Projet Stubborn

1. **Introduction**

Ce projet consiste à créer un site e-commerce pour la marque de sweat-shirts *Stubborn*. L'application repose sur le framework Symfony, avec une base de données MySQL. Cette documentation présente les étapes d'installation, la structure du projet, ainsi que les fonctionnalités développées pour répondre aux besoins de la boutique en ligne.

2. **Prérequis**

Avant de commencer, assurez-vous d'avoir les éléments suivants installés sur votre machine :

- PHP >= 8.0
- Composer
- MySQL
- Symfony CLI
- Un compte Stripe pour la gestion des paiements
- Un serveur local (WAMP, MAMP, ou LAMP)

3. **Installation et Configuration**

3.1. Clonage du projet

- git clone https://github.com/Dada-nol/Project-Stubborn.git
- cd Project-Stubborn

3.2. Installation des dépendances

- composer install

3.3. Configuration des variables d'environnement

Dupliquez le fichier .env pour créer .env.local, puis modifiez-le avec vos paramètres de base de données et Stripe :

- DATABASE_URL="mysql://root:@localhost:3306/Stubborn_project"
- STRIPE_SECRET_KEY="votre_clé_stripe_secrète"
- STRIPE_PUBLIC_KEY="votre_clé_stripe_publique"

3.4. Création et migration de la base de données

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

3.5. Démarrer le serveur Symfony

- ./start.cmd

3.6. Gestion des mails

Dans notre cas cette manipulation n'est pas nécessaire car cela se lance au démarage du server symfony.

Les mails étant stockés de manière asynchrone, pour activer les mails la commande suivante est nécessaire :

- symfony console mailer:consume async

4. **Structure du projet**

- **src/Entity** : Contient les entités principales telles que User, SweatShirts, Cart, CartItem, et TailleSweat, Stock.
- **src/Controller** : Contient les contrôleurs pour gérer les différentes routes (SweatShirtController, CartController, BackOfficeController, HomeController, LoginController, RegistrationController, PaymentController).

- **src/Repository** : Contient les classes de gestion des requêtes SQL pour chaque entité.
- **src/Security** : Contient la méthode de login et de vérification d'email.
- **src/Service** : Contient les services de cart, de stripe et la méthode pour l'upload des images.
- **templates/** : Dossiers contenant les vues Twig pour chaque page (accueil, sweatshirt, panier, back-office, login, register, reset password).
- **config/** : Contient les fichiers de configuration, comme les routes et les services.

5. **Authentification et Rôles**

Le projet utilise le composant Security de Symfony pour la gestion des utilisateurs. Deux rôles principaux existent :

- **ROLE_USER** : Accès aux pages utilisateur telles que la page d'accueil, la boutique, et le panier.
- **ROLE_ADMIN** : Accès aux pages du back-office permettant de gérer les produits.

Les pages d'inscription (/register) et de connexion (/login) sont disponibles.
L'inscription génère un e-mail de confirmation avant d'activer le compte de l'utilisateur.

6. **Gestion des Produits et Filtrage**

- **Page d'accueil** (/) : Affiche une sélection de produits mis en avant.
- **Page produits** (/product) : Permet à l'utilisateur de consulter tous les produits, avec un système de filtrage basé sur une fourchette de prix (10-29€, 30-35€, 35-50€).
- **Page produit spécifique** (/product/{id}) : Affiche les détails d'un sweat-shirt et propose l'ajout au panier, avec la possibilité de choisir une taille.

7. **Gestion du Panier**

- **Page panier** (/cart) : Permet à l'utilisateur de visualiser son panier, d'ajouter ou de supprimer des articles, et de calculer le prix total.
- **Valider le panier** : Une fois le panier validé, l'utilisateur peut procéder au paiement.

8. **Paiement avec Stripe**

Le paiement est géré via Stripe. Un service Stripe a été créé pour gérer les transactions en mode développement.

8.1. Création d'une session de paiement

Lorsqu'un utilisateur valide son panier, une session Stripe est générée pour gérer le paiement. Après un paiement réussi, le stock des produits est mis à jour.

8.2. Intégration de Stripe

Le bundle Stripe est intégré dans le projet pour faciliter la gestion des paiements.

- composer require stripe/stripe-php

Le service Stripe interagit avec l'API pour créer les sessions de paiement .

9. **Gestion des Stocks**

Après un achat réussi, le stock des produits est automatiquement mis à jour en fonction des quantités achetées par l'utilisateur.

10. **Page Back-Office**

La page admin (/admin) est réservée aux administrateurs. Elle permet de :

- Ajouter un nouveau produit.
- Modifier un produit existant.
- Supprimer un produit.

11. **Tests Unitaires**

Des tests unitaires ont été mis en place pour valider les fonctionnalités critiques :

- **Tests du panier** : Assurent que les articles sont bien ajoutés au panier.
- **Tests de paiement** : Vérifient que le paiement via Stripe fonctionne correctement.

Ces tests se lancent au moment où le serveur Symfony est lancé.

12. **Hébergement Local**

La commande de démarage du serveur Symfony lance le dit serveur, Mailer ainsi que les tests. Pour héberger et tester le projet localement :

- Démarrer le serveur Symfony : ./start.cmd
- Accéder au site via : http://localhost:8000
- Simuler un achat en utilisant le mode sandbox de Stripe pour tester les paiements sans argent réel.