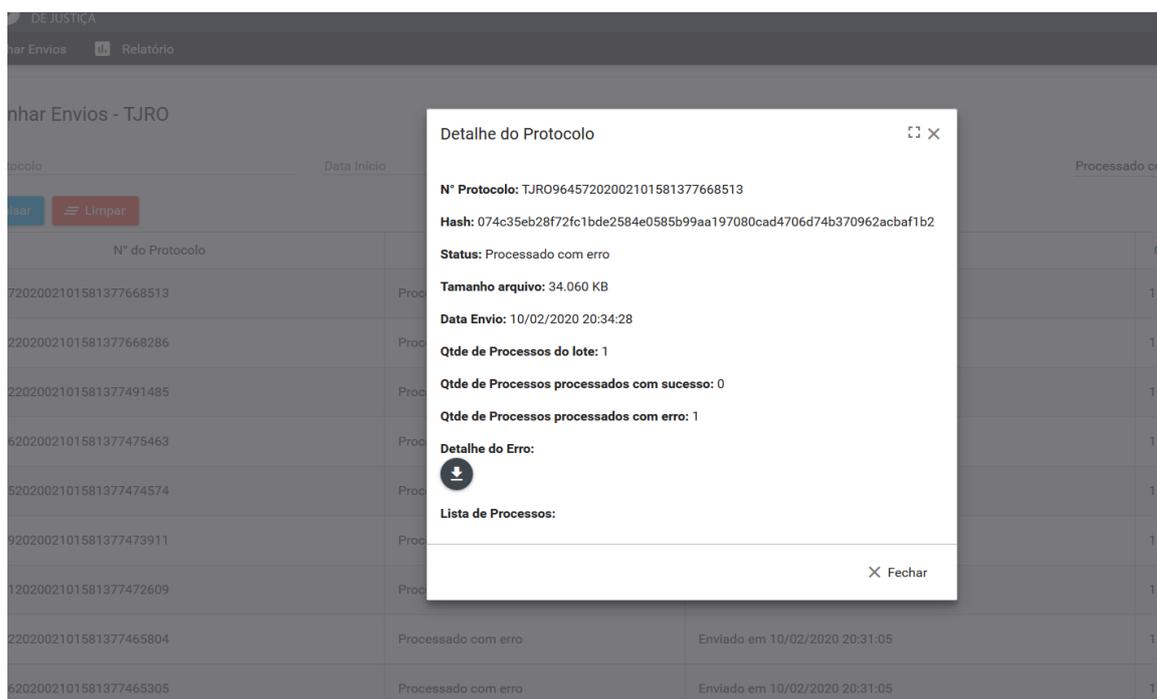
Cada uma das linhas representa 1 envio feito pelo Tribunal. Na primeira coluna, o código de protocolo, que foi explicado na Seção 2.1; na segunda coluna, um status de processamento, que pode assumir um dos seguintes valores. Os valores são cronológicos, e representam etapas no ciclo de vida do protocolo de processamento:

Status	Descrição
Enviado	Arquivo foi recém enviado, e ainda não foi iniciado o processamento do mesmo.
Aguardando Processamento	Iniciada a etapa de processamento para esse protocolo. Até esse momento, não existem ainda informações sobre os processos nos painéis de acompanhamento no Kibana
Processado com Sucesso	Dados sobre os processos foram processados sem erros estruturais graves, e estão disponíveis no ElasticSearch/Kibana

Modelo de Transferência de Dados

Processado com Erro	Do processamento do protocolo surgiram erros negociais, como por exemplo, número de processo que fuja ao padrão da Numeração Única (Resolução 65), ou código de órgão inválido (não existente)
Erro no Arquivo	Erro estrutural, ou seja, no formato do arquivo. Pode também ser resultante de problemas de recepção e/ou transmissão.

