

**Symshop**

**La boutique e-commerce**

**Symshop**

**Directed by Nicolas**

**BACHELOR D&I**

**(Développement & Innovation)**

**ENSITECH – Groupe ENSUP 2023 – 2024**

**Diplôme :**

**Concepteur développeur d’applications**

**(CDA)**



Table des matières

[Remerciements 5](#_Toc167304910)

[Introduction 5](#_Toc167304911)

[Project Summary 6](#_Toc167304912)

[Résumé du projet 6](#_Toc167304913)

[Expression des besoins 7](#_Toc167304914)

[1. Cahier des charges 7](#_Toc167304915)

[2. Fonctionnalités par profils 8](#_Toc167304916)

[3. M V P (Minimum Viable Projet) 9](#_Toc167304917)

[Spécifications techniques, fonctionnelles & non-fonctionnelles 10](#_Toc167304918)

[Glossaire 11](#_Toc167304919)

[UX/UI Design 14](#_Toc167304920)

[1. Introduction 14](#_Toc167304921)

[2. Choix des couleurs 14](#_Toc167304922)

[3. Présentation du logo 14](#_Toc167304923)

[Wireframes 15](#_Toc167304924)

[Zonings 16](#_Toc167304925)

[Conception d’application 17](#_Toc167304926)

[Modèle de conception détaillé 17](#_Toc167304927)

[Cadre technique 18](#_Toc167304928)

[1. Spécification Fonctionnelle 18](#_Toc167304929)

[1. Présentation d’ensemble du projet 18](#_Toc167304930)

[2. Les contraintes et difficultés rencontrées 18](#_Toc167304931)

[3. La méthodologie choisie 19](#_Toc167304932)

[2. Spécification Technique 20](#_Toc167304933)

[La méthode de travail, outils et langages informatiques utilisés 20](#_Toc167304934)

[3. La conception et l’architecture de la base de données 22](#_Toc167304935)

[MCD : 22](#_Toc167304936)

[MLD 23](#_Toc167304937)

[MPD 24](#_Toc167304938)

[Capture d’écran de la base de données 28](#_Toc167304939)

[Modélisation UML 29](#_Toc167304940)

[1. Diagramme des cas d’utilisation 29](#_Toc167304941)

[2. Diagramme de classes 30](#_Toc167304942)

[3. Diagramme de séquence 31](#_Toc167304943)

[4. Diagrammes d’activités 33](#_Toc167304944)

[Développement 34](#_Toc167304945)

[1. Authentification 34](#_Toc167304946)

[2. Inscription 36](#_Toc167304947)

[3. Panier gestion 39](#_Toc167304948)

[4. Payement 44](#_Toc167304949)

[5. Réinitialisation du mot de passe en cas d’oubli 47](#_Toc167304950)

[6. Changer son mot de passe depuis ses informations : 50](#_Toc167304951)

[7. Barre de recherche 53](#_Toc167304952)

[8. Gestion admin 54](#_Toc167304953)

[Sécurité 56](#_Toc167304954)

[1. Sécurité RGPD 56](#_Toc167304955)

[Comment la mettre en place ? 56](#_Toc167304956)

[2. Attaque FORCE BRUTE 59](#_Toc167304957)

[La solution mise en place : 59](#_Toc167304958)

[3. Sécurité renforcée 61](#_Toc167304959)

[1. Sécuriser des formulaires 61](#_Toc167304960)

[2. Sécurité des données des utilisateurs 62](#_Toc167304961)

[3. Traitement des données & routes sensibles 64](#_Toc167304962)

[4. CORS (Cross-Origin Resource Sharing) 65](#_Toc167304963)

[La solution mise en place : 65](#_Toc167304964)

[Référencement 66](#_Toc167304965)

[Veille technologique 67](#_Toc167304966)

[Perspective 68](#_Toc167304967)

[Conclusion 68](#_Toc167304968)

# Remerciements

En cette étape cruciale de mon parcours, je tiens à exprimer ma profonde gratitude envers mon professeur principal, M. Mocknèche. Son dévouement et son soutien ont été essentiels pour mon développement académique et personnel.

Je remercie également mes autres professeurs, Mme. Ali, Mme. Tessier, MM. Mokhneche, Mme. Borredon et Mme. Louar, pour leur enseignement tout au long de mon parcours.

Un grand merci également à toute l'équipe pédagogique pour avoir créé un environnement propice à mon développement.

Merci à tous pour votre contribution à mon parcours scolaire.

Cordialement,

# Introduction

Bienvenue dans l'univers de la mode chez Symshop, votre destination en ligne pour trouver les dernières tendances, les pièces essentielles et les looks qui reflètent votre style unique. Chez Symshop, nous sommes passionnés par la mode et nous sommes déterminés à vous offrir une expérience de shopping en ligne exceptionnelle.

Explorez notre vaste sélection de vêtements soigneusement sélectionnés pour répondre à tous les styles, toutes les tailles et toutes les occasions. Que vous recherchiez des tenues décontractées pour le quotidien, des pièces élégantes pour une occasion spéciale ou des accessoires tendance pour compléter votre look, vous trouverez tout cela et bien plus encore chez nous.

Nous croyons en la puissance de la mode pour exprimer votre individualité et votre créativité. C'est pourquoi nous nous engageons à vous offrir des vêtements de haute qualité, des marques renommées et des designs uniques qui vous aideront à vous démarquer où que vous soyez.

Chez Symshop, nous croyons en la puissance de la technologie pour simplifier et améliorer nos vies. C'est pourquoi nous nous engageons à offrir une expérience de shopping en ligne fluide, sécurisée et transparente à chacun de nos clients.

Rejoignez-nous chez et laissez-nous vous aider à créer un look qui vous ressemble et qui vous inspire. Merci de nous avoir choisis pour vos besoins en matière de mode en ligne. Nous avons hâte de vous voir briller dans nos vêtements.

Symshop - Votre destination mode en ligne préférée.

# Project Summary

I undertook a project as part of my Application Developer Designer diploma program, focusing on "Development and Innovation." My task was to create a multi-layered application using a technical stack of my choice. My project, named Symshop, aimed to develop an online boutique where users can explore and purchase a variety of clothing items.

Working solo, I dedicated myself to the development of Symshop throughout the duration of my diploma program. Leveraging my skills and knowledge, I tackled each aspect of the project with diligence and determination. Effective task management and close monitoring of progress ensured I stayed on track despite the challenges encountered along the way.

Symshop provides users with a user-friendly interface to browse products and make purchases seamlessly. My goal was to implement features such as product categorization, search functionality, and secure payment processing to enhance the overall user experience.

Furthermore, my project involved creating a comprehensive requirements document outlining the expected deliverables and functionalities of Symshop. Through meticulous planning and execution, I aimed to ensure the successful realization of my vision for the online boutique.

Despite being a solo project, I collaborated effectively with myself and utilized my individual strengths to bring Symshop to fruition. I am proud of my accomplishment and the valuable skills I have gained throughout this endeavor.

# Résumé du projet

Symshop est une application web conçue pour répondre aux besoins d'un chef d’entreprise dans le domaine de la boutique de prêt-à-porter. L'objectif principal de Symshop est d'élargir la portée de cette entreprise en attirant davantage de clients grâce à une présence en ligne.

En utilisant le framework Symfony, reconnu pour sa polyvalence et sa robustesse, j’ai développé une plateforme conviviale permettant aux utilisateurs de parcourir différents catalogues d'articles de prêt-à-porter. L'application offre la possibilité aux clients de créer un compte, de se connecter et de gérer leurs informations personnelles.

Symshop permet également aux utilisateurs d'ajouter des produits à leur panier, de passer des commandes en toute sécurité et de bénéficier d'une expérience de shopping en ligne fluide et sécurisée. Pour faciliter la gestion du site, Symshop est doté d'un compte administrateur offrant des fonctionnalités avancées telles que la gestion des produits, des catégories et des commandes des clients.

L'application permet aux utilisateurs de filtrer leurs recherches en fonction de leurs préférences, facilitant ainsi la découverte de produits adaptés à leurs besoins. De plus, les utilisateurs peuvent gérer leur compte en toute simplicité, avec la possibilité de réinitialiser leur mot de passe en cas de besoin.

En résumé, Symshop offre une solution complète et personnalisée pour permettre au directeur de la boutique de prêt-à-porter d'étendre sa présence en ligne et d'attirer de nouveaux clients grâce à une expérience d'achat en ligne intuitive et sécurisée.

L'application comprend également une base de données pour stocker et mettre à jour les données des utilisateurs, qui est gérée à l'aide de PHPMyAdmin et de Wampserver. La persistance des données est assurée grâce à l'utilisation de l'ORM Doctrine.

# Expression des besoins

## 1. Cahier des charges

**1. Finalité du projet**

Faciliter l'expansion de la clientèle : Symshop vise à offrir une plateforme en ligne permettant au propriétaire de la boutique de prêt-à-porter d'atteindre de nouveaux clients potentiels qui recherchent des produits similaires en ligne. En fournissant une expérience de shopping en ligne fluide et intuitive, Symshop contribuera à élargir la portée de l'entreprise et à accroître sa visibilité.

**2. Objectifs**

* Développer une plateforme de commerce électronique intuitive et conviviale.
* Offrir une expérience d'achat en ligne sécurisée pour les utilisateurs.
* Permettre la visite en tant qu'invité pour les utilisateurs sans compte.
* Permettre aux utilisateurs inscrits de récupérer leur mot de passe et de modifier leurs données personnelles.
* Permettre la gestion efficace des produits, des catégories et des commandes par l'administrateur.
* Intégrer des fonctionnalités de filtrage pour faciliter la recherche de produits.
* Proposer un processus de paiement sécurisé pour les transactions.

