

Compte rendu de projet : Développement d'un site e-commerce avec Symfony

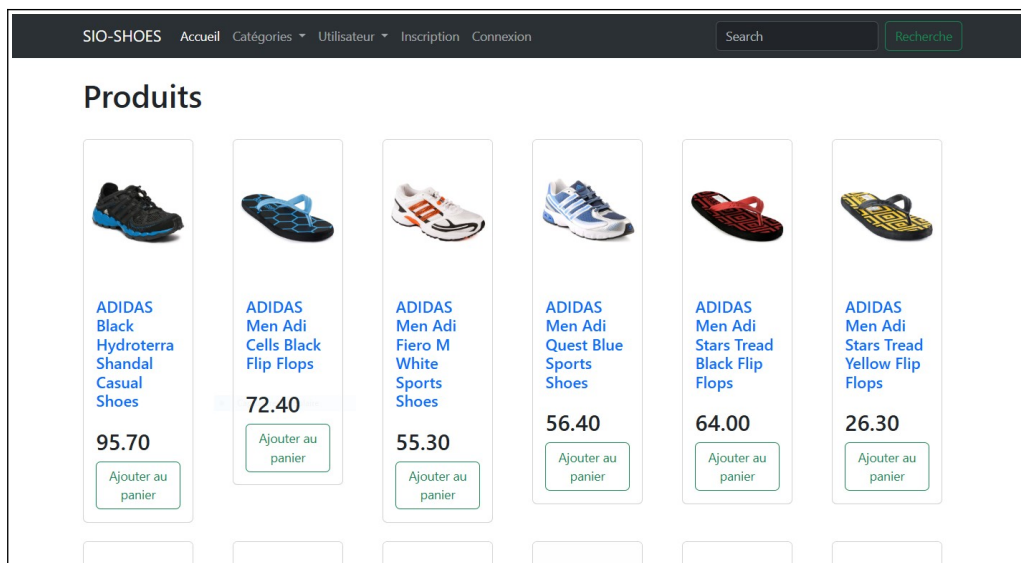
Table des matières

1. Présentation du projet.....	2
2. Fonctionnalités principales.....	2
2.1 La base de données.....	3
2.2 Les rôles.....	3
2.3 La création des entités et les migrations.....	3
3. Architecture.....	4
3.1 Gestion des routes.....	4
4. Développement.....	5
4.1 Contrôleur Produit et gestion des stocks.....	5
4.2 Formulaire Symfony.....	6
4.4 Gestion du panier et des commandes.....	6
4.5 Paiement stripe et envoi d'un email.....	7
4.6 Utilisation de bootstrap.....	8
4.7 La pagination.....	8
4.8 Le moteur de recherche.....	9
5. Outils utilisés.....	9
6. Compétences techniques mises en œuvre.....	10
7. Conclusion.....	10

1.Présentation du projet

Dans le cadre de ma formation en BTS SIO option SLAM, j'ai développé un site e-commerce (SIO-SHOES) de vente de chaussures à l'aide du framework Symfony. L'objectif était de concevoir un site web e-commerce permettant aux utilisateurs de naviguer parmi différents produits, de créer un compte, de gérer leur panier, de passer commande et d'effectuer un paiement en ligne.

Le site devait également offrir aux administrateurs la possibilité de gérer les produits, les commandes et les frais de livraison par ville, tout en maintenant une interface claire et sécurisée.



Page d'accueil de SIO-SHOES

2.Fonctionnalités principales



































































Le site affiche les produits classés par catégories et sous-catégories, avec un système de pagination pour faciliter la navigation. Les utilisateurs peuvent créer un compte, se connecter et accéder à leur panier pour ajouter, modifier ou supprimer des articles.

Les rôles définis (USER,EDITOR, ADMIN) permettent de restreindre ou d'autoriser certaines actions, comme la gestion des produits ou des utilisateurs.

Les commandes validées sont enregistrées en base de données et un e-mail de confirmation est envoyé automatiquement, avec un PDF généré pour le bon de commande.

Le paiement est sécurisé grâce à l'API Stripe, et le montant total prend en compte les frais de livraison calculés automatiquement selon la ville choisie par le client. L'interface administrateur permet de gérer tous les aspects du site : produits, commandes, utilisateurs avec leurs rôles et villes avec leurs frais de livraison.

