Knowledge Learning - Une plateforme e-learning basée sur Symfony

Ce document fournit une vue d'ensemble du projet *Knowledge Learning*, incluant son contexte, ses fonctionnalités, son architecture technique et les détails de son implémentation.

Sommaire

---

2. Résumé du projet

Mission & Contexte

La société *Knowledge*, spécialisée dans l'édition de livres de formation, souhaite étendre son offre en proposant une plateforme e-learning. Cette plateforme permettra aux clients d'acheter des cours et de les suivre à leur rythme.

L'objectif de ce projet est de développer le **back-end** du site *Knowledge Learning*, comprenant :

- Un système d'authentification avec activation par email.
- Une gestion des rôles (Administrateur / Client).
- Des fonctionnalités e-commerce pour l'achat de formations et de cours.
- Un système de certification pour les utilisateurs complétant un cursus.

Fonctionnalités principales

✔ Gestion des utilisateurs (Inscription, Activation, Connexion, Back-office Admin)
✔ Système e-commerce (Achat de cours et leçons)
✔ Accès restreint (Seuls les utilisateurs activés peuvent acheter)
✔ Suivi de la progression et obtention de certifications
✔ Administration des formations
✔ Paiements sécurisés avec Stripe

Technologies utilisées

- **Back-end** : Symfony 7.2.3, PHP 8.2.12
- **Base de données** : MySQL
- **Front-end** : Twig, Bootstrap

- **Tests** : PHPUnit pour les tests unitaires et fonctionnels
- **Paiements** : Stripe API

---

3. Conception du projet & Référentiel GitHub

Le code source, les étapes d'installation et la documentation sont disponibles sur GitHub.

🔗 **Dépôt GitHub** : *https://github.com/FeuMangue7639/knowledge-learning*

Lien du site : http://knowledge-learning.alwaysdata.net/

Structure du projet

knowledge-learning/
├── **assets/** → Fichiers frontend (CSS, JS, images)
├── **bin/** → Scripts exécutables Symfony
├── **config/** → Fichiers de configuration Symfony
├── **migrations/** → Migrations Doctrine
├── **public/** → Point d'entrée pour le serveur web
├── **src/** → Code principal de l'application
├── **templates/** → Fichiers Twig pour les vues frontend
├── **tests/** → Tests unitaires et fonctionnels
├── **var/** → Cache et logs
├── **vendor/** → Dépendances PHP installées via Composer
├── **.env** → Configuration de l'environnement
├── **composer.json** → Dépendances gérées par Composer
├── **symfony.lock** → Fichier de verrouillage d'installation Symfony
└── **README.md** → Guide d'installation du projet

