Devoir Symfony

Documentation de l'application Stubborn E-commerce

Hugo Coelho

4 février 2025

Sommaire

1. Introduction

Stubborn E-commerce est une plateforme de vente en ligne développée avec **Symfony** et intégrant **Stripe** pour la gestion des paiements en ligne. Elle permet aux utilisateurs d'ajouter des produits à leur panier, de finaliser leur commande et d'effectuer des paiements sécurisés.

---

2. Technologies utilisées

- **Backend** : Symfony 6, PHP 8
- **Frontend** : Twig, Bootstrap
- **Base de données** : MySQL
- **Paiement** : Stripe API
- **Tests** : PHPUnit

---

3. Fonctionnalités

3.1 Gestion des utilisateurs

✅ Inscription et connexion via **Symfony Security**
✅ Gestion de session pour l'authentification

3.2 Gestion des produits

✅ Liste des produits affichée sur la page d'accueil et la boutique
✅ Filtrage des produits par **prix**

3.3 Gestion du panier

✅ Ajout de produits au panier
✅ Mise à jour des quantités
✅ Suppression des articles

3.4 Paiement avec Stripe

✅ Redirection vers Stripe pour finaliser le paiement
✅ Confirmation de paiement et affichage du **message de succès**
✅ Annulation de paiement avec **redirection vers la boutique**

---

4. Structure des fichiers

📂 **stubborn-ecommerce/**
│── 📂 **src/**
│ │── 📂 **Controller/** (Gestion des routes et de la logique métier)
│ │ ├── `ProductController.php` (Gestion des produits)
│ │ ├── `CartController.php` (Gestion du panier)
│ │ ├── `PaymentController.php` (Gestion des paiements avec Stripe)
│ │── 📂 **Service/**
│ │ ├── `CartService.php` (Gestion des actions liées au panier)
│ │── 📂 **Entity/** (Définition des entités de la base de données)
│
│── 📂 **templates/** (Fichiers Twig pour l'affichage des pages)
│ │── 📂 **cart/**
│ │ ├── `checkout.html.twig` (Page de finalisation de commande)
│ │── 📂 **product/**
│ │ ├── `list.html.twig` (Page affichant les produits)
│
│── 📂 **tests/** (Tests unitaires avec PHPUnit)
│ │── `PaymentControllerTest.php` (Tests du paiement)
│ │── `CartServiceTest.php` (Tests du panier)

---

5. Installation et Configuration

5.1 Prérequis

- PHP 8.x
- Composer
- Symfony CLI

- MySQL
- Un compte **Stripe**

5.2 Installation du projet

1️⃣ Cloner le projet

git clone https://github.com/ton-repo/stubborn-ecommerce.git

cd stubborn-ecommerce

2️⃣ Installer les dépendances

composer install

3️⃣ Configurer l'environnement

Créer un fichier .env.local et y ajouter les clés Stripe :
STRIPE_PUBLIC_KEY=pk_test_ta_cle_publique

STRIPE_SECRET_KEY=sk_test_ta_cle_secrete

4️⃣ Créer la base de données

php bin/console doctrine:database:create

php bin/console doctrine:migrations:migrate

5️⃣ Lancer le serveur Symfony

symfony server:start

Ouvrir http://127.0.0.1:8000 dans le navigateur.

6. Gestion des paiements avec Stripe

6.1 Configuration dans `PaymentController.php`

Le paiement est géré par l'API Stripe Checkout. Lorsqu'un utilisateur clique sur "Payer avec Stripe", voici ce qui se passe en détail :

1. **Création de la session de paiement Stripe**
    - **Le contrôleur PaymentController.php reçoit une requête pour initier un paiement.**
    - **Il récupère les articles présents dans le panier de l'utilisateur en session.**
    - **Un tableau `line_items` est généré pour inclure les produits avec leur nom, prix (en centimes) et quantité.**
2. **Envoi des données à Stripe**
    - **Une session de paiement est créée via `Stripe\Checkout\Session::create()`.**
    - **Cette session contient les informations nécessaires pour le paiement, comme :**
        - **Le mode de paiement (mode = 'payment')**
        - **Les méthodes acceptées ('payment_method_types' => ['card'])**
        - **Les détails de la commande (line_items)**
        - **Les URLs de redirection en cas de succès ou d'échec du paiement.**
3. **Redirection vers la page de paiement Stripe**
    - **Une réponse JSON contenant l'ID de la session Stripe est renvoyée au navigateur.**
    - **Côté frontend, Stripe est initialisé avec la clé publique, et l'utilisateur est redirigé vers la page de paiement Stripe via `redirectToCheckout()`.**
4. **Retour vers le site après paiement**
    - **Si le paiement réussit, l'utilisateur est redirigé vers une page de confirmation (`/payment-success`).**
    - **En cas d'annulation, il est renvoyé vers la page d'échec (`/payment-cancel`).**

Cette approche permet d'assurer la sécurité et la fiabilité du paiement, sans stocker d'informations sensibles sur le serveur.

7. Tests unitaires

7.1 Lancer les tests

php bin/phpunit

7.2 Exemple de test unitaire pour le paiement

```
public function testCreateCheckoutSession(): void

{

        $client = static::createClient();

        $client->request('POST', '/create-checkout-session', [], [], ['CONTENT_TYPE' =>
'application/json'], json_encode([

        ['name' => 'Produit Test', 'price' => 2999, 'quantity' => 1]

        ]));



        $this->assertResponseIsSuccessful();

        $this->assertJson($client->getResponse()->getContent());

}
```

8. Déploiement en production

1️⃣ Construire l'application

composer install --no-dev --optimize-autoloader

2️⃣ Configurer Stripe avec les vraies clés de production

STRIPE_PUBLIC_KEY=pk_live_ta_cle_publique

STRIPE_SECRET_KEY=sk_live_ta_cle_secrete

3️⃣ Exécuter les migrations et vider le cache

php bin/console doctrine:migrations:migrate

php bin/console cache:clear

9. Conclusion

Stubborn E-commerce est une application Symfony complète intégrant un système de panier, un paiement en ligne sécurisé avec Stripe et des tests unitaires pour garantir la robustesse du projet.