**Q-Ware Collab**

**Manual de Desenvolvimento**

**Ministério do Meio Ambiente – MMA**

**Biosfera**

**SUMÁRIO**

[1. Estrutura de diretórios – Aplicação PHP 3](#_Toc489272883)

[Raiz da Aplicação 3](#_Toc489272884)

[Diretório app 4](#_Toc489272885)

[Diretório data 5](#_Toc489272886)

[Diretório lib 5](#_Toc489272887)

[Diretório node\_modules 6](#_Toc489272888)

[Diretório src 7](#_Toc489272889)

[Diretório vendor 7](#_Toc489272890)

[Diretório web 8](#_Toc489272891)

[2. Estrutura dos Bundles 9](#_Toc489272892)

[3. Estrutura de diretórios – Aplicação PushD 11](#_Toc489272893)

[4. Kernel da Aplicação PHP 12](#_Toc489272894)

[5. Arquivos de Assets dos Bundles 13](#_Toc489272895)

[6. Arquivos de mapeamento de Tour 15](#_Toc489272896)

[7. Comandos CLI 16](#_Toc489272897)

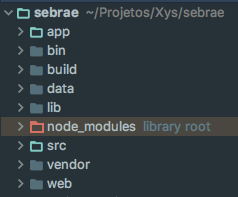
1. **INTRODUÇÃO.**

**Este documento tem por objetivo esclarecer aos desenvolvedores os procedimentos necessários para instalação e configuração da Plataforma Biosfera.**

# ESTRUTURA DE DIRETÓRIOS – Aplicação PHP.

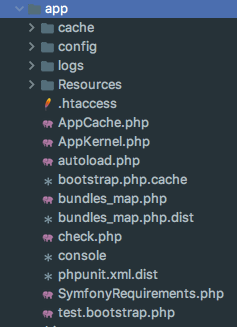
1. **Raiz da Aplicação**.

Na raiz do diretório da aplicação temos os seguintes diretórios:



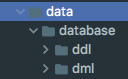
* **app -** diretório que possui os arquivos de configuração, gerenciamento de rotas, Kernel da aplicação dentre outros arquivos gerais a aplicação.
  + ***Não deve ser exposto para a web***
* **bin** – diretório que possui os binários para instalação de vendors, testes, e ferramentas de análise de código.
  + ***Não deve ser exposto para a web***
* **build** – diretório que possui arquivos de configuração utilizados pela integração contínua.
  + ***Não deve ser exposto para a web***
* **data** – diretório com scripts DDL e DML necessários para a aplicação
  + ***Não deve ser exposto para a web***
* **lib** – diretório com utilitários desenvolvidos em versão prévia do Q-Ware Collab ainda necessários para a versão atual.
  + ***Não deve ser exposto para a web***
* **node\_modules** – diretório com módulos do node necessários para automações de tarefas via node como por exemplo manipulação de assets.
  + ***Não deve ser exposto para a web***
* **src** – diretório com Bundles responsáveis pelas funcionalidades da plataforma.
  + ***Não deve ser exposto para a web***
* **vendor** – diretório com os códigos-fonte de terceiros utilizados pela plataforma tais como symfony, doctrine entre outros.
  + ***Não deve ser exposto para a web***
* **web** – diretório que possui os arquivos públicos da aplicação

1. **Diretório APP**.



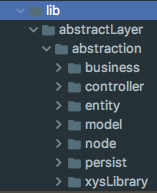
* **cache -** diretório que armazena os arquivos físicos de cache.
* **config** – diretório que contém os arquivos de parâmetros da aplicação, arquivos de configuração de cache, arquivos de configuração de segurança e arquivos de configuração de rotas da aplicação.
* **logs** – diretório que possui arquivos de logs da aplicação.
* **Resources** – diretório com views básicas do sistema, páginas de exceção e arquivo do yuicompressor.

1. **Diretório Data**.



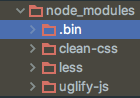
* **database/ddl -** diretório que armazena os arquivos com scripts ddl da plataforma.
* **database/dml -** diretório que armazena os arquivos com scripts dml da plataforma.

1. **Diretório Lib**.



