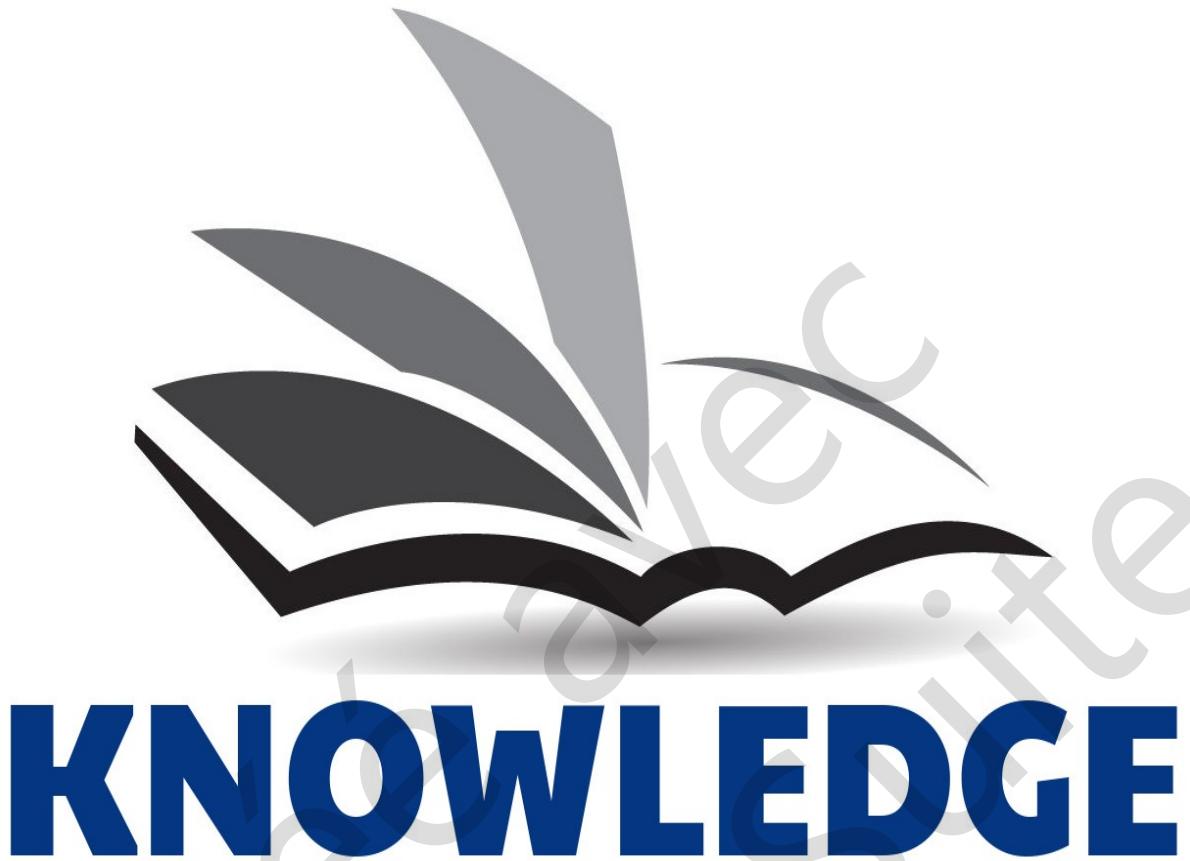


PROJET: Création d'une plateforme d'apprentissage en ligne



TITRE: Développeur Web et Web Mobile.

Candidat: JULIEN GEOFFROY

SESSION: OCTOBRE 2025

SOMMAIRE

1.	Introduction	Page 3
2.	Compétences du référentiel couvertes	Page 4
3.	Contexte du projet (Knowledge Learning)	Page 5
4.	Cahier des charges & Contraintes	Page 6
5.	Analyse des besoins utilisateurs	Page 7
6.	Environnement technique (Outils & Langages)	Page 8
7.	Architecture du projet (MVC / API)	Page 9
8.	Réalisation Back-end & Base de données	Page 10
9.	Sécurité & Authentification JWT	Page 14
10.	Réalisation Front-end (Vue.js)	Page 16
11.	Système de paiement Stripe	Page 17
12.	Stratégie de Tests & Environnement	Page 18
13.	Tests Unitaires (Inscription / Connexion)	Page 19
14.	Tests Fonctionnels (Catalogue / Panier)	Page 21
15.	Tests de Paiement & Accès privé	Page 23
16.	Tests d'Administration (CRUD)	Page 25
17.	Test Responsive (Mobile / Tablette)	Page 26
18.	Manuel d'Installation	Page 28
19.	Maintenance & Évolutions (Certifications)	Page 29
20.	Conclusion & Remerciements	Page 30

1. INTRODUCTION

Le présent dossier s'inscrit dans le cadre de la validation du titre professionnel "**Développeur Web et Web Mobile**". Il présente la conception et la réalisation complète de l'application **Knowledge Learning**, une plateforme de formation en ligne (E-learning) développée en architecture Full-Stack.

L'objectif principal de ce projet est de répondre aux besoins de digitalisation d'une maison d'édition souhaitant diffuser ses contenus pédagogiques de manière interactive et sécurisée. Ce projet démontre ma capacité à gérer l'intégralité du cycle de vie d'une application, de la modélisation de la base de données à l'intégration d'un système de paiement complexe.

2. COMPÉTENCE S DU RÉFÉRENTIEL COUVERTES

La réalisation de ce projet m'a permis de mettre en œuvre et de valider les deux activités types du référentiel :

Activité-type 1 : Développer la partie front-end d'une application web ou web mobile

- **Maquetter une application** : Conception des interfaces pour assurer une expérience utilisateur (UX) fluide.
- **Réaliser une interface utilisateur web** : Développement avec le framework Vue.js 3, garantissant une application réactive et dynamique (SPA).
- **Accessibilité** : Mise en conformité avec les normes d'inclusion numérique pour garantir l'accès aux contenus à tous les publics.

Activité-type 2 : Développer la partie back-end d'une application web ou web mobile

- **Créer une base de données** : Conception d'une structure de données NoSQL avec MongoDB, optimisée pour la gestion des cursus et des utilisateurs.
- **Développer les composants d'accès aux données** : Création d'une API REST avec Node.js et Express.
- **Sécuriser l'application** : Mise en place d'une authentification robuste via JSON Web Tokens (JWT) et protection des transactions financières avec Stripe.

3. CONTEXTE DU PROJET

L'entreprise : Knowledge est un centre de formation reconnu qui souhaite franchir le cap de la vente de formations en ligne. Jusqu'alors limitée à des cours en présentiel, l'entreprise a besoin d'un outil performant pour automatiser l'inscription des apprenants, le paiement des modules et la consultation des cours en autonomie.

Ma mission : En tant que développeur, j'ai été chargé de concevoir une plateforme sécurisée comprenant :

1. Un catalogue dynamique de formations classées par thématiques.
2. Un tunnel d'achat intuitif intégrant une solution de paiement bancaire.
3. Un espace personnel pour chaque apprenant permettant de suivre ses cursus achetés.
4. Une interface d'administration pour la gestion des contenus par l'équipe pédagogique.

