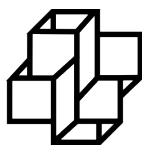


---

# Flowman

## Manual da Ferramenta

---



---

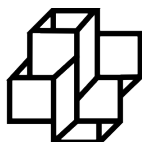
Copyright © 2014 Christian M. S. Mury

<http://comcidis.lncc.br>

Licensed under the Creative Commons Attribution 4.0 International License. You may not use this file except in compliance with the License. You may obtain a copy of the License at <http://creativecommons.org/licenses/by/4.0/>. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.



**Criado:** September 30, 2014  
**Atualizado:** 6 de Novembro de 2014



ComCiDis - Computação Científica Distribuída

---

# Conteúdo

<b>1</b>	<b>Apresentação</b>	<b>4</b>	3.7	upload_file.php . . . . .	9
1.1	Requisitos Mínimos . . . . .	4	3.8	css/ . . . . .	9
<b>2</b>	<b>Instalação</b>	<b>5</b>	3.9	font/ . . . . .	9
2.1	Bibliotecas . . . . .	5	3.10	images/ . . . . .	9
2.2	Amphitrite . . . . .	5	3.10.1	images/nodes/ . . . . .	9
2.3	Poseidon . . . . .	5	3.10.2	images/icon/ . . . . .	10
2.4	Criar Container . . . . .	6	3.11	js/ . . . . .	10
2.5	Pasta Compartilhada . . . . .	7	3.11.1	js/filesaver/ . . . . .	10
<b>3</b>	<b>Amphitrite</b>	<b>8</b>	3.11.2	js/joint/ . . . . .	10
3.1	index.php . . . . .	8	3.11.3	js/amphitrite.js . . . . .	10
3.2	post.php . . . . .	8	3.11.4	js/customNodes.js . . . . .	11
3.3	file.php . . . . .	8	<b>4</b>	<b>Poseidon</b>	<b>12</b>
3.4	delete.php . . . . .	8	4.1	Poseidon.java . . . . .	12
3.5	deleteAll.php . . . . .	9	4.2	ActivityComparator.java . . . . .	12
3.6	uploadFile.php . . . . .	9	4.3	Activity.java . . . . .	13
			<b>5</b>	<b>Nav bars</b>	<b>14</b>
			5.1	index nav bar . . . . .	14
			5.2	file nav bar . . . . .	15
			5.3	upload nav bar . . . . .	16

# Capítulo 1

## Apresentação

Este documento apresenta como configurar, manter, atualizar e administrar a ferramenta.

### 1.1 Requisitos Mínimos

Para prosseguir com a configuração, há a necessidade de alguns requisitos prévios. A lista abaixo apresenta quais são eles:

- Servidor deve estar com o Sistema Operacional LINUX;
- A biblioteca LXC deve estar instalada;
- Apache2 e PHP devem estar configurados; e
- Java deve estar instalado;

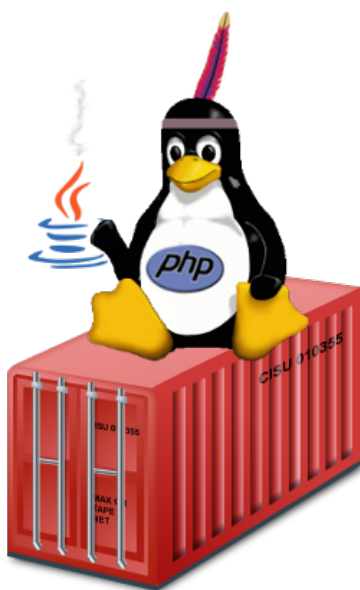


Figura 1.1: Tecnologias necessárias

# Capítulo 2

## Instalação

Esse manual ensinará como instalar o Flowman e seus requisitos no sistema operacional Ubuntu. Para este tutorial foi utilizado a versão 14.04.

### 2.1 Bibliotecas

Para poder utilizar a tecnologia de *containers* deve instalar a biblioteca do Linux Containers(LXC). Para isso instale o pacote LXC do repositório:

```
$ sudo apt-get install lxc
```

Outra necessidade do Flowman é o Apache2 e PHP para a pagina e comunicação com núcleo.

```
$ sudo apt-get install apache2 php5
```

O ultimo requisito é ter o Java instalado, para isso instale o OpenJDK do repositório ou o Java da Oracle. O OpenJDK já vem pré-instalado nas distribuições Ubuntu.