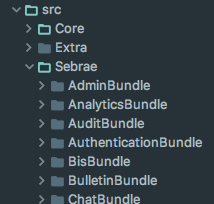
* **abstractionLayer -** diretório que possui os códigos da camada de abstração legada do Q-Ware Collab mas ainda necessária na aplicação.
* **abstractionLayer/abstraction/business -** diretório que armazena os arquivos com abstração para camada de negócios.
* **abstractionLayer/abstraction/controller -** diretório que armazena os arquivos com abstração para camada de controller.
* **abstractionLayer/abstraction/entity -** diretório que armazena os arquivos com abstração para camada de entidades.
* **abstractionLayer/abstraction/model -** diretório que armazena os arquivos com abstração para camada de modelo.
* **abstractionLayer/abstraction/node -** diretório que armazena o arquivo da classe utilizada para comunicação de notificações para o PushD.
* **abstractionLayer/abstraction/persist -** diretório que armazena os arquivos com abstração para camada de persistência.
* **abstractionLayer/abstraction/xysLibrary -** diretório que armazena os arquivos de libs de terceiros necessárias para o funcionamento específico de alguns recursos da plataforma.

1. **Diretório Node\_Modules**.



Contém os binários necessários para os componentes utilizados na compilação dos assets da aplicação.

1. **Diretório SRC**.



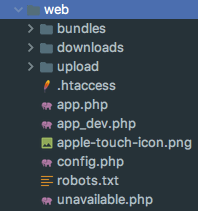
Diretório responsável pelos códigos-fonte da aplicação Q-Ware Collab e customizações específicas para o Ministério do Meio Ambiente.

* **Core -** diretório que possui os códigos do núcleo da aplicação.
* **Extra -** diretório que possui os códigos de funcionalidades extras da aplicação.
* **Biosfera –** diretório que possui os códigos de funcionalidades específicas do Ministério do Meio Ambiente ou customizações e sobrescritas de códigos do Core.
* **Social** – diretório que possui os códigos da camada social do do Q-Ware Collab.
* **Padrão –** diretório que possui os códigos da camada de layout padrão do Q-Ware Collab.

1. **Diretório Vendor**.

Apenas libs e frameworks de terceiros.

1. **Diretório *W*eb**.

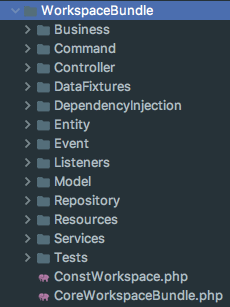


Diretório responsável por toda a parte pública da plataforma, este diretório deve ser a raiz do virtual host.

* **bundles -** diretório que possui os códigos exportados da parte pública de cada bundle.
* **upload –** diretório que possui os arquivos de upload da plataforma.

1. **ESTRUTURA DOS BUNDLES**.

O Bundle representa um conjunto de códigos dentro de um contexto de uma funcionalidade ou de uma ação. O Bundle contém várias sub-estruturas sendo algumas obrigatórias e outras opcionais. Neste passo mostraremos e especificaremos todas as estruturas utilizadas na arquitetura proposta. O Bundle em questão representa as funcionalidades do módulo *Comunidades*.



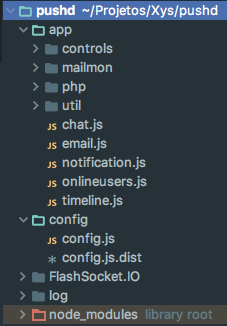
* **Business -** diretório que possui os códigos referentes a regras de negócio no modelo Legado.
* **Command -** diretório que possui os códigos das classes com comandos para a interface de linha de comando.
* **Controller –** diretório que possui os códigos das classes de Controller responsáveis por prover dados e algumas visualizações através de suas ações.
* **DataFixtures** – diretório que possui os códigos legados de geração de massa de dados para testes.
* **DependencyInjection –** diretório que possui os códigos de Extensão e pré-compiladores sendo alguns códigos gerados pelo Symfony.
* **Entity** – diretório que possui os códigos de classes do mapeamento objeto relacional.
* **Event** – diretório que possui os códigos das classes de Eventos lançados pelas funcionalidades.
* **Listeners** – diretório que possui os códigos das classes de Listeners de eventos do backend.
* **Model** – diretório que possui os códigos das classes de Modelo por onde os dados eram consultados no modelo legado.
* **Repository** – diretório que possui os códigos das classes de Repositório por onde os dados são consultados na arquitetura nova.
* **Resources** – diretório que possui os códigos dos arquivos de configuração específicos do bundle bem como arquivos de rotas, arquivos de mapeamento de assets, arquivos de assets como javascript, css, imagens, templates e arquivos de tradução.
* **Services –** diretório que possui os códigos das classes Services que são responsáveis pelas regras de negócio da funcionalidade no modelo de arquitetura atual.
* **Tests** – diretório com que possui os códigos das classes de Testes e massas de testes.

