Documentation du Projet Stubborn

1. Introduction

Ce projet est un site web réalisé avec Symfony (version 7.1) et une base de données MySQL. Il permet à l'entreprise de ventre sur internet ces produits, ce site dispose d'un système d'authentification pour effectuer des achats ainsi que d'une connexion spécifique pour un ou des administrateurs pour permettre l'ajout et la suppression de produits ainsi que la gestion des stocks.

2. Prérequis

Avant de démarrer, assurez-vous que les éléments suivants sont installés sur votre machine :

- PHP 8.2 ou plus récent
- Composer pour la gestion des dépendances PHP
- MySQL (ou MariaDB) pour la base de données
- Symfony CLI (facultatif mais recommandé pour la gestion du projet)

3. Installation

Étape 1 : Cloner le projet

git clone https://github.com/Tintin4522/stubborn

cd stubborn

Étape 2 : Installer les dépendances

composer install

Étape 3 : Configurer les variables d'environnement

J’ai laisser le fichier .env dans le rpository GitHub pour l’accès en base de donnée, assurez-vous d’utiliser la même configuration que moi pour la connection en local :

DATABASE_URL="mysql://root:@127.0.0.1:3306/Stubborn"

C’est-à-dire, pas de mot de passe pour l’utilisateur root et le port 3306 pour MySQL.
Si ce n’est pas le cas changer le port dans le fichier .env et ajoutez le mot de passe entre « : » et « @ » pour que la connexion fonctionne.

Étape 4 : Créer la base de données et exécuter les migrations

Créez la base de données avec :

php bin/console doctrine:database:create

Puis exécutez les migrations :

php bin/console doctrine:migrations:migrate

Étape 5 : Charger les données de départ

Dans le repository GitHub, j’y joint la sauvegarde de la base de donnée, avec les données (produits et utilisateurs) que j’ai créé.

Dans ce fichier, l’utilisateur administrateur est enregistré comme suis pour permettre la connection :

Nom d’utilisateur : admin

Mot de passe : mdpadmin

Pour l’utilisateur test :

Nom d’utilisateur : test

Mot de passe :mdptest

4. Structure du Projet

Le projet suit la structure Symfony standard, avec des dossiers spécifiques pour le code source, les templates, les migrations, et les assets.

- **src/** : Code source (contrôleurs, entités, services, formulaires).
- **templates/** : Vues (templates Twig).
- **migrations/** : Fichiers de migration pour la base de données.
- **public/** : Contient les fichiers publics (images, CSS, JS), ainsi que le point d'entrée index.php.

5. Dépendances et Packages

Principales dépendances Symfony

- **symfony/asset** : Gestion des assets (CSS, JS, etc.).
- **symfony/console** : Composant pour exécuter des commandes CLI.
- **symfony/doctrine-messenger** : Intégration de Doctrine avec le composant Messenger.
- **symfony/dotenv** : Chargement des variables d'environnement.

- **symfony/form** : Création et gestion des formulaires.
- **symfony/mailer** : Envoi d’e-mails.
- **symfony/monolog-bundle** : Journalisation avec Monolog.
- **symfony/security-bundle** : Sécurité et gestion des utilisateurs.
- **symfony/twig-bundle** : Intégration de Twig pour les templates.
- **symfony/ux-turbo** : UX Turbo pour des fonctionnalités réactives et temps réel.
- **symfony/validator** : Validation des données.
- **symfony/web-link** : Gestion des liens dans les headers HTTP.
- **symfony/yaml** : Manipulation des fichiers YAML.

Autres packages notables

- **doctrine/orm** : Utilisé pour la gestion des entités et la communication avec la base de données.
- **stripe/stripe-php** : Intégration de l'API Stripe pour les paiements.
- **phpdocumentor/reflection-docblock** et **phpstan/phpdoc-parser** : Utilisés pour la documentation et la gestion des annotations.
- **symfony/verify-email-bundle** : Pour la vérification des adresses e-mail.

Dépendances de développement

- **symfony/maker-bundle** : Génération de code pour une configuration plus rapide.
- **phpunit/phpunit** : Framework de tests.
- **symfony/phpunit-bridge** : Intégration de PHPUnit avec Symfony.
- **symfony/debug-bundle** et **symfony/web-profiler-bundle** : Outils de débogage et de profilage pour le développement.

6. Commandes Utiles

Démarrer le serveur de développement

Pour lancer le serveur Symfony en local :

symfony server :start

Autres commandes importantes

- **Générer une migration** :

  php bin/console doctrine:migrations:diff

- **Exécuter les migrations** :

  php bin/console doctrine:migrations:migrate

- **Vider le cache** :

  php bin/console cache:clear

7. Sécurité

Le projet utilise le composant **Security** pour la gestion de l'authentification et de la protection des routes. Les règles de sécurité se trouvent dans le fichier config/packages/security.yaml. Les utilisateurs et rôles sont gérés de manière à protéger l'accès aux ressources sensibles.

8. API Stripe

Ce projet inclut l'API Stripe pour le traitement des paiements.

Comme indiqué par la documentation de l'API Stripe, j'ai intégré les clé suivantes pour effectuer des paiement fictif sur mon site et vérifié que ce service fonctionnait.

STRIPE_SECRET_KEY=sk_test_VePHdqKTYQjKNInc7u56JBrQ

STRIPE_PUBLIC_KEY=pk_test_oKhSR5nslBRnBZpjO6KuzZeX

Pour effectuer un paiement fictif sur le site en mode développement, et permettre de valider un faux paiement voici les informations supplémentaire que vous devrez rentrer sur le site pour valider un paiement :

- numéro de carte: 4242 4242 4242 4242
- date: ultérieur au jour de test
- CVC: 3 chiffres aléatoire
- code postal: celui qu'on veut