NF_V3

Fernando H. Crozetta

27 de junho de 2017

Resumo

Com a reescrita dos programas de comunicação com webservices do governo, foram alteradas as estruturas do projeto.

Por conta das grandes alterações, este documento serve como um guia para o desenvolvedor que irá realizar a manutenção do fonte.

Sumário

1 Conceito

A partir de versões anteriores de comunicação com webservices e diversos programas diferentes, que em essência realizam a mesma tarefa, foi pensada uma solução que pudesse atender a todas as chamadas de webservices do governo brasileiro, que utilizem o SOAP para comunicação.

Partindo do princípio de escrever uma vez, para que funcione em diversos programas, as rotinas foram modularizadas, e separadas em categorias, de acordo com sua natureza. Algumas permitem a chamada via linha de comando, outras são funções a serem chamadas apenas pelas interfaces, etc...

A partir da estrutura de diretórios de arquivos de configuração de clientes, foi redesenhada a forma de organizar estes arquivos. Foi realizada uma separação entre dados e programas, permitindo uma fácil manutenção e atualização em clientes.

A partir desta versão, todo e qualquer caminho precisa ser parametrizado, permitindo a mudança de diretórios sempre que for necessário.[1].

Após a separação entre dados e programas; e separação de cada tipo de programa dentro da plataforma, a organização fica simples para leitura.

O conjunto de programas que realizam as tarefas de comunicação, junto com suas ferramentas, estrutura de diretórios, arquivos temporários,etc. Serão chamados neste documento de *plataforma*, para melhor entendimento.

O sistema foi criado para ser auxiliar e secundário. Ou seja, ele apenas faz a troca de mensagens entre o webservice, e um outro programa. Neste documento, o programa chamador também será referenciado como *erp*,

2 Estruturação de Diretórios

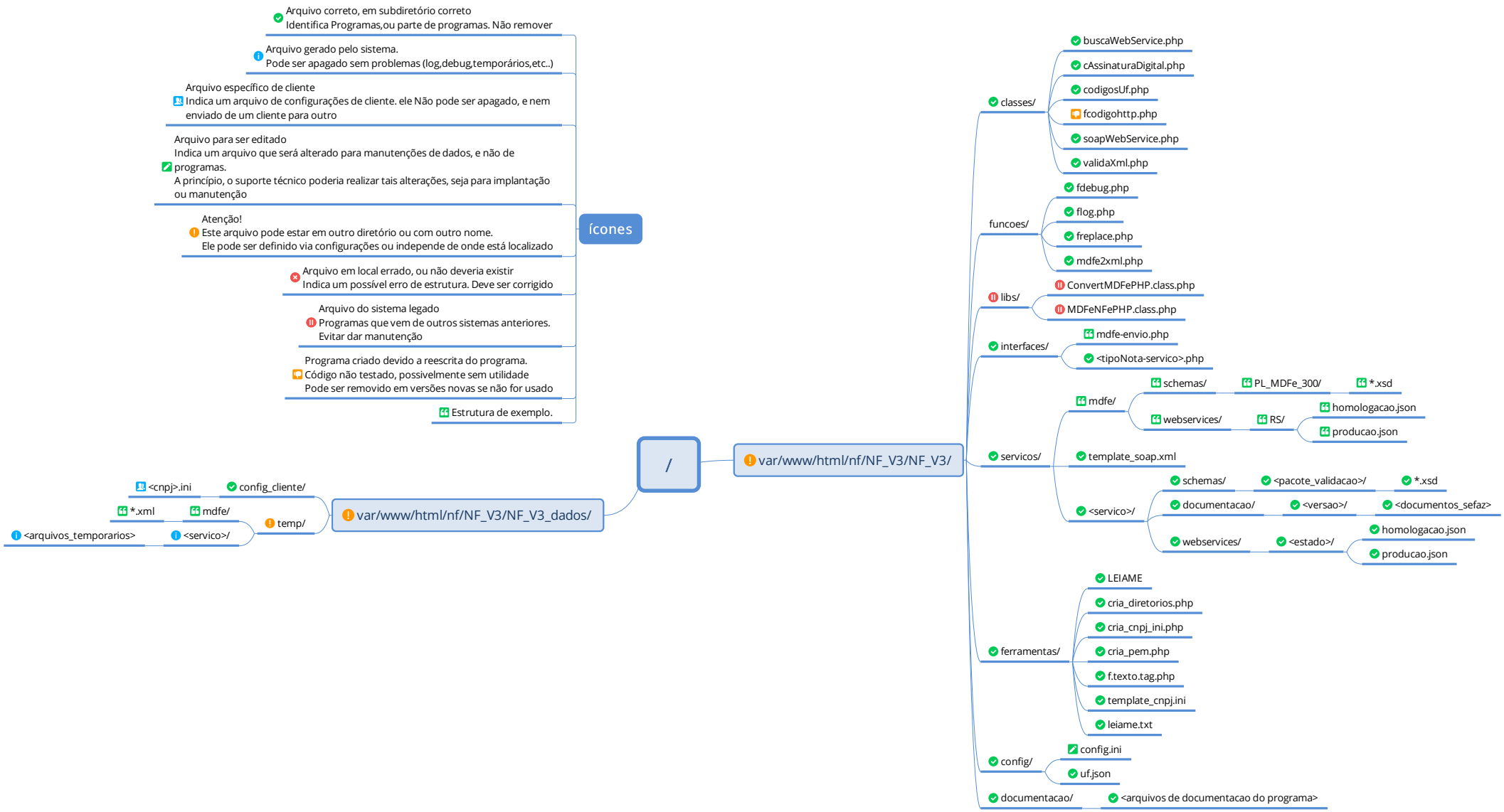
Por padrão, existem apenas dois diretórios principais:

1. NF_V3
2. NF_V3_dados

Sendo o primeiro para programas e o segundo para os dados dos clientes

[1] Esta alteração também permite a personalização de acordo com a infra estrutura do cliente

NF_V3


/
var/www/html/nf/NF_V3/NF_V3/
classes/
buscaWebService.php
cAssinaturaDigital.php
codigosUf.php
fcodigohttp.php
soapWebService.php
validaXml.php
funcoes/
fdebug.php
flog.php
freplace.php
mdfe2xml.php
libs/
ConvertMDFePHP.class.php
MDFeNFePHP.class.php
interfaces/
mdfe-envio.php
<tipoNota-servico>.php
servicos/
mdfe/
schemas/
PL_MDFe_300/
*.xsd
webservices/
RS/
homologacao.json
producao.json
template_soap.xml
<servico>/
schemas/
<pacote_validacao>/
*.xsd
documentacao/
<versao>/
<documentos_sefaz>
webservices/
<estado>/
homologacao.json
producao.json
ferramentas/
LEIAME
cria_diretorios.php
cria_cnpj_ini.php
cria_pem.php
f.texto.tag.php
template_cnpj.ini
leiame.txt
config/
config.ini
uf.json
documentacao/
<arquivos de documentacao do programa>
var/www/html/nf/NF_V3/NF_V3_dados/
config_cliente/
<cnpj>.ini
mdfe/
*.xml
temp/
<servico>/
<arquivos_temporarios>
Ícones
Arquivo correto, em subdiretório correto
Identifica Programas,ou parte de programas. Não remover
Arquivo gerado pelo sistema.
Pode ser apagado sem problemas (log,debug,temporários,etc..)
Arquivo específico de cliente
Indica um arquivo de configurações de cliente. ele Não pode ser apagado, e nem enviado de um cliente para outro
Arquivo para ser editado
Indica um arquivo que será alterado para manutenções de dados, e não de programas.
A princípio, o suporte técnico poderia realizar tais alterações, seja para implantação ou manutenção
Atenção!
Este arquivo pode estar em outro diretório ou com outro nome.
Ele pode ser definido via configurações ou independe de onde está localizado
Arquivo em local errado, ou não deveria existir
Indica um possível erro de estrutura. Deve ser corrigido
Arquivo do sistema legado
Programas que vem de outros sistemas anteriores.
Evitar dar manutenção
Programa criado devido a reescrita do programa.
Código não testado, possivelmente sem utilidade
Pode ser removido em versões novas se não for usado
Estrutura de exemplo.

2.1 NF_V3_dados

Dentro do diretório de dados do cliente temos, por padrão, um diretório *config_cliente*, e um diretório *temp*, que servem para armazenas os dados persistentes e temporários de cada cliente.