1. **ESTRUTURA DE DIRETÓRIOS – Aplicação PushD**.

A aplicação PushD é responsável por todos os eventos para operações em real-time tais como ações do chat e notificações da plataforma. A tecnologia escolhida para essa funcionalidade foi o NodeJS utilizando os frameworks Socket.IO e Express.

Esta aplicação se subdivide em duas partes básicas, um servidor de notificações via push para a web e um servidor web para capturar as requisições enviadas pela aplicação PHP seguindo o modelo webhook, ou seja, todas as notificações a serem disparadas para usuários online são enviadas pelo PHP para o NodeJS através de um POST, o Node recebe a informação, realiza os tratamentos necessários, identifica quais canais socket precisam receber as notificações e realiza a emissão dos eventos individualmente ou via broadcast.

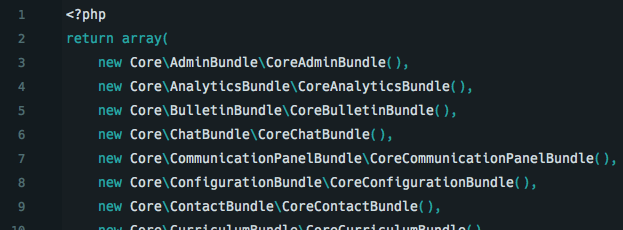
O PushD pode ser iniciado como daemon ou dentro de um screen ou pode ter a sua saída movida para um arquivo e ser iniciado com nohup. Toda a persistência das informações enquanto trafegadas no Node são realizadas em memória o que impossibilita um balanceamento de carga que não utilize a estratégia de failover.



* **app** – diretório com toda a implementação lógica da aplicação
  + **controls** – Controles de sockets, usuários logados, sessões, usuários online entre outros.
  + **mailmon** – módulo de monitoramento de mensagens via protocolo imap.
  + **php** – integração com a ApiRest da plicação php
  + **util** – classes utilitárias
  + **chat.js** – manipula eventos do canal do chat
  + **email.js** – manipula eventos do canal de email
  + **notification.js** – manipula eventos do canal de notificações
* **config –** arquivos de configuração para a aplicação pushd
* **FlashSocket.IO –** Integração legada para clientes que usam tecnologia Flex
* **log –** arquivos de log da aplicação
* **node\_modules –** arquivos de bibliotecas de terceiros.

1. **Z;a 90-Kernel da Aplicação PHP**.

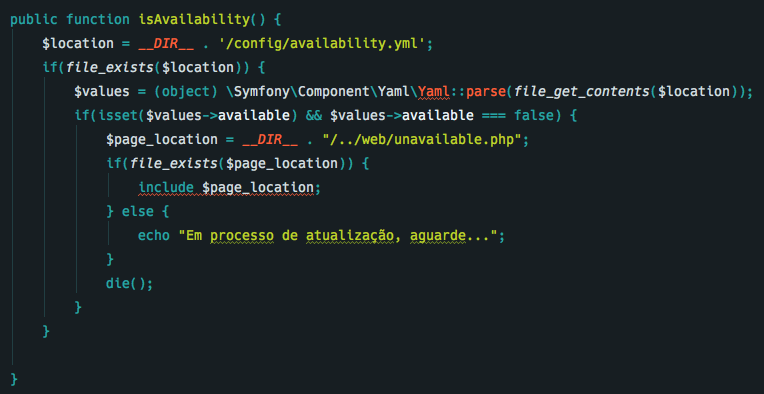
O Kernel da aplicação define os Bundles registrados para uso da aplicação. Para registrarmos os nossos bundles entretanto escrevemos um arquivo com um array de bundles para nossas dependências internas. Este arquivo é o *app/bundles\_map.php.*



Todos os Bundles desenvolvidos para o projeto da Biosfera pela Padrão-iX estão registrados neste array.

