

# JOIE-EPICE

Application e-commerce de vente d'épice

*Projet réalisé dans le cadre de la présentation au*  
**Titre Professionnel Développeur Web et Web Mobile**

*présenté par*

**Kanthio DOUCOURE**

Centre Européen de Formation



# SOMMAIRE

INTRODUCTION .....	4
LISTE DES COMPÉTENCES COUVERTES PAR LE PROJET .....	5
I. Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité.....	5
A. <i>Maquetter une application</i> .....	5
B. <i>Réaliser une interface utilisateur web statique et adaptable</i> .....	5
C. <i>Développer une interface utilisateur web dynamique</i> .....	5
II. Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité.....	5
A. <i>Créer une base de données</i> .....	5
B. <i>Développer les composants d'accès aux données</i> .....	6
C. <i>Développer la partie back-end d'une application web ou web mobile</i> .....	6
RÉSUMÉ DU PROJET .....	7
CAHIER DES CHARGES .....	8
I. Besoins et objectifs de l'application .....	8
A. Besoins.....	8
B. Objectifs .....	8
II. Structure du site.....	8
III. Fonctionnalités.....	12
IV. Base de données .....	13
V. Évolutions potentielles .....	14
CHARTRE GRAPHIQUE ET LOGO.....	15
RÉALISATIONS .....	16
1. Fonctionnalité principale.....	16
2. Ajout au Panier .....	18
3. Affichage du Panier .....	19
4. Passer une Commande .....	20
5. Gestion des rôles .....	21
SPÉCIFICATIONS TECHNIQUES .....	22
A. <i>Architecture du projet</i> :.....	22
B. <i>Gestion des utilisateurs et de l'authentification</i> : .....	22
C. <i>Gestion du panier</i> :.....	22
D. <i>Sécurisation des formulaires et prévention des attaques</i> : .....	22
E. <i>Gestion des images et des fichiers</i> :.....	22

<i>F. Performance et optimisations :</i>	23
<i>G. Gestion des erreurs et des logs :</i>	23
<i>H. Tests et déploiement :</i>	23
<i>I. Interface utilisateur (UI) et expérience utilisateur (UX) :</i>	23
Routes front-end et back-end	23
A. Back-end	23
B. Front-end	25
Création de la base de données	26
A. MCD	26
B. MLD	26
VULNÉRABILITÉS DE SÉCURITÉ ET VEILLE	27
CONCLUSION	28
ANNEXE	29

# INTRODUCTION

Depuis mon enfance, l'informatique a toujours été une passion. Je me souviens particulièrement de l'émission "Les Enquêtes impossibles" que je regardais avec ma mère, où j'étais fascinée par les techniques utilisées pour résoudre des affaires criminelles grâce à la technologie. Cette curiosité m'a naturellement orientée vers un baccalauréat scientifique, spécialité mathématiques, physique et chimie, suivi d'une première année en ingénierie des systèmes informatiques.

Souhaitant me spécialiser davantage, j'ai suivi une formation en développement web et web mobile au Centre Européen de Formation, où j'ai acquis des compétences en programmation et en conception d'applications. Dans le cadre de cette formation, j'ai réalisé le projet "Joie-Épice", un site e-commerce dédié à la vente d'épices. Ce projet m'a permis de développer un site web avec Symfony, en prenant en charge l'interface graphique, la création du logo, le développement back-end, l'envoi d'e-mails via SendGrid et l'intégration du paiement via Stripe.

Après l'obtention de mon titre professionnel, je souhaite poursuivre mes études en intégrant une licence 3 en informatique, afin d'approfondir mes connaissances et de me préparer à une carrière dans le domaine.

Cette expérience m'a permis de renforcer mes compétences en développement web, tout en me préparant à collaborer efficacement au sein d'une équipe sur des projets concrets.

