**Mapas Culturais > Deploy**

Neste guia faremos o deploy do Mapas Culturais utilizando o nginx + php-fpm em um sistema Ubuntu 18 Server, ou superior, recém instalado somente com o OpenSSH Server. O Banco de dados e a aplicação rodarão no mesmo servidor e usuário.

Não abordaremos as configurações de autenticação, seja com ID da Cultura, seja com Login Cidadão. Ao final do guia teremos a aplicação rodando com o método de autenticação Fake.

As linhas que começam com **mapas@server$** são executadas com o usuário criado para rodar a aplicação e as linhas que começam com **root@server#** são executadas com o usuário *root*.

**1. Softwares Requeridos**

Primeiro vamos instalar os pacotes necessários para o funcionamento do Mapas Culturais.

# Atualize os repositórios de referência de sua máquina

root@server# apt-get update

// Instale as dependências diversas

\*\*root@server# apt-get install git curl npm ruby2.3 ruby2.3-dev

// Atualizar referências para a versão de ruby 2.3

\*\*root@server# update-alternatives --install /usr/bin/ruby ruby /usr/bin/ruby2.3 10

\*\*root@server# update-alternatives --install /usr/bin/gem gem /usr/bin/gem2.5 10

// Instale a versão stable mais nova do nodejs

root@server# curl -sL https://deb.nodesource.com/setup\_4.x | sudo -E bash -

root@server# sudo apt-get install -y nodejs

// Instale o postgresql e postgis

root@server# apt-get install postgresql postgresql-contrib postgis postgresql-10-postgis-2.5 postgresql-10-postgis-2.5-scripts

// Inicialize o postgresql

/etc/init.d/postgresql start

// Instale o php, php-fpm e extensões do php utilizadas no sistema

root@server# apt-get install php7.2 php7.2-gd php7.2-cli php7.2-json php7.2-curl php7.2-pgsql php-apc php7.2-fpm imagemagick libmagickcore-dev libmagickwand-dev php-imagick

// Instale o nginx

root@server# apt-get install nginx

// Instale o gerenciador de dependências do PHP Composer

root@server# curl -sS https://getcomposer.org/installer | php

root@server# mv composer.phar /usr/local/bin/composer.phar

Também é importante ter o pacote zip instalado no seu servidor. Ele é usado para gerar o pacote com os anexos enviados nas inscrições. Caso ainda não tenha:

root@server# apt-get install zip

No Ubuntu o executável do NodeJS se chama *nodejs*, porém para o correto funcionamento das bibliotecas utilizadas, o executável deve se chamar *node*. Para isto criamos um link simbólico com o comando abaixo

root@server# update-alternatives --install /usr/bin/node node /usr/bin/nodejs 10

Instalando os minificadores de código Javascript e CSS: uglify-js, uglifycss e autoprefixer

root@server# npm install -g uglify-js2 uglifycss autoprefixer

root@server# update-alternatives --install /usr/bin/uglifyjs uglifyjs /usr/bin/uglifyjs2 10

Instalando o SASS, utilizado para compilar os arquivos CSS

root@server# gem install sass -v 3.4.22

**2. Clonando o Repositório**

Primeiro vamos criar o usuário que rodará a aplicação e que será proprietário do banco de dados, definindo sua home para */srv* e colocando-o no grupo *www-data*.

root@server# useradd -G www-data -d /srv/mapas -m mapas

Vamos clonar o repositório usando o usuário criando, então precisamos primeiro "logar" com este usuário.

root@server# su - mapas

Agora faça o clone do repositório.

mapas@server$ git clone https://github.com/mapasculturais/mapasculturais.git

E alterne para o branch v2 ou alguma tag de release, disponível em [https://github.com/mapasculturais/mapasculturais/releases](https://github.com/hacklabr/mapasculturais/releases). Se for uma instalação de teste, você pode pular esta etapa.