Abaixo detalhes de um arquivo processado com erro:



Ao clicar no botão na seção de “Detalhe do Erro”, o conteúdo desse arquivo (no formato JSON) detalhará o ocorrido:

```
[
{
  "tipoErro": "O arquivo foi processado, porém alguns dados não são válidos.",
  "numeroProtocolo": "TJR096457202002101581377668513",
```

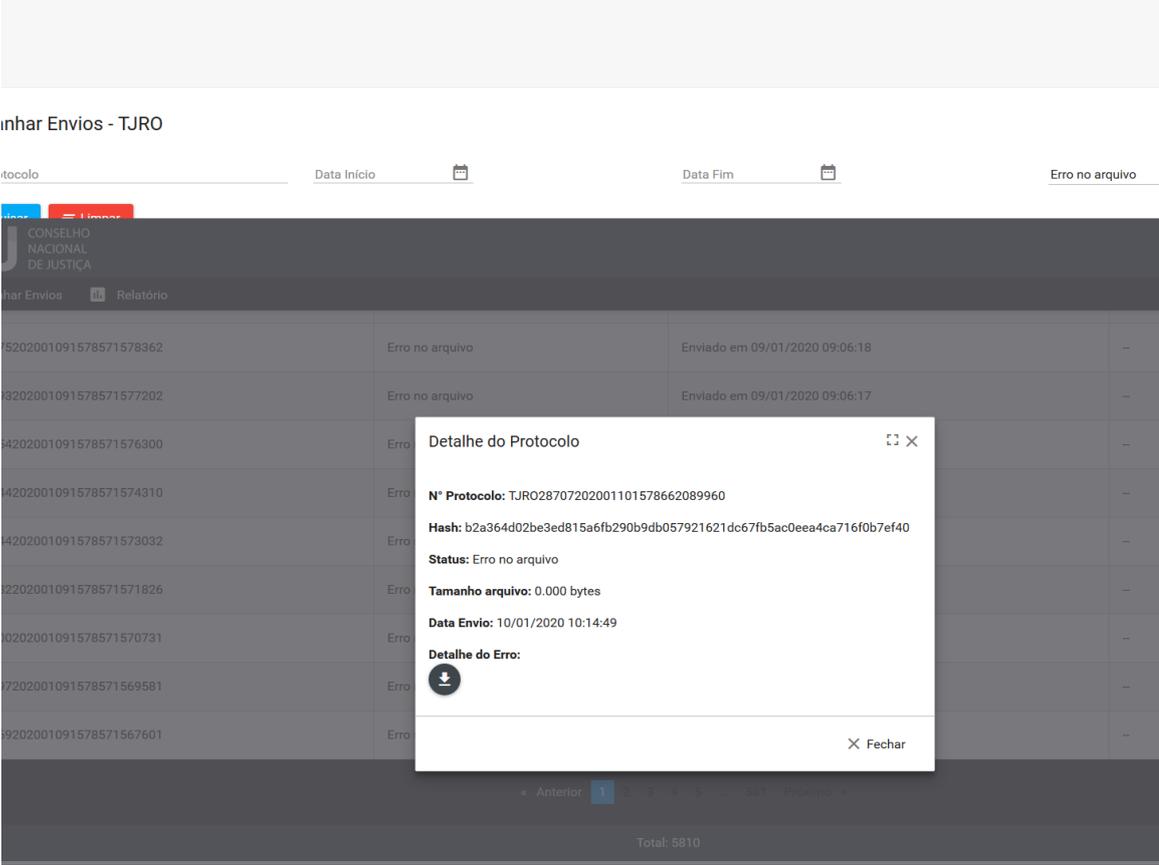
Modelo de Transferência de Dados

```

    "erroProcessamento": "Número de processo inválido: 70077204320198220001"
  }
]

```

Caso o status seja o de “Erro no Arquivo”, o detalhamento trará um indicativo da razão (arquivo sem conteúdo – tamanho de zero bytes):