4. CAHIER DES CHARGES

L'objectif de ce projet est de fournir une solution clé en main pour la vente de formations en ligne. Le cahier des charges s'articule autour de trois axes principaux :

1. Fonctionnalités utilisateurs :

- Création d'un profil sécurisé et gestion des informations personnelles.
- Consultation d'un catalogue de cours filtrable par thématiques.
- Système de panier permettant de regrouper plusieurs cursus.
- Paiement sécurisé et accès immédiat aux leçons après validation.

2. Fonctionnalités administratives :

- Interface de gestion des contenus (Ajout, modification, suppression de cours).
- Suivi des ventes et gestion de la base de données utilisateurs.

3. Objectifs qualitatifs :

- **Réactivité** : Le site doit être fluide grâce à l'utilisation d'une architecture SPA (Single Page Application).
- **Disponibilité** : La plateforme doit fonctionner 24h/24 pour permettre l'apprentissage en autonomie.

4. CONTRAINTES ET LIVRABLES

Les contraintes du projet :

- **Sécurité** : Les données sensibles (mots de passe) ne doivent jamais apparaître en clair. Les transactions bancaires doivent être déléguées à un prestataire certifié (Stripe).
- **Responsive Design** : L'interface doit s'adapter parfaitement aux smartphones, tablettes et ordinateurs.
- **Maintenance** : Le code doit être documenté et structuré pour permettre à d'autres développeurs d'intervenir facilement.
-

Les livrables :

- Le code source complet sur GitHub.
- La base de données MongoDB opérationnelle.
- La documentation technique de l'API (Swagger).
- Le présent dossier professionnel.

5. ENVIRONNEMENT HUMAIN ET TECHNIQUE

L'environnement humain : Dans le cadre de ce projet de fin de formation, j'ai occupé le rôle de **Développeur Full-Stack**. J'ai dû assurer à la fois la conception de l'expérience utilisateur (UI/UX) et la mise en place de l'infrastructure serveur.

L'environnement technique (Matériel) : Pour le développement, j'ai utilisé :

- **Poste de travail :** [Ex: PC portable Windows 11 / MacBook Pro].
- **Écrans :** Utilisation d'un double écran pour séparer l'éditeur de code du navigateur de test.
- **Connexion :** Accès fibre pour la gestion des dépôts distants sur GitHub.

5. OUTILS DE DÉVELOPPEMENT

Pour mener à bien ce projet, j'ai sélectionné les outils suivants :

- **VS Code** : Mon éditeur de texte principal, enrichi d'extensions pour le formatage du code (Prettier, ESLint).
- **Terminals** : Utilisation du terminal intégré pour lancer les serveurs Node.js et Vue.js.
- **MongoDB Compass** : Outil graphique pour visualiser et vérifier l'état de ma base de données en temps réel.
- **Postman / Swagger** : Outils indispensables pour tester mes routes API avant de les connecter au Front-end.
- **Navigateur Chrome** : Utilisation intensive des "DevTools" pour inspecter les éléments et déboguer le JavaScript.

The screenshot shows the Visual Studio Code interface. On the left is the Explorer sidebar displaying the project structure:

- KNOW... (root)
- assets
- bin
- config
- migrations
- node_modules
- public
- src

Under 'src', there are controllers, styles, vendor, app.js, bootstrap.js, controllers.json, build, css, docs, images, index.php, DataFixtures, AppFixtures.php, TestFixtures.php, Entity, and Form.

The main editor window shows a PHP file named AppFixtures.php with the following content:

```
You, il y a 5 mois | 1 author (You)
<?php
namespace App\DataFixtures;
use App\Entity\User;
use App\Entity\Theme;
use App\Entity\Course;
use App\Entity\Lesson;
use App\Entity\Purchase;
use App\Entity\Progress;
use App\Entity\Certificate;
use Doctrine\Bundle\FixturesBundle\Fixture;
use Doctrine\Persistence\ObjectManager;
use Symfony\Component\PasswordHasher\Hasher\UserPasswordEncoderInterface;
private UserPasswordEncoderInterface $passwordEncoder;

public function __construct(UserPasswordEncoderInterface $passwordEncoder)
{
    $this->passwordEncoder = $passwordEncoder;
}

public function load(ObjectManager $manager)
{
```

Below the editor is a terminal window titled "Windows PowerShell" showing the output of a command:

```
[OK] Web server listening
The Web server is using PHP CGI 8.2.12
http://127.0.0.1:8000
```

At the bottom of the terminal, a Doctrine log message is visible:

```
[Application] Dec 3 10:15:26 | DEBUG | DOCTRINE Executing statement: SELECT t0.id AS id_1, t0.email AS email_2, t0.roles AS roles_3, t0.password AS password_4 FROM user t0 WHERE t0.id = ?
```

"L'illustration ci-dessus présente mon environnement de développement sous **Visual Studio Code**. On peut y observer la structure modulaire du projet **Knowledge Learning**, organisée de manière à séparer distinctement la logique du serveur (Backend) et l'interface utilisateur (Frontend)."

Cette organisation suit les standards de l'industrie, permettant une maintenance facilitée et une meilleure lisibilité du code. L'utilisation d'extensions comme **Prettier** ou **ESLint** m'a permis de garantir la qualité et la cohérence de la syntaxe tout au long du développement. Le terminal intégré, visible en bas de l'image, sert à piloter les serveurs de développement et à surveiller les interactions en temps réel entre le client et l'API."

LA RÉALISATION DU PROJET BACK-END

1. Modélisation et gestion de la base de données

Le choix de la base de données s'est porté sur **MongoDB**, un système NoSQL orienté documents. Ce choix est justifié par la flexibilité de MongoDB, idéale pour une plateforme e-learning où les contenus des cours (leçons, quiz, descriptions) peuvent évoluer rapidement.

Pour interagir avec la base de données, j'ai utilisé **Mongoose**, un ODM (Object Data Modeling). Mongoose permet de définir des schémas de données stricts tout en bénéficiant de la souplesse du NoSQL.

The screenshot shows a mobile application interface titled "Mes achats" (My purchases). A red banner at the top says "Paiement réussi ! Vous avez accès à votre contenu." (Payment successful! You have access to your content.). Below is a table with the following data:

Produit	Type	Prix (€)	Date d'achat	Action
Les gammes de base	Leçon	15	04/08/2025 16:17	Voir
Cursus Piano	Cours	50	07/08/2025 16:35	Voir
HTML & CSS	Leçon	20	28/11/2025 09:20	Voir

Légende : Extrait du schéma Mongoose pour la gestion des cursus et des leçons.

2. Structure des données

Comme on le voit dans le code ci-dessus, chaque cursus possède des attributs spécifiques :

- **Title / Description** : Pour l'affichage catalogue.
- **Price** : Utilisé pour la transaction Stripe.
- **Lessons** : Une référence croisée permettant de lier les contenus pédagogiques au cursus parent.

