# Documentation du Projet Knowledge

## 1. Introduction

**Knowledge** est un site web développé avec Symfony (version 7.1) et une base de données MySQL.

Il offre une plateforme pour gérer des cours éducatifs et divisés en leçons, avec un système d'authentification pour les utilisateurs et une intégration avec Stripe pour les paiements.

## 2. Prérequis

Avant de commencer, assurez-vous que les éléments suivants sont installés sur votre machine :

- PHP 8.2 ou une version ultérieure.
- Composer pour gérer les dépendances PHP.
- MySQL (ou MariaDB) pour la base de données.
- Symfony CLI (optionnel mais recommandé pour gérer le projet).

## 3. InstallationÉtape 1 : Cloner le projet

```
git clone https://github.com/Tintin4522/knowledge.git
cd knowledge
```

### Étape 2 : Installer les dépendances

```
composer install
```

### Étape 3 : Configurer les variables d'environnement

Copiez le fichier `.env` en `.env.local` et configurez la connexion à votre base de données.

```
DATABASE_URL=« mysql://.
nomutilisateur:motdepasse@127.0.0.1:3306/knowledge »
```

`Remplacer nom utilisateur et motdepasse par vos données de votre configuration local.`

### Étape 4 : Créer la base de données et exécuter les migrations

```
php bin/console doctrine:database:create
php bin/console doctrine:migrations:migrate
```

### Étape 5 : Charger les données de départ (optionnel)

Ajoutez des données initiales si nécessaires via le fichier SQL se trouvant dans le repository Github.

# 4. Structure du Projet

Le projet suit la structure Symfony standard :

- **src/** : Code source (contrôleurs, entités, services, formulaires).
- **templates/** : Templates Twig pour les vues.
- **migrations/** : Fichiers pour les migrations de base de données.
- **public/** : Point d'entrée `index.php`. et le fichier `.htaccess` qui contient les paramètres pour le serveur Apache lors du déploiement
- **Assets/** : Fichier CSS, JS et images

# 5. Dépendances et Packages

### Principaux Packages Symfony

- `symfony/asset` : Gestion des assets (CSS, JS, etc.).
- `symfony/console` : Commandes CLI.• `symfony/security-bundle` : Gestion des utilisateurs et de la sécurité.
- `symfony/twig-bundle` : Templates Twig.
- `symfony/validator` : Validation des données.

### Autres Packages Notables

- `doctrine/orm` : ORM pour gérer les entités.
- `stripe/stripe-php` : Intégration de l'API Stripe pour les paiements.
- `phpunit/phpunit` : Framework de tests.

# 6. Commandes Utiles

- **Lancer le serveur local** :

```
symfony server:start
```

- **Générer une migration** :

```
php bin/console doctrine:migrations:diff
```

- **Exécuter les migrations** :

```
php bin/console doctrine:migrations:migrate
```

- **Vider le cache** :

```
php bin/console cache:clear
```

# 7. Sécurité

Le projet utilise le composant `Security` pour protéger les routes et gérer les utilisateurs. Les règles de sécurité se trouvent dans `config/packages/security.yaml`.

# 8. Intégration Stripe

**Knowledge** intègre l'API Stripe pour les paiements. En mode développement, utilisez les clés de test suivantes :

- **Clé secrète** : `sk_test_VePHdqKTYQjKNInc7u56JBrQ`
- **Clé publique** : `pk_test_oKhSR5nslBRnBZpjO6KuzZeX`

Pour tester les paiements, entrez les informations suivantes :

- Numéro de carte : `4242 4242 4242 4242`
- Date d'expiration : n'importe quelle date future.
- CVC : trois chiffres aléatoires.
- Code postal : n'importe quel code.

# 9. Identifiant de connection

Pour ce connecter en tant qu'administrateur sur le site :

- **E-mail** : admin@example.com
- **Mot de passe** : mdpdeadmin

Pour ce connecter en tant qu'utilisateur sur le site :

- **E-mail** : test@test.fr
- **Mot de passe** : mdpdetest

Ou n'hésitez pas à créer votre propre compte utilisateur directement sur le site sous l'onglet « s'inscrire »