## 2. Fonctionnalités par profils

**Fonctionnalités par Utilisateur :**

|  |  |
| --- | --- |
| En tant que … | Je peux pouvoir |
| Visiteur | Me rendre sur le site |
| Visiteur | Visiter les différents catalogues de produit dans différentes catégories |
| Visiteur | Créer un compte |
| Visiteur | Accéder au mentions légales |
| Visiteur | Utiliser les fonctionnalités de filtrage |
| Visiteur | Ajouter des produits au panier |
| Visiteur | Gérer son panier |

|  |  |
| --- | --- |
| En tant que … | Je peux pouvoir |
| Utilisateur inscrit | Me rendre sur le site |
| Utilisateur inscrit | Me connecter / me déconnecter |
| Utilisateur inscrit | Gérer mes informations personnelles |
| Utilisateur inscrit | Réinitialiser mon mot de passe |
| Utilisateur inscrit | Ajouter un produit au panier |
| Utilisateur inscrit | Parcourir les différents catalogues d’articles |
| Utilisateur inscrit | Gérer son panier |
| Utilisateur inscrit | Accéder au mentions légales |
| Utilisateur inscrit | Utiliser les fonctionnalités de filtrage |
| Utilisateur inscrit | Gérer le formulaire de livraison |
| Utilisateur inscrit | Effectuer les transactions pour régler |
| Utilisateur inscrit | Voir ses anciennes commandes |

|  |  |
| --- | --- |
| En tant que … | Je peux pouvoir |
| Administrateur | Modifier un produit |
| Administrateur | Supprimer un produit |
| Administrateur | Modifier une catégorie |
| Administrateur | Supprimer une catégorie |
| Administrateur | Créer un produit |
| Administrateur | Créer une catégorie |

## 3. M V P (Minimum Viable Projet)

MVP (Minimum Viable Projet) est une version du projet avec un ensemble minimum de fonctionnalités nécessaires pour valider les hypothèses clés et répondre aux besoins initiaux des utilisateurs. Pour Symshop, notre boutique en ligne de prêt-à-porter, le MVP se concentre sur les fonctionnalités essentielles nécessaires pour lancer la plateforme et recueillir les premiers retours des utilisateurs.

Voici notre MVP pour Symshop :

* Consultation des produits : Les utilisateurs peuvent parcourir les produits disponibles sur Symshop, consulter leurs descriptions, leurs prix et leurs images.
* Création de compte : Les utilisateurs ont la possibilité de créer un compte sur Symshop en fournissant des informations telles que leur nom, leur adresse e-mail et un mot de passe.
* Gestion du panier : Les utilisateurs peuvent ajouter des produits à leur panier, en modifier les quantités et procéder à l'achat.
* Processus de commande : Les utilisateurs sont guidés à travers un processus de commande fluide, où ils fournissent leurs informations de livraison et de paiement.

Notre MVP nous permettra de lancer rapidement Symshop sur le marché, en fournissant une base solide pour construire et itérer par la suite en fonction des retours des utilisateurs et des besoins du marché.

### Spécifications techniques, fonctionnelles & non-fonctionnelles

#### **Besoins fonctionnels**

Les besoins fonctionnels définissent les actions que le système doit effectuer pour être opérationnel. Pour Symshop, notre boutique en ligne de prêt-à-porter, les besoins fonctionnels incluent les éléments suivants :

Consultation des produits : Les utilisateurs doivent pouvoir parcourir les produits disponibles sur Symshop, consulter leurs descriptions, leurs prix et leurs images.

Création de compte : Les utilisateurs doivent avoir la possibilité de créer un compte sur Symshop en fournissant des informations telles que leur nom, leur adresse e-mail et un mot de passe.

Gestion du panier : Les utilisateurs doivent pouvoir ajouter des produits à leur panier, en modifier les quantités et procéder à l'achat.

Processus de commande : Les utilisateurs doivent être guidés à travers un processus de commande fluide, où ils fournissent leurs informations de livraison et de paiement.

Recherche de produits : Les utilisateurs doivent pouvoir rechercher des produits spécifiques en utilisant des filtres tels que la catégorie, la taille, la couleur, etc.

#### **Besoins non fonctionnels**

Les besoins non fonctionnels définissent les contraintes internes et externes du système. Pour Symshop, les besoins non fonctionnels comprennent :

Code clair et évolutif : Le code de Symshop doit être bien structuré et documenté pour permettre des évolutions et des améliorations futures.

Ergonomie et convivialité : L'interface utilisateur de Symshop doit être conviviale, intuitive et responsive, garantissant ainsi le confort de l'utilisateur lors de sa navigation sur la plateforme.

Sécurité des données : Symshop doit garantir la confidentialité des données des utilisateurs en mettant en place des mesures de sécurité robustes pour assurer l'intégrité et la cohérence des données à chaque mise à jour ou insertion.

# Glossaire

**Repository (Répertoire) :** Un repository est une classe chargée de la gestion de la persistance des entités dans une base de données. Il fournit des méthodes pour récupérer, créer, mettre à jour et supprimer des entités. Dans Symfony, les repositories sont généralement créés pour chaque entité et sont basés sur le patron de conception Repository. Ils sont responsables de l'interaction avec la couche d'accès aux données (ORM, base de données, etc.).

**Entity (Entité) :** Une entité représente une classe qui est mappée à une table dans une base de données relationnelle. Elle correspond généralement à un objet métier ou à une structure de données spécifique dans votre application. Les entités dans Symfony sont généralement des classes PHP qui sont annotées avec des métadonnées spécifiques pour définir leur mappage avec la base de données à l'aide d'un ORM (comme Doctrine).

**Form (Formulaire) :** Un formulaire est un composant qui permet de représenter et de gérer des données de manière interactive. Dans Symfony, il est possible de créer des formulaires en utilisant le composant Form. Les formulaires permettent de générer des champs de saisie, de valider les données soumises, de gérer les erreurs de validation, et de lier les données du formulaire à des entités ou à d'autres structures de données.

**Templates (Modèles) :** Les templates sont des fichiers de vue qui définissent la présentation des pages web dans votre application. Ils utilisent généralement un moteur de templates (comme Twig) pour générer le HTML dynamiquement en utilisant des variables, des boucles, des conditions, etc. Les templates dans Symfony sont utilisés pour afficher les données et les résultats du traitement côté serveur dans une structure HTML.

**Assets :** Les assets (ressources) font référence à tous les fichiers statiques utilisés dans votre application, tels que les fichiers CSS, les fichiers JavaScript, les images, les polices, etc. Dans Symfony, vous pouvez organiser et gérer vos assets à l'aide du composant Asset. Il fournit des fonctionnalités pour gérer le versioning, la compression et l'inclusion des assets dans vos templates et vues.

**Controller :** Les contrôleurs sont destinés à regrouper la logique de traitement des demandes associée au sein d'une seule classe.

**Migrations :** Une migration permet de créer et de mettre à jour un schéma de base de données.

Autrement dit vous pouvez créer des tables, des colonnes dans ces tables, en supprimer, créer des index... Tout ce qui concerne la maintenance de vos tables peut être prise en charge par cet outil.

**ORM :** Un ORM (Object-Relational Mapping) est un outil ou une technique de programmation qui facilite la conversion de données entre les systèmes de gestion de base de données relationnelles et les langages de programmation orientés objet. L'idée est de "mapper" les tables de la base de données à des classes spécifiques dans le code de l'application, permettant ainsi aux développeurs de manipuler les données de la base de données comme s'ils manipulaient des objets dans leur langage de programmation préféré.

**Type de requête HTTP :**

* **GET** : Elle est utilisée pour demander une ressource, des paramètres peuvent être passés dans l’URL.

* **POST** : Si vous souhaitez envoyer de grandes quantités de données, par exemple des images, ou des données confidentielles de formulaires au serveur, des paramètres également peuvent être passés dans l’URL.

**Request :** La classe Request représente une requête HTTP entrante et fournit de nombreuses méthodes utiles pour récupérer et manipuler les informations de la requête, telles que les paramètres, les en-têtes, les cookies, les fichiers téléchargés, etc

**WAMP** : WampServer est un outil d'administration de base de données possédant un éditeur SQL et un constructeur de requête.

Il a été développé et optimisé pour être utilisé avec le SGBD relationnel phpMyadmin disponible commercialement ou gratuitement. Quand WampServer est lancé, l’application ouvre automatiquement un gestionnaire de session pour se connecter en local ou sur un serveur distant.

**Composant :** Un composant est ainsi un élément isolé, autonome, qui possède des propriétés et des comportements et qu'on peut utiliser directement dans le HTML.

**Web Profiler :** Web Profiler est un outil de débogage puissant qui permet d'afficher les requêtes SQL générées par l'ORM de Symfony. Le Web Profiler est particulièrement utile pour optimiser les performances d’une application et résoudre les problèmes liés à la base de données. Il nous fournit une vision claire et détaillée des requêtes SQL envoyées à la base de données, ce qui nous permet d'analyser les requêtes, de repérer les goulets d'étranglement et d'optimiser en conséquence notre utilisation de la base de données.

**MVC :** MVC qui signifie Modèle-Vue-Contrôleur, est un modèle de conception d'architecture logicielle qui sépare une application en trois composants logiques principaux :

Le Modèle, la Vue et le Contrôleur. C'est une manière d'organiser le code de manière logique et efficace, et il est largement utilisé dans le développement web et de logiciels.

* Modèle : Le Modèle gère les données, la logique et les règles de l'application. Il peut s'agir d'un ou de plusieurs types de structures de données, avec la logique pour lire, écrire, et modifier ces données. En quelque sorte, c'est le cerveau de l'application.
* Vue : La Vue est la représentation de l'information, c'est-à-dire ce que l'utilisateur voit à l'écran. Elle utilise les données du Modèle pour se créer. En général, cela se présente sous forme de graphiques, de diagrammes, de tableaux ou tout simplement de texte.
* Contrôleur : Le Contrôleur agit comme une interface entre le Modèle et la Vue. Il traite les entrées de l'utilisateur et effectue les appels au Modèle pour l'exécution des actions correspondantes. Il prend ensuite les données du Modèle et les présente à la Vue. Le but de l'architecture MVC est de séparer la logique de l'application des interactions de l'utilisateur et de la présentation visuelle. Cette séparation permet une meilleure organisation du code, une plus grande flexibilité et une maintenance plus facile.