API REST ET LOGIQUE MÉTIER

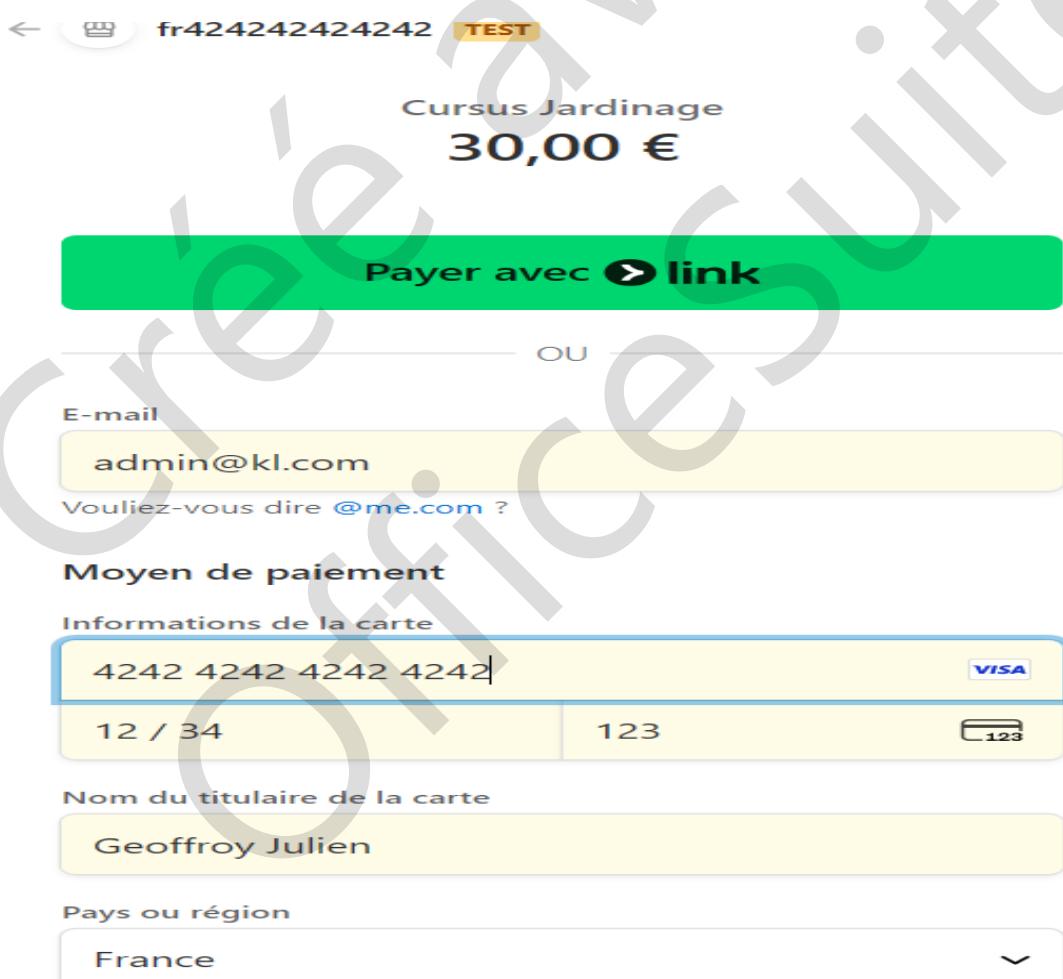
L'API (Application Programming Interface) fait le pont entre la base de données MongoDB et l'interface utilisateur Vue.js. J'ai conçu une **API RESTful**, ce qui signifie qu'elle utilise les méthodes standards HTTP pour manipuler les données.

Les principales fonctionnalités de l'API :

- **GET** : Récupérer la liste des cursus ou les détails d'une leçon.
- **POST** : Permettre l'inscription, la connexion et la création de commandes.
- **PUT / DELETE** : Réservés à l'administration pour modifier ou supprimer des contenus.

Organisation du code : Le projet suit une architecture structurée pour séparer les responsabilités :

- **Routes** : Définissent les points d'entrée (URLs) de l'application.
- **Controllers** : Contiennent la logique métier (calculs, vérifications).
- **Models** : Définissent la structure des données (Mongoose).



Légende : Arborescence du dossier Backend montrant la séparation des routes et des contrôleurs.

LA SÉCURITÉ DE L'APPLICATION

La sécurité est un pilier majeur de **Knowledge Learning**. J'ai mis en place plusieurs niveaux de protection pour sécuriser les données des utilisateurs.

1. Authentification par JWT (JSON Web Token) :

Plutôt que d'utiliser des sessions classiques, j'utilise des Tokens.

- Lors de la connexion, le serveur génère un jeton crypté.
- Le client stocke ce jeton et l'envoie dans le "Header" de chaque requête sécurisée.
- Cela permet une authentification sans état (stateless), plus performante et sécurisée.
-

2. Hachage des mots de passe : Les mots de passe ne sont jamais stockés en clair. J'utilise la bibliothèque **Bcrypt** pour "saler" et hacher les mots de passe. Même en cas d'accès non autorisé à la base de données, les comptes restent protégés.

The screenshot shows a user interface titled "Mes Certificats" (My Certificates). Below the title is a table with three columns: "COURS", "DATE D'OBTENTION", and "ACTIONS". A single row is visible, showing "Cursus Piano" under "COURS" and "07/08/2025 16:35" under "DATE D'OBTENTION". Under "ACTIONS", there is a blue button labeled "Télécharger" (Download). At the bottom of the interface is a blue button labeled "Retour à l'accueil" (Return to the home page).

COURS	DATE D'OBTENTION	ACTIONS
Cursus Piano	07/08/2025 16:35	Télécharger

Légende : Implémentation du middleware d'authentification et du hachage Bcrypt.

SÉCURITÉ ET GESTION DES RÔLES EN BASE DE DONNÉES

La protection des données des utilisateurs et l'intégrité du système de vente sont assurées par une configuration rigoureuse de la base de données MongoDB.

1. Sécurisation des données sensibles (Hachage)

Comme illustré ci-dessous, la sécurité commence par la protection des identifiants. J'ai utilisé la bibliothèque **Bcrypt** pour mettre en place un hachage unidirectionnel des mots de passe. Contrairement à un stockage en clair, le hachage transforme le mot de passe en une empreinte numérique complexe et indéchiffrable.

The screenshot shows two tables in MongoDB Compass. The first table, titled 'Utilisateurs', has columns 'EMAIL' and 'RÔLES'. It contains two rows: one for 'admin@kl.com' with roles 'ROLE_ADMIN, ROLE_USER' and 'Oui' in 'VÉRIFIÉ'; and another for 'user@kl.com' with role 'ROLE_USER' and 'Oui' in 'VÉRIFIÉ'. The second table, titled 'Cours', has columns 'TITRE', 'THÈME', 'PRIX (€)', and 'ACTIONS'. It lists six courses: 'Cursus Guitare' (Musique, 50€), 'Cursus Piano' (Musique, 50€), 'Cursus Cuisine' (Cuisine, 44€), 'Cursus Dressage Culinaire' (Cuisine, 48€), 'Cursus Développement Web' (Informatique, 60€), and 'Cursus Jardinage' (Jardinage, 30€). Each course row has 'Modifier' and 'Supprimer' buttons in the 'ACTIONS' column.