Utilizando o branch V2:

mapas@server$ cd mapasculturais

mapas@server$ git checkout v2

Utilizando um release (Ex: 5.0):

mapas@server$ cd mapasculturais

mapas@server$ git checkout 5.0

Agora vamos instalar as dependências de PHP utilizando o Composer.

mapas@server$ cd ~/mapasculturais/src/protected/

mapas@server$ composer.phar install

**3. Banco de Dados**

Vamos voltar ao usuário *root* para criar o banco de dados.

root@server# exit

mapas@server$

Primeiro vamos criar o usuário no banco de dados com o mesmo nome do usuário do sistema

root@server# sudo -u postgres psql -c "CREATE USER mapas"

Agora vamos criar a base de dados para a aplicação com o mesmo nome do usuário

root@server# sudo -u postgres createdb --owner mapas mapas

Criar as extensões necessárias no banco

root@server# sudo -u postgres psql -d mapas -c "CREATE EXTENSION postgis;"

root@server# sudo -u postgres psql -d mapas -c "CREATE EXTENSION unaccent;"

Volte a "logar" com o usuário criado e importar o esquema da base de dados

root@server# su - mapas

mapas@server$ psql -f mapasculturais/db/schema.sql

**4. Configurações de instalação**

Primeiro crie um arquivo de configuração copiando o arquivo de template de configuração. Este arquivo está preparado para funcionar com este guia, utilizando o método de autenticação Fake.

mapas@server$ cp mapasculturais/src/protected/application/conf/config.template.php mapasculturais/src/protected/application/conf/config.php

**Criando diretórios de log, files e estilo**

Como root, crie a pasta para os arquivos de log:

$ exit

root@server# mkdir /var/log/mapasculturais

root@server# chown mapas:www-data /var/log/mapasculturais

Com o usuário criado, crie a pasta para os assets, para os uploads e para os uploads privados (arquivos protegidos, como anexos de inscrições em oportunidades):

root@server# su - mapas

mapas@server$ mkdir mapasculturais/src/assets

mapas@server$ mkdir mapasculturais/src/files

mapas@server$ mkdir mapasculturais/private-files

**Configuração do nginx**

Precisamos criar o *virtual host* do nginx para a aplicação. Para isto crie, como root, o arquivo **/etc/nginx/sites-available/mapas.conf** com o conteúdo abaixo: (Acrescentar cabeçalhos de segurança e configuração do https)

server {

set $site\_name www.meudominio.gov.br;

listen \*:80;

listen 443 ssl;

server\_name www.meudominio.gov.br;

ssl\_certificate /etc/nginx/ssl/nginx.crt;

ssl\_certificate\_key /etc/nginx/ssl/nginx.key;

access\_log /var/log/mapasculturais/nginx.access.log;

error\_log /var/log/mapasculturais/nginx.error.log;

index index.php;

root /var/www/mapas/mapasculturais/src/;

location / {

try\_files $uri $uri/ /index.php?$args;

}

location ~ /files/.\*\.php$ {

deny all;

return 403;

}

location ~\* \.(js|css|png|jpg|jpeg|gif|ico|woff)$ {

expires 1w;

log\_not\_found off;

}

location ~ \.php$ {

try\_files $uri =404;

include fastcgi\_params;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

fastcgi\_pass unix:/var/run/php/php7.2-fpm-www.meudominio.gov.br.sock;

client\_max\_body\_size 10M;

}

location ~\* \.(eot|ttf|woff|woff2)$ {

add\_header Access-Control-Allow-Origin \*;

add\_header Content-Security-Policy "default-src 'self'";

add\_header Strict-Transport-Security "max-age=31536000; includeSubdomains" always;

add\_header X-Frame-Options "SAMEORIGIN";

add\_header X-Xss-Protection "1; mode=block";

add\_header X-Content-Type-Options "nosniff";

add\_header Referrer-Policy "no-referrer-when-downgrade";

add\_header Feature-Policy "geolocation 'none'; midi 'none'; notifications 'none'; push 'self'; sync-xhr 'none'; microphone 'none'; camera 'none'; magnetometer 'none'; gyroscope 'none'; speaker 'none'; vibrate 'self'; fullscreen 'self'; payment 'self'";