The screenshot shows a web application interface for monitoring data transfers. At the top, there are filters for 'Protocolo', 'Data Início', 'Data Fim', and 'Erro no arquivo'. Below the filters is a table with columns for 'Protocolo', 'Status', 'Data Envio', and 'Erro no arquivo'. A modal window titled 'Detalhe do Protocolo' is open, displaying the following information:

- Nº Protocolo: TJRO28707202001101578662089960
- Hash: b2a364d02be3ed815a6fb290b9db057921621dc67fb5ac0eea4ca716f0b7ef40
- Status: Erro no arquivo
- Tamanho arquivo: 0.000 bytes
- Data Envio: 10/01/2020 10:14:49
- Detalhe do Erro: (with a download icon)

The table below shows several rows of data, all with the status 'Erro no arquivo'.

Protocolo	Status	Data Envio	Erro no arquivo
5202001091578571578362	Erro no arquivo	Enviado em 09/01/2020 09:06:18	-
83202001091578571577202	Erro no arquivo	Enviado em 09/01/2020 09:06:17	-
44202001091578571576300	Erro		-
44202001091578571574310	Erro		-
44202001091578571573032	Erro		-
42202001091578571571826	Erro		-
40202001091578571570731	Erro		-
7202001091578571569581	Erro		-
8202001091578571567601	Erro		-

E ao visualizar o arquivo com o “Detalhe do Erro”:

```

[
  {
    "tipoErro": "Erro na estrutura do arquivo enviado. O arquivo XML não é válido.",
    "numeroProtocolo": "TJRO28707202001101578662089960",
    "erroProcessamento": "com.fasterxml.jackson.databind.exc.MismatchedInputException:
No content to map due to end-of-input\n at [Source: (File); Line: 1, column: 0]"
  }
]

```

5. Aplicativo Validador de XMLs

O objetivo deste aplicativo é permitir que os tribunais executem localmente as suas rotinas de validação de dados XML que serão enviados para o serviço REST do Modelo de Transferência de Dados do CNJ (<https://www.cnj.jus.br/modelo-de-transferencia-de-dados/v1>).

1.1. Configuração

Para a aplicação funcionar, é necessário instalar o Docker (<https://www.docker.com>). Existem versões do Docker para Windows, Linux e MacOS.

Após instalar o Docker, faça download dos arquivos de configuração dos containers:

URL: <https://www.cnj.jus.br/owncloud/index.php/s/PCvxAj6qdqr6I5e>

Senha: validador

Ao descompactar o arquivo, será exibida a estrutura abaixo:

Nome	Data de modificaç...	Tipo	Tamanho
 build	09/03/2020 14:06	Pasta de arquivos	
 sql_create	09/03/2020 14:06	Pasta de arquivos	
 docker-compose.yaml	09/03/2020 16:19	Arquivo YAML	1 KB
 Dockerfile	06/03/2020 17:07	Arquivo	1 KB

Nesse diretório há um diretório build, no qual está localizada a aplicação, que é um microsserviço feito em Spring Boot e desenvolvido em Java. No diretório sql_create estão os scripts de criação do banco que será usado para as ações de validação dos arquivos XML e enriquecimento (veja Seção 3.2). O arquivo docker-compose.yaml é um arquivo de configuração para orquestração de serviços e será necessário para subir a instância MySQL e o container com o backend REST. O arquivo Dockerfile é um arquivo que faz o build da imagem da aplicação backend REST.

No mesmo diretório onde foi descompactado e tendo já instalado o Docker (ou o Docker Desktop para usuários MacOS e Windows), pode-se executar o comando seguinte:

docker-compose up

Esse comando subirá 2 containers Docker: uma instância do MySQL (alimentado com os dados das tabelas processuais unificadas e outras tabelas de apoio) e um backend com uma interface REST com alguns endpoints implementados. A intenção é que esse backend responda da mesma forma que a solução em produção que funciona em <https://www.cnj.jus.br/modelo-de-transferencia-de-dados/>.

