

Organisation des fichiers

Castopod s'appuie sur le framework CodeIgniter, qui possède un modèle MVC consistant à diviser les différentes parties d'un projet sous trois formes :

- Les vues
- Les contrôleurs
- Les modèles

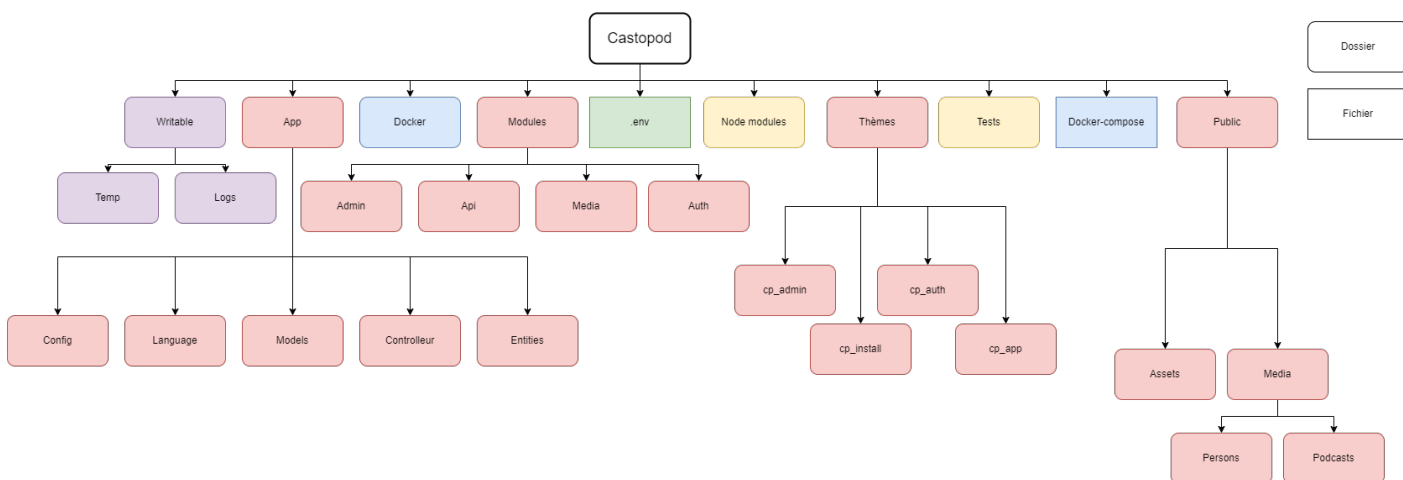
Il existe également les entités qui sont des classes permettant de manipuler plus facilement des objets comme des utilisateurs ou des fichiers.

Pour en savoir plus sur ce modèle, nous vous invitons à vous diriger vers [la page correspondante](#) à l'explication de CodeIgniter ainsi qu'à la [documentation officielle](#).

Une fois ce modèle bien compris, nous pouvons donc analyser la structure de Castopod, qui possède ces différents éléments. Ici, nous nous concentrons sur l'organisation des fichiers en [mode développement](#), nous ne nous attardons pas sur les fichiers en mode production qui peuvent présenter des différences.

Structure générale

Voici la structure générale du projet Castopod, qui sera détaillé plus en profondeur ensuite :



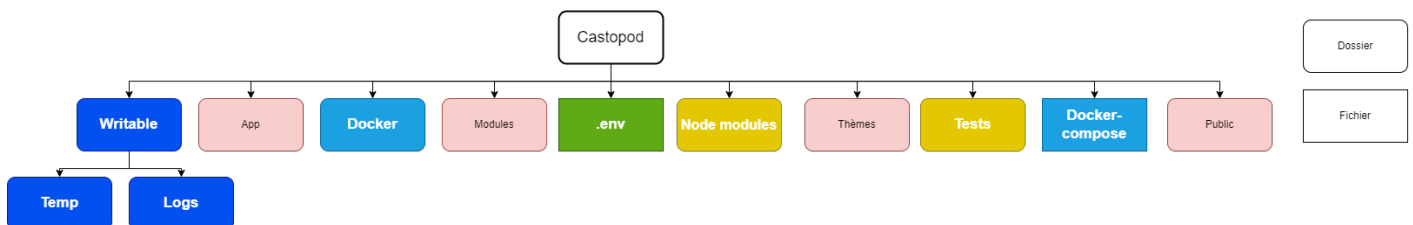
Vous remarquerez ici que nous nous sommes concentré sur une partie des fichiers, et que nous n'allons pas décrire l'ensemble des fichiers/dossiers. Beaucoup de dossiers sont liés à CodeIgniter ou à Git, mais ils n'ont normalement pas besoin d'être configuré. Cependant, si vous souhaitez

malgré tout vous pencher sur la configuration de Codelgniter, nous vous invitons à accéder [au chapitre expliquant la configuration du projet](#).

