|  |
| --- |
| **Documentation** |
| Version 1 du CMS |
|  |
| Gestionnaire de contenu gratuit pour la généalogie. |
|  |
| **CYRIL HAVRET** |
| **02/06/2021** |
|  |

**Geneager**

Sommaire

[1 – Généralités 3](#_Toc73535593)

[1.1 - Introduction 3](#_Toc73535594)

[1.2 – Licence et droits 3](#_Toc73535595)

[2 – Structure des fichiers 3](#_Toc73535596)

[2.1 – Généralités 3](#_Toc73535597)

[2.2 – L’autoloader 4](#_Toc73535598)

[3 – Convention de nommage 4](#_Toc73535599)

[3.1 - Généralités 4](#_Toc73535600)

[3.2 – Models, views et controllers 5](#_Toc73535601)

[3.3 – Noms réservés 5](#_Toc73535602)

[3.3.1 – Les fichiers 5](#_Toc73535603)

[3.3.2 – Les variables et constantes PHP 5](#_Toc73535604)

[3.4 – Mots clés appréciés 5](#_Toc73535605)

[4 – La base de données 6](#_Toc73535606)

[4.1 – Généralités 6](#_Toc73535607)

[5 – PHP 6](#_Toc73535608)

[5.1 – Créer une page 6](#_Toc73535609)

[6 – Mods et dépendances 7](#_Toc73535610)

[6.1 - Apache2 7](#_Toc73535611)

[6.2 - PHP mods 7](#_Toc73535612)

[7 – Liens 7](#_Toc73535613)

# 

# 1 – Généralités

## 1.1 - Introduction

Gestionnaire de contenu GENEAGER est destiné pour la généalogie et ayant pour but de démocratiser d’avantage l’indépendance numérique dans le monde de la généalogie: offrant ainsi donc entre-autres un meilleur contrôle des données saisies (la révocation par exemple).

A terme un installateur sera inclus pour démocratiser d’avantage « en ciblant » les utilisateurs plus novices.

## 1.2 – Licence et droits

L’ensemble des codes du projet sont sous licence Creative Commons 4.0 (CC BY-NC-SA 4.0), cela signifie que :

**- Vous pouvez partager** (copier et distribuer et communiquer le matériel par tous moyens et sous tous formats)

- **vous pouvez adapter:** vous devez distribuer vos contributions sous la même licence que l'original.

- **Vous ne pouvez pas** utiliser les réalisations à des fins commerciales

Plus d’informations sur la licence sur la page suivante: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

# 2 – Structure des fichiers

## 2.1 – Généralités

L’architecture du CMS est de type MCV.

Dossier « root » (contient le .htaccess)

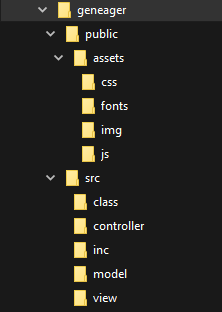
Accessible par le client (via l’URL)

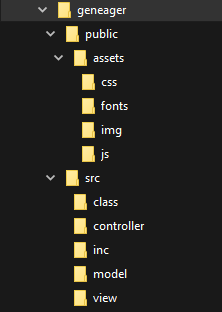
Inaccessible via le terminal client (réécriture à l’aide du fichier .htaccess).

Si le visiteur tape: *https://monsite.tld/class/index.class.php*

Il accède en réalité au fichier: */public/class/index.class.php*

🡪 En occurrence ici, il aura une page « 404 ».





**Contient les dossiers:**

- CSS (feuilles de style)

- Fonts (Police d’écriture)

- Img (icônes, sprites, …)

- JS (Javascript)

**Contient:**

- class (class appelées manuellement (sans autoloader))

- controller (programmes de logiques)

- inc (les fichiers includes (fonctions))

- model (class chargées automatiquement (autoloader))

- view (contient principalement le code HTML (la vue)): appelé avec l’autoloader si le « controller » est absent.

## 2.2 – L’autoloader

L’autoloader est une fonctionnalité qui permet de charger les bons fichiers en fonction de l’URL.

Par exemple « *https://monsite.tld/home* » appellera les fichiers suivants :

- /src/model/home.model.php

- /src/view/home.view.php (si le « controller » est absent)

- /src/controller/home.controller.php (programme logique devant appeler la « view »)

Les fichiers « models » et « views » ainsi que « controllers » sont liés ensemble notamment à l’aide du « router PHP ».