# LISTE DES COMPÉTENCES COUVERTES PAR LE PROJET

## I. Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

### *A. Maquetter une application*

Pour le projet "Joie Épice", bien que je n'aie pas réalisé de wireframes formels, j'avais une vision claire de l'interface souhaitée. J'ai conçu le logo à l'aide de Canva, en sélectionnant une image représentative et en y ajoutant le nom de ma marque. Ensuite, j'ai élaboré l'identité graphique en choisissant la police "Bebas Neue" et une palette de couleurs composée de :

- Orange (#f28705)
- Gris foncé (#2c2c2c)
- Bleu (code hexadécimal non spécifié)
- Blanc (#f2f2f2)
- Rouge (#f20530)

Ces choix ont permis de définir l'esthétique générale de l'application.

### *B. Réaliser une interface utilisateur web statique et adaptable*

L'application "Joie Épice" étant destinée à une clientèle variée, il était essentiel de concevoir une interface simple et intuitive, accessible à tous les utilisateurs, quel que soit leur niveau de compétence en informatique. De plus, l'application étant prévue pour une utilisation facile, elle devait être compatible avec les écrans mobiles.

### *C. Développer une interface utilisateur web dynamique*

Pour dynamiser l'interface, j'ai utilisé des technologies web modernes permettant d'ajouter des interactions et des animations, notamment pour le carrousel d'images et les transitions entre les sections. Ces éléments interactifs enrichissent l'expérience utilisateur en rendant la navigation plus engageante.

## II. Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

### *A. Créer une base de données*

J'ai structuré la base de données de l'application en définissant les entités suivantes :

- **User (Utilisateur)** : représente les clients de l'application.
- **Product (Produit)** : contient les informations sur les épices proposées.
- **Cart (Panier)** : gère les produits que l'utilisateur souhaite acheter.
- **Order (Commande)** : enregistre les transactions effectuées par les utilisateurs.
- **OrderItem (Élément de Commande)** : détaille les produits inclus dans chaque commande.

Ces entités sont interconnectées pour assurer une gestion cohérente des données.

### *B. Développer les composants d'accès aux données*

Pour interagir avec la base de données, j'ai utilisé l'EntityManager de Doctrine, ce qui facilite la gestion des entités et des requêtes. Cette approche assure une communication efficace entre l'application et la base de données.

### *C. Développer la partie back-end d'une application web ou web mobile*

Le back-end de l'application a été développé en utilisant le framework Symfony. J'ai mis en place des contrôleurs pour gérer les différentes fonctionnalités, telles que l'affichage des produits, la gestion du panier et le traitement des commandes. Des mesures de sécurité, comme l'authentification des utilisateurs et la validation des données, ont été intégrées pour protéger les informations sensibles.

Cette structure assure une séparation claire entre le front-end et le back-end, facilitant ainsi la maintenance et l'évolution de l'application.

# RÉSUMÉ DU PROJET

Joie-Épice est une plateforme en ligne spécialisée dans la vente d'épices de qualité, permettant à nos clients d'enrichir leurs préparations culinaires sans se déplacer en magasin. Nous proposons une sélection d'épices provenant du Niger, notamment le fakou, ainsi que des mélanges exclusifs alliant cannelle, vanille, poivres, curcuma et gingembre etc... Ces produits sont disponibles en petites boîtes et en grands sachets, répondant ainsi aux besoins variés de nos clients.

# CAHIER DES CHARGES

## I. Besoins et objectifs de l'application

### A. Besoins

Le projet "Joie Epice" vise à développer une plateforme e-commerce dédiée à la vente d'épices de qualité supérieure. Les besoins identifiés sont :

- 1. Interface utilisateur intuitive : Permettre aux clients de naviguer aisément, de découvrir les produits et de finaliser leurs achats sans difficulté.
- 2. Gestion efficace des produits : Offrir à l'administrateur la possibilité d'ajouter, modifier ou supprimer des produits, ainsi que de gérer les stocks.
- 3. Processus de commande simplifié : Faciliter l'ajout de produits au panier, la modification des quantités et la finalisation de la commande, incluant un système de paiement sécurisé.
- 4. Gestion des utilisateurs : Permettre aux clients de créer un compte, de se connecter, de réinitialiser leur mot de passe et de consulter l'historique de leurs commandes.
- 5. Sécurité des transactions : Assurer la protection des données personnelles et des informations de paiement des utilisateurs.