Por padrão, o backend REST subirá na porta 8080, e o MySQL subirá na porta 3306. Caso não seja possível usar essas portas no seu ambiente operacional, basta alterar o arquivo docker-compose.yaml nos pontos marcados pelas setas vermelhas:

```
1 version: '3.7'
2 services:
3
4   rn-mysql:
5     container_name: rn-mysql
6     image: mysql/mysql-server:5.7
7     environment:
8       MYSQL_ROOT_USER: root
9       MYSQL_ROOT_PASSWORD: 12345
10      MYSQL_ROOT_HOST: '%'
11      MYSQL_DATABASE: corporativo
12     expose:
13       - 3306
14     volumes:
15       - ./sql_create:/docker-entrypoint-initdb.d
16     restart: always
17     networks:
18       - banco
19     healthcheck:
20       test: "/usr/bin/mysql --user=root --password=12345 --execute \"SHOW DATABASES;\""
21       interval: 2s
22       timeout: 20s
23       retries: 10
24
25   app:
26     restart: always
27     build:
28       context: ./
29       dockerfile: Dockerfile
30     working_dir: /app
31     expose:
32       - "8080"
33     ports:
34       - "8080:8080"
35     networks:
36       - banco
37     links:
38       - rn-mysql
```

A diretiva **ports** faz um “port forwarding”, e dessa forma realiza mapeamento de uma porta externa com uma porta interna do container.

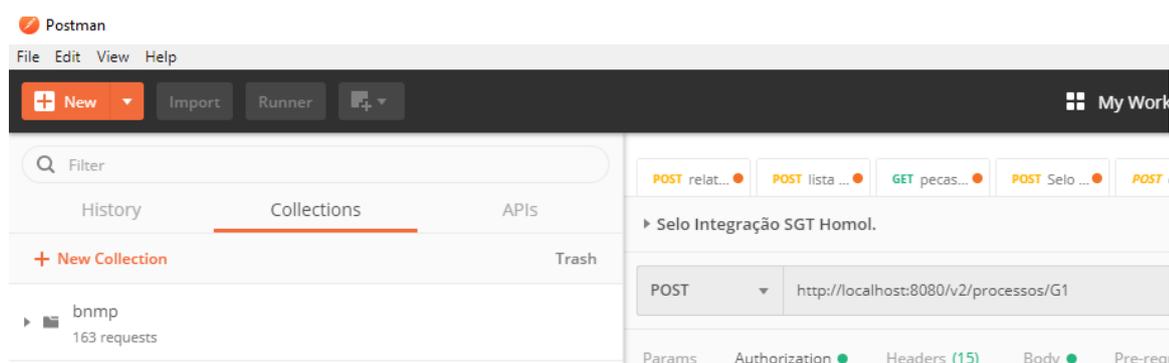
Após alterar para a porta desejada, basta salvar o arquivo e executar novamente o comando *docker-compose up*.

1.2. Uso do Validador

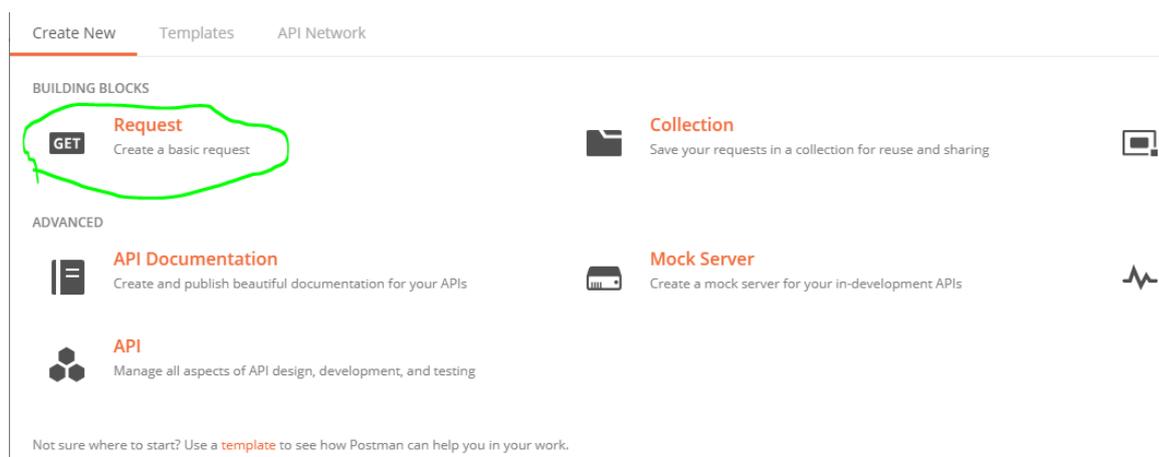
O validador roda como um serviço na máquina, na porta 8080. Pra simular o uso dessa aplicação, faremos uso do Postman (<https://www.postman.com/>).