**SGBDR :** SGBDR signifie Système de Gestion de Base de Données Relationnelle. C'est un type de système de gestion de base de données qui stocke les données sous forme de tableaux relationnels ou de tables, ce qui rend les données plus faciles à comprendre et à utiliser pour les utilisateurs.

**Relations (entre les tables) :** Le modèle relationnel est basé sur le concept de relation qui est dérivé des principes de la théorie des ensembles en mathématiques. Les tables dans une base de données relationnelle sont liées par des clés communes, qui sont utilisées pour créer des liens entre les données.

**Clé primaire :** Une clé primaire est un champ ou une combinaison de champs dans une table de base de données qui est utilisée pour identifier de manière unique chaque enregistrement dans cette table. Les clés primaires sont une caractéristique fondamentale des systèmes de gestion de bases de données relationnelles (SGBDR), où elles sont utilisées non seulement pour l'identification unique, mais aussi pour créer des relations entre différentes tables.

**Clé étrangère :** Une clé étrangère est un champ ou un ensemble de champs dans une table qui est utilisé pour établir un lien vers la clé primaire d'une autre table. Les clés étrangères jouent un rôle essentiel dans les bases de données relationnelles en assurant des liens entre les tables et en maintenant l'intégrité des données.

# UX/UI Design

## 1. Introduction

L'interface utilisateur (UI) et l'expérience utilisateur (UX) de Symshop doivent répondre à des exigences graphiques spécifiques pour offrir une expérience utilisateur optimale :

* Le logo de Symshop doit être présent de manière claire et visible sur toutes les pages du site.
* Utilisation des couleurs et des graphiques conformément à la charte graphique de la marque.

## 2. Choix des couleurs

Pour Symshop, j’ai choisi une palette de couleurs qui reflète l'identité de la marque et qui favorise une expérience visuelle agréable pour les utilisateurs :

* Le bleu est une couleur souvent associée à la confiance et à la fiabilité, ce qui en fait un choix idéal pour une boutique en ligne. Il transmet un sentiment de sécurité et de professionnalisme aux utilisateurs.
* Le blanc est utilisé comme couleur secondaire pour créer un contraste optimal avec le bleu principal. Il contribue également à une apparence épurée et moderne tout en améliorant la lisibilité du contenu.

## 3. Présentation du logo

Logo du site

Pour Symshop, j’ai conçu un logo distinctif qui reflète l'essence de notre boutique en ligne. Le logo a été créé en mettant en valeur les initiales de la marque, ce qui le rend facilement reconnaissable et mémorable pour nos clients. Les couleurs choisies ont été sélectionnées pour leur harmonie avec la palette de couleurs de la marque, créant ainsi une cohérence visuelle à travers tout le site. Ce logo simple mais efficace renforce l'identité de la marque et laisse une impression durable auprès de nos clients.

### Wireframes

Les wireframes sont des outils essentiels dans le processus de conception d'une interface utilisateur. Ils servent à visualiser la structure et la disposition des éléments d'une manière simplifiée, permettant de planifier efficacement la navigation et les fonctionnalités d'une application ou d'un site web.

Dans le cadre de mon projet, j'ai utilisé des wireframes pour conceptualiser et structurer l'interface utilisateur. Les wireframes, sous forme de schémas simplifiés, ont servi à définir la disposition et l'organisation des éléments de l'interface sans s'attarder sur les détails visuels. Cette approche m'a permis de planifier efficacement la navigation et les fonctionnalités de mon application, en me concentrant sur l'essentiel de la conception dès le départ.

**Wireframes pages d’accueil :**

**Une image contenant Rectangle, diagramme, ligne, croquis

Description générée automatiquement**

### Zonings

Le zoning est une technique utilisée dans la conception d'interfaces utilisateur pour diviser visuellement une page ou une application en zones distinctes, chacune ayant un but spécifique. En utilisant des zonings, j'ai pu définir la structure de mon interface en identifiant les zones principales telles que les en-têtes, les menus de navigation, les zones de contenu et les pieds de page. Cette approche m'a permis de planifier efficacement la disposition des éléments et la navigation de mon application, en garantissant une expérience utilisateur cohérente et intuitive. Les zonings servent également de guide pour la disposition des informations et des fonctionnalités, facilitant ainsi la compréhension de l'architecture de l'interface et la communication des idées avec d'autres parties prenantes. En résumé, les zonings sont un outil précieux dans le processus de conception d'interfaces, permettant de définir la structure de manière claire et organisée pour une meilleure expérience utilisateur.

**Zoning page d’accueil :**

**Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquement**

# Conception d’application

## Modèle de conception détaillé

Pour Symshop, j’ai opté pour le modèle de conception MVC (Modèle-Vue-Contrôleur) avec le framework Symfony. Ce modèle est largement utilisé dans le développement web pour structurer efficacement le code et séparer les différentes préoccupations d'une application web.

Modèle (Model) : Le Modèle représente les données de l'application, la logique métier et les règles qui gèrent l'accès et les modifications de ces données. Dans le cas de Symshop, le modèle comprend toutes les entités Doctrine qui représentent les objets métier tels que les utilisateurs, les produits, les commandes, etc.

Vue (View) : La Vue est responsable de la présentation des données à l'utilisateur. Dans Symfony, les templates Twig sont utilisés pour générer le HTML à partir des données fournies par les contrôleurs. La Vue formate les données de manière à ce qu'elles soient présentées de manière claire et attrayante à l'utilisateur.

Contrôleur (Controller) : Le Contrôleur agit comme un intermédiaire entre le Modèle et la Vue. Il reçoit les requêtes de l'utilisateur, interagit avec le modèle pour traiter ces requêtes et prépare les données à afficher avant de les transmettre à la vue. Dans Symfony, les contrôleurs sont généralement représentés par des classes PHP contenant des méthodes pour gérer les différentes actions de l'application.

Pour donner une analogie, vous pouvez penser au modèle MVC comme à un restaurant. Le "Modèle" est la cuisine où les plats sont préparés (les données sont traitées), le "Contrôleur" est le serveur qui prend votre commande et apporte les plats à votre table (il traite les requêtes et transmet les données entre le modèle et la vue), et la "Vue" est la présentation du plat lui-même (l'affichage des données à l'utilisateur).

Outre le modèle MVC, il est crucial de comprendre les différentes architectures utilisées dans le développement logiciel :

* Architecture 1-tier : Toutes les couches de l'application sont regroupées sur la même machine, ce qui simplifie le déploiement mais peut rendre la maintenance plus complexe.
* Architecture 2-tier : L'application est divisée en deux niveaux distincts : le client et le serveur. Cette division offre une meilleure évolutivité et une maintenance simplifiée par rapport à une architecture 1-tier.
* Architecture 3-tier : L'application est divisée en trois niveaux distincts : la présentation, la logique métier et le stockage des données. Cette approche favorise une meilleure modularité, évolutivité et flexibilité dans le déploiement et la gestion de l'application.

En résumé, le modèle MVC de Symshop organise efficacement le code de l'application, séparant les préoccupations et facilitant ainsi la maintenance et l'évolution. Comprendre les différentes architectures, telles que le 1-tier, le 2-tier et le 3-tier, permet de choisir l'approche la plus appropriée en fonction des besoins spécifiques du projet en termes de performance, de maintenabilité et de scalabilité.

# Cadre technique

## Spécification Fonctionnelle

### 1. Présentation d’ensemble du projet

Introduction

Le but principal de Symshop est de créer une véritable communauté entre les amateurs et les professionnels de la mode, offrant un accès rapide et simple à une boutique en ligne sécurisée. L'authentification par login et mot de passe ainsi qu'un formulaire d'inscription seront mis en place pour garantir la sécurité et la facilité d'accès à l'application web.

Fonctionnalités principales :

Catalogue de produits : Les utilisateurs peuvent parcourir une large sélection d'articles de prêt-à-porter, avec des descriptions détaillées, des images et des prix.

Création de compte : Les utilisateurs peuvent s'inscrire sur Symshop en fournissant leurs informations personnelles, permettant ainsi une expérience personnalisée et sécurisée.

Gestion du panier : Les utilisateurs peuvent ajouter des articles à leur panier, en modifier les quantités et finaliser leurs achats de manière sécurisée.

Processus de commande : Symshop guide les utilisateurs à travers un processus de commande fluide, leur permettant de saisir leurs informations de livraison et de paiement de manière sécurisée.

Recherche de produits : Une fonctionnalité de recherche avancée permet aux utilisateurs de trouver rapidement les articles qu'ils recherchent en utilisant des filtres tels que la catégorie, la taille, la couleur, etc.

### 2. Les contraintes et difficultés rencontrées

Difficultés rencontrées :

J’ai rencontré des difficultés dans l'implémentation de certaines fonctionnalités, notamment la fonctionnalité du filtre qui reste encore à finir, en raison de contraintes de temps et de ressources limitées.

Contraintes :

Apprentissage du framework : La courbe d'apprentissage du framework Symfony a été une contrainte initiale, avec des concepts parfois complexes à appréhender et des erreurs fréquentes.

Documentation et bugs : La documentation parfois complexe et les bugs rencontrés ont ajouté des défis supplémentaires au développement du projet.

Définition des fonctionnalités essentielles : Il a été parfois difficile de déterminer les fonctionnalités essentielles de la boutique en ligne et de les traduire en exigences techniques claires, nécessitant des recherches complémentaires.

Malgré ces obstacles, j’ai persévéré en étudiant la documentation, en recherchant des solutions sur les forums et en adaptant mon approche pour surmonter les difficultés rencontrées.