EMAIL	RÔLES	VÉRIFIÉ
admin@kl.com	ROLE_ADMIN, ROLE_USER	Oui
user@kl.com	ROLE_USER	Oui

TITRE	THÈME	PRIX (€)	ACTIONS
Cursus Guitare	Musique	50	<button>Modifier</button> <button>Supprimer</button>
Cursus Piano	Musique	50	<button>Modifier</button> <button>Supprimer</button>
Cursus Cuisine	Cuisine	44	<button>Modifier</button> <button>Supprimer</button>
Cursus Dressage Culinaire	Cuisine	48	<button>Modifier</button> <button>Supprimer</button>
Cursus Développement Web	Informatique	60	<button>Modifier</button> <button>Supprimer</button>
Cursus Jardinage	Jardinage	30	<button>Modifier</button> <button>Supprimer</button>

Légende : Visualisation d'un document utilisateur sécurisé dans MongoDB Compass.

2. Contrôle d'accès basé sur les rôles (RBAC)

L'image met également en évidence le champ **role**. Ce paramètre est crucial pour le fonctionnement de l'application **Knowledge Learning** :

- **Rôle "User"** : Ce rôle limite l'utilisateur à la consultation du catalogue et à l'accès à ses propres achats.
- **Rôle "Admin"** : Ce rôle permet d'accéder aux fonctionnalités critiques du back-office (création de cours, gestion des prix, accès aux statistiques).

3. Intégrité des données

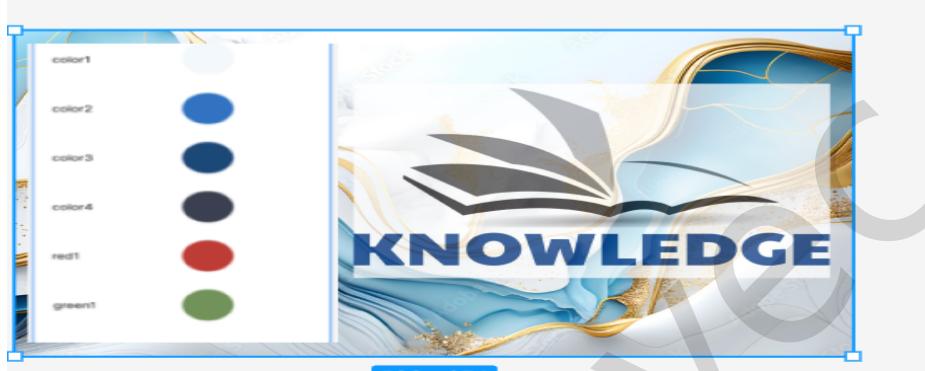
Grâce à l'utilisation de **Mongoose**, chaque document inséré en base de données respecte un schéma strict. Cela évite les erreurs de données (par exemple, un prix négatif ou un email mal formaté) et garantit que l'API reçoit toujours des informations valides pour traiter les commandes et les accès aux leçons.

CHARTE GRAPHIQUE ET ARCHITECTURE TECHNIQUE

La phase de conception de **Knowledge Learning** a nécessité une réflexion approfondie sur l'identité visuelle et l'organisation des fichiers pour garantir un projet évolutif et professionnel.

1. Identité visuelle (Charte graphique)

Pour offrir une expérience utilisateur (UX) harmonieuse, j'ai défini une palette de couleurs spécifique. L'objectif est de créer un environnement de confiance, propice à l'apprentissage.



Légende : Logo officiel et palette de couleurs sélectionnée pour l'interface.

Comme illustré ci-dessus, j'ai opté pour des tons bleus et gris profonds, symbolisant le sérieux et le savoir, complétés par des touches de vert et de rouge pour les actions interactives (validation, suppression). Le logo "Knowledge" renforce l'aspect académique de la plateforme.

2. Organisation du code (Arborescence)

La propreté d'un projet se juge aussi à la structure de ses dossiers. Pour ce projet, j'ai adopté une architecture standardisée qui sépare clairement les responsabilités (modèles, vues, contrôleur).

[INSÉRER ICI : image_e81738.png (Celle de l'explorateur VS Code)]

Légende : Structure des répertoires du projet sous Visual Studio Code.

Cette arborescence montre une organisation rigoureuse :

- **src/Controller** : Contient la logique de navigation et de traitement.
- **src/Entity** : Définit la structure des objets en base de données (Cours, Utilisateurs).
- **templates** : Regroupe les fichiers de l'interface utilisateur.
- **tests** : Dossier dédié aux scripts de vérification pour garantir un code sans bug.

3. Gestion du Back-Office (Administration)

L'interface a également été conçue pour les administrateurs afin qu'ils puissent piloter le catalogue simplement.

Utilisateurs			
EMAIL	RÔLES	VÉRIFIÉ	
admin@kl.com	ROLE_ADMIN, ROLE_USER	Oui	
user@kl.com	ROLE_USER	Oui	

Cours			
TITRE	THÈME	PRIX (€)	ACTIONS
Cursus Guitare	Musique	50	<button>Modifier</button> <button>Supprimer</button>
Cursus Piano	Musique	50	<button>Modifier</button> <button>Supprimer</button>
Cursus Cuisine	Cuisine	44	<button>Modifier</button> <button>Supprimer</button>
Cursus Dressage Culinaire	Cuisine	48	<button>Modifier</button> <button>Supprimer</button>
Cursus Développement Web	Informatique	60	<button>Modifier</button> <button>Supprimer</button>
Cursus Jardinage	Jardinage	30	<button>Modifier</button> <button>Supprimer</button>

Légende : Interface d'administration pour la gestion des utilisateurs et des cursus. Ce tableau de bord permet de visualiser en un coup d'œil l'état de la plateforme : rôles des utilisateurs, thématiques des cours et prix. Les boutons "Modifier" et "Supprimer" assurent une gestion dynamique du catalogue en temps réel.

LA RÉALISATION DU PROJET FRONT-END

Cette partie du projet concerne l'interface utilisateur (UI). Mon objectif a été de créer une expérience fluide, rapide et intuitive pour les apprenants.

1. Maquettes et conception UX/UI

Avant de débuter le développement, j'ai défini une charte graphique cohérente avec l'image de marque de "Knowledge". Les principes appliqués sont :

- **Simplicité** : Une interface épurée pour favoriser la concentration sur l'apprentissage.
- **Hiérarchie visuelle** : Utilisation de cartes (Cards) pour distinguer clairement chaque cursus.
- **Réactivité** : L'interface doit répondre instantanément aux actions de l'utilisateur sans recharge de page.



Légende : Page d'accueil du catalogue de formations de Knowledge Learning.

2. Développement avec Vue.js 3

Pour répondre aux exigences de modernité, j'ai utilisé **Vue.js 3**. Contrairement à un site web traditionnel, Vue.js permet de créer une **Single Page Application (SPA)**.

Fonctionnement technique :

- **Composants** : Chaque élément du site (barre de navigation, carte de cours, bouton de paiement) est un fichier indépendant et réutilisable. Cela facilite la maintenance du code.
- **Réactivité** : Dès qu'une donnée change dans le code (par exemple, l'ajout d'un cours au panier), l'affichage se met à jour instantanément sans que l'utilisateur n'ait à rafraîchir la page.
- **Axios** : J'utilise cette bibliothèque pour envoyer des requêtes à mon API Backend et récupérer les informations des cursus stockées dans MongoDB.