2.1 La base de données

Table	Action	Lignes	Type	Interclassement	Taille	Perte
<input type="checkbox"/> add_product_history	★      	0	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> category	★      	2	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> city	★      	2	InnoDB	utf8mb4_unicode_ci	16,0 kio	-
<input type="checkbox"/> doctrine_migration_versions	★      	15	InnoDB	utf8_unicode_ci	16,0 kio	-
<input type="checkbox"/> messenger_messages	★      	0	InnoDB	utf8mb4_unicode_ci	64,0 kio	-
<input type="checkbox"/> order	★      	2	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> order_products	★      	2	InnoDB	utf8mb4_unicode_ci	48,0 kio	-
<input type="checkbox"/> product	★      	1 580	InnoDB	utf8mb4_unicode_ci	160,0 kio	-
<input type="checkbox"/> product_sub_category	★      	2 174	InnoDB	utf8mb4_unicode_ci	208,0 kio	-
<input type="checkbox"/> sub_category	★      	11	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> user	★      	2	InnoDB	utf8mb4_unicode_ci	32,0 kio	-

Base de données : my_shop_dev

2.2 Les rôles

Les utilisateurs authentifié sont Users, Admin et/ou EDITOR sur le site, ils peuvent utiliser différentes fonctionnalités.

Users : Ils peuvent naviguer sur le site , consulter les différentes catégories, ajouter des articles a leur panier et procéder aux paiements, et aussi s'inscrire et se déconnecter.

Admin : Ils ont tous les droits des utilisateurs et des éditeurs mais ils peuvent aussi gérer tout les utilisateurs de la base de données , les supprimer ou encore leur attribuer des rôles.

Editor : Ils ont tous les droits des utilisateurs mais en plus ils peuvent ajouter, supprimer, des produits, des catégories et des villes et ont accès a l'interfaces qui affiche les commandes de tous les utilisateurs.

2.3 La création des entités et les migrations

Symfony console make:entity Product | J'ai utilisée cette commande pour créer une entité dans symfony. Cette entité correspond a la table Product a laquelle ont peut attribuer des champs en l'occurrence id, name, price , image et stock.ce qui va permettre a symfony de générer automatiquement les méthodes get et set.

Symfony console make : migration | Cette commande génère un fichier de migration qui contient toutes les requêtes SQL pour créer ou modifier la table product, mais elle n'est pas encore envoyées dans la base, elle prepare juste le script.

Symfony console doctrine:migration:migrate | Cette commande applique les migrations à la base de données en exécutant le fichier généré par la commande précédente.

3. Architecture

Le projet utilise l'architecture MVC de Symfony pour organiser le code et séparer la logique métier, les données et la vue. Les entités Doctrine permettent de gérer les données et les relations avec la base MySQL, tandis que Twig et Bootstrap sont utilisés pour l'affichage des pages et le design responsive. Le composant Form de Symfony est utilisé pour la gestion des formulaires.

3.1 Gestion des routes

Le site utilise le système de route de Symfony pour associer chaque page à une URL. Les routes permettent de définir à quelles pages (URL) un utilisateur va pouvoir accéder. Certaines routes utilisent des paramètres dynamiques, comme l'identifiant d'un produit ou d'une commande, et l'accès à certaines pages est restreint selon les rôles utilisateurs.

Par exemple, l'URL `/editor/{id}` permet d'afficher le détail d'un produit spécifique, tandis que `/product/new` est réservée à la création de nouveaux produits par les administrateurs ou éditeurs.

```
#[Route('/editor/{id}', name: 'app_product_show', methods: ['GET'])]
public function show(Product $product): Response
{
    return $this->render('product/show.html.twig', [
        'product' => $product,
    ]);
}
```

```
#[Route('/editor/new', name: 'app_product_new', methods: ['GET', 'POST'])]
public function new(Request $request, EntityManagerInterface
$entityManager, SluggerInterface $slugger): Response
{
    $product = new Product();
    $form = $this->createForm(ProductType::class, $product);
    $form->handleRequest($request);
    ...
}
```

Code de la route : `app_product_new` et de `app_product_show`

4. Développement

4.1 Contrôleur Produit et gestion des stocks

Le ProductController gère toutes les opérations liées aux produits, y compris l'affichage de la liste, la création, la modification, la suppression, et la consultation détaillée d'un produit. Pour sécuriser l'accès aux fonctionnalités sensibles, j'ai mis en place une vérification des rôles : seuls les utilisateurs avec le rôle ADMIN ou EDITOR peuvent effectuer ces actions. La création et la modification des produits passent par des formulaires Symfony. Une fonctionnalité spécifique permet également d'ajouter du stock à un produit existant, tout en mettant à jour l'historique des ajouts.