### B. Objectifs

Les objectifs du projet sont :

- 1. Développer une plateforme e-commerce fonctionnelle : Créer un site web opérationnel permettant la vente en ligne d'épices.
- 2. Assurer une expérience utilisateur optimale : Garantir une navigation fluide, une présentation attrayante des produits et un processus de commande sans friction.
- 3. Intégrer un système de paiement sécurisé : Mettre en place une solution de paiement fiable et conforme aux normes de sécurité en vigueur.
- 4. Fournir des outils de gestion pour l'administrateur : Offrir des fonctionnalités permettant la gestion des produits, des commandes et des utilisateurs.
- 5. Respecter les délais et le budget : Livrer le projet dans les délais impartis et en respectant le budget alloué.

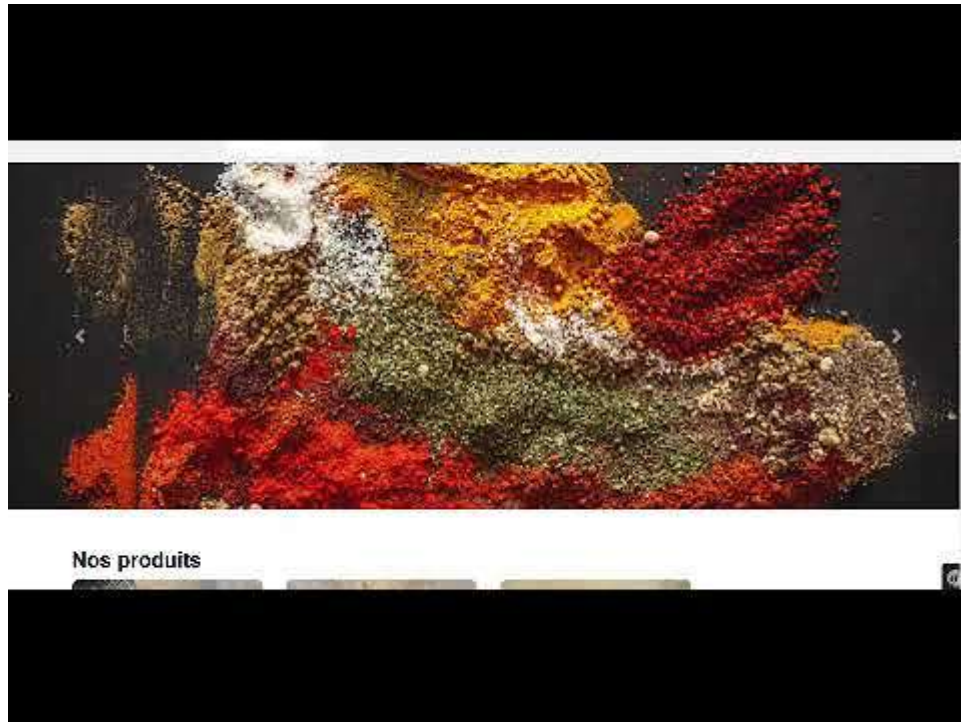
## II. Structure du site

- **Page d'accueil :**

- En-tête avec le logo et des icônes de navigation (panier, profil, historique des commandes).
- Carrousel d'images défilantes mettant en avant des produits phares.



- Présentation des produits sous forme de cartes Bootstrap avec nom, prix et bouton "Découvrir".