---

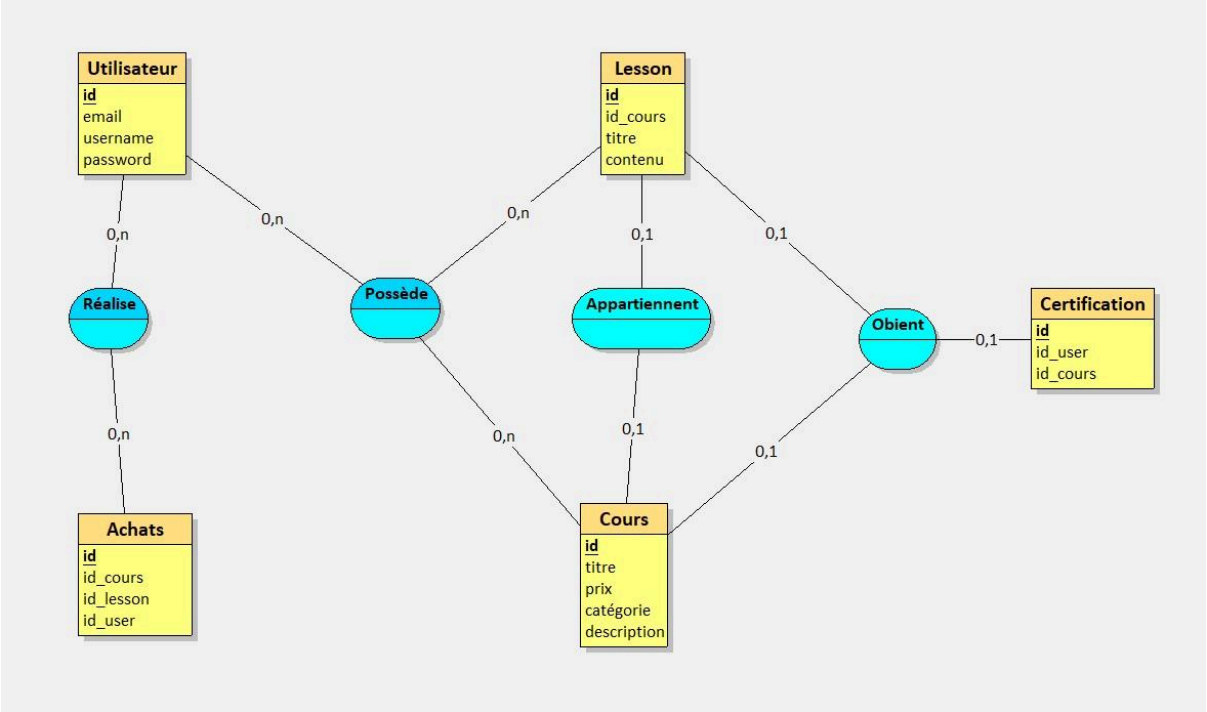
4. Guide d'installation & de déploiement

4.1 Installation en local

Pré-requis

- PHP 8.2.12
- Composer
- Symfony CLI
- MySQL
- Node.js & npm (pour compiler les assets, si nécessaire)

4.2 MPD de la BDD



📌 Entités principales :

1. **Utilisateur** : Représente les apprenants ou les utilisateurs de la plateforme.
    - Attributs : `id`, `email`, `username`, `password`.
2. **Cours** : Contient les cours disponibles sur la plateforme.
    - Attributs : `id`, `titre`, `prix`, `catégorie`, `description`.
3. **Lesson** : Correspond aux leçons d'un cours.
    - Attributs : `id`, `id_cours` (référence vers `Cours`), `titre`, `contenu`.
4. **Achats** : Gère les cours achetés par les utilisateurs.
    - Attributs : `id`, `id_cours`, `id_lesson`, `id_user`.
5. **Certification** : Indique qu'un utilisateur a obtenu une certification pour un cours.
    - Attributs : `id`, `id_user`, `id_cours`.

🔗 Relations :

1. **Réalise (Utilisateur ↔ Achats)** :
    - Un utilisateur peut réaliser plusieurs achats.
    - Un achat appartient à un seul utilisateur.
    - (Cardinalité : `0,n` côté `Utilisateur`, `0,n` côté `Achats`).
2. **Possède (Utilisateur ↔ Cours)** :
    - Un utilisateur peut posséder plusieurs cours.
    - Un cours peut être possédé par plusieurs utilisateurs.
    - (Cardinalité : `0,n` côté `Utilisateur`, `0,n` côté `Cours`).
3. **Appartiennent (Lesson ↔ Cours)** :
    - Une leçon appartient à un seul cours.
    - Un cours peut contenir plusieurs leçons.
    - (Cardinalité : `0,1` côté `Lesson`, `0,n` côté `Cours`).
4. **Obtient (Utilisateur ↔ Certification)** :
    - Un utilisateur peut obtenir plusieurs certifications.
    - Une certification est liée à un seul utilisateur et un seul cours.
    - (Cardinalité : `0,1` côté `Utilisateur`, `0,1` côté `Cours`).