De plus, la description des dossiers et fichiers peuvent changer en fonction des versions du projet, nous nous appuyons ici sur la version 1.9.0 du 31 janvier 2024.

Configuration et mise en place du projet

Nous allons décrire dans cette partie les dossiers et fichiers permettant l'installation et le paramétrage du projet Castopod.



Ces différents éléments permettent au projet de fonctionner selon l'envie de l'utilisateur, mais peuvent tout à fait être laissé par défaut si le développeur souhaite travailler uniquement sur les pages Web du projet.

Docker

Pour gérer Docker, Castopod s'appuie sur des fichiers décrivant les différents conteneurs à construire pour gérer le projet.

On a un fichier nommé *docker-compose.yml* qui décrit les conteneurs à construire pour le projet, qui sont au nombre de 5 :

- Le conteneur Castopod : Il va contenir l'ensemble des fichiers de Codelgniter qui vont permettre de gérer le front-end et le back-end du projet
- Le conteneur Redis : C'est un système de gestion de base de données. Il stocke les données en mémoire vive, ce qui le rend extrêmement rapide pour les opérations de lecture et d'écriture.
- Le conteneur MariaDB : On retrouve la base de données générale du projet. Contrairement à Redis, elle est utilisée pour gérer des ensembles de données complexes avec des relations entre les tables.
- Le conteneur PHP My Admin : Ce conteneur va permettre d'avoir une gestion graphique de la base de données, en accédant à la page *localhost:8888* (modifiable dans le *.env* du

projet)

- Le conteneur S3 Mock : C'est un outil qui simule le service de stockage d'objets d'Amazon Web Services (AWS) appelé Amazon S3

Ce fichier va permettre de gérer les ports sur lequel les conteneurs vont être reliés à la machine hôte, de gérer différents paramètres des applications, les commandes qui vont être lancés au démarrage et le nom des volumes.

On a également un dossier Docker, qui contient un fichier dans le sous-dossier développement, qui va décrire la construction du conteneur de Castopod.

Pour en savoir plus sur la gestion des fichiers Docker, nous vous invitons à lire [la documentation officielle de Docker](#), qui contiendra plus d'informations sur le déploiement des conteneurs.

Composer

Composer est un logiciel gestionnaire de dépendances libre écrit en PHP. Castopod s'appuyant sur de nombreuses bibliothèques pour fonctionner, Composer lui permet lors de son déploiement d'installer ces bibliothèques.

Pour cela, les bibliothèques PHP nécessaire sà Castopod sont listés dans un fichier nommé *composer.json*, qui indique également les versions à installer de ces bibliothèques, et qui créé des alias pour des commandes liées à ces bibliothèques dans la partie *scripts* du fichier.

Ces bibliothèques seront ensuite installées dans le dossier **vendor** une fois cette commande utilisée :

```
composer install
```

Encore une fois, pour avoir de plus ample informations, nous vous renvoyons vers [la documentation de Composer](#) qui vous décrira son fonctionnement.

Node

Node est, à l'instar de Composer, un gestionnaire de paquet mais en javascript, qui permet l'installation de bibliothèques en JS lors du déploiement du projet, ainsi que la gestion côté serveur.

L'ensemble de ces paquets sont détaillés dans le fichier *package.json*, ainsi que les alias qui pourront être utilisé via [npm](#). Il permet par exemple l'utilisation de Vite, une bibliothèque permettant le développement en local de site internet.

Le dossier **node_modules** contient l'ensemble des bibliothèques JS installées, qui peuvent être téléchargées et installées via la commande :

```
npm install
```

Fichier *.env*

Ce fichier va décrire la manière donc le projet doit fonctionner. C'est un fichier phare du projet, qui est directement lié à CodeIgniter car c'est un fichier que l'on retrouve sur les projets utilisant ce framework.

Dans ce fichier, vous allez pouvoir indiquer des paramètres pour le fonctionnement de Castopod, comme :

- Le type de déploiement (production ou développement, ce qui va jouer les performances du projet)
- L'URL du projet (celle de l'application, et celle où les médias seront enregistrés)
- Les informations de la base de données
- Le préfix pour les enregistrement dans la base de données
- Les identifiants SMTP pour la gestion des emails
- La gestion d'une [API REST](#) sur Castopod
- Des configurations pour les conteneurs du projet, tels que S3 ou Redis

Si vous souhaitez avoir plus d'informations sur ce fichier, nous vous invitons à vous diriger vers [la documentation de CodeIgniter](#), qui décrit plus en profondeur les possibilités de ce fichier.