Créer un nouveau produit

Nom du produit :

Description :

Prix :

Stock :

Sous-catégories :

Chaussures plates

Chaussures à talons

Chaussures décontractées

Chaussures de sport

Image :

Choisir un fichier

Aucun fichier choisi

Enregistrer

[Retour à la liste](#)

Page pour la création d'un produit

Produits						
Id	Name	Description	Prix	Stock	Actions	
1	Nike Men Air Zoom Century Shoes		62.50	30	Afficher	Ajout stock
2	Nike Men White Cricket Shoes		94.10	45	Afficher	Ajout stock
3	Reebok Men's Ventilator Ubiq Shoe		97.30	12	Afficher	Ajout stock
4	Reebok Men's Frisker LP Shoe		21.80	Stock épuisé	Afficher	Ajout stock

Fonctionnalités de la page produit

4.2 Formulaire Symphony

J'utilise les formulaires pour gérer l'inscription des utilisateurs, l'ajout de produits, la saisie des commandes et la gestion des catégories ou des villes. Chaque formulaire est lié à l'entité correspondante.

Par exemple, le formulaire CategoryType permet de créer ou modifier des catégories avec un champ pour le nom, lié directement à l'entité Category.

4.4 Gestion du panier et des commandes

Mon panier fonctionne de la manière suivante :

Chaque utilisateur authentifié peut ajouter des produits à son panier, peut modifier les quantités, peut supprimer des articles et peut finaliser sa commande.

Les données de l'utilisateur sont stockées dans une session symphony ce qui lui permet de conserver son panier même si il change de page et qu'il navigue sur le site jusqu'à sa déconnexion.

SIO-SHOES

Accueil

Catégories

Administrateur

Déconnexion

Panier

Search

Recherche

Votre panier

Nom du produit	Quantité	Prix	Total	Actions
Catwalk Women Black Ballerinas	2 <div>+</div> <div>-</div>	83.80 €	167.60 €	<div>Supprimer</div>

Total : 167.60€

Valider et Payer

Vider le panier

Panier avec ses différentes fonctionnalités

Les utilisateurs avec les rôles ADMIN et EDITOR ont accès à une page qui répertorie toutes les commandes qui ont été passés par tous les utilisateur avec les informations client et il peut valider si la livraison à été effectuée.


Commande n° 3 du 10/10/2025 11:32:30

Commande en attente de livraison

Informations client :

Id	Nom	Prénom	Adresse	Ville	Téléphone
3	kimpolo	divin	31 rue de cottenchy	80000 AMIENS	13134383

Produits commandés :

Image produit	Nom produit	Prix unitaire	Quantité	Prix total
	Carlton London Women Gold Wedge Sandal	60.00 €	2	120,00 €

Frais de port : 12.00 €

Total à payer : 132.00 €

Marquer comme livrée

Supprimer

Page d'administrateur pour la vue des commandes

4.5 Paiement stripe et envoi d'un email

Pour le paiement j'ai intégré l'Api Stripe qui permet de gérer les transactions de manière sécurisée sans stocker les données bancaires de l'utilisateur. Une fois que le paiement est validée, un email de confirmation est envoyée grâce à stmp4dev ce message contient un bon de commande au format PDF.

Confirmation de commande			
Bonjour divin kimpolo. Votre commande est confirmée.			
Commande n° 3 du 10/10/2025 11:32:30			
Produits commandés :			
Nom produit	Prix unitaire	Quantité	Prix Total
Carlton London Women Gold Wedge Sandal	60.00 €	2	120 €
Frais de port : 12.00 €			
<hr/>			
Total à payer : 132 €			

Bon de commande (format pdf) envoyé par mail

```
$payment = new StripePayment();  
$payment->startPayment($order);  
return $this->redirect($payment->getStripeRedirectUrl());
```

Code pour le paiement Stripe

On crée un objet qui va gérer le paiement avec stipe, puis on envoie les informations de la commande (contenu dans \$order) à Stripe et enfin on redirige l'utilisateur sur la page Stripe pour qu'il puisse payer en sécurité.

Pour utiliser l'api Stripe il faut créer un compte sur Stripe qui met à disposition un environnement de test pour les développeurs qui peuvent alors effectuer des paiements à l'aide d'une clé secrète et d'une de plusieurs codes de cartes bancaire qui permettent d'obtenir différents résultats.