Essas são as bibliotecas e linguagens necessárias para o funcionamento do Flowman.

### 2.2 Amphitrite

Amphitrite é a pagina Web do Flowman, para instalar-la, basta apenas copiar seus arquivos para a pasta do Apache2, usualmente situada em `"/var/www/html/ "`.

### 2.3 Poseidon

Poseidon é o núcleo do Flowman, é ele quem de fato particiona o XML e executa os *containers*. Flowman é feito em Java e suas classes compiladas devem estar na pasta `/home/user/bin/`.

## 2.4 Criar Container

Para criar um *container* com LXC:

```
$ sudo lxc-create -t FLAVOR -n NOME
```

-t FLAVOR: Significa o sistema operacional a ser criado. Por exemplo “-t ubuntu”.

-n NOME: Nome do *container* a ser criado.

Para iniciar o *container* use:

```
$ sudo lxc-start -n NOME -d
```

O parâmetro “-d” inicia ele em background, não precisando o console permanecer aberto.

Esses são os passos para criação e inicialização de um *container* com LXC. Após criado e inicializado em background o *container* precisa de uma pasta para ser compartilhada e um *script* para ser inicializado pelo Poseidon.

Para abrir o *container* use o comando:

```
$ sudo lxc-console -n NOME
```

Agora você está dentro do *container*, na primeira inicialização ele pedirá login e senha, o login e senha padrão do flavor ubuntu é o próprio nome do sistema.

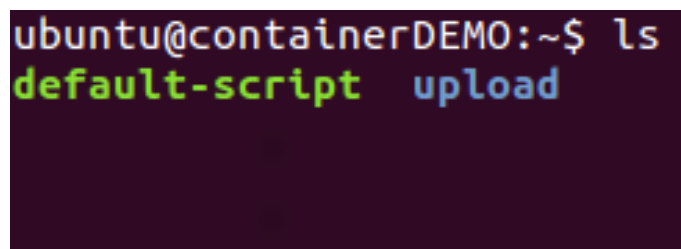
Login : ubuntu

Senha : ubuntu

O Poseidon precisa de mais duas coisas para poder utilizar o *container*, a primeira é uma pasta chamada upload no próprio home do sistema. Para isso use o comando:

```
$ mkdir /home/ubuntu/upload
```

O último item necessário é um script que será executado pelo Poseidon, esse script deve estar na pasta do home junto com a pasta de upload. O script deve conter o nome: “default-script”



```
ubuntu@containerDEMO:~$ ls
default-script  upload
```

Figura 2.1: Arquivos que devem conter no *container*

Para sair de um *container* e manter-lo executando, utilize ctrl+a q. (ctrl+a, solte, q)

## 2.5 Pasta Compartilhada

O sistema Host deve conter uma pasta compartilhada chamada "upload". Essa pasta deve ser compartilhada com todos os *containers*, e listada pelo PHP. Para compartilhar a pasta precisa criar o arquivo `fstab` do *container*, na qual normalmente fica em:

```
$ sudo vim /var/lib/lxc/NOME/fstab
```

Onde NOME é o nome do *container* a ser compartilhado. Criando esse arquivo precisa adicionar uma linha de configuração a ele:

```
        /home/host/upload /var/lib/lxc/division/rootfs/home/ubuntu/upload /  
none    bind      0          0
```

Na qual o primeiro argumento, no exemplo `/home/host/upload`, é a pasta do host a ser compartilhada.

Segundo argumento é a pasta do *container* a ser compartilhada, essa pasta é usada por padrão pelo Poseidon. (`/var/lib/lxc/division/rootfs/home/ubuntu/upload/`).

# Capítulo 3

## Amphitrite

Amphitrite é toda a página Web do Flowman, toda a interação com o usuário, desde a criação do *workflow* até *upload* de arquivos. Amphitrite esta dividido nos seguintes arquivos:

### 3.1 index.php

Esse arquivo é a página principal do Flowman, nela é onde esta presente o quadro de desenhos. A única função dessa página é importar outras paginas e *javascripts*.

