## Documentation technique

Ce site web est fait pour être déployé sur un serveur Linux.

### Config

Les fichiers d'environnement sont des fichiers qui contiennent des informations qui doivent être changées en fonction de l'environnement dans lequel l'application tourne.

Les fichiers terminant par `.dist` sont des fichiers d'exemple qu'il faut copier en enlevant le .dist pour déployer l'application.

`.env` Contient les ports des services

`db.env` Contient les informations que la base de données utilise pour les mots de passe et les utilisateurs

**ATTENTION** ces informations permettent aussi d'accéder à la base de données avec Adminer, il faut donc faire attention à ne pas les divulguer, sinon la base de données risque d'être attaquée.

Le fichier `./api/config/db.ini` contient les informations utilisées par l'API pour se connecter à la base de données, elles doivent être cohérentes avec ce qui est dans `db.env`

Le fichier `secret.env` contient le secret utilisé par le token JWT

`./Front/src/environment.ts` contient l'adresse de l'API utilisée par le Front-end pour se connecter à l'API, c'est l'URL de l'endroit où votre serveur API est déployé.

### Docker

L'application utilise des conteneurs Docker pour déployer l'application. Une fois que les fichiers de configuration sont copiés, on peut passer au lancement des conteneurs.

On commence par installer les dépendances PHP avec la commande

```
docker compose run php sh -c "composer install && composer upgrade"
```

Puis on lance les conteneurs avec la commande

```
docker compose up -d --remove-orphans;`
```

Une fois les conteneurs lancés, si la base de données n'est pas initialisée, on peut le faire avec la commande

```
docker compose exec amap.db bash -c "psql -U user amap < /var/sql/dump_2802.sql"
```

Cela insérera les tables ainsi que les données de base de l'application.
Normalement à ce moment-là l'application est fonctionnelle.

## Maintenance

Pour maintenir l'application, il vous faut des connaissances techniques différentes en fonction de ce que vous voulez modifier.
Je ne rentrerai pas dans les détails du fonctionnement de l'application, mais je vais donner les grandes lignes.
Le site fonctionne avec un front-end en Angular et une API en PHP, qui se connecte à une base de données PostgreSQL. L'application front-end fait des requêtes HTTP à l'API qui fait des requêtes SQL à la base de données pour récupérer les données afin de les afficher.

**Front-end**
Le front-end est fait avec Angular, il est dans le dossier `./Front`
Si vous changez quelque chose dans le front-end, il faut relancer le conteneur front-build avec la commande

```
docker compose up front-build
```

Cela aura pour effet de rebuilder l'application, qui est servie statiquement par un serveur Nginx dans le conteneur front-web.

Chaque composant est divisé en 3 fichiers, un fichier HTML pour le contenu, un fichier CSS pour le style et un fichier TS pour le comportement.

Si vous voulez modifier certaines parties du texte affichées, comme les titres des menus ou les textes des boutons, vous devez aller modifier les fichiers HTML.

Je vous conseille d'utiliser un outil de recherche textuelle sur plusieurs fichiers pour trouver ce que vous cherchez.

Si vous voulez changer les couleurs de l'application, les couleurs principales sont dans le fichier styles.scss. Structure de fichier:

Si vous voulez rajouter de nouveaux composants, il faut le faire au travers de l'outil en ligne de commande, il modifiera des fichiers de configuration tout seul et c'est très pratique

```
└── Front
    ├── public //contient les différents assets statiques qui ne changent pas, comme
les logos
    └── src // contient le code source de l'application
        └── app
            ├── Components //Contient les composants de l'application
            ├── Guard //Contient les guards
            ├── Interfaces //contient la structure des objets typés custom
TypeScript, utile pour les objets de l'API
            ├── router //Config des routes,
            └── Services //service d'appel à l'API en fonction du sujet appelé
                └── store //store d'auth pour conserver le token d'auth
```

**Liens docs:**
Angular Composant
Objet typé custom TypeScript
Guards
CLI

**Back-end**
Le back-end est dans le fichier `./api` Il est fait en PHP avec la librairie Slim pour le routing et Doctrine pour l'accès base de données.
Vous n'avez pas besoin de recompiler le back-end.
Pour la configuration, tout est dans le dossier config, il contient le bootstrap et les déclarations de dépendance pour l'injection de dépendances Injection de dépendance. Il est important de noter que l'application suit (du mieux qu'elle peut) le principe d'architecture hexagonale, ce qui veut dire que les différentes parties de l'application sont isolées.

Donc le côté applicatif, c'est là où on reçoit la requête HTTP et où on renvoie le JSON.
Le métier (core) c'est là où réside la logique de l'application (très peu dans notre cas).
L'infrastructure c'est l'accès aux services extérieurs, ici simplement la base de données.

```
├── api
│   ├── config //fichiers de configuration (bootstrap, routes, dépendances)
│   ├── src // fichiers source
│   │   ├── application //face publique de l'application
│   │   │   └── action //les actions se chargent de traiter la requête, et de
retourner les bonnes données
│   │   ├── core
│   │   │   ├── dto //format des données entre l'application et le core
│   │   │   ├── entities //format des données dans la base de données
│   │   │   └── service //là où la logique de l'application est
│   │   ├── infrastructure //l'accès à la base de données
│   │   │   ├── entities
│   │   │   └── repository //chaque classe s'occupe d'une différente partie de la BD
│   │   └── middleware //middleware pour l'authentification et l'autorisation d'accès
aux ressources
```

**Base de données**

Le site ne possède pas de système d'administration, si vous voulez enlever des recettes, il faudra le faire directement sur la base de données.

Pour ça il y a un site Adminer dans le conteneur. Il vous permet de vous connecter à la base de données. **ATTENTION** il n'y a pas de système de backup, si vous supprimez des données, elles sont perdues.

Pour vous connecter à la base de données, il vous faut entrer les infos suivantes:

- System: PostgreSQL
- Server: amap.db (c'est le nom du service dans le docker-compose)
- Username: le nom d'utilisateur du fichier `db.env`
- Password: le mot de passe du fichier `db.env`
- Database: la database du fichier `db.env` Il est conseillé d'avoir des connaissances en SQL pour modifier la base de données. Cependant il est possible de modifier directement les tables avec les outils d'Adminer sans écrire de requête SQL.