### 3. La méthodologie choisie

Dans le cadre du développement de Symshop, j'ai choisi d'utiliser Trello pour la gestion des tâches et la planification du projet. Trello est un outil de gestion de projet en ligne basé sur le système kanban, offrant une organisation claire et flexible des tâches, idéale pour les méthodes de développement Agile.

En adoptant une approche Agile, j'ai pu suivre un processus itératif et incrémental pour le développement de Symshop. Cette méthodologie m'a permis de réagir rapidement aux changements et aux nouvelles informations découvertes tout au long du projet. Chaque tâche est représentée par une carte Trello, organisée dans différents tableaux correspondant aux différentes phases du cycle de développement telles que "À faire", "En cours" et "Terminé".

En tant que seul membre de l'équipe, j'ai été responsable de toutes les cartes et tâches sur Trello. Cela m'a permis de maintenir une clarté dans mes responsabilités et de gérer efficacement la répartition du travail. De plus, cette approche m'a permis de suivre l'avancement des tâches individuelles et du projet dans son ensemble.

En résumé, l'utilisation de Trello et de la méthodologie Agile m'a permis une gestion de projet efficace pour Symshop, en favorisant la communication, en augmentant la transparence et en me permettant de réagir rapidement aux changements tout au long du développement de la boutique en ligne.

## 2. Spécification Technique

### La méthode de travail, outils et langages informatiques utilisés

**Méthode de travail :**

Dans le cadre de mon projet e-commerce développé avec Symfony, j'ai suivi une méthodologie de travail rigoureuse basée sur le modèle MVC (Modèle-Vue-Contrôleur) pour garantir une structure claire et une gestion efficace des données. Voici comment j'ai intégré cette approche dans ma méthode de travail :

* Analyse des besoins : J'ai commencé par analyser les besoins de mon application e-commerce, en identifiant les fonctionnalités essentielles telles que la gestion des utilisateurs, des produits et des commandes.
* Conception du modèle de données : Sur la base de ces besoins, j'ai défini le modèle de données de mon application en créant des entités pour représenter les différentes composantes, telles que les utilisateurs, les produits et les commandes. J'ai utilisé Symfony et son ORM Doctrine pour configurer ces entités et définir les relations entre elles.
* Implémentation du contrôleur : Ensuite, j'ai mis en place la logique de mon application en développant les contrôleurs pour gérer les requêtes des utilisateurs.
* Création des vues : Enfin, j'ai développé les vues de mon application en utilisant le moteur de templating de Symfony pour afficher les données de manière conviviale pour l'utilisateur.

En suivant cette méthodologie de travail basée sur le modèle MVC et en utilisant les fonctionnalités de Symfony, j'ai pu développer mon projet e-commerce de manière efficace et structurée, en garantissant une gestion optimale des données et une expérience utilisateur agréable.

#### **Les outils et langages informatiques utilisés**

Pour la réalisation d'une application de ce type, j’ai commencé par mettre en place un serveur web WAMP (Windows, Apache, MySQL, PHP). Ce serveur me permet de développer et de tester mon application localement avant de la déployer sur un serveur en production.

En ce qui concerne l'IDE (Integrated Development Environment) ou l'éditeur de code, j’ai choisi d'utiliser Visual Studio Code. Cet éditeur est très apprécié car il offre une grande flexibilité et de nombreuses fonctionnalités. J’ai pu améliorer mon flux de travail en installant des extensions qui facilitent la rédaction du code et offrent des fonctionnalités supplémentaires. De plus, Visual Studio Code propose une intégration native de Git, ce qui m’a permis de gérer facilement mon versionnage de code directement depuis l'éditeur.

* **Système de gestion de base de données**

J’ai fait le choix pour la base de données d’utiliser PhpMyAdmin pour des contraintes de génération de table avec Symfony mais également par rapport à la performance de celui- ci par rapport à MariaDB.

* **Front-end**

Pour le front-end de templates, j’ai choisi Twig. Twig est un moteur de templates pour le langage PHP qui facilite la création de vues claires et structurées. Il m’a permis de séparer efficacement la logique de présentation du code PHP, ce qui a rendu mon code plus lisible et maintenable.

* **Backend**

J’ai commencé par utiliser Wampserver, un environnement de développement web pour Windows. Wampserver m’a permis d'installer et de gérer de manière centralisée notre serveur web Apache, la base de données MySQL et l'interpréteur PHP. Cet outil offre un environnement de développement complet qui permet de tester et de déployer facilement des applications web.

Pour la gestion de nos données, j’ai choisi MySQL comme système de gestion de base de données. MySQL est un choix populaire en raison de sa fiabilité, de sa capacité à gérer efficacement de grands volumes de données et de sa flexibilité pour s'adapter à une grande variété d'applications web.

Afin de faciliter l'administration de notre base de données MySQL, j’ai utilisé phpMyAdmin. Cet outil offre une interface utilisateur graphique qui permet de gérer facilement les bases de données, les tables, les colonnes, les relations, les index, les utilisateurs, les permissions, etc. Il offre également des fonctionnalités pour l'exécution des requêtes SQL, l'importation et l'exportation de données, et bien plus encore.

Pour manipuler la base de données, j’ai utilisé l'ORM Doctrine, qui est intégré avec Symfony. L'ORM (Object-Relational Mapping) est une technique de programmation qui permet de manipuler les données stockées dans la base de données comme si elles étaient des objets dans le code. Doctrine facilite l'interaction avec la base de données, permettant la création, la lecture, la mise à jour et la suppression de données de manière plus intuitive et orientée objet.

Il offre également des fonctionnalités supplémentaires, comme la gestion des relations entre les tables et l'abstraction de la base de données, ce qui rend le code plus portable et maintenable.

## La conception et l’architecture de la base de données

### MCD :

Un Modèle Conceptuel de Données (MCD) est une représentation visuelle des entités et des relations entre elles dans un système d'information. Il offre une vue abstraite et simplifiée de la structure des données, indépendamment de la technologie de stockage utilisée.

Le MCD est essentiel dans le processus de conception des bases de données car il permet de clarifier les besoins en données du système, d'identifier les entités principales ainsi que leurs attributs, et de définir les relations entre ces entités.

L'utilisation d'un MCD aide à communiquer efficacement avec les différentes parties prenantes du projet,en fournissant une représentation claire et partagée de la structure des données. Cela permet de s'assurer que toutes les parties comprennent les exigences et les spécifications du système de manière cohérente.

En résumé, l'utilisation d'un MCD facilite la conception, la communication et la mise en œuvre d'une base de données efficace et cohérente.

Une image contenant texte, diagramme, Plan, Dessin technique

Description générée automatiquement

### MLD

Le Modèle Logique de Données (MLD) est une représentation schématique des données d'une base de données, qui se concentre sur les entités, leurs attributs et leurs relations. Contrairement au Modèle Conceptuel de Données (MCD), qui capture les concepts et les relations entre eux sans se soucier des contraintes de mise en œuvre, le MLD spécifie la structure logique des données, y compris les types de données, les clés primaires, les clés étrangères et les relations entre les tables.

Le MLD prend en compte les exigences spécifiques du système et les traduit en un modèle de base de données fonctionnel. Il fournit une vue détaillée de la manière dont les données seront organisées et stockées dans la base de données réelle. En outre, le MLD inclut souvent des détails tels que les index, les contraintes d'intégrité référentielle et d'autres règles de validation des données.

Par rapport au MCD, qui reste à un niveau conceptuel plus élevé, le MLD offre une spécification plus détaillée et plus précise de la structure des données. Il constitue une étape intermédiaire essentielle dans le processus de conception de bases de données, permettant aux concepteurs de passer des concepts abstraits à une représentation concrète des données qui peut être mise en œuvre dans un système de gestion de base de données.

Une image contenant texte, diagramme, Plan, schématique

Description générée automatiquement

### MPD

Le Modèle Physique de Données (MPD) est une représentation concrète et détaillée de la structure de données d'une base de données, optimisée pour une implémentation dans un système de gestion de base de données spécifique. Contrairement au Modèle Logique de Données (MLD), qui se concentre sur la structure logique des données sans considérer les détails d'implémentation, le MPD spécifie les types de données, les index, les contraintes d'intégrité, les partitions et d'autres aspects techniques nécessaires à la mise en œuvre effective de la base de données.

Le MPD constitue la dernière étape de la conception de la base de données avant sa mise en œuvre. Il traduit les concepts abstraits du MCD et les structures logiques du MLD en un schéma concret qui peut être directement implémenté dans un système de gestion de base de données. Le MPD fournit les instructions précises nécessaires pour créer les tables, les colonnes et les contraintes dans la base de données physique, garantissant ainsi une cohérence entre le modèle de données et sa représentation réelle dans le système de base de données.

