KNOWLEDGE

PLATFORME E-LEARNING

KNOWLEDGE LEARNING

SOMMAIRE

1. Introduction

Dans le cadre de ce projet, j'aurais l'opportunité de développer et de renforcer mes compétences dans plusieurs domaines du développement d'applications web. Il me permettra d'approfondir mes connaissances théoriques, mais également de les appliquer à des situations pratique et concrètes. Les compétences suivantes seront maitrisées :

- Ecrire un algorithme
- Développer dans un langage objet
- Développer la partie dynamique de l'application avec des composants serveurs, dans un style défensif, et éventuellement en asynchrone
- Appeler dans le composant serveur des services web
- Gérer la sécurité de l'application (authentification, permissions…) dans la partie serveur
- Réaliser un jeu de tests de l'application web
- Effectuer un achat test sur la boutique en ligne
- Publier l'application web sur un serveur
- Utiliser un outil de gestion de versions de code source

2. Présentation

Ce projet a pour but de donner la possibilité aux utilisateurs d'acheter un cursus de cours ou des cours individuel en ligne. Pour cela, nous devrons donner la possibilité de visualiser les cursus et le détail des cours individuel pour les utilisateurs non connecté.

Les utilisateurs pourront également créer leur compte et confirmer la création de ce dernier par l'intermédiaire d'un lien dans le mail.

Une fois connecté, les utilisateurs pourront acheter des cursus et des cours individuels directement depuis ma plateforme. Pour cela, nous avons choisi STRIPE comme moyen de paiement en ligne en raison de sa fiabilité, de sa sécurité et de sa facilité d'intégration avec Symfony. Grâce à STRIPE, nous pouvons offrir une expérience de paiement fluide et sécurisée, permettant ainsi aux utilisateurs de finaliser leurs transactions rapidement et en toute confiance.

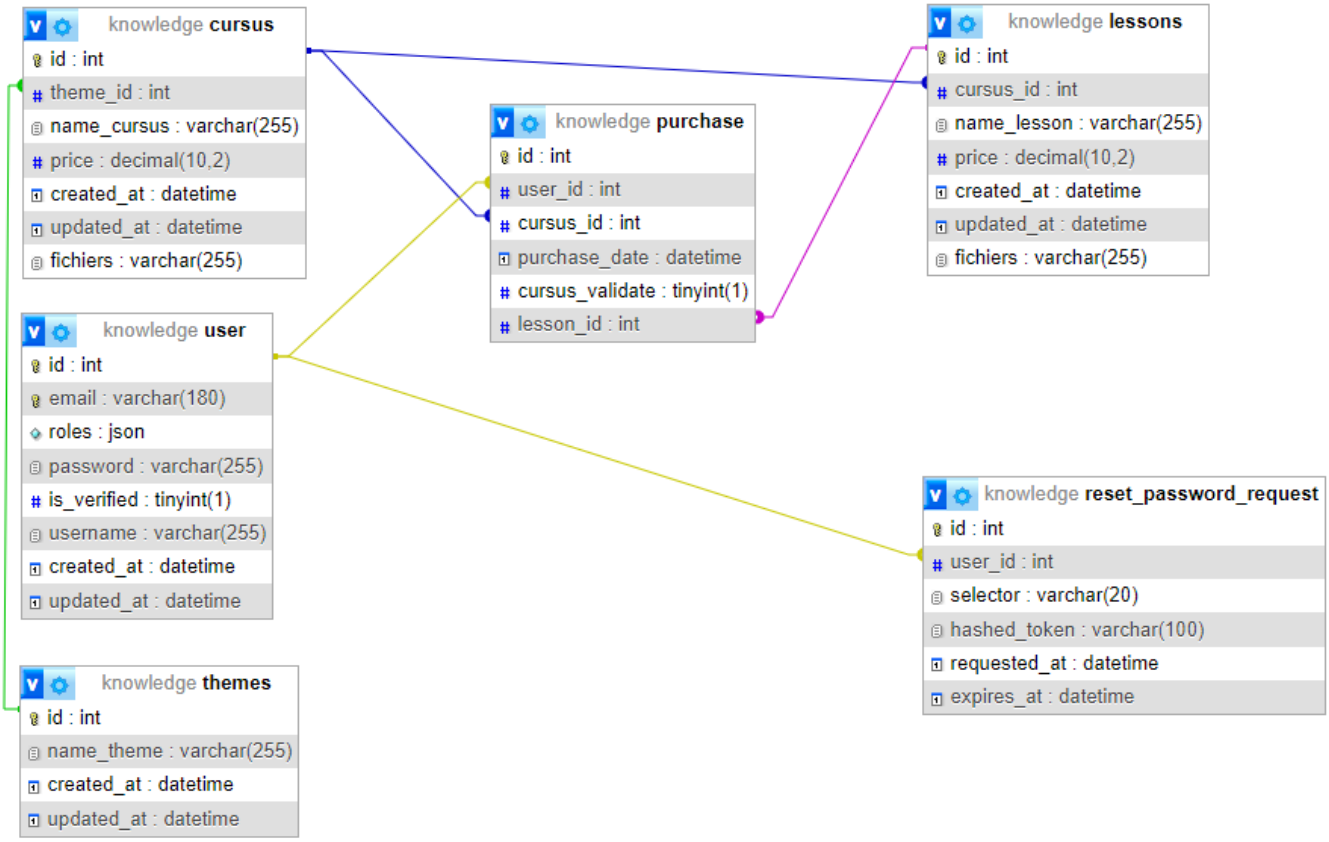
Une fois l'achat réalisé, l'utilisateur aura accès à l'intégralité du cursus et/ou du cours qu'il aura acheté.