1.3. Fazendo uma Requisição – Simulando um Envio de XML

Ao entrar no Postman, clique no botão “New”, como mostra a tela a seguir.



Depois, clique em “Request”:



Dê um nome para a requisição:

SAVE REQUEST

Requests in Postman are saved in collections (a group of requests).
[Learn more about creating collections](#)

Request name

Teste Valida XML

Request description (Optional)

Make things easier for your teammates with a complete request description.

Descriptions support [Markdown](#)

Na próxima tela, escolha o método HTTP como POST, e a URL, como está apresentado a seguir (supondo que você esteja rodando o Postman na mesma máquina que roda o Validador):

Teste Valida XML

POST http://localhost:8080/v2/processos/G1

Send

Params Authorization Headers Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Response

Lembrando que o parâmetro G1, que representa o Grau, pode ser substituído e assumir os valores seguintes:

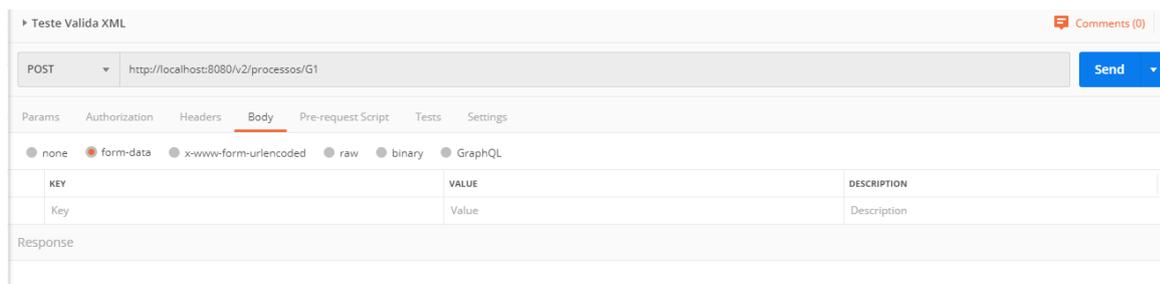
- **SUP** para Tribunais Superiores;
- **G2** para 2º grau;
- **G1** para 1º grau (justiça comum);
- **TR** para Turmas Recursais;
- **JE** para Juizados especiais;