### 3.2 post.php

Essa página é executada quando o usuário clica em *run* na página principal, essa página particiona o XML, grava ele na pasta *upload* e executa o Poseidon com o XML criado. Após a execução a pagina é carregada mostrando um botão de *download* na qual o usuário pode usar para ver o resultado obtido.

### 3.3 file.php

Essa página mostra ao usuário todos os arquivos presentes na pasta upload. também estão presentes a função de deletar um determinado arquivo pela pagina delete.php ou deletar todos os arquivos na pasta pela pagina deleteAll.php.

### 3.4 delete.php

Pagina que deleta determinado arquivo selecionado na página file.php.



## 3.5 deleteAll.php

Pagina que deleta todos os arquivos presentes na pasta upload. Essa função é chamada por um botão presente na pagina file.php.

## 3.6 uploadFile.php

Nessa pagina é onde o usuário faz o *upload* de seus arquivos, chamando a pagina upload\_file.php.

## 3.7 upload\_file.php

Pagina responsável por realizar o *upload* do arquivo requisitado pela pagina uploadFile.php.

## 3.8 css/

Nessa pasta estão presentes todos os css necessários para o funcionamento da pagina; bootstrap, jointjs, fontawesome, sbadmin.

## 3.9 font/

Nessa pasta estão presentes todos os arquivos para o funcionamento correto do bootstrap e fontawesome.

## 3.10 images/

Pasta de imagens do Flowman.

### 3.10.1 images/nodes/

Nessa pasta estão presentes todas as imagens dos módulos do *workflow*.

### 3.10.2 images/icon/

Nessa pasta está presente o ícone da página.

## 3.11 js/

Pasta contendo todos os *javascripts* do Flowman.

### 3.11.1 js/filesaver/

Nessa pasta está presente o *javascript* que auxilia o download de arquivos do Amphitrite.

### 3.11.2 js/joint/

Nessa pasta estão presentes todas as *javascripts* necessários para o jointjs funcionar corretamente.

### 3.11.3 js/amphitrite.js

Esse *javascript* é todo o núcleo de funções *javascript* do Amphitrite. Dentre as funções presentes são elas:

- Creating drawing paper: Função que cria um papel de desenhos do jointjs para desenhar o diagrama de *workflows*.
- Right button menu: Função que cria a modal de botão direito dos módulos do *workflow*.
- Test node connections: Função que testa as conexões dos módulos do *workflow*, usando bolinhas verdes para ver o fluxo de dados do *workflow*.
- Show Download Modal: Mostra a modal de download para XML ou arquivo da ferramenta .flow
- Show Upload Modal: Mostra a modal de *upload* para o usuário poder fazer *upload* de seu *workflow* previamente criado no formato .flow, usando a função Upload Diagram.
- Insert File: Função que serve para inserir o caminho de um arquivo na activity de um módulo file.
- Upload Diagram: Função usada para fazer o *upload* do diagrama de *workflow* previamente criado pelo usuário.
- Post XML: Função usada para enviar o esquema de *workflow* para a página que particionará o XML.
- Export JSON: Função usada para gerar o arquivo .flow e o download para o usuário.
- Creating XML: Função utilizada para criar o XML de seu esquema e o download para o usuário.
- Making links dont pass each other: Função usada para as setas não ocuparem o mesmo espaço na criação de um *workflow*.

### 3.11.4 js/customNodes.js

Esse *script* é o responsável por criar as funções usadas para chamar os módulos para a criação do *workflow*.

```
function nomeDeFuncao() {
    var node = new joint.shapes.basic.Image({
        position: {
            x: 20,
            y: 10
        },
        size: {
            width: height / 30 + width / 30,
            height: height / 30 + width / 30
        },
        attrs: {
            text: {
                text: 'Texto que representa o modulo',
                magnet: true
            },
            image: {
                'xlink:href': 'images/nodes/imagemDoModulo.png',
                width: 50,
                height: 50
            },
            '.': {
                magnet: false
            },
        },
        activity: '',
        name: 'containerDEMO'
    });
    graph.addCell(nomeDeFuncao);
}
```

position: posição na qual o módulo é criado

size: tamanho da imagem do módulo, atualmente dinamicamente redimensionado com o tamanho da tela.

attrs: Atributos dos módulos.

text: texto que será mostrado ao usuário.

magnet é a função que deixa o módulo poder ser linkado por uma seta.

image: Local da imagem usada pelo módulo.