}

charset utf-8;

}

# server {

# listen \*:80;

# server\_name www.meudominio.gov.br /;

# return 301 $scheme:// www.meudominio.gov.br /$request\_uri;

# }

Crie o link para habilitar o virtual host

root@server# ln -s /etc/nginx/sites-available/mapas.conf /etc/nginx/sites-enabled/mapas.conf

Remover o arquivo default da pasta /etc/nginx/sites-available/

root@server# rm /etc/nginx/sites-available/default

**Configurações pool do php-fpm**

Crie o arquivo **/etc/php7.2/fpm/pool.d/mapas.conf** com o conteúdo abaixo:

[mapas]

listen = /var/run/php/php7.2-fpm-www.meudominio.gov.br.sock

listen.owner = mapas

listen.group = www-data

user = mapas

group = www-data

catch\_workers\_output = yes

pm = dynamic

pm.max\_children = 10

pm.start\_servers = 1

pm.min\_spare\_servers = 1

pm.max\_spare\_servers = 3

pm.max\_requests = 500

chdir = /var/www/mapas

; php\_admin\_value[open\_basedir] = /var/www/mapas:/tmp

php\_admin\_value[session.save\_path] = /tmp/

php\_admin\_value[error\_log] = /var/log/mapasculturais/php.error.log

php\_admin\_flag[log\_errors] = on

php\_admin\_value[display\_errors] = 'stderr'

**5. Concluindo**

Para finalizar, precisamos popular o banco de dados com os dados iniciais e executar um script que entre outras coisas compila e minifica os assets, otimiza o autoload de classes do composer e roda atualizações do banco.

root@server# su - mapas

mapas@server$ psql -f mapasculturais/db/initial-data.sql

mapas@server$ ./mapasculturais/scripts/deploy.sh

Reinicie os serviços do **nginx** e **php-fpm**

root@server# service nginx restart

root@server# service php7.2-fpm restart

**6. Configurando serviço de cron para recriar cache de permissões**

Veja [Caches de Permissão (pcache)](https://github.com/mapasculturais/mapasculturais/blob/develop/documentation/docs/mc_permission_cache.md)

**7. Pós-instalação > API de CEP (versão atualizada do viacep)**

No arquivo de configuração da aplicação (mapasculturais/src/protected/application/conf/config.php), há possibilidade de setar um token para consulta de uma API que ajuda na geolocalização de endereço. O administrador da plataforma deve entrar no portal cep aberto ([http://www.cepaberto.com](http://www.cepaberto.com/)), efetuar o cadastro e inserir o token no arquivo de configuração, descomentando a linha.

// 'cep.token' => '[token]',

**8. Pós-instalação > Criando super admin**

Para criar super usuários, é necessário mudar o status de um usuário já criado, deixando-o como superadmin. Você pode proceder da seguinte forma:

1 - Crie um usuário pelo painel;

2 - Entre no postgres e conecte-se na base. Caso esteja usando socket, basta que você esteja logado no terminal com o usuário do sistema em questão (se tiver seguido a documentação, esse usuário será o 'mapas') e digite psql. Isso irá mudar o terminal para:

$ mapas =>

3 - Verifique o número do ID do usuário criado. Você pode ver pelo painel do mapas, na url do usuário ou ainda usar um select.

$ mapas => select id,status, email from usr where email='digite o endereço de email do usuário criado';

Quando executar essa linha você vai pegar o id.

4 - Dê um insert ou um update na tabela *role*.

Insert:

$ mapas => INSERT INTO role (id, usr\_id, name) VALUES (nextval('role\_id\_seq'), ID-DO-USUARIO, 'superAdmin');

Update:

UPDATE role SET name = 'saasAdmin' WHERE id = 'ESCREVA SEU ID AQUI';

5 - Caso queira verificar o sucesso da ação, dê um select na tabela role.