Modelo de Transferência de Dados

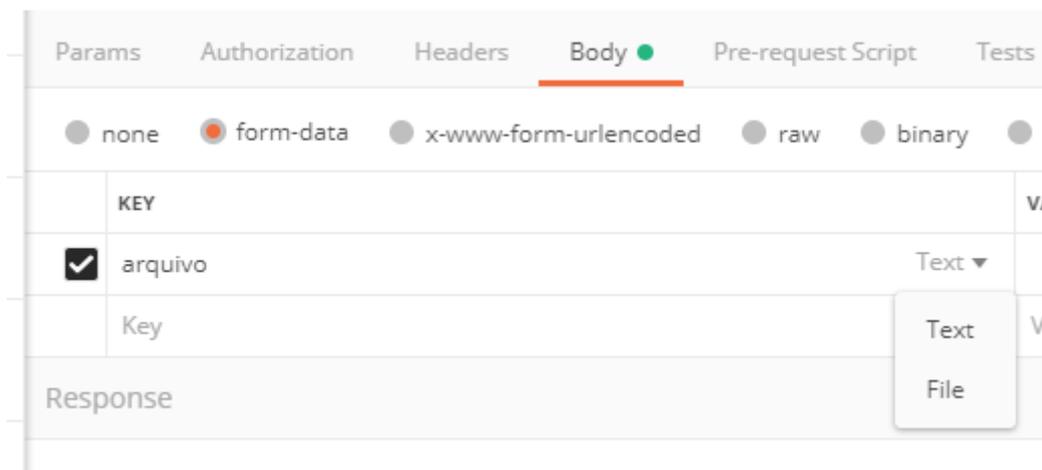
19

- **TRU** para Turmas Regionais de Uniformização;
- **TNU** para Turmas Nacionais de Uniformização.

Vá para a aba “Body” e selecione o modo “form-data”:



Depois, acrescente um campo para carregar o arquivo XML. Para isso, será necessário dar um nome qualquer a essa chave (key) e selecionar o tipo do dado, que nesse caso deverá ser “File”:



Selecione o arquivo na coluna “Value” (o arquivo, obviamente, deve estar aderente ao XSD do Modelo de Transferência de Dados) e depois clique no botão “Send”, conforme figura abaixo.

Modelo de Transferência de Dados

20

Teste Valida XML

POST http://localhost:8080/v2/processos/G1

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> arquivo	TJMS_G2_20200228.xml X	
Key	Value	Description

Body Cookies (1) Headers (5) Test Results Status: 200 OK Time: 8.38s Size: 232 B Save

```

1 |
2 |   "status": "SUCESSO",
3 |   "protocolo": "TJMS78726202003091583790663756"
4 |

```

O mesmo resultado seria obtido caso a requisição apontasse para a solução em produção, uma vez que elas implementam o *endpoint* da mesma maneira.

1.4. Operação de Validação

Com essa operação, é possível simular um envio e obter como resultado um conteúdo JSON com o XML processado, incluindo as validações com os dados tabelados em banco e os campos acrescentados por meio do que chamamos de “Rotinas de enriquecimento de dados”. Esse mesmo processo de enriquecimento é usado na solução do CNJ no processamento dos dados XML, e esses campos servem para diversos fins, como calcular a quantidade de movimentos com códigos não aderentes às TPUs (Tabelas Processuais Unificadas), por exemplo.

O endpoint <http://localhost:8080/v2/valida> irá realizar essa transformação e retornar o JSON enriquecido:

Teste Valida XML

POST http://localhost:8080/v2/valida/

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> arquivo	TJMS_G2_20200228.xml X	
Key	Value	Description

Body Cookies (1) Headers (5) Test Results Statu

Ao clicar em “**Send**”, o resultado será:

Modelo de Transferência de Dados

```
1
2 "processos": [
3   {
4     "protocoloRN": "TJMS42666202003091583791362197",
5     "dadosBasicos": {
6       "polo": [
7         {
8           "parte": [
9             {
10              "pessoa": {
11                "outroNome": [],
12                "documento": [
13                  {
14                    "codigoDocumento": "00104875732",
15                    "emissorDocumento": "SSP",
16                    "tipoDocumento": "NB",
17                    "nome": null
18                  }
19                ]
20              },
21              "endereco": [
22                {
23                  "logradouro": "Avenida Afonso Pena",
24                  "numero": "3297",
25                  "complemento": null,
26                  "bairro": "Centro",
27                  "cidade": "Campo Grande",
28                  "codCidade": null,
29                  "estado": "MS",
30                  "pais": "BR",
31                  "cep": "79002072"
32                }
33              ]
34            }
35          ]
36        }
37      ]
38    }
39  ]
40 }
```