Une image contenant texte, diagramme, Plan, Police

Description générée automatiquement

**Voici l’architecture de notre base de données :**

Table **USER**

**id** INT PRIMARY KEY AUTO\_INCREMENT,

**email** VARCHAR(180) UNIQUE,

**roles** JSON,

**password** VARCHAR(60),

**full\_name** VARCHAR(60),

**reset\_password\_token** VARCHAR(100)

**consent\_given** tinyint(1)

Table **CATEGORY**

**id** INT PRIMARY KEY AUTO\_INCREMENT,

**name** VARCHAR(60),

**slug** VARCHAR(60)

Table **DOCTRINE\_MIGRATION\_VERSIONS**

**version** VARCHAR(191) PRIMARY KEY AUTO\_INCREMENT,

**executed\_at** DATETIME,

**execution\_time** INT

Table **PRODUCT**

**id** INT PRIMARY KEY AUTO\_INCREMENT,

**name** VARCHAR(60),

**price** INT,

**slug** VARCHAR(60),

**category\_id** INT FOREIGN KEY (category\_id) REFERENCES category(id),

**short\_description** LONGTEXT

**main\_picture** VARCHAR(255)

**brand\_id** INT FOREIGN KEY (brand\_id) REFERENCES brand(id),

Table **PURCHASE**

**id** INT PRIMARY KEY AUTO\_INCREMENT,

**user\_id** INT FOREIGN KEY (user\_id) REFERENCES user(id),

**full\_name** VARCHAR(60),

**address** VARCHAR(60),

**postal\_code** VARCHAR(20),

**city** VARCHAR(60),

**total** INT,

**status** VARVHAR(255),

**purchase\_at** DATETIME

Table **PURCHASE\_ITEM**

**id** INT PRIMARY KEY AUTO\_INCREMENT,

**product\_id** INT FOREIGN KEY (product\_id) REFERENCES product(id),

**purchase\_id** INT, FOREIGN KEY (purchase\_id) REFERENCES purchase(id),

**product\_name** VARCHAR(60),

**product\_price** VARCHAR(60),

**quantity** INT,

**total** INT

Table **COLOR**

**Id** INT PRIMARY KEYAUTO\_INCREMENT,

**name** Varchar(30)

Table **PRODUCT\_COLOR**

product\_id **INT,**

color\_id **INT,**

PRIMARY KEY (product\_id, color\_id),

FOREIGN KEY (product\_id) REFERENCES PRODUCT(id),

FOREIGN KEY (color\_id) REFERENCES COLOR(id)

Table **PRODUCT\_SIZE**

product\_id **INT,**

size\_id **INT,**

PRIMARY KEY (product\_id, size\_id),

FOREIGN KEY (product\_id) REFERENCES PRODUCT(id),

FOREIGN KEY (size\_id) REFERENCES SIZE(id)

Table **SIZE**

**Id** INT PRIMARY KEYAUTO\_INCREMENT,

**name** Varchar(30)

Table **SIZE\_SHOES**

**Id** INT PRIMARY KEYAUTO\_INCREMENT,

**name** Varchar(30)

Table **PRODUCT\_SIZE\_SHOES**

product\_id **INT**,

size\_shoes\_id **INT,**

PRIMARY KEY (product\_id, size\_shoes\_id),

FOREIGN KEY (product\_id) REFERENCES PRODUCT(id),

FOREIGN KEY (size\_shoes\_id) REFERENCES SIZE\_SHOES(id)

Table **BRAND**

**Id** INT PRIMARY KEYAUTO\_INCREMENT,

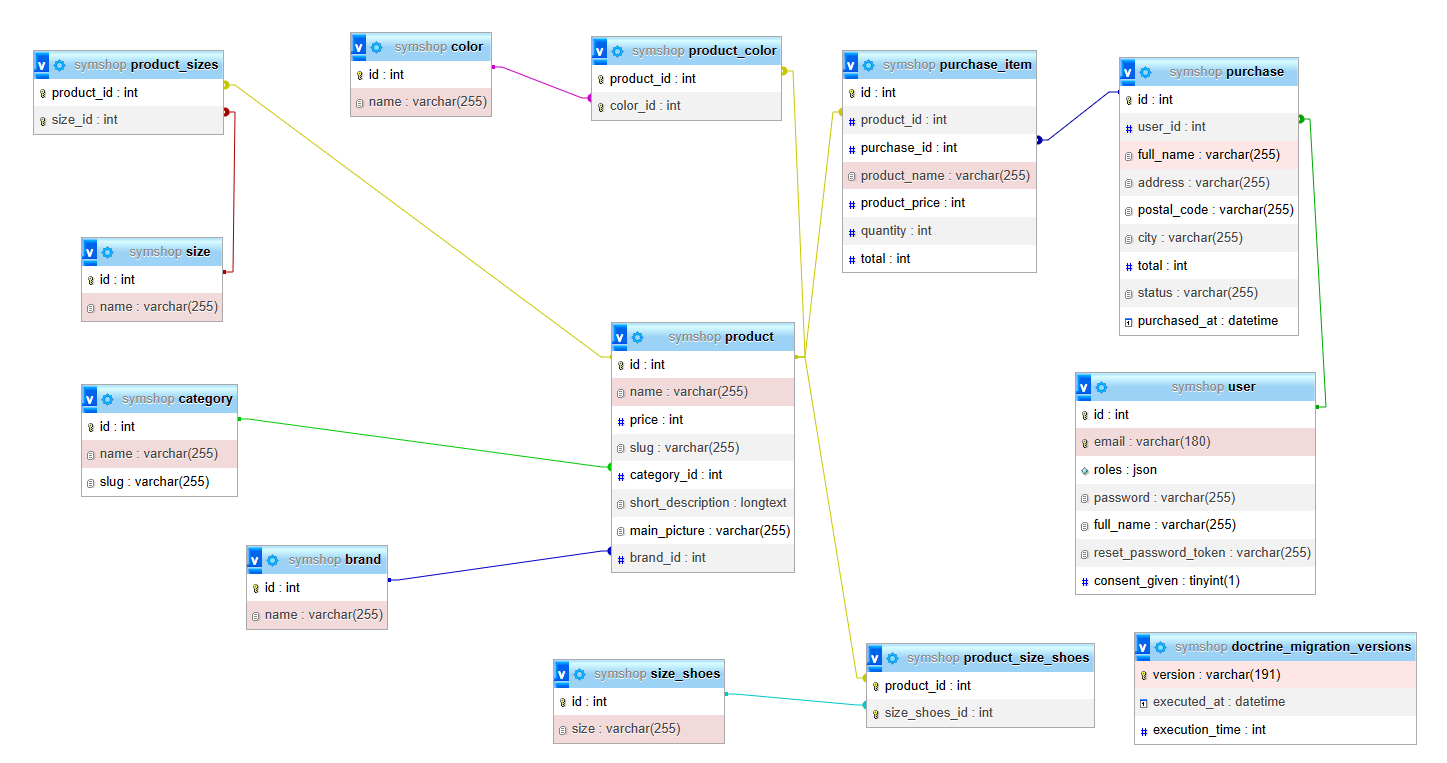
**name** Varchar(30)

Exemple de requêtes SQL (affichée grâce Web Profiler):

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

### Capture d’écran de la base de données



# Modélisation UML

## Diagramme des cas d’utilisation

Une image contenant diagramme, Plan, Dessin technique, schématique

Description générée automatiquement

Un diagramme de cas d'utilisation est une représentation graphique qui illustre les interactions entre les acteurs (utilisateurs ou systèmes externes) et un système donné. Il met en évidence les fonctionnalités ou actions offertes par le système du point de vue des utilisateurs.

Pour commencer, l’utilisateur peut parcourir le site et rechercher les produits dans différentes catégories. Il a à sa disposition une barre de recherche et un filtre pour faciliter ses recherches, le tout sans avoir besoin de se connecter. Il peut voir tous les produits, les ajouter au panier et gérer l'état de celui-ci. Cependant, pour valider son panier, il devra se connecter s’il possède déjà un compte ou s’inscrire s'il est nouveau sur le site.

Si l’utilisateur décide de s’inscrire, lors de l’affichage du formulaire, la base de données va vérifier si l’adresse e-mail n’est pas déjà enregistrée ; sinon, l’utilisateur est enregistré. Dans le cas où l’utilisateur se connecterait, il pourra remplir le formulaire de livraison et procéder au paiement. De plus, il aura la possibilité de gérer les informations de son compte ou encore de visualiser ses commandes effectuées.

L'administrateur dispose de fonctionnalités supplémentaires, notamment la gestion des produits et des catégories. Il peut ajouter de nouveaux articles, catégories, modifier des informations ou encore les supprimer.

## Diagramme de classes

Une image contenant texte, diagramme, Plan, Parallèle

Description générée automatiquement

Le diagramme de classe représente la structure statique de notre système, composé de plusieurs entités principales : Brand, Category, Color, User, Size, Purchase et SizeShoes.

La classe Brand représente les marques de produits. Chaque marque est caractérisée par un identifiant unique (id) qui agit comme clé primaire. De plus, elle possède un nom (name) et une relation avec la classe Product à travers une clé étrangère reliant les produits associés à cette marque.

La classe Category représente les catégories de produits. Chaque catégorie est identifiée par un identifiant unique (id) qui sert de clé primaire. Elle possède également un nom (name) et un slug (slug) utilisé pour les URL conviviales. Cette classe est liée à la classe Product par le biais d'une clé étrangère reliant les produits à leurs catégories respectives.

La classe Color représente les couleurs des produits. Chaque couleur est définie par un identifiant unique (id) qui agit comme clé primaire et un nom (name). Les produits peuvent avoir plusieurs couleurs, ce qui est représenté par une relation à travers une table intermédiaire. Cette table intermédiaire utilise des clés étrangères pour lier les produits aux couleurs correspondantes.

La classe User représente les utilisateurs du système. Chaque utilisateur possède un identifiant unique (id) qui sert de clé primaire, un nom d'utilisateur (username), une adresse e-mail (email) et un mot de passe (password). Les utilisateurs peuvent interagir avec d'autres parties du système, telles que les produits, les achats, etc., à travers des relations définies par des clés étrangères.

Les classes Size, Purchase et SizeShoes ont été ajoutées au diagramme pour enrichir la représentation du système. Size représente les tailles des produits, Purchase représente les achats effectués par les utilisateurs, et SizeShoes représente les tailles spécifiques aux chaussures. Chacune de ces entités possède ses propres attributs et relations avec les autres entités du système.

Ce diagramme de classe permet de visualiser la structure du système, y compris les relations entre ses différentes entités et l'utilisation des clés étrangères pour établir ces relations.

