



Doc. Admin. – Canap-Gest

« CANAP-GEST » APPLICATION WEB DE GESTION DE
CANDIDATURES D'APPRENTISSAGE EPFL –
DOCUMENTATION ADMINISTRATEUR

Sommaire

SOMMAIRE	1
1 INSTALLATION	2
1.1 DÉPENDANCES EN DÉVELOPPEMENT	2
1.2 DÉPENDANCES EN PRODUCTION	2
2 CONFIGURATION	2
2.1 ENVIRONNEMENT DE DÉVELOPPEMENT	2
2.2 PRÉPARATION À LA PRODUCTION	3
2.3 STRUCTURE	5
2.4 FICHIERS	5
2.5 BASE DE DONNÉES	7

1 Installation

1.1 Dépendances en développement

L'environnement de développement pour faire tourner l'application nécessite les dépendances suivantes :

- WampServer (avec PHP 7.2) (<http://www.wampserver.com/>)
- Composer (<https://getcomposer.org/>)
- Node.js (<https://nodejs.org/en/>)
- @vue/cli (<https://cli.vuejs.org/>)
- Git (<https://git-scm.com/>)

1.2 Dépendances en production

Le serveur de production devra être équipé de :

- Service Web (IIS, apache etc.)
- PHP 7.2
- Composer

2 Configuration

2.1 Environnement de développement

Pour commencer, clonez la dernière version stable du projet en local :

```
C:\..\> git clone https://c4science.ch/source/canapgest.git
```

Pour faire tourner l'application localement dans le cadre du développement, il faut établir quelques configurations. Pour commencer il faut créer la base de données, pour ce faire il suffit d'exécuter le script *create_db_script.sql* qui se trouve en annexe, puis d'exécuter la requête suivante, en modifiant d'abord la valeur <password> :

```
USE canap_db;  
CREATE USER 'canap-gest-user'@'localhost' IDENTIFIED BY '<password>';  
GRANT SELECT, UPDATE, INSERT, DELETE ON * TO 'canap-gest-user'@'localhost';
```

L'utilisateur 'canap-gest-user' dispose uniquement des opérations dont il a besoin pour les interactions avec la DB.

Au niveau de l'API, renommer le fichier API/.env.example en .env, puis remplir les champs ainsi :

```
APP_ENV=local
APP_DEBUG=true
APP_KEY=<chaîne de caractère aléatoire>
APP_TIMEZONE=UTC

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=canap_db
DB_USERNAME=canap-gest-user
DB_PASSWORD=<mot de passe précédemment choisi>

CACHE_DRIVER=file
QUEUE_DRIVER=sync

JWT_SECRET==<chaîne de caractère aléatoire>
```

La configuration de l'application vue est optionnelle. L'application a originellement été développée en utilisant la configuration suivante, dans le fichier Site/vue.config.js

```
module.exports = {
  devServer: {
    host: 'canap-gest-dev.local',
    port: 8080,
    https: true
  }
}
```

Pour utiliser un nom d'hôte comme ici, il faut éditer la configuration dns locale (fichier host) se trouvant dans C:\Windows\System32\drivers\etc\hosts et y ajouter les lignes suivantes :

```
127.0.0.1 canap-gest-dev.local
::1 canap-gest-dev.local
```

2.2 Préparation à la production

Afin de préparer la production, certains éléments définis lors du développement doivent être modifiés.

Pour commencer, clonez la dernière version stable du projet en local :

```
C:\..\> git clone https://c4science.ch/source/canapgest.git
```

Au niveau de l'interface, il suffit uniquement de modifier la configuration de Axios afin d'utiliser l'url en production de l'API :

Il faut donc éditer le fichier Site/src/plugins/axios.js, et de modifier l'instance Axios ainsi :

```
const instance = axios.create({
  headers: { 'Authorization': "Bearer " + localStorage.getItem('stored_token')
},
  baseURL: 'https://canap-gest.epfl.ch/api'
})
```

Pour finir, optimiser et générer l'application Vue.js pour la production :

```
C:\..\> cd Site
C:\..\Site> npm install
C:\..\Site> npm run build
```

Le résultat se trouve comme indiqué dans le dossier « dist ».

Au niveau de l'API il faut modifier plusieurs éléments : tout d'abord, modifier l'URL de redirection Tequila.