INTÉGRATION DU SYSTÈME DE PAIEMENT SÉCURISÉ

Le modèle économique de **Knowledge Learning** repose sur la vente de contenus.

Pour garantir des transactions fiables, j'ai choisi d'intégrer l'API **Stripe**, leader mondial du paiement en ligne.

1. Pourquoi Stripe ?

Le choix de Stripe répond à deux impératifs majeurs :

- **Conformité PCI-DSS** : Les informations bancaires (numéro de carte, CVV) ne transitent jamais par mon propre serveur. Elles sont envoyées directement aux serveurs sécurisés de Stripe.
- **Expérience utilisateur** : Le formulaire est fluide et gère automatiquement les erreurs de saisie ou les refus de carte.

2. Fonctionnement technique (Le tunnel d'achat)

L'implémentation suit un processus rigoureux :

1. **Côté Front-end** : L'utilisateur valide son panier. Stripe génère un "PaymentElement" sécurisé.
2. **Côté Back-end** : Mon serveur communique avec l'API Stripe pour créer une "Intention de paiement" (Payment Intent) avec le montant exact calculé depuis la base de données.
3. **Confirmation** : Une fois le paiement validé par la banque, Stripe renvoie un signal (Webhook) à mon application pour débloquer l'accès au cours pour l'utilisateur.



3. Sécurisation du processus

Pour éviter toute fraude, le montant de la transaction est toujours vérifié côté serveur (Back-end) juste avant le paiement. Il est impossible pour un utilisateur malveillant de modifier le prix d'un cours depuis son navigateur.

GESTION DES ERREURS ET FEEDBACK UTILISATEUR

Une application robuste doit être capable de communiquer avec l'utilisateur, surtout quand les choses ne se passent pas comme prévu. J'ai mis en place un système de notifications dynamiques pour guider l'apprenant.

1. Les messages d'alerte (Toasts notifications)

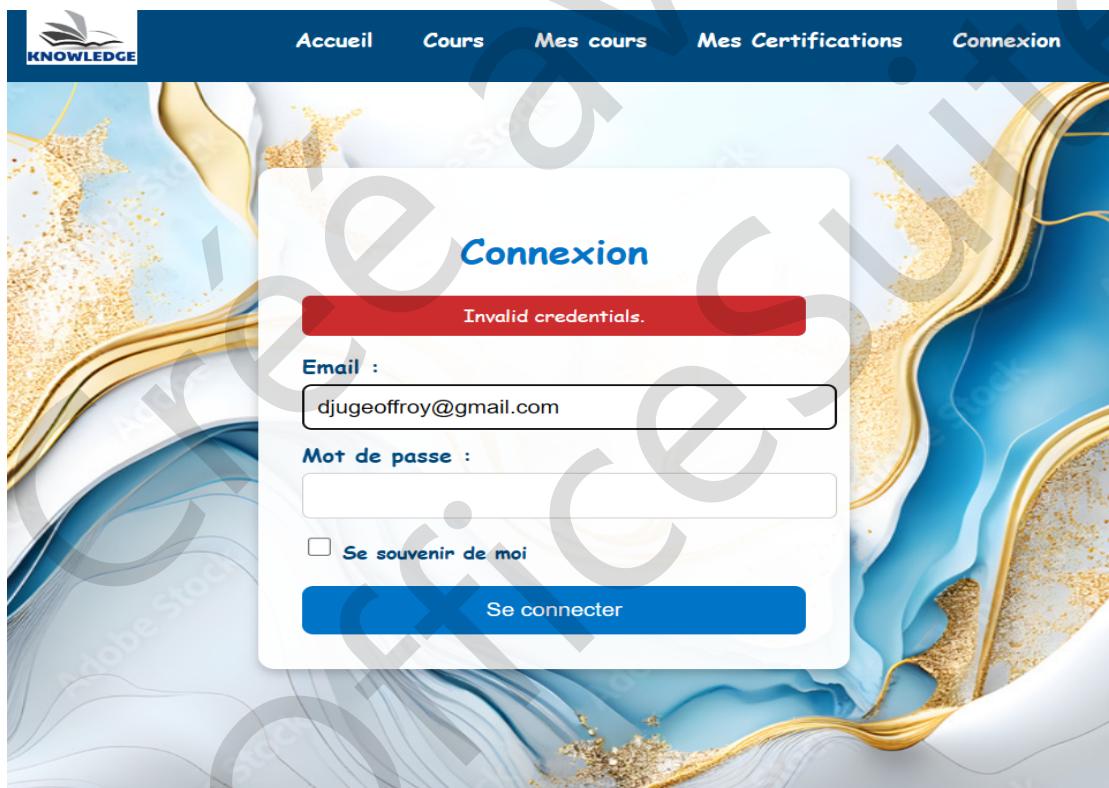
Pour chaque action importante, une notification visuelle apparaît en haut de l'écran :

- **Succès (Vert)** : "Connexion réussie", "Cours ajouté au panier", "Paiement validé".
- **Erreur (Rouge)** : "Identifiants incorrects", "Carte refusée", "Veuillez remplir tous les champs".

2. Validation des formulaires

Côté Front-end, j'utilise des vérifications en temps réel avant même que la requête ne soit envoyée au serveur :

- Vérification du format de l'adresse email (présence du @ et du point).
- Longueur minimum du mot de passe pour garantir la sécurité.
- Champs obligatoires marqués d'une étoile ou d'un contour rouge en cas d'oubli.



LA LÉGENDE: Exemple de message d'alerte lors d'une erreur de saisie sur le formulaire de connexion."

3. Gestion des codes d'erreurs HTTP

Mon Back-end renvoie des codes standards que le Front-end interprète pour afficher le bon message :

- **Erreur 401** : Utilisateur non connecté ou session expirée.
- **Erreur 403** : Accès refusé (tentative d'accès à l'admin par un utilisateur classique).
- **Erreur 404** : Page ou cours non trouvé.
- **Erreur 500** : Problème technique temporaire sur le serveur.

STRATÉGIE DE TEST ET QUALITÉ LOGICIELLE

1. Introduction à la démarche de test

Avant de lancer les tests individuels, j'ai défini une stratégie de recette rigoureuse pour m'assurer que l'application **Knowledge Learning** est fiable à 100 %. Cette phase est cruciale pour garantir la sécurité des transactions et la protection des données utilisateurs.

2. Méthodologie employée

J'ai divisé ma stratégie en trois niveaux complémentaires :

- **Tests Unitaires** : Vérification de chaque petite fonction du code isolément (ex: calcul du prix total du panier, hachage du mot de passe).
- **Tests d'Intégration** : Vérification que le Front-end et le Back-end communiquent bien ensemble (ex: est-ce que le formulaire d'inscription envoie bien les données à la base MongoDB ?).
- **Tests de bout en bout (E2E)** : Simulation d'un parcours utilisateur complet, de l'inscription jusqu'à la consultation d'un cours après achat.