## Diagramme de séquence

Une image contenant texte, diagramme, Parallèle, capture d’écran

Description générée automatiquement

Un diagramme de séquence est une représentation graphique qui illustre la séquence chronologique des messages échangés entre les objets ou les acteurs dans un scénario donné. Il montre comment les objets interagissent les uns avec les autres au fil du temps, en mettant l'accent sur la séquence d'exécution des actions et des messages. Le diagramme de séquence permet de visualiser le déroulement temporel des interactions et de comprendre le flux de contrôle entre les différents éléments du système. Dans le cas où l’utilisateur souhaiterait s’inscrire, il devra commencer par accéder au site Symshop, devra ensuite cliquer sur le bouton pour s’inscrire pour que le site lui renvoie l’affichage du formulaire à remplir. L’utilisateur remplira les différents champs à compléter pour que le site envoie les informations complétées à la base de données. Cependant, il y a une alternative au moment de l’envoi des données et de leur traitement. Dans le cas où l’utilisateur serait déjà inscrit, la base de données va dire au site qu’un compte est déjà enregistré avec cette adresse mail, et donc le site renverra à l’utilisateur un message d’alerte lui signalant que cette dernière est déjà utilisée. Dans le cas où les champs seraient complétés sans problèmes d’email déjà utilisé, les informations sont enregistrées en base de données, le compte est créé et le site renvoie un message comme quoi l’utilisateur est bien enregistré.

Une image contenant texte, diagramme, Parallèle, reçu

Description générée automatiquement

Dans le cas où l’utilisateur serait déjà inscrit et souhaite se connecter, il devra commencer par accéder au site Symshop, devra ensuite cliquer sur le bouton pour se connecter pour que le site lui renvoie l’affichage du formulaire à remplir. L’utilisateur renseigne son nom mail et son mot de passe pour que le site envoie les informations complétées à la base de données et vérifie si l’utilisateur est bien enregistré. Cependant, durant l’envoi des informations, l’utilisateur peut avoir complété un mot de passe incorrect. Dans ce cas, l’utilisateur remplira le formulaire associé à la perte de son mot de passe envoyé par le site, le site enverra les nouvelles données à la base de données qui mettra à jour les informations de l’utilisateur. Enfin, l’utilisateur se connectera et pour de nouveau accéder au site.

## Diagrammes d’activités

Une image contenant diagramme, texte, Plan, Dessin technique

Description générée automatiquement

Un diagramme d'activité est une représentation graphique qui illustre le flux d'activités, d'actions et de décisions dans un processus ou un algorithme. Il met l'accent sur les étapes séquentielles et parallèles, ainsi que sur les conditions et les décisions qui guident le déroulement du processus. Le diagramme d'activité permet de visualiser le comportement dynamique d'un système, en montrant les actions qui se déroulent, les décisions prises et les transitions entre les différentes étapes. Il est largement utilisé pour modéliser des processus métier, des algorithmes ou des workflows.

# Développement

### Authentification

Le **LoginFormAuthenticator** est responsable de la gestion de l'authentification via le formulaire de connexion. Lorsqu'un utilisateur soumet ses informations de connexion, le **LoginFormAuthenticato**r vérifie la validité de ces informations en comparant le mot de passe fourni avec celui stocké en base de données. En cas de succès, l'utilisateur est redirigé vers la page d'accueil. En cas d'échec, des messages d'erreur appropriés sont renvoyés pour informer l'utilisateur des problèmes rencontrés. Le **SecurityController** gère les routes liées à l'authentification, telles que /login pour afficher le formulaire de connexion et **/logout** pour déconnecter l'utilisateur. Lorsque l'utilisateur accède à la page de connexion, le **SecurityController** affiche le formulaire de connexion à l'aide de **LoginType**. En cas d'échec de connexion, les messages d'erreur sont récupérés et affichés à l'utilisateur pour l'informer des raisons de l'échec de la connexion.

* Page connexion :

Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement

Formulaire de connexion :

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Template de connexion :

Une image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement

### Inscription

Le contrôleur **RegistrationController** gère le processus d'enregistrement des utilisateurs dans une application web. Lorsqu'un utilisateur accède à l'URL **"/register",** il est présenté avec un formulaire d'inscription où il peut saisir ses informations. Le contrôleur valide les données soumises par l'utilisateur, s'assurant qu'aucun robot n'est impliqué dans le processus grâce à une vérification de captcha. Une fois les données validées, le mot de passe de l'utilisateur est crypté avant d'être enregistré dans la base de données. Une redirection vers une page de succès est effectuée lorsque l'enregistrement est terminé avec succès.

* Page d’inscription :

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

Formulaire d’inscription :

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Template d’inscription :

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Extrait du contrôler d’inscription pour la validation de l’inscription :

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Requête grâce à Symfony Profiler :

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

### Panier gestion

Le contrôleur **CartController** orchestre les interactions entre les utilisateurs et leur panier d'achats au sein de l'application. Lorsque les utilisateurs ajoutent des articles à leur panier, le contrôleur valide leur sélection, assure l'intégrité des données, et fournit un retour visuel via des messages de succès. En affichant le contenu du panier, il simplifie l'expérience utilisateur en pré-remplissant les informations d'achat basées sur les commandes précédentes, et en proposant des options de suppression ou de modification de la quantité des articles sélectionnés. En assurant une interaction fluide et sécurisée, le **CartController** contribue à une expérience d'achat agréable et efficace pour les utilisateurs de l'application.

Page d’un produit ajouté au panier :

Une image contenant texte, capture d’écran, Site web, Page web

Description générée automatiquement

Méthode de l’ajout du produit au panier :

Une image contenant texte, capture d’écran

Description générée automatiquement

Page du panier :

Une image contenant texte, logiciel, Police, nombre

Description générée automatiquement

Extrait du template du panier :

Une image contenant texte, capture d’écran

Description générée automatiquement

* Fonctionnalité de décrémentation d’un produit dans le panier :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* Formulaire si l’utilisateur est connecté :

Page du formulaire :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Template du formulaire pour la livraison :

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

Extrait formulaire de livraison :

Une image contenant texte, capture d’écran

Description générée automatiquement

* Fonctionnalité pour le pré-remplissage du formulaire en fonction de la dernière commande :

Page formulaire :

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

Extrait de la fonction d’affichage du formulaire :

Une image contenant texte, capture d’écran

Description générée automatiquement

Exemple de requête grâce à Symfony Profiler :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

### Payement

Le processus de paiement dans mon application commence par l'affichage du formulaire de paiement **Stripe** grâce à **PurchasePaymentController**. Lorsque l'utilisateur soumet ce formulaire, le paiement est traité et validé par **Stripe**. En cas de succès, le contrôleur **PurchasePaymentSuccessController** confirme le paiement, met à jour le statut de la commande à "PAYÉE", vide le panier de l'utilisateur et déclenche un événement de succès de paiement. Enfin, l'utilisateur est redirigé vers une page de confirmation avec un message indiquant que le paiement a été effectué avec succès. Cette approche assure une expérience fluide et sécurisée pour les utilisateurs effectuant des achats sur ma plateforme.

Page de payement :

Une image contenant texte, Police, capture d’écran, Page web

Description générée automatiquement

Template de la page :

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Script java pour le paiement avec STRIPE :

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Extrait de la fonction du succès du paiement :

Une image contenant texte, capture d’écran

Description générée automatiquement

Page des commandes passées :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

### Réinitialisation du mot de passe en cas d’oubli

Le processus de réinitialisation du mot de passe de mon application est assez simple. Tout d'abord, lorsque les utilisateurs demandent une réinitialisation, ils saisissent leur adresse e-mail dans un formulaire dédié. Ce formulaire est géré par le fichier **ResetPasswordRequestType.php**. Une fois l'e-mail envoyé, un lien de réinitialisation est inclus dans l'e-mail, qui contient un jeton unique. Ce lien est généré grâce au fichier **ResetPasswordController.php**. Lorsque les utilisateurs cliquent sur ce lien, ils sont redirigés vers un autre formulaire où ils peuvent saisir leur nouveau mot de passe. Ce formulaire est géré par le fichier **ResetPasswordType.php**. Lorsque ce formulaire est soumis, le jeton est vérifié par le contrôleur, et si tout est bon, le mot de passe est mis à jour dans la base de données.

Page de demande :

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

Extrait du controller qui se charge de l’envoi du mail :

Une image contenant texte, capture d’écran

Description générée automatiquement

Si l’adresse électronique existe bien :

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

Template du mail :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Extrait du formulaire de réinitialisation :

Une image contenant texte, capture d’écran

Description générée automatiquement

Extrait de la fonction du changement de mot de passe :

Une image contenant texte, capture d’écran

Description générée automatiquement

Extrait template :

Une image contenant texte, capture d’écran

Description générée automatiquement

### Changer son mot de passe depuis ses informations :

Lorsque l'utilisateur accède à la page de changement de mot de passe, il est d'abord invité à remplir un formulaire de vérification. Ce formulaire, défini dans **VerificationFormType**, demande à l'utilisateur de saisir son mot de passe actuel ainsi que la réponse à une question secrète préalablement définie. Ces informations sont nécessaires pour s'assurer que seul l'utilisateur légitime peut modifier son mot de passe. Une fois le formulaire de vérification soumis, le contrôleur **UserController** vérifie la validité des informations fournies par l'utilisateur. Si le mot de passe actuel et la réponse à la question secrète sont corrects, l'utilisateur est autorisé à accéder au formulaire de changement de mot de passe.

Le formulaire de changement de mot de passe, défini dans **ChangePasswordFormType**, permet à l'utilisateur de saisir son nouveau mot de passe. Les règles de validation garantissent que le nouveau mot de passe respecte certaines conditions, telles que sa longueur et la présence de caractères spéciaux. Une fois que l'utilisateur soumet le formulaire de changement de mot de passe, le contrôleur **UserController** procède à l'encodage du nouveau mot de passe et le stocke dans la base de données associée à l'utilisateur. En cas de succès, l'utilisateur est redirigé vers sa page de profil avec un message de confirmation.