Refus de paiement générique	4000 0000 0000 0002 	card_declined	generic_decline
Refus de paiement pour cause de fonds insuffisants	4000 0000 0000 9995 	card_declined	insufficient_funds
Refus de paiement pour cause de perte de carte	4000 0000 0000 9987 	card_declined	lost_card
Refus de paiement pour cause de vol de carte	4000 0000 0000 9979 	card_declined	stolen_card
Refus de paiement pour cause de carte expirée	4000 0000 0000 0069 	expired_card	s.o.
Refus de paiement pour cause de code CVC incorrect	4000 0000 0000 0127 	incorrect_cvc	s.o.

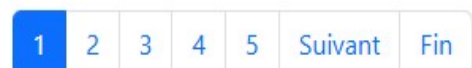
Les différentes cartes avec les résultats attendus

4.6 Utilisation de bootstrap

J'ai utilisé la bootstrap pour la partie front-end du site SIO- SHOES notamment pour la navbar, les boutons du site et le responsive. La bibliothèque bootstrap m'a permis d'optimiser mon efficacité.

4.7 La pagination

Pour faciliter la navigation dans les différentes pages, j'ai implémenté un système de pagination pour afficher les produits sur plusieurs pages.



Les différents produits sont aussi classés par ordres alphabétiques, et il y a un nombre limite de 12 produits par pages.

```
#[Route('/', name: 'app_home', methods:['GET'])]
public function index(ProductRepository $productRepository, CategoryRepository
$categoryRepository, Request $request, PaginatorInterface $paginator ): Response
{
    $data = $productRepository->findBy([], ['name'=>"ASC"]);
    $products =$paginator->paginate(
        $data,
        $request->query->getInt('page', 1),
        12
    );
}
```

Code de la route : app_home

4.8 Le moteur de recherche

J'ai mis en place dans ma barre de navigation la possibilité d'effectuer des recherches.

Par exemple : Si j'écris nike dans la barre de recherche tous les éléments qui comporte le mot :nike vont s'afficher à l'écran.

La route suivante va gérer le moteur de recherche quand l'utilisateur va taper un mot dans la barre de recherche var stocker le mot tapé par l'utilisateur puis elle va chercher tous les produits dont le nom correspond à celui que l'utilisateur a tapé.

les résultats sont ensuite paginés donc affichées sur plusieurs pages si nécessaire.

```
#[Route('/search/engine', name: 'app_search_engine')]
public function index(ProductRepository $productRepository, CategoryRepository
$categoryRepository, PaginatorInterface $paginator, Request $request): Response
{
    if ($request -> isMethod('GET')){
        $data = $request->query->all();

        $word = $data['word'];

        $data = $productRepository->searchEngine($word);

        $products = $paginator->paginate(
            $data,
            $request->query->getInt('page',1),
        );
    }
    return $this->render('search_engine/index.html.twig', [
        'products' => $products,
        'categories' => $categoryRepository ->findAll(),
    ]);
}
```

Code de la route : app_search_engine

5. Outils utilisés

Le projet a été réalisé avec Symfony pour le back-end, Twig pour les templates et Bootstrap pour le front-end responsive. La base de données MySQL a été gérée avec Doctrine ORM. Stripe a été utilisé pour le paiement en ligne, smtp4dev pour tester l'envoi d'e-mails, Dompdf pour générer automatiquement les PDF des commandes, et github pour stocker mes fichiers.

6. Compétences techniques mises en œuvre

Ce projet m'a permis de renforcer mes compétences en développement web full-stack avec Symfony. J'ai conçu des contrôleurs et routes pour gérer la logique métier, utilisé Doctrine pour les entités et relations, créé et validé des formulaires, sécurisé les actions selon les rôles utilisateurs, intégré Stripe pour le paiement, manipulé les sessions pour le panier, généré et envoyé des PDF, et créé un système de gestion des rôles par l'administrateur. J'ai également travaillé sur le front-end avec Twig et Bootstrap pour un rendu responsive et moderne, et j'ai stocker le code sur Git-hub.

7. Conclusion

Ce projet m'a permis de mettre en pratique mes compétences en développement web full-stack avec Symfony, en intégrant le front-end et le back-end de manière cohérente. J'ai conçu un site complet, sécurisé et fonctionnel, tout en approfondissant ma maîtrise du framework, de la gestion des routes, des formulaires, des entités et de l'intégration d'API externes. L'expérience acquise me donne une base solide pour le développement de projets web professionnels.