2.1.1 config_cliente

Este diretório deve possui apenas arquivos de configuração de cada cnpj de empresa. Apesar do caminho padrão poder ser alterado, recomenda-se que sempre fique dentro do *NF_V3_dados*. cada arquivo deve possui a seguinte nomenclatura:

```
<cnpj>.ini
EX:12345678901234.ini
```

Dentro do arquivo, devem estar presentes, obrigatoriamente, os seguintes blocos de dados[2]:

```
[empresa]
cnpj='12345678901234'

[certificado]
caminho_certificado="/var/www/html/nf/nfse/certificados/"
arquivo_certificado="/var/www/html/nf/nfse/certificados/12345678901234.pfx"
raiz_certificado='/var/www/html/nf/nfse/certificados/12345678901234'
senha="senha123"

[curl]
conexao_segura=S
verbose=0
porta=443
header=1
sslversion=3
ssl_verifyhost=0
ssl_verifypeer=0
connecttimeout=
timeout=
maxredirs=
followlocation=0
post=1
returntransfer=1
fresh_connect=0

[proxy]
servidor=
porta=00000
usuario=
```

[2] os dados corresponde a uma empresa de cnpj 12345678901234, cuja senha do certificado é senha123

```
senha=


[outros_dados]
cnpj="ERRO! Load de arquivo errado"
```

A princípio não há uma ordem obrigatória para os blocos de informação, entretanto, o bloco *[outros_dados]* deve ficar por último, pois caso o arquivo seja lido de forma errada, o cnpj trará uma mensagem de erro para o desenvolvedor .

Além dos dados obrigatórios, existem também os dados específicos de cada tipo de nota. Estes dados devem ser criados pelo desenvolvedor da interface da *plataforma* para o *erp* .

Abaixo, um exemplo de um arquivo já preenchido com os dados de dois serviços:

```
[empresa]
cnpj="12345678901234"

[certificado]
caminho_certificado="/var/www/html/nf/nfse/certificados/"
arquivo_certificado="/var/www/html/nf/nfse/certificados/12345678901234.pfx"
raiz_certificado='/var/www/html/nf/nfse/certificados/12345678901234'
senha="senha123"

[mdfe]
versao="3.00"
pacote="PL_MDFe_300"
versao_soap="2"
dir_retorno='/var/www/html/nf/NF_V3/NF_V3_dados/temp/mdfe/'

[nfse]
ip="e-gov.betha.com.br"
ambiente=2
url_producao="http://e-gov.betha.com.br/e-nota-contribuinte-ws/nfseWS?wsdl"
url_homologacao="http://e-gov.betha.com.br/e-nota-contribuinte-test-ws/nfseWS?w

[dnfe]
versao="1.01"
pacote="PL_NFeDistDFe_102"
versao_soap="2"
dir_retorno='/var/www/html/nf/NF_V3/NF_V3_dados/temp/dnfe/'
dir_procNFe='/user/nfe/12345678901234/DEST/'
dir_resEvento='/var/www/html/nf/NF_V3/NF_V3_dados/temp/dnfe/resEvento/'
dir_resNFe='/var/www/html/nf/NF_V3/NF_V3_dados/temp/dnfe/resNFe/'
dir_procEventoNFe='/var/www/html/nf/NF_V3/NF_V3_dados/temp/dnfe/procEventoNFe/'
dir_naoImplementado='/var/www/html/nf/NF_V3/NF_V3_dados/temp/dnfe/naoImplementa
dir_saida_cobol='/user/nfe/12345678901234/CaixaSaida/Sefaz/'

[curl]
conexao_segura=S
```

```
verbose=0
porta=443
header=1
sslversion=3
ssl_verifyhost=0
ssl_verifypeer=0
connecttimeout=
timeout=
maxredirs=
followlocation=0
post=1
returntransfer=1
fresh_connect=0

[proxy]
servidor=
porta=00000
usuario=
senha=


[outros_dados]
cnpj="ERRO! Load de arquivo errado"
```

2.1.2 temp

O diretório temp identifica o diretório de trabalho da *plataforma* . Este diretório *NÃO* pode ficar dentro do diretório de programas, pois corre o risco de ser atualizado em clientes.

Dentro deste diretório se encontram subdiretórios referentes ao servido de nota buscado: *temp/mdfe* e *temp/dnfe*, por exemplo.

Tudo o que estiver sob o diretório temp pode ser apagado, sem exceção. Por este motivo, tudo o que for necessário ser salvo, deve ser colocado em parâmetros dentro do config do cliente pelo desenvolvedor .