O Kernel possui uma implementação para indisponibilizar o sistema, esta lógica lê um arquivo de disponibilidade que fica em *app/config/availability.yml*, este arquivo se possuir uma propriedade *available* com o valor *false* carregará uma página php localizada em *web/unavailable.php* caso este arquivo não seja encontrado, apresentamos a mensagem fixa: “*Em processo de atualização, aguarde...”*.

A implementação encontra-se em *app/AppKernel.php* no método *isAvailability*.



1. **ARQUIVOS DE ASSETS DOS BUNDLES**.

Os assets dos Bundles ficam nos diretórios públicos:

**src/<MacroName>/*<NomeDoBundle>/Resources/public***

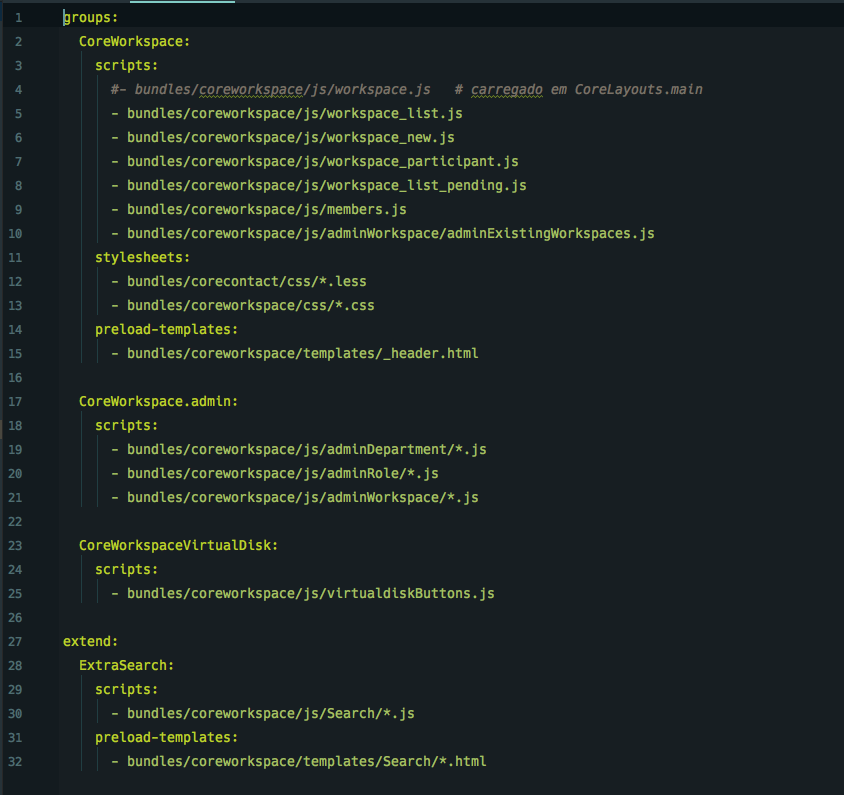
Os javascripts e folhas de estilos que formarem grupos de assets devem ser registrados nos arquivos de mapeamento de grupos, os templates podem ser pré-cacheados também nesses arquivos que ficam em:

**src/<MacroName>/*<NomeDoBundle>/Resources/config/assets.yml***

A nomenclatura e localização dos arquivos deverá ser exatamente a especificada acima, para que o mecanismo de captura de assets funcione.

Ao final do processo os scripts serão unificados e comprimidos por um comando CLI desenvolvido pela Padrão-iX e os arquivos serão disponibilizados no ambiente público da aplicação, formando assim o novo modelo de arquitetura para assets. O processo legado usa o modo de compilação de assets padrão do Symfony.