Voici le lien pour bien voir la page d'accueil :

<https://www.youtube.com/watch?v=ni6Ufe9s8lq>

#### Page produit :

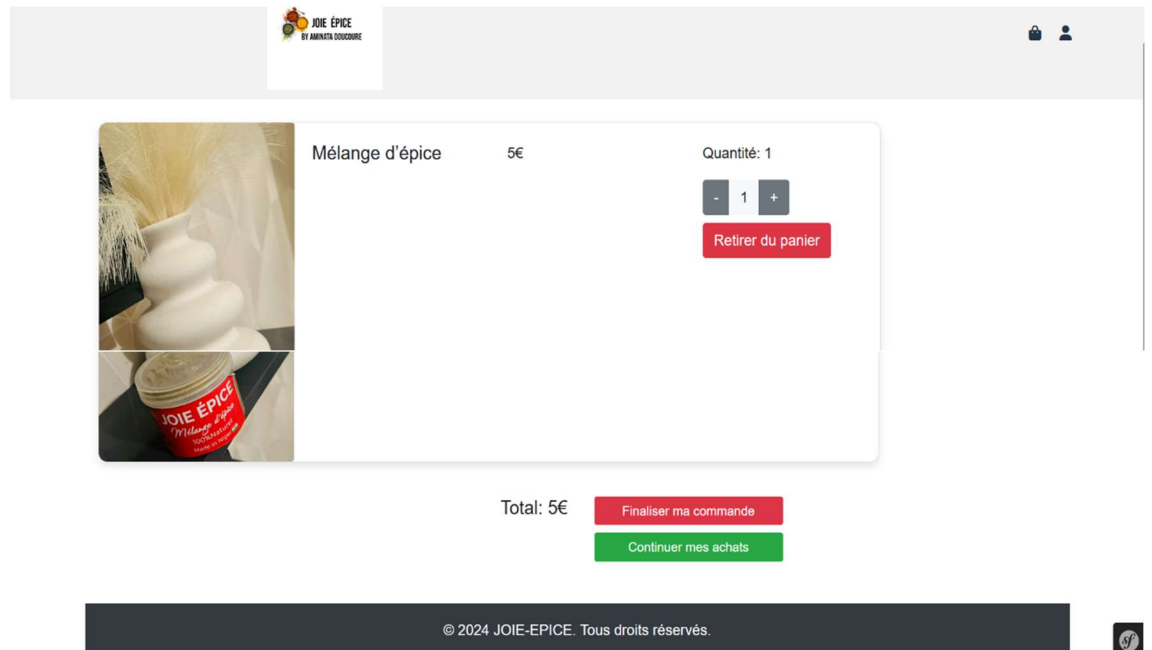
- Détails du produit sélectionné : nom, prix, image, description.
- Bouton "Ajouter au panier".



#### • Panier :

- Liste des produits ajoutés avec image, nom, prix et options de quantité (+/-).
- Bouton "Retirer du panier".

- Total de la commande avec boutons "Finaliser la commande" et "Continuer mes achats".



- **Page de connexion :**

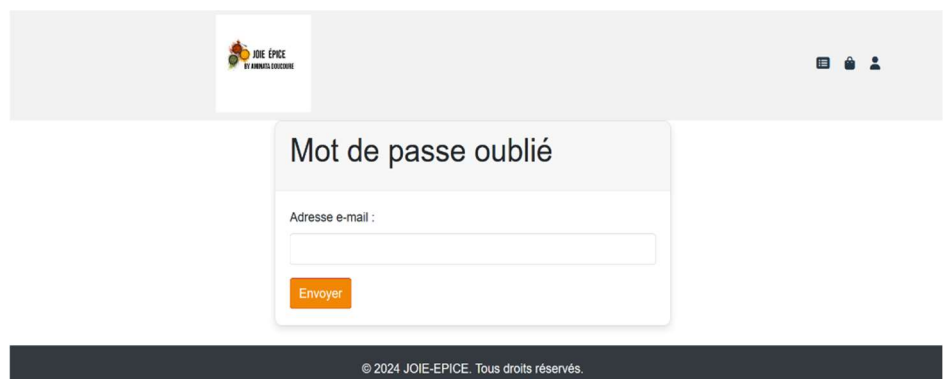
- Formulaire de connexion avec champs "Nom d'utilisateur", "Mot de passe" et lien "Mot de passe oublié ?".