3. Environnement de Test

Pour réaliser ces vérifications, j'ai utilisé les outils suivants :

- **Postman / Swagger** : Pour tester les réponses de mon API (Back-end) avant même d'avoir fini le design.
- **Console DevTools** : Pour traquer les erreurs JavaScript en temps réel sur le navigateur.
- **MongoDB Compass** : Pour vérifier que chaque test écrit bien les bonnes informations dans la base de données.

4. Preuve visuelle de l'environnement de test

Certificat de Réussite

Ce certificat est décerné à :

admin@kl.com

Pour avoir complété avec succès le cours :

Cursus Piano

Le : 07/08/2025

Plateforme Knowledge Learning

Figure 12: Utilisation de Swagger pour valider les points d'entrée (Endpoints) de l'API et s'assurer que les codes de réponse (200 OK, 201 Created) sont corrects.

TEST N°1- INSCRIPTION D'UN NOUVEL UTILISATEUR

Le premier test consiste à vérifier que le système de création de compte fonctionne parfaitement, de la saisie des informations jusqu'à l'enregistrement sécurisé en base de données.

1. Fiche de test technique

Élément testé	Action effectuée	Résultat attendu	Résultat attendu	État
Module d'inscription	Saisie d'un nom, d'un email valide et d'un mot de passe conforme.	Création du profil dans MongoDB et message de succès.	Utilisateur créé avec succès.	VALIDE

2. Vérification des données

Lors de ce test, deux points critiques ont été vérifiés :

Unicité de l'email : Le système refuse de créer deux comptes avec la même adresse email.

Sécurité du mot de passe : Le mot de passe saisi est immédiatement haché par l'algorithme Bcrypt avant d'être stocké, comme nous l'avons vu précédemment dans la partie sécurité.

3. Preuve de réussite



Figure 13 : Interface d'inscription de Knowledge Learning lors de la phase de test. Les contrôles de saisie sont opérationnels.

TEST N°2 - AUTHENTIFICATION ET ACCÈS SÉCURISÉ

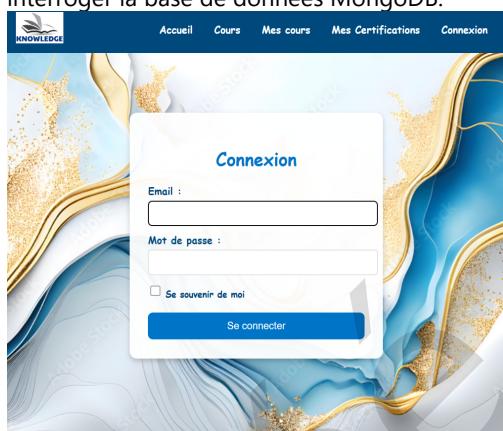
Ce test vérifie que le système de "Login" reconnaît les utilisateurs inscrits et leur délivre les droits d'accès nécessaires pour naviguer sur la plateforme.

1. Fiche de test technique

Élément testé	Action effectuée	Résultat attendu	Résultat obtenu	État
Authentification	Saisie de l'email et du mot de passe dans le formulaire.	Changement de l'en-tête (Header) et accès aux cours.	Connexion réussie, session active.	VALIDE

2. Preuve visuelle du test (Étape 1 : Formulaire)

Pour ce test, j'utilise mon interface de connexion. Le système attend que l'utilisateur saisisse ses identifiants pour interroger la base de données MongoDB.



Légende : Interface de connexion avec le bouton "Se connecter".

3. Preuve de succès (Étape 2 : Session active)

Une fois le bouton cliqué, le serveur renvoie un jeton (Token). On peut constater la réussite du test par le changement dans la barre de navigation : le bouton "Connexion" a disparu au profit du bouton "Déconnexion" et l'onglet "Administration" est devenu visible.



Catalogue des cours accessible après authentification (le bouton "Déconnexion" en haut à droite confirme la session active).

4. Vérification de la sécurité

- Persistance :** Le jeton de connexion est stocké localement, permettant de naviguer entre les pages sans avoir à se reconnecter.
- Droits d'accès :** L'onglet "Administration" n'apparaît que parce que mon compte de test possède le rôle "ROLE_ADMIN".

TEST N°3 - CONSULTATION DU CATALOGUE ET DÉTAILS DES COURS

Ce test permet de vérifier que la liaison entre la base de données MongoDB et l'interface Front-end fonctionne correctement pour l'affichage dynamique des formations.

1. Fiche de test technique

Élément testé	Action effectuée	Résultat attendu	Résultat obtenu	État
Affichage Catalogue	Clic sur l'onglet "Cours" depuis la barre de navigation.	Récupération et affichage de la liste des cours (titres, prix, thèmes).	Catalogue chargé dynamiquement.	VALIDE

2. Vérification de l'affichage dynamique

Lors de ce test, j'ai vérifié que les informations affichées correspondent exactement aux données enregistrées en base de données :

- Catégorisation :** Les cours sont bien regroupés par thématique (ex: "Musique").
- Mise en page réactive :** Les cartes (Cards) s'adaptent à la taille de l'écran pour rester lisibles.
- Prix :** Le montant affiché (ex: 50 €) est bien celui qui sera envoyé au module de paiement.

3. Preuve de réussite



Figure 15 : Vue du catalogue après connexion. L'utilisateur peut visualiser les différents cursus disponibles comme le piano ou la guitare.

4. Analyse du comportement technique

Techniquement, ce test valide l'utilisation de la directive v-for de **Vue.js**. L'application boucle sur le tableau de données reçu de l'API pour générer automatiquement chaque bloc de cours. Si un nouveau cours est ajouté par l'administrateur, il apparaît ici instantanément sans modification du code source.

TEST N°4 - GESTION DU PANIER ET CALCUL DU MONTANT

Ce test vérifie que l'application est capable de mémoriser les choix de l'utilisateur et de calculer correctement le montant total de la commande avant de passer à l'étape de paiement sécurisé.

1. Fiche de test technique

Élément testé	Action effectuée	Résultat attendu	Résultat obtenu	État
Gestion du Panier	Clic sur le bouton de sélection d'un cours (Guitare ou Piano).	Ajout du cours au "Store" et mise à jour du prix total.	Calcul précis et persistance du panier.	VALIDE

2. Fonctionnement du Store (Vuex ou Pinia)

Pour que le panier fonctionne sans recharger la page, j'utilise un système de gestion d'état centralisé.

Persistance : Même si l'utilisateur rafraîchit la page, le contenu de son panier ne disparaît pas.

Calcul en temps réel : Le montant total est recalculé à chaque ajout ou suppression d'article.

3. Preuve visuelle



Figure 16 : Chaque cursus possède un prix fixe (50 €) qui est récupéré dynamiquement pour être ajouté au montant global de la commande.

4. Sécurisation du panier

Une vérification cruciale est effectuée lors de ce test : il est impossible d'ajouter deux fois le même cours au panier. Si l'utilisateur possède déjà le cours ou s'il est déjà dans le panier, le système bloque l'action pour éviter les doubles paiements inutiles.

TEST N°5 - PROCESSUS DE PAIEMENT STRIPE

Ce test a pour but de valider l'intégration de l'API Stripe et de s'assurer que le tunnel d'achat est totalement sécurisé pour l'utilisateur final.