Abaixo exemplo de arquivo assets:



1. **ARQUIVOS DE MAPEAMENTO DE TOUR**.

Os tours virtuais são configurados nos arquivos yml que ficam em:

***src/<MacroName>/<NomeDoBundle>/Resources/config/tour.yml***



1. **COMANDOS CLI**.

Symfony version 2.4.10 - app/dev/debug

Usage:

  command [options] [arguments]

Options:

  --help           -h Display this help message

  --quiet          -q Do not output any message

  --verbose        -v|vv|vvv Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug

  --version        -V Display this application version

  --ansi              Force ANSI output

  --no-ansi           Disable ANSI output

  --no-interaction -n Do not ask any interactive question

  --shell          -s Launch the shell.

  --process-isolation    Launch commands from shell as a separate process.

  --env            -e The Environment name.

  --no-debug          Switches off debug mode.

Available commands:

  help                                        Displays help for a command

  list                                        Lists commands

api

  api:doc:dump

assetic

  assetic:dump                                Dumps all assets to the filesystem

assets

  assets:install                              Installs bundles web assets under a public web directory

bazinga

  bazinga:expose-translation:dump             Dumps all translation files into a given directory.

cache

  cache:clear                                 Clears the cache

  cache:warmup                                Warms up an empty cache

config

  config:dump-reference                       Dumps default configuration for an extension

container

  container:debug                             Displays current services for an application

coreanalytics

  coreanalytics:logstash:dump-config          Emite a configuração do logstash

coreconfiguration

  coreconfiguration:compile:file\_parameters   Inicializa labels default do usuário.

coreexternaluser

  coreexternaluser:reminder:participation     notificar expiração do espaço de trabalho

  coreexternaluser:remove:participation       Remover participação do espaço de trabalho

coremessage

  coremessage:deletemessage:trash             Deleta Mensagens por um período em dias

  coremessage:labels:initializer\_default      Inicializa labels default do usuário.

coreportal

  coreportal:components:initialize            Inicializa os componentes dos portais

coresystem

  coresystem:assets:compile                   Comprime os assets p/ produção

  coresystem:assets:debug                     Lista os assets conhecidos

  coresystem:bootstrapinfo:debug              Emite os dados enviados à aplicação

  coresystem:data:load                        Carrega os dados nas tabelas de domínio

  coresystem:frontend:debug:routes            Lista as rotas de front-end encontradas

  coresystem:import:sql                       Executa as instruções do(s) arquivo(s) SQL para dentro do Banco de Dados.

  coresystem:mer:generator                    Criar estrutura de tabelas e colunas para realiza busca relacionando a entidade (exemplo: --entity=News)

  coresystem:worker:remove                    Remove os workers do sistema

  coresystem:worker:run                       Inicia os workers do sistema

coretag

  coretag:tags:sync\_vcs                       Inicializa sincronização entre Tags e VCS.

coretask

  coretask:email:daily                        Email diário de tarefas

coretour

  coretour:list                               Lista os tours conhecidos

  coretour:register                           Atualiza a tabela de tours, sincronizando-a com a lista de tours no sistema

coreupload

  coreupload:uploadfile:updatesize            Atualiza o tamanho dos arquivos que estão vazios ou 0 bite.

coreworkspace

  coreworkspace:components:list               Lista os WorkspaceComponents disponíveis

  coreworkspace:components:register           Atualiza a tabela de WorkspaceComponents

coreworkspacepoll

  coreworkspacepoll:notifypoll:current        Notifica os membros do espaço de trabalho sobre uma enquete que entrou em vigor.

doctrine

  doctrine:cache:clear-metadata               Clears all metadata cache for an entity manager

  doctrine:cache:clear-query                  Clears all query cache for an entity manager

  doctrine:cache:clear-result                 Clears result cache for an entity manager

  doctrine:database:create                    Creates the configured databases

  doctrine:database:drop                      Drops the configured databases

  doctrine:ensure-production-settings         Verify that Doctrine is properly configured for a production environment.