- **Réinitialisation du mot de passe quand vous oubliez le mot de passe :**

- **Processus de demande :**

L'utilisateur clique sur le lien "Mot de passe oublié ?" sur la page de connexion.

Il est redirigé vers une page où il doit saisir l'adresse email associée à son compte.



Après soumission, si le courriel est bien celui associé à votre compte il vous redirige vers une page pour modifier votre e-mail.






### Confirmer la réinitialisation du mot de passe


Nouveau mot de passe :



Confirmer le nouveau mot de passe :


Confirmer

© 2024 JOIE-EPICE. Tous droits réservés.

- Option "S'inscrire" pour les nouveaux utilisateurs.





[Déconnexion](#)



Vous êtes connecté en tant que Mina, Déconnexion

### Connexion

Nom d'utilisateur


Mot de passe

[Mot de passe oublié ?](#)

Se connecter

- **Page historique des commandes** pour un utilisateur admin:
  - Liste des commandes passées avec ID, date, articles et total.
  - Bouton "Retour à l'accueil".



### Historique des Commandes

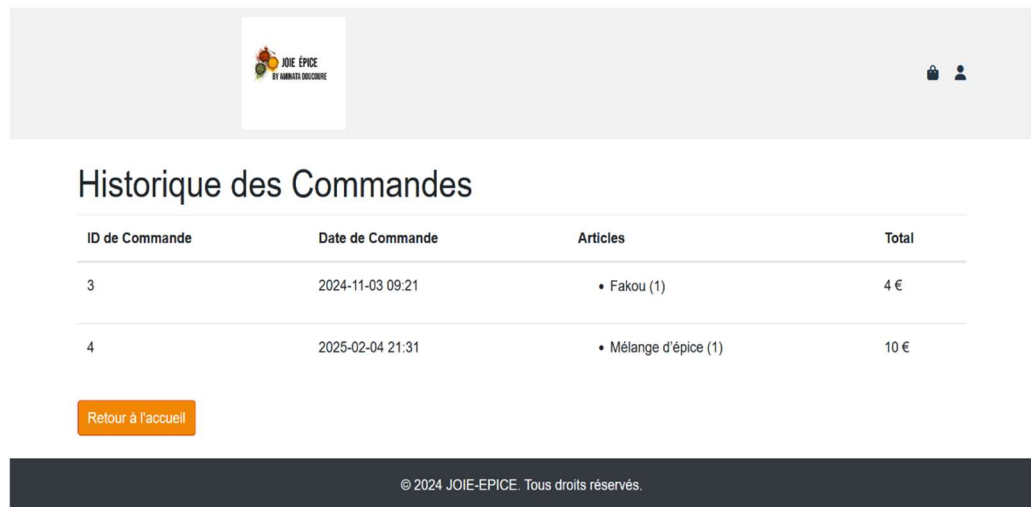
ID de Commande	Date de Commande	Nom d'Utilisateur	Articles	Total
3	2024-11-03 09:21	Mina	• Fakou (1)	4 €
4	2025-02-04 21:31	Mina	• Mélange d'épice (1)	10 €

Retour à l'accueil

© 2024 JOIE-EPICE. Tous droits réservés.

- **Page historique des commandes** pour un utilisateur client:

- Liste des commandes passées avec ID, date, articles et total.
- Bouton "Retour à l'accueil".