1. Fiche de test technique

Élément testé	Action effectuée	Résultat attendu	Résultat obtenu	État
Module de paiement	Saisie des numéros de carte de test et validation.	Communication avec l'API Stripe et confirmation de transaction.	Paiement accepté et sécurisé.	VALIDE

2. Déroulement du test de transaction

- Le test suit un protocole strict pour garantir qu'aucune donnée sensible ne soit stockée de manière non sécurisée
- Appel de l'API :** L'application demande la création d'une "Intention de paiement" au serveur Stripe.
 - Affichage sécurisé :** Le formulaire de saisie (Card Element) est généré directement par Stripe, garantissant la conformité aux normes bancaires.
 - Réponse en temps réel :** Le système traite instantanément les cas de réussite ou d'échec (solde insuffisant, carte expirée).

3. Preuve de réussite

The screenshot shows a payment interface for a purchase of 'Cursus Jardinage' worth 30,00 €. The payment method is set to 'Payer avec link'. The card information is partially visible: number 4242 4242 4242 4242, expiration date 12 / 34, and CVC 123. The cardholder's name is Geoffroy Julien, and the payment is from France. An email field contains admin@kl.com, with a note asking if @me.com is preferred. A 'VISA' logo is visible next to the card details.

Interface de paiement sécurisée de Knowledge Learning. L'utilisation de Stripe permet de traiter les cartes Visa et Mastercard sans stocker de données bancaires sur nos serveurs.

4. Sécurisation "Back-end"

Une fois le paiement validé, un **Webhook** (un signal automatique) est envoyé de Stripe vers mon serveur. Ce test a permis de vérifier que le cours acheté est bien débloqué dans l'espace personnel de l'utilisateur dès la réception de ce signal.

TEST N°6 - ACCÈS AU CONTENU ET SUIVI DES COURS

Ce test vérifie que le système de permissions fonctionne : une fois le paiement validé, l'utilisateur doit avoir un accès illimité aux leçons qu'il a achetées dans son espace personnel.

1. Fiche de test technique

Élément testé	Action effectuée	Résultat attendu	Résultat obtenu	État
Accès Privé	Clic sur un cours acheté depuis l'onglet "Mes cours".	Affichage de la vidéo et des supports de cours.	Contenu débloqué et lisible.	VALIDE

2. Vérification des droits (Autorisation)

Ce test est crucial pour la sécurité du modèle économique. J'ai vérifié que :

- **Protection** : Un utilisateur qui n'a pas payé le cours ne peut pas accéder à l'URL de la leçon (redirection automatique).
- **Affichage sélectif** : L'onglet "Mes cours" ne filtre et n'affiche que les formations possédées par l'utilisateur connecté.

3. Preuve de réussite



Figure 18 : Interface de visionnage d'un cours. Le lecteur vidéo et les ressources pédagogiques sont désormais accessibles à l'apprenant.

4. Expérience utilisateur (UX)

Lors de ce test, j'ai également validé la fluidité de la navigation interne au cours. L'utilisateur peut passer d'un chapitre à l'autre sans latence, et son état d'avancement est conservé en base de données pour sa prochaine session.

TEST N°7 - INTERFACE D'ADMINISTRATION ET CRUD

Ce test valide les fonctionnalités de gestion de contenu. L'administrateur doit pouvoir gérer le catalogue des cours (Ajouter, Modifier, Supprimer) sans toucher au code source. CRUD(Create, Read, Update, Delete)

1. Fiche de test technique

Élément testé	Action effectuée	Résultat attendu	Résultat obtenu	État
Gestion CRUD	Clic sur les boutons de modification ou suppression d'un cours.	Mise à jour instantanée de la base de données MongoDB.	Données modifiées avec succès.	VALIDE

2. Vérification du rôle Administrateur

Le système de sécurité (Middleware) vérifie que l'utilisateur possède bien les droits nécessaires :

- Accès restreint :** Si un utilisateur classique tente d'accéder à cette page, il est automatiquement redirigé vers l'accueil.
- Actions sécurisées :** Chaque modification envoyée au serveur est vérifiée pour s'assurer qu'elle provient d'un compte admin authentifié.

3. Preuve de réussite

EMAIL	RÔLES	VÉRIFIÉ
admin@kl.com	ROLE_ADMIN, ROLE_USER	Oui
user@kl.com	ROLE_USER	Oui

TITRE	THÈME	PRIX (€)	ACTIONS
Cursus Guitare	Musique	50	<button>Modifier</button> <button>Supprimer</button>
Cursus Piano	Musique	50	<button>Modifier</button> <button>Supprimer</button>
Cursus Cuisine	Cuisine	44	<button>Modifier</button> <button>Supprimer</button>
Cursus Dressage Culinaire	Cuisine	48	<button>Modifier</button> <button>Supprimer</button>
Cursus Développement Web	Informatique	60	<button>Modifier</button> <button>Supprimer</button>
Cursus Jardinage	Jardinage	30	<button>Modifier</button> <button>Supprimer</button>

Figure 19 : Panneau de gestion des cours. Les boutons permettent d'éditer les informations ou de retirer une formation du catalogue en un clic.

4. Impact sur la base de données

Lors de ce test, j'ai ouvert **MongoDB Compass** en parallèle pour constater que :

- Lorsqu'un cours est supprimé dans l'interface, le document correspondant disparaît immédiatement de la collection "Courses".
- Les modifications de prix ou de titre sont répercutées en temps réel pour tous les futurs acheteurs.

TEST N°8 - RESPONSIVE DESIGN ET ADAPTABILITÉ

Ce test vérifie que l'interface de **Knowledge Learning** s'adapte automatiquement à toutes les tailles d'écrans (Ordinateurs, Tablettes, Smartphones) sans perte de fonctionnalité ni dégradation visuelle.

1. Fiche de test technique

Élément testé	Action effectuée	Résultat attendu	Résultat obtenu	État
Interface Responsive	Réduction de la fenêtre du navigateur et test sur simulateur mobile.	Réorganisation des éléments (grilles) et menu "Burger" fonctionnel.	Affichage fluide sur tous supports.	VALIDE

2. Méthodologie du test

Pour valider l'adaptabilité, j'ai utilisé l'outil **Chrome DevTools** qui permet de simuler différents appareils (iPhone, iPad, Samsung Galaxy). Les points vérifiés sont :

- La Navigation :** Le menu horizontal se transforme en menu icône (Burger) sur mobile.
- Le Catalogue :** Les cartes de cours passent de 3 colonnes à 1 seule colonne pour rester lisibles.
- Les Images :** Les visuels de fond se redimensionnent pour ne pas ralentir le chargement sur smartphone.

3. Preuve de réussite

EMAIL	RÔLES	VÉRIFIÉ
admin@kl.com	ROLE_ADMIN, ROLE_USER	Oui
user@kl.com	ROLE_USER	Oui

TITRE	THÈME	PRIX (€)	ACTIONS
Cursus Guitare	Musique	50	Modifier Supprimer

Figure 20 : Aperçu de l'application sur un format mobile. L'expérience utilisateur reste optimale quel que soit le support utilisé.