Tests

Dans ce dossier, vous allez retrouver les différents tests qui seront effectués durant le développement du projet. Pour cela, Castopod s'appuie sur la bibliothèque [PHP Unit](#) qui permet d'effectuer des tests unitaires sur des projets PHP.

Dans ce dossier, vous pouvez configurer votre propre test dans ce dossier, et des tests sont déjà effectués par défaut par PHP Unit. Le fichier *README* du dossier vous explique les étapes à effectuer pour lancer les tests, ou pour créer les votre directement.

Il existe deux manières de lancer ces tests, selon votre système d'exploitation :

```
./phpunit # Sur Linux et Mac  
vendor\bin\phpunit # Sur Windows
```

Et vous pouvez spécifier un dossier particulier pour lancer les tests, en rajoutant en paramètre de la commande l'emplacement du dossier.

Writable

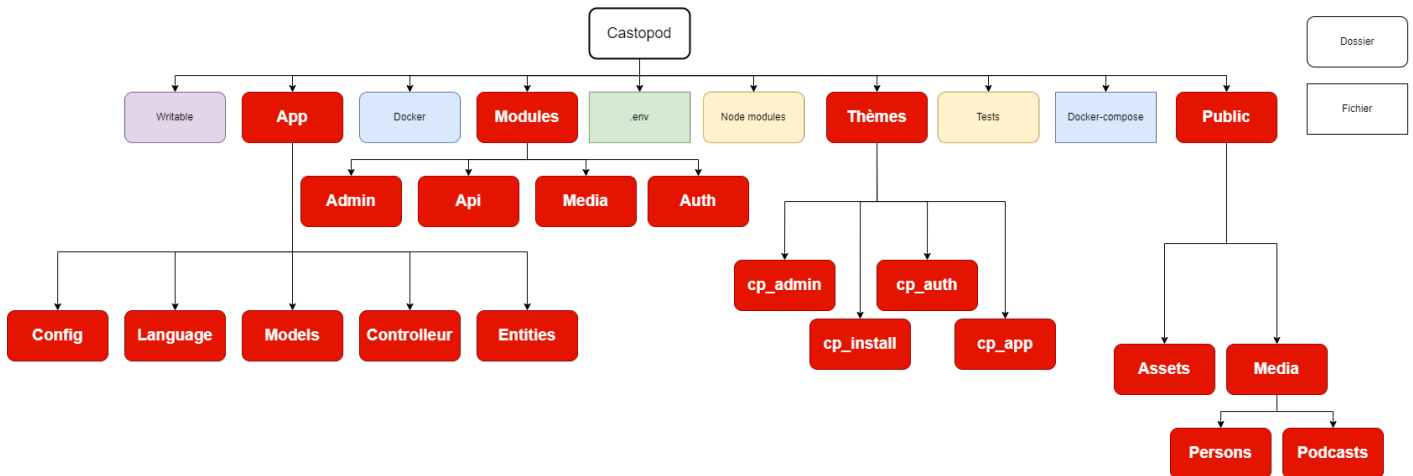
Ce dossier permet d'écrire temporairement des données sur le conteneur Castopod et d'accéder à des logs.

On retrouve dedans un dossier **temp**, qui peut être utilisé pour stocker des fichiers temporairement, ainsi qu'un dossier uploads qui lui peut être utilisé pour un stockage plus long. Attention, ce n'est cependant pas ici que Castopod enregistre les médias liés aux podcasts ou aux utilisateurs. Eux sont enregistrés dans le dossier **public** dont nous parlerons plus loin.

Enfin, on retrouve également des logs, dans le dossier **logs**, ainsi que des logs des sessions où l'on va retrouver les variables des différentes sessions dans le dossier **session**.

Développement de Castopod

Nous allons maintenant vous décrire l'architecture des dossiers et fichiers qui permettent d'éditer le front-end ainsi que le back-end des différentes pages de Castopod.



Comme expliqué au départ, vous allez retrouver la structure MVC dans cette organisation. Ainsi, si vous n'êtes pas sûr d'avoir bien compris en quoi cela consistait, nous vous invitons à aller sur [Présentation CodeIgniter](#), la page décrivant ce modèle.

Public

Le dossier **public** est utilisé pour stocker les fichiers audios et les images utilisés par Castopod.