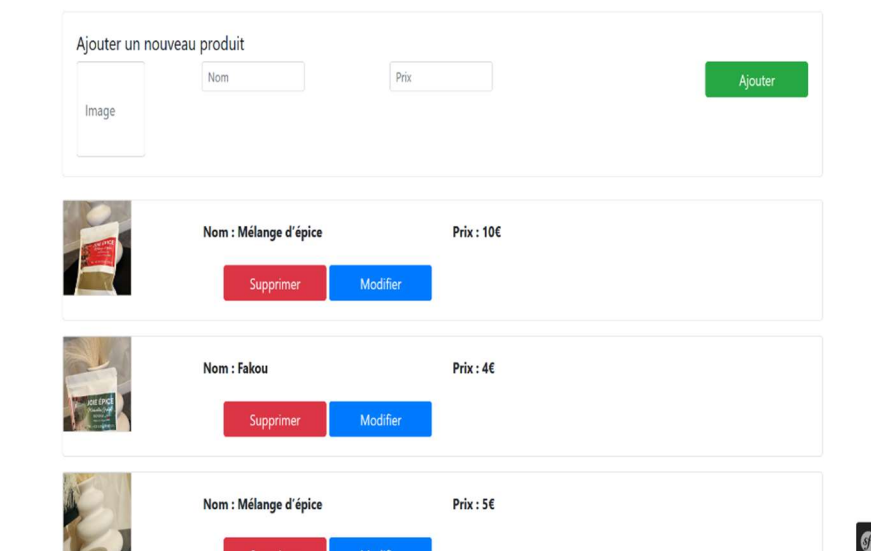
ID de Commande	Date de Commande	Articles	Total
3	2024-11-03 09:21	• Fakou (1)	4 €
4	2025-02-04 21:31	• Mélange d'épice (1)	10 €

Retour à l'accueil

© 2024 JOIE-EPICE. Tous droits réservés.

### III. Fonctionnalités

- **Gestion des utilisateurs :**
  - Création de compte, connexion, réinitialisation de mot de passe.
  - Gestion du profil utilisateur.
- **Gestion des produits :**
  - Ajout, modification, suppression de produits (réservé à l'administrateur).
  - Gestion des catégories de produits.