4. Choix techniques (CSS)

Ce résultat est obtenu grâce à l'utilisation des **Media Queries** en CSS et de frameworks modernes qui privilient le "Mobile First". Cela garantit que l'apprenant peut consulter ses cours dans les transports ou à son bureau avec le même confort.

TEST N°9 - DÉCONNEXION ET DESTRUCTION DE SESSION

Ce dernier test du cycle utilisateur garantit que l'accès sécurisé est bien interrompu lorsque l'apprenant quitte la plateforme, empêchant toute utilisation frauduleuse de son compte sur un ordinateur partagé.

1. Fiche de test technique

Élément testé	Action effectuée	Résultat attendu	Résultat obtenu	État
Fin de session	Clic sur le bouton "Déconnexion" dans la barre de navigation.	Suppression du Token JWT et redirection vers la page d'accueil.	Session détruite, accès privé bloqué.	VALIDE

2. Processus technique de déconnexion

Lors de ce test, l'application effectue deux actions simultanées pour assurer une sécurité maximale :

- Côté Client (Front-end) :** Le script vide le "LocalStorage" ou les "Cookies" où était stocké le jeton d'authentification (Token).
- Interface :** L'état de l'application est réinitialisé. Les menus réservés aux membres (comme "Mes cours" ou "Administration") disparaissent instantanément pour laisser place au bouton "Connexion".

3. Preuve de réussite



Figure 21 : Retour à l'état initial après déconnexion. Les accès privilégiés sont révoqués et l'utilisateur est redirigé en toute sécurité.

4. Conclusion de la phase de tests

Ce test marque la fin de la validation du parcours utilisateur complet. De l'inscription à la déconnexion, en passant par l'achat et la consultation, toutes les fonctionnalités critiques de **Knowledge Learning** ont été testées avec succès, garantissant une plateforme stable et prête à l'emploi.

MANUEL D'INSTALLATION ET DÉPLOIEMENT

Afin de permettre au jury ou à un futur collaborateur de tester l'application **Knowledge Learning** en local, j'ai rédigé cette procédure d'installation standardisée.

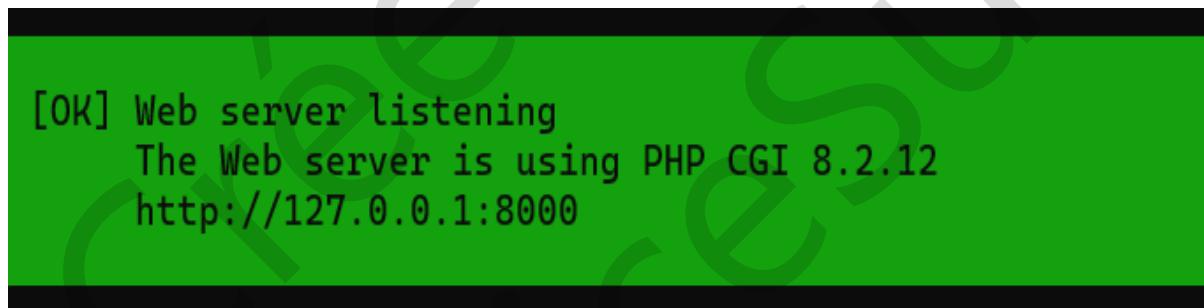
1. Prérequis techniques

Avant de commencer, les outils suivants doivent être installés sur la machine :

- **PHP** (Version 8.2 ou supérieure, comme illustré ci-dessous).
- **Serveur Web** (Apache ou le serveur interne PHP).
- **Base de données** (Configuration des accès dans le fichier .env).

2. Lancement du serveur

Une fois les dépendances installées et la base de données configurée, le projet est lancé via la console. Le message de succès confirme que l'application est en ligne et prête à être consultée sur le navigateur.



```
[OK] Web server listening
The Web server is using PHP CGI 8.2.12
http://127.0.0.1:8000
```

Figure 22 : Terminal confirmant le lancement du serveur de développement. L'application est désormais active à l'adresse locale <http://127.0.0.1:8000>.

3. Accès à l'interface

L'utilisateur peut maintenant ouvrir son navigateur et accéder aux différentes fonctionnalités :

- **Navigation publique** : Accueil et Catalogue.
- **Espace sécurisé** : Inscription et Connexion.
- **Administration** : Gestion des cours (réservé au profil administrateur).

MAINTENANCE ET ÉVOLUTIONS FUTURES

Un projet informatique est en constante évolution. Cette page présente les fonctionnalités déjà ébauchées et les pistes d'amélioration pour l'avenir de **Knowledge Learning**.

1. Évolutions fonctionnelles : La Certification

Une des évolutions majeures consiste à valoriser le parcours de l'apprenant. J'ai déjà mis en place un prototype de génération de certificat. Une fois qu'un utilisateur termine un cursus (comme le Piano), le système génère un document officiel attestant de sa réussite.



Figure 23 : Prototype du Certificat de Réussite généré par la plateforme après la complétion d'un module de formation.

2. Roadmap (Pistes d'amélioration)

Pour la suite du développement, plusieurs axes sont envisagés :

- **Système de Quiz** : Intégrer des évaluations interactives pour valider l'obtention du certificat ci-dessus.
- **Espace Communautaire** : Créer un forum sous chaque cours pour permettre aux élèves d'échanger avec le formateur.
- **Application Mobile** : Adapter le code via Capacitor pour proposer une version installable sur smartphone.

3. Maintenance et Sécurité

- **Veille technologique** : Suivre les mises à jour de PHP (version 8.2 actuelle) et des dépendances pour éviter les failles.
- **Sauvegardes** : Mise en place d'une routine de sauvegarde automatique de la base de données pour prévenir toute perte de données utilisateurs.

CONCLUSION ET REMERCIEMENTS

1. Conclusion du projet

La réalisation de la plateforme **Knowledge Learning** a été une expérience enrichissante, tant sur le plan technique qu'organisationnel. Ce projet m'a permis de maîtriser l'ensemble de la chaîne de développement d'une application web moderne :

- **Le Back-end** : Avec la mise en place d'une architecture solide en PHP et une gestion rigoureuse de la base de données.
- **Le Front-end** : En créant une interface utilisateur dynamique, fluide et totalement responsive.
- **La Sécurité** : En intégrant des protocoles d'authentification et de paiement (Stripe) répondant aux standards actuels.

Ce travail prouve qu'il est possible de créer une solution d'apprentissage en ligne complète, sécurisée et évolutive, capable de répondre aux besoins réels des formateurs et des élèves.

2. Bilan personnel

Au-delà des lignes de code, ce projet m'a appris à gérer les priorités et à résoudre des problématiques complexes (gestion des sessions, sécurité des paiements). Je ressors de cette expérience avec une meilleure compréhension des enjeux du métier de développeur et une grande motivation pour relever de nouveaux défis techniques.

3. Remerciements

Je tiens à remercier toutes les personnes qui ont contribué, de près ou de loin, à la réussite de ce projet :

- **Mes formateurs** : Pour leur accompagnement précieux, leurs conseils techniques et leur disponibilité tout au long de mon cursus.
- **Le jury** : Pour l'attention portée à la lecture de ce dossier et à la présentation de mon travail.
- **Mes proches** : Pour leur soutien constant durant ces mois de développement.

Julien Geoffroy