Dans le dossier général, on retrouve les bannières ainsi que l'icone de Castopod, et un filtre robots pour l'indexation sur des sites externes (dans le fichier *robots.txt*).

Dans le fichier *.well-know/GDPR.yml*, on retrouve une description de [la politique RGPD](#) utilisée dans Castopod.

Dans **assets**, on retrouve différents fichiers svg pouvant être utilisés par les pages de Castopod.

Et enfin, dans **media**, on retrouve les fichiers médias utilisés pour la gestion des utilisateurs et des podcasts. Dans **podcasts**, on va par exemple retrouver les fichiers audios liés aux épisodes, et les illustrations du podcast et des épisodes.

Thèmes

Dans le dossier **themes**, on va retrouver les vues réparties en 4 catégories :

- **cp_admin** : Les pages accessibles depuis la page d'administration. Par défaut, cette page est située à *URL/cp-admin*
- **cp_app** : Les pages accessibles aux utilisateurs classiques, qui souhaitent écouter les podcasts ou réagir avec
- **cp_auth** : La page de connexion, qui apparait lorsque l'on souhaite interagir ou accéder à la page d'administration
- **cp_install** : La page d'installation, qui est utilisé lors de la mise en place de Castopod et qui permet de finaliser l'installation et de créer un compte super-administrateur pour la première fois. Cette page est accessible par défaut à *URL/cp-install*

Vous pourrez remarquer que certains fichiers sont préfixés avec `_`. Ces fichiers sont des éléments qui sont présent sur plusieurs pages, comme par exemple le bandeau d'administration présent sur le gauche des pages de *cp_admin* :



Pour éviter de devoir recopier ce bandeau, il y a un fichier nommé *_sidebar.php* qui est inséré dans toutes les vues de *cp_admin*.

App

Dans le dossier **app**, on retrouve les contrôleurs, modèles et entités qui peuvent être utilisés par tous les utilisateurs (et pas seulement par les administrateurs). C'est dans ce dossier que la gestion de la page d'accueil et de l'écoute des épisodes est décrite.

Tout d'abord, nous avons les routes qui sont décrites dans `./Config/Routes.php`. On retrouve par exemple la route vers la page d'accueil décrite au début :

```
$routes->get('/', 'HomeController', [  
    'as' => 'home',  
]);
```

Nous décrivons le fonctionnement des routes sur Castopod dans [cette partie de la documentation](#).

Nous avons ensuite :

- Les modèles présents dans le dossier **Models**
- Les entités dans le dossier **Entity**
- Les contrôleurs dans le dossier **Controllers**.

Enfin, pour permettre à Castopod d'être utilisable dans plusieurs langues, on retrouve dans le dossier **Language** la traduction des différents textes des pages.

Modules

Dans le dossier **modules**, on va retrouver les contrôleurs qui sont utilisés pour un cas spécifique, qui ne concerne pas tous les utilisateurs (contrairement au dossier **app** vu juste auparavant).

Dans ce dossier, on va retrouver différents dossiers liés à des services. On peut notamment retrouver comme dossiers :

- Le dossier **Admin**, qui décrit les routes et les contrôleurs du panneau d'administration
- Le dossier **API**, qui permet d'avoir un système API Rest pour accéder aux informations des podcasts depuis un service extérieur
- Le dossier **Auth**, qui permet de gérer l'authentification et les pages accessibles pour un utilisateur connecté
- Le dossier **Media**, qui permet de gérer l'enregistrement des fichiers médias en paramétrant par exemple le dossier d'enregistrement, et qui gère aussi l'enregistrement sur la base de données des informations des médias

Il existe d'autres dossiers qui permettent de gérer d'autres services de Castopod, comme l'installation (**Install**) ou les podcasts premium (**PremiumPodcasts**).

Les dossiers possèdent une structure similaire avec un dossier **Config** qui décrit les routes, et généralement un autre fichier pour paramétrer le service. Et on retrouve également un dossier **Controllers**, qui va gérer le fonctionnement de ce service.

Il peut ensuite y avoir un dossier **Models** lorsque le service a besoin de communiquer avec la base de données, et un dossier **Entity** lorsque l'on souhaite pouvoir manipuler une classe liée à ce dossier.

Enfin, comme avec le dossier **app**, on retrouve un dossier **Language** lorsque le service peut afficher une page, pour gérer la traduction de cette page dans différentes langues.

Revision #22

Created 9 February 2024 14:57:37 by tanguy

Updated 10 February 2024 20:52:18 by tanguy