Toutes les classes générales (utilisées dans plusieurs « controllers ») devront-être rangés dans le dossier dédié nommé « class » (situé dans le dossier « src »).

# 3 – Convention de nommage

## 3.1 - Généralités

Nous utiliserons l’**anglais** et **uniquement des caractères alphanumériques** (chiffre et lettres **non-accentuées**) pour l’ensemble des nommages.

Nous commencerons chaque nommage tel qu’il soit par une minuscule. Le 2ème mot (particule) commencera avec une majuscule et sans espace ni « underscore » (tiret bas).

*Exemples:*

*- maClassIdentity*

*- exampleFile.php*

On évitera également les déterminants du type « the ».

*Exemples:*

*- ~~theIdentityCard~~*

*- ~~theExampleFile.php~~*

Et privilégierons le singulier au pluriel.

*Exemple: ~~identityCards.php~~ 🡪 identityCard.php*

## 3.2 – Models, views et controllers

Chaque fichier MCV devra être rangé dans son dossier attitré avec la bonne double extension.

*Exemples:*

*- /model/model/home.model.php*

*- /src/view/home.view.php*

*- /src/controller/home.controller.php*

## 3.3 – Noms réservés

### 3.3.1 – Les fichiers

Les noms de fichiers à 3 chiffres dans les dossiers MVC (controller, model, view) **sont réservés aux pages erreurs** (principalement 2xx, 4xx, 5xx).

Exemple: /src/view/403.view.php doit-être réservé pour la page d’erreur « 403 – Access Forbidden »

### 3.3.2 – Les variables et constantes PHP

#### 3.2.2.1 – Constantes

|  |  |
| --- | --- |
| Constantes | Commentaire |
| MVC | Contient le chemin d’accès des fichiers MVC, mais aussi des classes « génériques » et les fichiers « include » |
|  |  |

#### 3.2.2.2 – Les variables

|  |  |
| --- | --- |
| Variables | Commentaire |
| $db |  |
|  |  |

## 3.4 – Mots clés appréciés

|  |  |  |
| --- | --- | --- |
| Mot-clé | Exemple | Commentaire |
| list | /src/class/userList.class.php | C’est plus évocateur de que « users.class.php » |
| view |  |  |
| search |  |  |

# 4 – La base de données

## 4.1 – Généralités

Pour les noms de tables nous reprendrons la même convention de nommage des fichiers (caractères alphanumériques non accentués uniquement, première lettre en minuscule, premières lettre en majuscule pour les noms composés).

Exemple:

- MysqlTable

- birthPlace

# 5 – PHP

## 5.1 – Créer une page

Pour créer une page il suffit de créer vos fichiers MCV dans leurs dossiers respectifs.

Exemple:

/src/model/cropImage.model.php 🡪 Le « model » (classes respectives à la page)

/src/view/cropImage.view.php 🡪 La « vue » (principalement le HTML)

/src/controller/cropImage.controller.php🡪 Le « controller » (programme logique)

Pour afficher la page ici nous devrons appeler l’adresse « https://monsite.tld/cropImage ».

🡪 Si le controller « cropImage.controller.php » est supprimé le « router » ouvrira le « model » (s’il existe) puis la « view ».

🡪 Si le controller existe il faudra appeler la vue dans le controller (en effet si le « controller » existe le « router » laissera alors le programme logique choisir).

Concernant les autres dossiers, pour rappel:

|  |  |  |
| --- | --- | --- |
| Dossier | Nommage | Commentaire |
| /src/inc/ | \*.inc.php | Fonctions ou vues commune (comme un menu par exemple commun à plusieurs pages) |
| /src/class/ | \*.class.php | Classes PHP communes à plusieurs fichiers |

# 6 – Mods et dépendances

## 6.1 - Apache2

|  |  |
| --- | --- |
| Mod | Commentaire |
| RewriteEngine | Doit-être **requis** |

## 6.2 - PHP mods

|  |  |
| --- | --- |
| Mod | Commentaire |
| Imagick | Fortement **recommandé** |
| curl | Fortement **recommandé** |

# 7 – Liens

|  |  |
| --- | --- |
| URL | Commentaire |
| <https://cyril.ovh> | Site du fondateur du projet |
| <https://www.apachefriends.org/> | *Mettez en place votre serveur maison facilement sous Windows avec* XAMPP (Apache, MySQL, PHP, PhpMyAdmin) |
| <https://visualstudio.microsoft.com/> | Visual Studio*: éditeur de codes complet et gratuit* |