Étapes d’installation

1️⃣ **Cloner le projet**

git clone https://github.com/your-repo/knowledge-learning.git

cd knowledge-learning

2️⃣ **Installer les dépendances**

composer install

3️⃣ **Configurer la base de données** dans le fichier `.env`

```
DATABASE_URL="mysql://utilisateur:motdepasse@127.0.0.1:3306/knowledge-learning"
```

4️⃣ **Créer et migrer la base de données**

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

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

5️⃣ **Charger les Fixtures (Données de test)**

Avant de démarrer le serveur, il est utile d’ajouter des **données de test** à la base de données.
Assure-toi que les fichiers **AppFixtures** et **CourseLessonFixtures** sont bien créés dans le dossier `src/DataFixtures/`.

Puis, exécute la commande suivante pour charger les données :

```
php bin/console doctrine:fixtures:load
```

6️⃣ **Démarrer le serveur Symfony**

Enfin, il faut lancer le serveur avec la commande suivante :

```
symfony server:start
```

4.2 Déploiement sur AlwaysData

Transfert des fichiers

- Envoyer tous les fichiers du projet dans le dossier `www/` via **FileZilla** ou **SSH**.

Mise à jour de la configuration de la base de données

Dans le fichier `.env`, modifier la connexion avec les informations AlwaysData :

DATABASE_URL="mysql://utilisateur:motdepasse@mysql-knowledge-learning.alwaysdata.net/nom_de_la_base"

Vérification du site

- S'assurer que le dossier `public/` est défini comme **racine du site** sur AlwaysData.
- Ouvrir l'URL du site pour vérifier son bon fonctionnement.

5. Documentation du code

5.1 Authentification des utilisateurs

- Inscription avec activation par email
- Connexion avec mot de passe sécurisé

📌 **Fichier :** `src/Controller/UserController.php`

```
/**
 * Inscription utilisateur avec activation par email.
 *
 * @Route("/register", name="app_register")
 */
public function register(Request $request, UserPasswordHasherInterface $passwordHasher, EntityManagerInterface $em): Response
{
    // Gestion du formulaire d'inscription
}
```

5.2 Paiement avec Stripe

- Gestion du paiement et redirection après succès
- Utilisation de l'API Stripe pour sécuriser les transactions

📌 **Fichier :** `src/Controller/PaymentController.php`

```
/**
 * Création d'une session de paiement Stripe.
 *
 * @Route("/checkout", name="app_stripe_payment", methods={"POST"})
 */
public function checkout(SessionInterface $session, CourseRepository $courseRepo, LessonRepository $lessonRepo): JsonResponse
{
        Stripe::setApiKey($this->getParameter('stripe_secret_key'));

        // Création d'une session Stripe avec les éléments du panier
}
```

5.3 Gestion des cours et leçons

- Les administrateurs peuvent ajouter/modifier des cours
- Les utilisateurs peuvent acheter et accéder aux contenus

📌 **Fichier :** `src/Controller/CourseController.php`

```
/**
 * Afficher la liste des cours disponibles.
 *
 * @Route("/courses", name="app_courses")
 */
public function listCourses(CourseRepository $courseRepo): Response
```

```
{
        $courses = $courseRepo->findAll();

        return $this->render('courses/index.html.twig', ['courses' => $courses]);

}
```

6. Informations complémentaires

Tests

Toutes les fonctionnalités principales sont couvertes par des tests unitaires et fonctionnels.

✅ Lancer les tests en local

php bin/phpunit

Sécurité

- Hashage des mots de passe avec bcrypt
- Gestion des rôles et restrictions d'accès
- Validation des entrées des formulaires

Améliorations futures possibles

- Ajout du suivi de progression pour chaque utilisateur
- Mise en place d'un système de commentaires et de discussions
- Amélioration du design avec des animations et une meilleure expérience utilisateur

---

Conclusion

Ce document couvre tous les aspects essentiels du projet *Knowledge Learning*. Il détaille ses fonctionnalités, son déploiement, sa configuration de base de données et sa documentation technique, garantissant une prise en main fluide.