2.2 NF_V3

Este diretório possui os programas e binários para a execução das rotinas de comunicação entre os webservices e o *erp* .

Os arquivos devem seguir os padrões de desenvolvimento definidas no diretório de documentação, e de acordo com os arquivos LEIAME, dentro do próprio diretório. Nos comentários do cabeçalho do programa, deve conter :

1. identificação do autor
2. data
3. descrição do programa
4. modo de uso (caso seja chamado via linha de comando)

Os programas foram separados em diretórios, que identificam sua função, e modos de uso:

1. interfaces
2. classes
3. funcoes
4. ferramentas
5. libs
6. servicos
7. config
8. documentacao

2.2.1 interfaces

Neste diretório, e apenas neste diretório, se localizam os programas a serem chamados pelo *erp* . Até o presente momento, é preciso realizar o comando *cd /caminho/para/interfaces/* e depois executar o comando php[3].

A nomenclatura de todos os programas aqui criados é:

<tipo de nota>–<servico>.php
EX: mdfe–consulta.php

Essa nomenclatura facilita para a localização dos serviços de cada tipo de nota

2.2.2 classes

Neste diretório devem ficar apenas as classes que são chamadas pelas interfaces e ferramentas. Nenhum programa deste diretório deve ser chamado por linha de comando.

2.2.3 funcoes

Neste diretório, os programas devem possuir apenas uma função dentro dele. Estes arquivos adicionam uma funcionalidade simples aos programas, servindo como apoio para o desenvolvedor que esta realizando a manutenção no código. Ele pode ser incluído nas interfaces com o comando *require_once()*

2.2.4 ferramentas

Os arquivos que estão aqui, não são necessariamente para uso interno da *plataforma* , mas permitem que o desenvolvedor e suporte técnico possam executar algumas tarefas que podem ser necessárias. Elas são funções complementares, e tem como objetivo auxiliar primariamente o suporte técnico a resolver possíveis problemas.

[3]Este é um bug conehcido e está na lista para correções

2.2.5 libs

Os programas deste diretório são programas legado que foram alterados apenas para que funcionem com diretórios locais. Nenhuma alteração de código foi realizada.

O desenvolvedor deve evitar realizar alterações nos programas deste diretório. Em teoria, este diretório ficará vazio com o passar do tempo.

2.2.6 servicos

Dentro deste diretório está localizada a estrutura de dados referentes aos webservices. Dentro dele estão os tipos de nota, separados por diretório; seguido de subdiretórios de schemas de validação, documentação proveniente da sefaz [4], e webservices. Este último, por sua vez é dividido em estados, e dentro de cada diretório de estado, existem dois arquivos:

1. homologacao.json
2. producao.json

Estes dois arquivos contém a estrutura necessária para envio de dados e montagem de xml

2.2.7 config

Neste diretório se encontram os arquivos de configuração do programa. O primeiro arquivo é uf.json, que possui os código de cada estado brasileiro e seu código IBGE. O segundo é config.ini, que possui os direcionamentos de caminhos para log, debug, servicos dos webservices, temp e diretório de dados(NF_V3_dados). Estes arquivos só precisam ser alterados se for necessária uma alteração grande nas estruturas do *plataforma* .

2.2.8 documentacao

Este diretório deve possuir a documentação referente a *plataforma* , sem conter manuais da sefaz e outros arquivos que não dizem respeito a construção e manutenção desta *plataforma* .

3 Desenvolvimento: classes/

3.1 buscaWebService.php

Instanciação de classe:

```
require_once("../classes/buscaWebService.php");
$var = BuscaWebService(($uf,$tipo_nota,$tipo_ambiente);
```

[4] Caso seja necessário adicionar documentação do *erp* , separar os diretórios.

3.1.1 construtor

```
function __construct($uf,$tipo_nota,$tipo_ambiente='homologacao')
```

- uf: Sigla do estado do webservice.Maiúsculo.
- tipo_nota: Tipo de nota a ser usada.EX:mdfe,dnfe,nfe,etc.Minúsculo.
- tipo_ambiente: string:homologacao/producao. homologacao, caso não seja passado o parâmetro

3.1.2 buscarServico

```
$var->buscarServico($nome_servico,$versao);
```

- nome_servico: Servico do tipo de nota, definido pelo desenvolvedor [5].
- versao: Versão do serviço buscado, definido pelo desenvolvedor no config do cliente

[5] Este nome é a chave dentro do arquivo de webservices