Dica: Usar sempre o magnet no text do módulo e não em sua imagem. Usando em sua imagem não é possível mover o modulo pelo papel.

activity: Parâmetro enviado para o *container*, normalmente inserido pelo usuário.

name: Nome do *container* a ser usado pelo módulo.

graph.addCell(node): adiciona o objeto ao papel.

# Capítulo 4

## Poseidon

Poseidon é o núcleo do Flowman, ele é o responsável por ler o XML e executar os *containers*. O Poseidon está dividido em 3 classes.

### 4.1 Poseidon.java

Essa é a classe principal do Poseidon, ela é a responsável por particionar o XML , ordenar as conexões do *workflow* usando a classe ActivityComparator.java e a inicializar e executar o *script* dentro do *container* usando a classe Activity.java 2

### 4.2 ActivityComparator.java

Essa classe é a responsável por ordenar as conexões do XML para serem eventualmente usadas no sistema de pesos para cada passo do *workflow*. Figura[4.1].

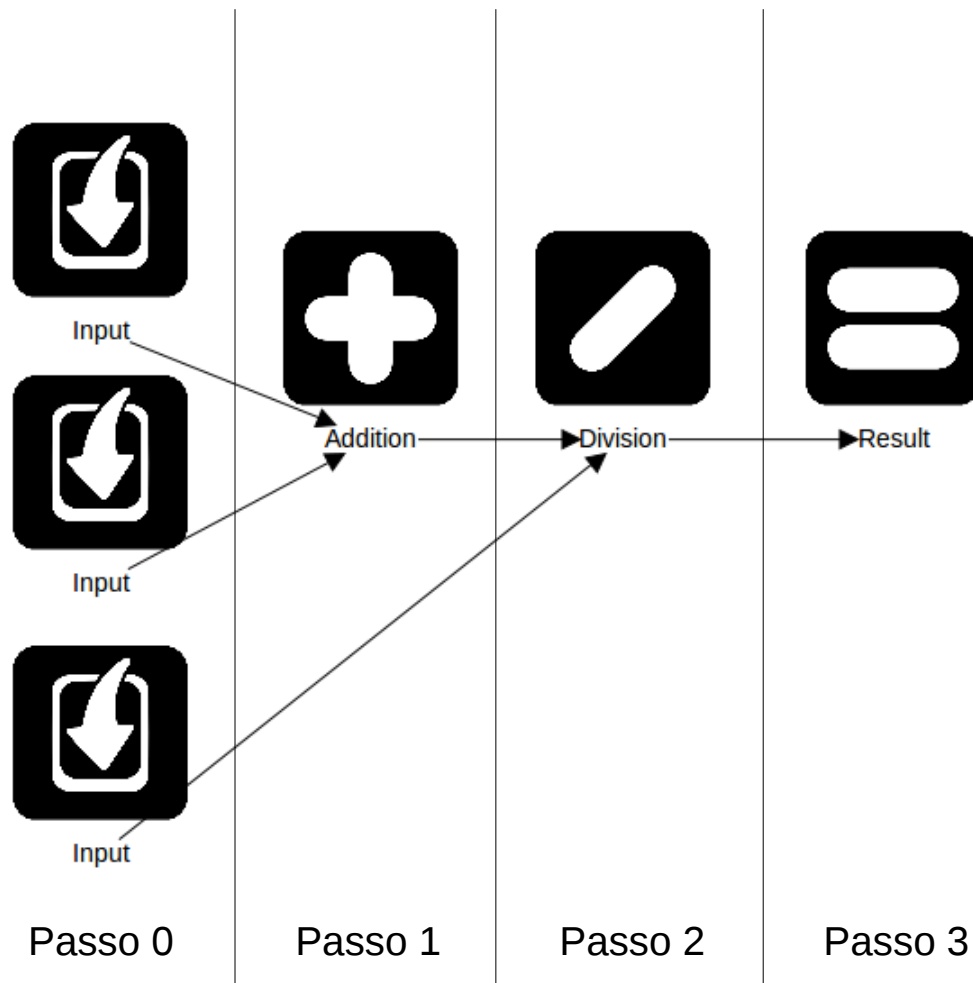


Figura 4.1: Sistema de passos

## 4.3 Activity.java

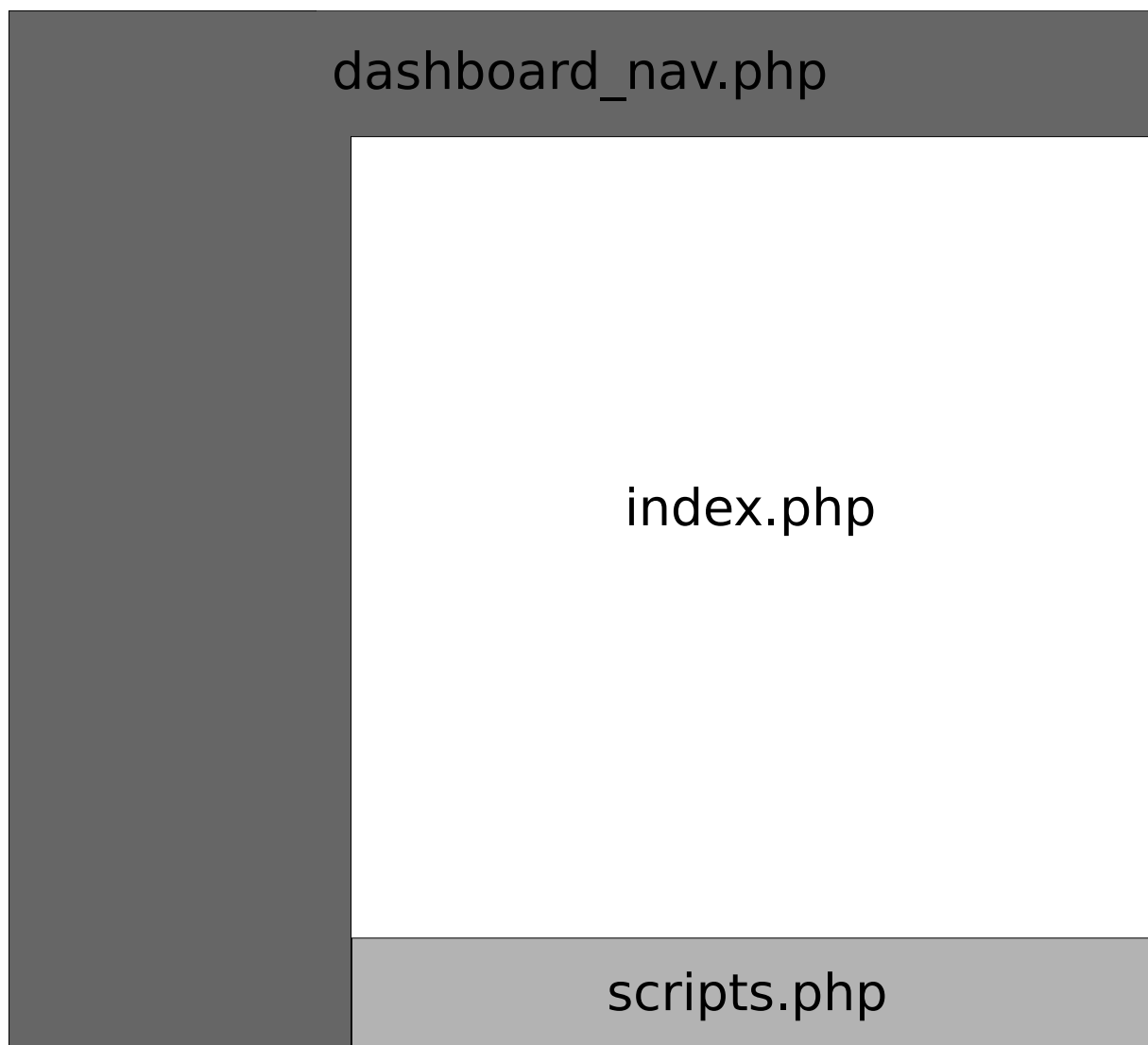
Essa classe é a responsável por criar o objeto da *activity* de cada módulo do *workflow*, ela também executa os *scripts* presentes dentro dos *containers* e grava sua saída.