Pour ce faire, éditer le fichier API/app/http/Controllers/AuthController.php en remplaçant

```
$oClient->SetApplicationURL('http://canap-gest-dev.local:8080');
                                par
```

```
$oClient->SetApplicationURL('https://canap-gest.epfl.ch');
```

Renommer ensuite le fichier API/.env.example en .env, puis remplir les champs ainsi :

```
APP_ENV=local
APP_DEBUG=false
APP_KEY=<chaîne de caractère aléatoire>
APP_TIMEZONE=UTC

DB_CONNECTION=mysql
DB_HOST=mysql-fac.epfl.ch
DB_PORT=33003
DB_DATABASE=canap_db
DB_USERNAME=<username>
DB_PASSWORD=<password>

CACHE_DRIVER=file
QUEUE_DRIVER=sync

JWT_SECRET==<chaîne de caractère aléatoire>
```

Les informations de connexion mySQL se trouvent dans le KeePass ENAC-IT2.

Installer ensuite les dépendances de l'API :

```
C:\..\> cd API
C:\..\Site> composer install
```

2.3 Structure

La structure du site, depuis sa racine web doit être structuré de cette manière :

- api
- www

Le dossier api contient donc le contenu du dossier « API », le www contient les fichiers obtenus après avoir effectué le « build » de l'application Vue.js (dossier « dist »).

Le dossier www doit être configuré en tant que destination par défaut.

2.4 Fichiers

Il faut ensuite ajouter quelques fichiers de configuration pour assurer le bon fonctionnement de l'application. Les exemples suivants sont valables pour une utilisation de IIS en tant que service Web.

Ajouter à la racine du dossier api, un fichier « web.config » contenant :

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <system.webServer>
    <handlers>
      <remove name="PHP_7.2" />
      <add name="PHP_7.2" path="*.php"
verb="GET,HEAD,POST,DEBUG,PUT,DELETE,PATCH,OPTIONS" type=""
modules="FastCgiModule" scriptProcessor="C:\Program Files (x86)\PHP\v7.2\php-
cgi.exe" resourceType="Either" requireAccess="Script" allowPathInfo="false"
preCondition="" responseBufferLimit="4194304" />
    </handlers>
    <rewrite>
      <rules>
        <rule name="Silex Front Controller" stopProcessing="true">
          <match url="^(.*)$" ignoreCase="false" />
          <conditions logicalGrouping="MatchAll">
            <add input="{REQUEST_FILENAME}" matchType="IsFile"
ignoreCase="false" negate="true" />
          </conditions>
        </rule>
      </rules>
    </rewrite>
  </system.webServer>
</configuration>
```

```
        <action type="Rewrite" url="/api/public/index.php"
appendQueryString="true" />
    </rule>
</rules>
</rewrite>
</system.webServer>
</configuration>
```

Cette configuration va rediriger toutes les requêtes API vers le routeur Lumen.

Ajouter ensuite à la racine du dossier www, un fichier « web.config » contenant :

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <system.webServer>
    <modules runAllManagedModulesForAllRequests="true">
      <remove name="WebDAVModule" />
    </modules>
    <defaultDocument enabled="true">
      <files>
        <clear />
        <add value="index.html" />
      </files>
    </defaultDocument>
    <rewrite>
      <rules>
        <clear />
        <rule name="Redirect to https" stopProcessing="true">
          <match url=".*" />
          <conditions>
            <add input="{HTTPS}" pattern="off" ignoreCase="true"
/>
            </conditions>
            <action type="Redirect"
url="https://{HTTP_HOST}:8443{REQUEST_URI}" redirectType="Permanent"
appendQueryString="false" />
          </rule>
        </rules>
      </rewrite>
      <handlers>
        <remove name="WebDAV" />
        <add name="PHP_7.2" path="*.php" verb="GET,HEAD,POST" type=""
modules="FastCgiModule" scriptProcessor="C:\Program Files (x86)\PHP\v7.2\php-
cgi.exe" resourceType="Either" requireAccess="Script" allowPathInfo="false"
preCondition="" responseBufferLimit="4194304" />
      </handlers>
    </system.webServer>
  </configuration>
```

Cette deuxième configuration permet de servir l'application si l'URL n'est pas une requête vers l'API.

2.5 Base de données

La base de données en production se trouve sur l'infrastructure MySQL EPFL (*mysql-fac.epfl.ch*). Les informations de connexion se trouvent dans le KeePass ENAC-IT2. Seul le serveur web *enacit2web1.epfl.ch* est autorisé à se connecter à ce serveur MySQL.

Pour créer la base de données, se connecter à *mysql-fac.epfl.ch*, puis exécuter le script *create_db_script.sql* qui se trouve en annexe.