* Page de vérification :

Une image contenant texte, ligne, Police, capture d’écran

Description générée automatiquement

Template :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Extrait de la fonction de vérification :

Une image contenant texte, capture d’écran

Description générée automatiquement

Page de modification si les informations sont correctes :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Template :

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Extrait formulaire :

Une image contenant texte, capture d’écran

Description générée automatiquement

Fonction pour le changement de mot de passe

Une image contenant texte, capture d’écran

Description générée automatiquement

### Barre de recherche

La fonction de recherche permet aux utilisateurs de trouver des produits en fonction des termes saisis dans la barre de recherche. Lorsque l'utilisateur soumet une requête de recherche, le contrôleur **RechercheController** récupère les termes de recherche à partir de la requête HTTP GET. Ensuite, il utilise le **ProductRepository** pour rechercher les produits correspondants en fonction de ces termes. Les résultats de la recherche sont ensuite renvoyés à la vue, où ils sont affichés pour l'utilisateur.

Vue :

Une image contenant texte, capture d’écran, Police, logo

Description générée automatiquement

Fonction de recherche :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Twig de la recherche :



Extrait du product repository :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

### Gestion admin

L'administrateur utilise le fichier **AdminController.php** pour gérer les catégories et les produits via des méthodes définies pour afficher, éditer et supprimer ces éléments. Les fichiers **Twig**, comme **dashboard.html.twig**, **category/create.html.twig** et **product/edit.html.twig,** fournissent l'interface utilisateur pour interagir avec ces fonctionnalités. Les formulaires Symfony, définis dans **CategoryType.php** et **ProductType.php**, facilitent la saisie et la modification des données. Enfin, les fichiers d'entités Doctrine comme **Category.php** et **Product.php** représentent la structure des données et les relations entre les objets. Ces fichiers fonctionnent ensemble pour permettre à l'administrateur de gérer efficacement le contenu du site.

Vue Dashboard :

Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement

Extrait du Controller AdminController :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

# Sécurité

## Sécurité RGPD

**C’est quoi ? :**

La sécurité RGPD, ou Règlement Général sur la Protection des Données, est une règlementation importante qui vise à protéger les informations personnelles des personnes sur Internet. Elle s'applique à toutes les entreprises et organisations qui collectent, utilisent ou stockent des données personnelles de citoyens de l'Union européenne. Le RGPD garantit que les individus ont le contrôle sur leurs données personnelles et que celles-ci sont traitées de manière sûre et transparente.

La CNIL, ou Commission Nationale de l'Informatique et des Libertés, est l'autorité française chargée de veiller à ce que les entreprises respectent les règles de protection des données. Elles sont composées de trois aspects principaux :

1. **Conditions d'utilisation** : Les conditions d'utilisation définissent les règles d'utilisation du site et les obligations légales de l'entreprise. Elles précisent notamment les droits et responsabilités des utilisateurs lors de l'utilisation du site.
2. **Mentions Légales** : Les mentions légales fournissent des informations sur l'entreprise, telles que son nom, son adresse, ses coordonnées, etc. Elles sont nécessaires pour garantir la transparence et la conformité juridique de l'entreprise.
3. **Protection des Données** : La protection des données concerne la manière dont l'entreprise collecte, utilise et protège les données personnelles des utilisateurs.

**A quoi ça sert ?**

Elle sert à garantir que les informations personnelles des individus sont traitées de manière sûre, transparente et équitable. Cela signifie que les entreprises doivent obtenir le consentement des personnes avant de collecter leurs données, protéger ces données contre les piratages et les fuites, et donner aux individus le contrôle sur leurs propres informations.

Sa création :

L'histoire de la sécurité RGPD remonte à 2016, lorsque le Parlement européen a adopté ce règlement pour répondre aux préoccupations concernant la protection de la vie privée et la sécurité des données en ligne. Le RGPD est entré en vigueur le 25 mai 2018, et depuis lors, il a eu un impact significatif sur la manière dont les entreprises traitent les données personnelles des utilisateurs, en Europe et dans le monde entier.

### Comment la mettre en place ?

Mon site Symshop est un e-commerce il doit donc être équipé de cette sécurité pour être aux normes.

Dans mon projet Symfony, je crée donc une nouvelle page dans mes vues pour afficher ma politique de confidentialité. Cette page explique comment je collecte, utilise et protège les données personnelles des utilisateurs. J’ajoute en bas de cette page deux boutons qui permettent d’accepter ces conditions ou de redigérer l’utilisateur sur une explication détaillée de l’utilisation des cookies sur mon site pour que les utilisateurs donnent leur consentement explicite à la collecte et à l'utilisation de leurs données. Je m'assure ensuite que les données personnelles collectées sont stockées de manière sécurisée.

Page RGPD lors de l’arrivée sur le site :

Une image contenant texte, capture d’écran, Pantalons, habits

Description générée automatiquement

Extrait du script rgpd :

Une image contenant texte, capture d’écran

Description générée automatiquement

Modal rgpd :

Une image contenant texte, capture d’écran

Description générée automatiquement

Fonction qui gère le consentement d’un utilisateur :

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

## 2. Attaque FORCE BRUTE

La force brute est une technique utilisée par les pirates informatiques pour tenter de deviner un mot de passe ou une clé en essayant de nombreuses combinaisons différentes, généralement de manière automatisée. Cela peut être fait en utilisant des programmes informatiques qui génèrent des milliers, voire des millions, de combinaisons de mots de passe jusqu'à ce que la bonne soit trouvée. En bref, la force brute consiste à chercher à trouver un mot de passe en essayant toutes les possibilités.

Dans certains cas, les attaquants peuvent également créer une multitude de comptes ou d'identités fictives dans le système cible dans le but de surcharger la base de données et les ressources du serveur. Cette attaque, connue sous le nom d'attaque par force brute distribuée, vise à épuiser les ressources du système, ce qui peut entraîner des temps de réponse plus longs pour les utilisateurs légitimes ou même causer une panne complète du serveur si les capacités de traitement sont dépassées.

### La solution mise en place :

Pour éviter ces attaques dans mon projet, je mets donc en place un service CAPTCHA en installant gregwar/captcha via composer pour que lors d’une inscription l’utilisateur ait à confirmer les caractères qu’il voit sur une image générée automatiquement. J’ai aussi mis une case pour confirmer le fait qu’il ne soit pas un robot afin de confirmer l’envoi du formulaire. J’utilise donc le composant CheckboxType pour créer une case à cocher dans mon formulaire. J’ajoute une étiquette explicative pour demander à l'utilisateur de cocher la case pour confirmer qu'il n'est pas un robot. Et pour finir j’ai aussi rajouté une case invisible dans le formulaire appelé honeypot. Le honeypot est utilisé pour détecter et décourager les activités de spam ou les attaques automatisées sur un formulaire web. En ajoutant un champ invisible au formulaire, caché aux utilisateurs légitimes mais identifiable par les robots, on peut identifier et bloquer les soumissions suspectes. Ensuite je vérifie que la case est bien cochée pour autoriser l’envoi du formulaire. Si la case anti-robot est cochée et que le CAPTCHA est correct alors le formulaire peut être soumis si tous les autres champs du formulaire sont valides également, l’utilisateur peut procéder à l'inscription.

Extrait du formulaire d’inscription :

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Extrait du controller d’inscription :

Une image contenant texte, capture d’écran

Description générée automatiquement

## 3. Sécurité renforcée

### 1. Sécuriser des formulaires

Dans mon application, la sécurisation des formulaires est gérée automatiquement par Symfony. Lorsque je saisis des informations dans un formulaire, Symfony vérifie automatiquement la validité de ces données à l'aide de la méthode **handleRequest()** dans les contrôleurs dédiés, tels que **ResetPasswordController** et **RegistrationController** ect. Symfony assure également la protection contre les attaques CSRF en générant automatiquement des jetons CSRF pour chaque formulaire créé. Ces jetons sont inclus par défaut lors de la création des formulaires, ce qui empêche les attaques de type CSRF en vérifiant l'origine des requêtes POST.

Dans l'application Symfony que j’ai développée, la sécurisation des formulaires POST est prise en charge par l'utilisation de contraintes de validation, comme dans la classe **User**. Dans le fichier **User.php** qui correspond à l’entité user, nous définissons des contraintes de validation à l'aide d'annotations Symfony, telles que **@Assert\Regex** et **@Assert\Length**. Ces contraintes permettent de définir des règles de validation pour les champs tels que le mot de passe et le nom complet de l'utilisateur. Par exemple, pour le champ du mot de passe, nous utilisons une expression régulière dans la contrainte **@Assert\Regex** pour spécifier que le mot de passe doit contenir au moins 7 caractères, une majuscule, un chiffre et un caractère spécial. De même, pour le champ du nom complet, nous utilisons une autre expression régulière dans la contrainte **@Assert\Regex** pour restreindre les caractères autorisés dans le nom complet à des lettres, des espaces, des tirets, des apostrophes et des caractères spéciaux accentués. Ces contraintes garantissent que seules les données conformes aux critères spécifiés peuvent être soumises via les formulaires, renforçant ainsi la sécurité de notre application Symfony.

Extrait de l’entité user :

Une image contenant texte, capture d’écran

Description générée automatiquement

Exemple de fonction régulière dans resetPasswordType :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Cette expression régulière définit les critères suivants pour le mot de passe :

* Au moins une lettre majuscule (**(?=.\*[A-Z])**)
* Au moins une lettre minuscule (**(?=.\*[a-z])**)
* Au moins un chiffre (**(?=.\*\d)**)
* Au moins un caractère spécial parmi **@$!%\*?&** (**(?=.\*[@$!%\*?&])**)
* Longueur minimale de 7 caractères (**{7,}**)

Cela garantit que le mot de passe doit respecter ces critères pour être considéré comme valide.