# Capítulo 5

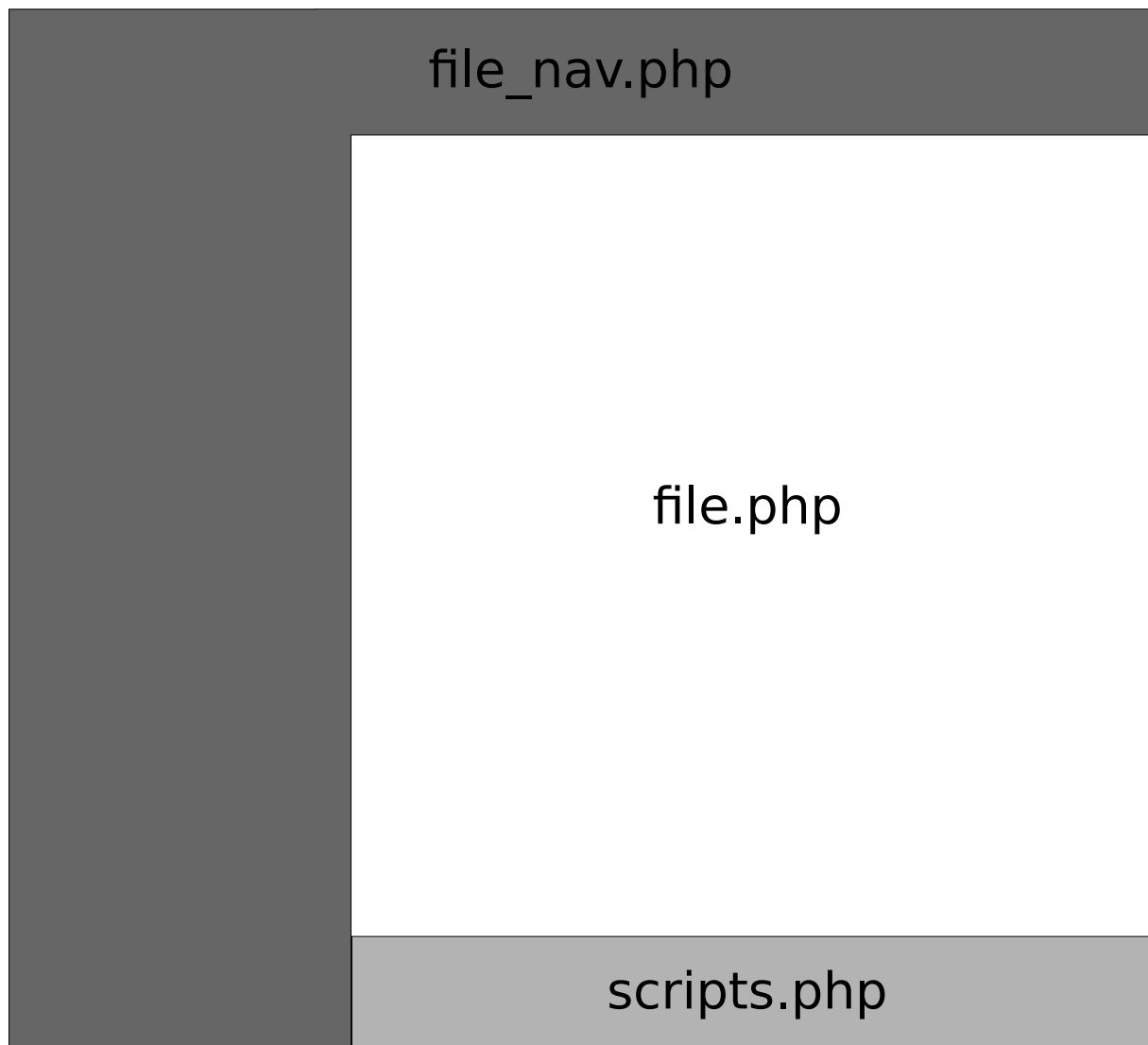
## Nav bars

Nesse capítulo será apresentada como estão arrumadas as *nav bars* das páginas do Amphitrite.

### 5.1 index nav bar



## 5.2 file nav bar



## 5.3 upload nav bar