Nous mettrons en place également un système d'obtention de certification. Le fonctionnement est le suivant, pour chaque utilisateur qui aura validé un cursus complet, une certification lui sera accordé et accessible sur sa fiche dès qu'il sera connecté.

Une plateforme d'administration a été développée avec EasyBundle pour la gestion des utilisateurs, les thèmes, les cursus ainsi que les cours. Les cursus et les cours seront intégré par notre plateforme sous forme de fichier au format TXT. Quant aux utilisateurs, ils pourront être également crée par cette plateforme.

3. Base de données

La base de données de notre application a été créé de la façon suivante.



3 tables (Themes, Cursus et Lessons) permettent de stocker les différents éléments des supports de cours. Elles sont liées entre elle par des clés étrangères.
La table Purchase est lié aux tables User, Cursus et Lessons, car elle permettra de stocker les achats réalisés par les utilisateurs et également le stockage de l'obtention des certifications

4. Moyen de paiement en ligne

La solution choisie pour le paiement en ligne a été STRIPE. Il met à disposition un mode développeur ce qui permet de réaliser des tests en situation réel. Le code mis en place est le suivant.

```php
#[Route('/checkout', name: 'app_checkout', methods: 'POST')]
public function checkout(SessionInterface $session): Response
{
    Stripe::setApiKey($this->getParameter('stripe_secret_key'));

    $items = $session->get('cart', []);
    $lineItems = [];

    foreach($items as $item){
        if($item['lesson']){
            $lineItems[] = [
                'price_data' => [
                    'currency' => 'eur',
                    'product_data' => [
                        'name' => $item['lesson']->getNameLesson(),
                    ],
                    'unit_amount' => $item['lesson']->getPrice() * 100,
                ],
                'quantity' => 1,
            ];
        }elseif($item['cursus']){
            $lineItems[] = [
                'price_data' => [
                    'currency' => 'eur',
                    'product_data' => [
                        'name' => $item['cursus']->getNameCursus(),
                    ],
                    'unit_amount' => $item['cursus']->getPrice() * 100,
                ],
                'quantity' => 1,
            ];
        }
    }

    $checkout_session = Session::create([
        'payment_method_types' => ['card'],
        'line_items' => $lineItems,
        'mode' => 'payment',
        'success_url' => $this->generateUrl('payment_success', [], UrlGeneratorInterface::ABSOLUTE_URL),
        'cancel_url' => $this->generateUrl('payment_cancel', [], UrlGeneratorInterface::ABSOLUTE_URL),
    ]);

    // Store the checkout session URL in the Symfony session
    $session->set('checkout_url', $checkout_session->url);
    // Redirection to intermediate route
    return $this->redirectToRoute('checkout_redirect');
}

#[Route('/checkout/redirect', name: 'checkout_redirect')]
public function checkoutRedirect(SessionInterface $session): Response
{
    // Retrieve the session URL
    $checkoutUrl = $session->get('checkout_url');

    // Return the template with the JavaScript redirection
    return $this->render('redirect.html.twig', [
        'checkout_url' => $checkoutUrl,
    ]);
}
```

4.1. Problème rencontré

Lors de la mise place de cette solution, nous avons rencontré une problématique avec l'url généré automatiquement par STRIPE. En effet, celle-ci doit nous rediriger sur leur plateforme afin de pouvoir renseigner les coordonnées bancaires. Dans cette url, le caractère (#) ou fragment tronque l'adresse se qui fait que cette page est introuvable. Nous verrons ensemble dans le prochain chapitre la solution apportée.

4.2. Solution apportée

Nous récupérons l'url avec le code suivant

```
         // Store the checkout session URL in the Symfony session
        $session->set('checkout_url', $checkout_session->url);
        // Redirection to intermediate route
        return $this->redirectToRoute('checkout_redirect');
```

que nous retournons dans la méthode, puis elle est rendue sur la template redirect.html.twig

```
#[Route('/checkout/redirect', name: 'checkout_redirect')]
    public function checkoutRedirect(SessionInterface $session): Response
    {
        // Retrieve the session URL
        $checkoutUrl = $session->get('checkout_url');

        // Return the template with the JavaScript redirection
        return $this->render('redirect.html.twig', [
            'checkout_url' => $checkoutUrl,
        ]);
    }
```

et enfin dans notre template, notre code javascript :

- va écouter l'évènement DOMContentLoaded.
- Récupérer l'url encodé pour le décodé
- et nous retournons sur le navigateur une url décodé.

```
    <p>Redirection vers Stripe, veuillez patienter...</p>
    <script type="text/javascript">
        document.addEventListener("DOMContentLoaded", function() {
            //Decode the URL before redirecting
            var encodedUrl = '{{ checkout_url }}';
            var decodedUrl = decodeURIComponent(encodedUrl);
            window.location.href = decodedUrl;
        });
    </script>
```

4.3. Simulation d'un paiement

Pour les tests, STRIPE met à disposition des cartes permettant la simulation de différents scénarios. Pour notre application nous utiliserons la carte de simulation suivante

N° de carte :
4242 4242 4242 4242

Date d'expiration :
Toute date supérieure à aujourd'hui

CVC :
Tout code à 3 chiffres

5. Lien utile

Le code de l'application est disponible sur github avec le lien suivant
https://github.com/tealc94/Knowledge

L'application est hébergée sur le serveur de alwaysdata, l'adresse du site est :
http://srv564566.hstgr.cloud/