$ mapas => select \* from role;

**9. Pós-instalação > Processo de autenticação**

O Mapas Culturais não tem um sistema próprio de autenticação, sendo seu funcionamento atrelado a um plugin ou a um sistema de autenticação terceiro. Atualmente, há um plugin e dois sistemas de autenticação que estão aptos e testados para essa tarefa:

* [Mapas Culturais Open ID](https://github.com/hacklabr/mapasculturais-openid)
* [Login Cidadão](https://github.com/redelivre/login-cidadao).
* [Multiple Local Auth](https://github.com/LibreCoopUruguay/MultipleLocalAuth) - Implementa um login nativo da aplicação, sem depender de serviços externos, e suporta login via redes sociais.

**9.1 Requisitos para implementação dos sistemas de autenticação de terceiros**

**Mapas Open ID Conect**

Este é um sistema em Python/Django e está ativo em algumas implementações, mas seu código tem pouca documentação e está descontinuado. Não recomenda-se a instalação com esse sistema a menos que o implementador possa contar com um time de desenvolvedores que impulsionem a retomada da ferramenta.

Fonte: <https://github.com/hacklabr/mapasculturais-openid>.

**Login Cidadão > Instalação Própria**

O Login Cidadão é um software que implementa um sistema de autenticação unificado em grande escala, unificando políticas de segurança, transparência e privacidade, e colocando o cidadão como ponto de convergência para a integração descentralizada dos dados e aplicações. Seu código é livre e é baseado, principalmente, no framework Symfony (php)

**Prós**

Os pontos positivos relativos aos aspectos de implementação de uma instalação própria são:

* Confidencialidade dos dados e soberania: todos os dados estarão fisicamente em posse do implementador;
* Maior controle técnico de customização de layout e features. A posse desse customização, desde que com conhecimento adequado, é do implementador;

**Contras**

* Necessidade de servidor próprio e dedicado a instalação;
* Manutenção com ônus financeiro uma vez que é necessário manter time (interno ou terceirizado) com conhecimentos técnicos adequado à operação técnica do software;
* Necessidade de endereço (url) dedicada e de certificado SSL implementado (o que também pode gerar custos uma vez 99% dos certificados são pagos anualmente);
* Comunidade pequena em torno do software, o que dificulta suporte espontâneo quando necessário;
* Versão 1.0 do software ainda não lançada;
* A aplicação não possui sistema de templates gerenciado via painel, o que gera necessidade de horas-técnicas para desenvolvimento/customização de tema no código;
* Documentação ainda incompleta, aumentando curva de aprendizado sobre o sistema;

**Fonte:**

<https://github.com/redelivre/login-cidadao>

**Documentação de instalação e parametrização técnica:** (incompleto)

<https://github.com/redelivre/login-cidadao/tree/master/doc>

**Documentação de operação:**

(inexistente)

**Portal:**

[http://logincidadao.org.br](http://logincidadao.org.br/)

**Login Cidadão > Instância MINC (ID da Cultura)**

**Prós**

* Confidencialidade dos dados e soberania protegidas por uma entidade federal (Ministério da Cultura);
* Dispensa necessidade de servidor próprio e dedicado a instalação;
* Manutenção sem ônus financeiro uma vez que equipe do Departamento de Tecnologia da Informação do Ministério da Cultura incorpora em seu workflow de trabalho as demandas de atualização do sistema;
* Dispensa necessidade de endereço (url) dedicado e de certificado SSL próprio;

**Contras**

* Menor controle técnico de customização de layout e features. A posse/soberania destas customização é do Ministério da Cultura e este deve ser acionado se necessário;
* Para implementação, é necessário acionar equipe do Minc/DTI para criar uma entrada de origem do sistema, uma vez que os administradores são membros do DTI. No entanto esse processo é rápido e deve acontecer apenas uma vez, no inicio da instalação ou em momento esporádico de eventual manutenção do sistema.

**Fonte:** [http://id.cultura.gov.br](http://id.cultura.gov.br/)