### Sécurité des données des utilisateurs

Tout d'abord, pour gérer l'authentification des utilisateurs, j'ai utilisé le composant **LoginFormAuthenticator**. Cet authenticateur personnalisé utilise les fonctionnalités de sécurité intégrées de Symfony pour vérifier les informations d'identification des utilisateurs lorsqu'ils se connectent. J'ai utilisé l'interface **UserPasswordEncoderInterface** pour encoder et vérifier les mots de passe des utilisateurs, assurant ainsi la confidentialité des mots de passe stockés dans la base de données. De plus, j'ai vérifié que les informations de connexion fournies correspondent à celles enregistrées dans la base de données, garantissant ainsi que seuls les utilisateurs autorisés peuvent accéder à l'application.

En ce qui concerne l'inscription des utilisateurs, j'ai utilisé le formulaire de création de compte (**RegistrationFormType**). Ce formulaire utilise également les fonctionnalités intégrées de Symfony pour valider les données saisies par l'utilisateur, y compris la vérification de la force du mot de passe et la confirmation du mot de passe. De plus, j'ai ajouté des mesures de sécurité supplémentaires, telles que la protection anti-robot avec un champ "Je ne suis pas un robot" et l'ajout d'un champ de leurre (honeypot) pour détecter les soumissions de formulaire automatisées.

Il est important de noter que Symfony gère automatiquement de nombreuses fonctionnalités de sécurité, telles que la gestion des sessions utilisateur, la protection CSRF, la gestion des cookies sécurisés, etc. Par conséquent, l'utilisation de Middlewares supplémentaires pour ces tâches spécifiques n'est pas nécessaire dans mon application pour le moment, car Symfony prend déjà en charge ces aspects de sécurité de manière transparente.

Extrait **LoginFormAuthenticator.php** lors du décryptage du mot de passe :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Dans cette méthode**, $this->encoder->isPasswordValid($user, $credentials['password'])** est utilisé pour vérifier si le mot de passe fourni correspond au mot de passe haché stocké dans la base de données pour l'utilisateur donné. Si les mots de passe correspondent, l'authentification est réussie. Sinon, une exception d'authentification est levée. C'est là que se produit le déchiffrement et la comparaison des mots de passe.

Extrait de la fonction register dans le controller d’inscription lors de l’encryptage des données :

Une image contenant texte, capture d’écran

Description générée automatiquement

Dans cette méthode, le mot de passe fourni par l'utilisateur est récupéré à partir du formulaire et est crypté à l'aide de **UserPasswordEncoderInterface**. La méthode **encodePassword** prend en charge le cryptage du mot de passe en utilisant un algorithme sécurisé (généralement bcrypt) avant de le stocker dans la base de données. Cela garantit que les mots de passe des utilisateurs sont stockés de manière sécurisée et non en clair.

### Traitement des données & routes sensibles

Pour sécuriser les traitements de données envoyées par les utilisateurs, notamment celles provenant des formulaires HTML dans les requêtes POST, plusieurs mesures peuvent être prises.

Tout d'abord, l'utilisation d'URL encodées n'est pas nécessaire dans une application Symfony, car le framework gère automatiquement la gestion des données de formulaire. Symfony offre des mécanismes intégrés pour valider et traiter ces données de manière sécurisée. Par conséquent, l'utilisation d'URL encodées comme méthode de sécurisation n'est pas nécessaire dans ce contexte.

En ce qui concerne la sécurisation des routes sensibles, telles que l'authentification et la connexion des utilisateurs, Symfony propose des fonctionnalités intégrées pour gérer ces aspects de sécurité. Par exemple, l'authentification des utilisateurs est gérée par des composants spécifiques comme le composant **LoginFormAuthenticator**, qui assure la vérification des informations d'identification des utilisateurs lorsqu'ils se connectent. De plus, Symfony offre des fonctionnalités telles que la protection CSRF (Cross-Site Request Forgery) pour garantir la sécurité des formulaires et des requêtes.

Quant à l'utilisation de middlewares pour sécuriser les routes sensibles, Symfony gère déjà de nombreux aspects de sécurité de manière transparente, y compris l'authentification, la validation des données et la gestion des erreurs. Les middlewares sont souvent utilisés dans d'autres frameworks pour gérer ces aspects, mais dans Symfony, ces fonctionnalités sont prises en charge par les composants intégrés du framework. Par conséquent, l'utilisation de middlewares supplémentaires pour ces tâches spécifiques n'est pas nécessaire dans une application Symfony.

Dans le fichier de configuration de sécurité (security.yaml), Symfony permet de définir des règles d'accès spécifiques pour les différentes routes de l'application. Cela inclut la restriction d'accès à certaines routes sensibles, telles que celles liées à l'authentification des utilisateurs. En configurant ces règles, Symfony assure une gestion sécurisée des accès aux ressources protégées de l'application.

En résumé, dans une application Symfony, la sécurisation des données envoyées par les utilisateurs et des routes sensibles est largement gérée par le framework lui-même, grâce à ses fonctionnalités intégrées. Il n'est donc pas nécessaire d'utiliser des URL encodées ou des middlewares supplémentaires pour renforcer la sécurité de l'application pour le moment.

Extrait du fichier sécurity.yaml :

Une image contenant texte, Police, capture d’écran, ligne

Description générée automatiquement

Dans cet exemple, la règle spécifie que seuls les utilisateurs ayant le rôle **ROLE\_ADMIN** sont autorisés à accéder à cette route. Cela garantit que seuls les administrateurs de l'application peuvent accéder à cette zone restreinte, renforçant ainsi la sécurité de l'application en limitant l'accès aux fonctionnalités sensibles.

## CORS (Cross-Origin Resource Sharing)

**C’est quoi ? :**

CORS est un mécanisme de sécurité utilisé par les navigateurs web pour contrôler les demandes d'accès aux ressources (telles que les fichiers, les polices, les images, etc.) d'un domaine à un autre. Cela vise à empêcher les attaques de type Cross-Site Request Forgery (CSRF) en spécifiant quelles origines sont autorisées à accéder à quelles ressources. En d'autres termes, CORS permet aux serveurs web de définir qui peut accéder à leurs ressources et comment les requêtes doivent être gérées pour éviter les problèmes de sécurité liés à l'accès entre différents domaines.

### La solution mise en place :

J’ai installé le bundle nelmio/cors-bundle via Composer en exécutant la commande suivante dans mon terminal : composer require nelmio/cors-bundle

Une fois installé, j’ai configuré le bundle en ajoutant les paramètres CORS dans le fichier de configuration :

Une image contenant texte, capture d’écran, Police, Logiciel multimédia

Description générée automatiquement

# Référencement

**Référencement SEO**

**C’est quoi ? :**

C'est quoi le référencement SEO ? Le référencement SEO, ou Search Engine Optimization, consiste à optimiser un site web pour qu'il soit mieux classé dans les résultats de recherche des moteurs comme Google.

**Pourquoi l’utiliser ? :**

Le référencement SEO est primordial car il rend un site plus facile à trouver sur Internet. En l'optimisant pour les moteurs de recherche comme Google, cela augmentera sa visibilité et attirera plus de visiteurs intéressés par vos produits ou services. Cela permet d'augmenter l’audience et de générer plus de ventes ou de conversions. De plus, le référencement SEO offre des résultats durables et rentables à long terme, ce qui en fait un investissement précieux pour une entreprise en ligne. En résumé, le référencement SEO est essentiel pour rester compétitif sur le marché en ligne et assurer la croissance et le succès d’un site.

J'utilise donc dans mon cas le référencement SEO pour augmenter la visibilité de Symshop en ligne, attirer plus de clients potentiels et ainsi booster les ventes de ma boutique en ligne.

**Comment la mettre en place ? :**

Pour optimiser le référencement SEO de ma page d'accueil Symshop dans Symfony, je commence par choisir des mots-clés pertinents liés à mes produits et services. Ensuite je mets les balises meta.

Les balises meta sont des informations spéciales que l'on ajoute à une page web pour donner des détails supplémentaires aux moteurs de recherche. Les balises meta jouent un rôle crucial dans le référencement SEO en fournissant des informations aux moteurs de recherche sur le contenu de ma page. Par exemple, la balise meta description fournit un court résumé du contenu de la page, ce qui aide les utilisateurs à décider s'ils veulent la visiter ou non.

Pour définir une balise meta description dans mon template, j'utilise le code suivant :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

De cette manière, je peux personnaliser les balises meta de mes pages en fonction de leur contenu et des mots-clés pertinents pour le référencement.

# Veille technologique

# Perspective

La mise en production du projet pourra être envisagée après la réalisation d’un certain nombre d’améliorations à apporter sur quelques fonctionnalités, notamment sur la fonction de filtrage (filtre intelligent…). La sécurité mise en place a été validée par le commanditaire du projet, toutefois, il m’a semblé intéressant de lui proposer de la renforcer ultérieurement grâce à la sécurité JWT.

# Conclusion

Le projet de développement de Symshop a été une expérience enrichissante et formatrice, me permettant de mettre en pratique et d'approfondir mes compétences en développement web, tant sur le plan technique que sur celui de la gestion de projet. En développant cette application de e-commerce de vêtements avec Symfony, j'ai pu intégrer et maîtriser des technologies et des outils essentiels tels que Symfony, Doctrine ORM, MySQL, et Twig, tout en adoptant une approche structurée et méthodique.

Tout au long du projet, j'ai développé une architecture en couches qui a permis de séparer les responsabilités entre la présentation, la logique métier et l'accès aux données, facilitant ainsi la maintenance et l'évolution de l'application. La mise en œuvre de bonnes pratiques de sécurité a été un aspect crucial de ce projet, pour garantir la protection des données sensibles des utilisateurs et la sécurité des transactions.

Ce projet m'a également permis de développer mes compétences en travail autonome et en gestion du temps.

En conclusion, Symshop n'est pas seulement une application fonctionnelle de e-commerce, mais également une démonstration tangible de ma capacité à gérer et à livrer un projet complexe de manière professionnelle et efficace.