  doctrine:generate:crud                      Generates a CRUD based on a Doctrine entity

  doctrine:generate:entities                  Generates entity classes and method stubs from your mapping information

  doctrine:generate:entity                    Generates a new Doctrine entity inside a bundle

  doctrine:generate:form                      Generates a form type class based on a Doctrine entity

  doctrine:mapping:convert                    Convert mapping information between supported formats.

  doctrine:mapping:import                     Imports mapping information from an existing database

  doctrine:mapping:info                       Shows basic information about all mapped entities

  doctrine:query:dql                          Executes arbitrary DQL directly from the command line.

  doctrine:query:sql                          Executes arbitrary SQL directly from the command line.

  doctrine:schema:create                      Executes (or dumps) the SQL needed to generate the database schema

  doctrine:schema:drop                        Executes (or dumps) the SQL needed to drop the current database schema

  doctrine:schema:update                      Executes (or dumps) the SQL needed to update the database schema to match the current mapping metadata

  doctrine:schema:validate                    Validates the doctrine mapping files

extracrudgenerator

  extracrudgenerator:crud:generator           Criar estrutura de arquivos e configuracoes

fos

  fos:elastica:populate                       Populates search indexes from providers

  fos:elastica:reset                          Reset search indexes

  fos:elastica:search                         Searches documents in a given type and index

  fos:js-routing:debug                        Displays currently exposed routes for an application

  fos:js-routing:dump                         Dumps exposed routes to the filesystem

gearman

  gearman:job:run                             Run given gearman job.

  gearman:worker:run                          Run given worker by filename.

generate

  generate:bundle                             Generates a bundle

  generate:controller                         Generates a controller

  generate:doctrine:crud                      Generates a CRUD based on a Doctrine entity

  generate:doctrine:entities                  Generates entity classes and method stubs from your mapping information

  generate:doctrine:entity                    Generates a new Doctrine entity inside a bundle

  generate:doctrine:form                      Generates a form type class based on a Doctrine entity

import

  import:data:elastic                         IMPORTAÇÃO DE DADOS DO ELASTICSEARCH PARA BANCO DE DADOS!!!!!!!!!!!!!!!!!!!!!!!!!

  import:datalogin:elastic                    IMPORTAÇÃO DE DADOS DO ELASTICSEARCH PARA BANCO DE DADOS!!!!!!!!!!!!!!!!!!!!!!!!!

init

  init:acl                                    Mounts ACL tables in the database

  init:jms-secure-random

orm

  orm:convert:mapping                         Convert mapping information between supported formats.

redis

  redis:flushall                              Flushes the redis database using the redis flushall command

  redis:flushdb                               Flushes the redis database using the redis flushdb command

router

  router:debug                                Displays current routes for an application

  router:dump-apache                          Dumps all routes as Apache rewrite rules

  router:match                                Helps debug routes by simulating a path info match

sebrae

  sebrae:change-status:sgus                   Atualiza o status da base de usuários do sgus

  sebrae:import:sgus                          Atualiza a nossa base de dados com as informações do SGUS

  sebrae:search:sgus                          Consulta usuários no sgus pelo cpf e importa se desejado

  sebrae:sgus:find-user                       Atualiza a nossa base de dados com as informações do SGUS

  sebrae:sgus:reset-sid                       Zera o sid de um usuário

  sebrae:sgus:test-auth                       Zera o sid de um usuário

  sebrae:update:sgus                          Atualiza a nossa base de dados com as informações do SGUS

sebraeanalytics

  sebraeanalytics:logstash:dump-config        Emite a configuração do logstash

sebraeenterprisepoll

  sebraeenterprisepoll:notifypoll:current     Notifica os membros do espaço de trabalho sobre uma enquete que entrou em vigor.

sebraenews

  sebraenews:portaluc:update                  Atualiza as notícias do PortalUC.

  sebraenews:schedulingsorting:update         Executa uma função para alterar a ordem de acordo com o agendamento de noticias.

server

  server:run                                  Runs PHP built-in web server

swiftmailer

  swiftmailer:debug                           Displays current mailers for an application

  swiftmailer:spool:send                      Sends emails from the spool

system

  system:availability                         Ativa/Inativa o portal

translation

  translation:update                          Updates the translation file

twig

  twig:lint                                   Lints a template and outputs encountered errors