Ajouter un nouveau produit

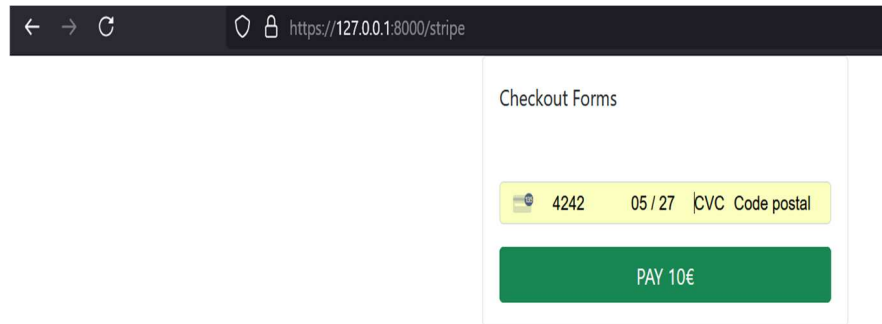
Image Nom Prix Ajouter

Nom : Mélange d'épice Prix : 10€ Supprimer Modifier

Nom : Fakou Prix : 4€ Supprimer Modifier

Nom : Mélange d'épice Prix : 5€ Supprimer Modifier

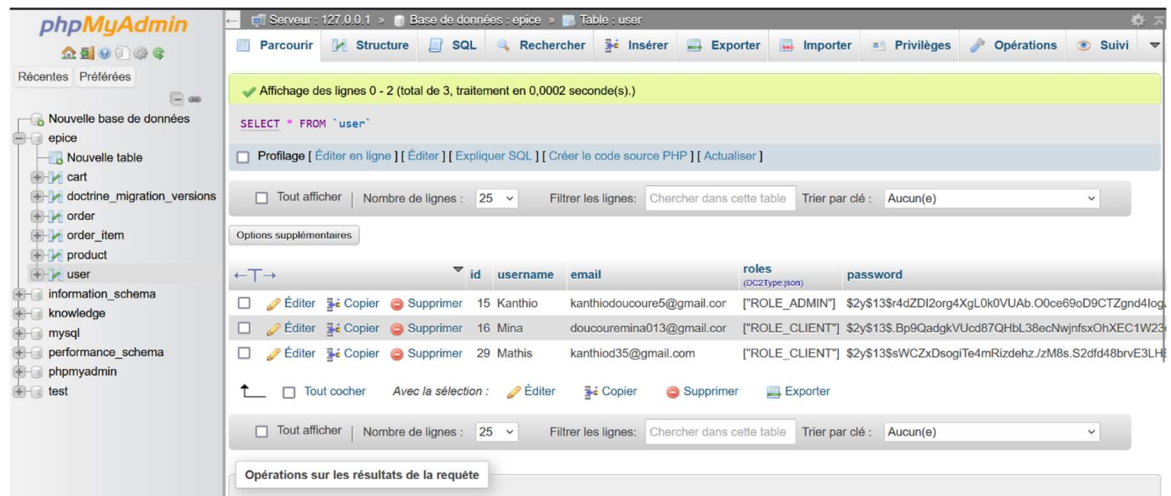
- **Paiement :**
  - Intégration de Stripe pour le traitement des paiements.



A screenshot of a web browser showing a Stripe checkout form. The browser's address bar at the top displays the URL `https://127.0.0.1:8000/stripe`. The page content is titled "Checkout Forms". Below the title, there is a light yellow input field containing a card icon, the number "4242", the expiration date "05 / 27", and the labels "CVC" and "Code postal". At the bottom of the form is a prominent green button with the text "PAY 10€".

## IV. Base de données

- **Tables principales :**
  - User : informations sur les utilisateurs.
  - Product : détails des produits.
  - Cart : paniers des utilisateurs.
  - Order : commandes passées.
  - OrderItem : éléments de chaque commande.



- **Relations :**

- Un utilisateur peut avoir plusieurs paniers et passer plusieurs commandes.
- Chaque panier appartient à un utilisateur et peut contenir plusieurs produits.
- Chaque commande est associée à un utilisateur et peut contenir plusieurs éléments de commande.
- Chaque élément de commande est lié à un produit spécifique.

## V. Évolutions potentielles

- **Bénévole :**

- Ajout de fonctionnalités de filtrage et de recherche avancée des produits.
- Intégration de recommandations personnalisées basées sur l'historique d'achat.

- **Administrateur :**

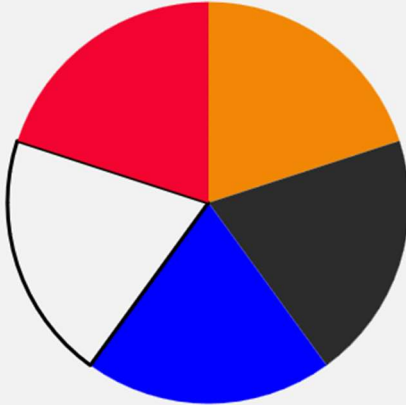
- Gestion des stocks et des promotions.
- Analyse des ventes et génération de rapports.






# CHARTRE GRAPHIQUE ET LOGO

<ul style="list-style-type: none"><li>• <b>Police d'écriture :</b> Bebas Neue</li></ul>	<p>Typography</p> <h2>Bebas Neue</h2> <table><tr><td>Regular</td><td>Bold</td></tr><tr><td>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</td><td>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</td></tr></table>	Regular	Bold	Lorem ipsum dolor sit amet, consectetur adipiscing elit.	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Regular	Bold				
Lorem ipsum dolor sit amet, consectetur adipiscing elit.	Lorem ipsum dolor sit amet, consectetur adipiscing elit.				

- Palette de couleurs

# Répartition des Couleurs



	#f28705		#2c2c2c
	#0000FF		#f2f2f2
	#f20530		

- **Logo :** Créé avec Canva, représentant une épice stylisée avec le nom de la marque et sa créatrice.



# RÉALISATIONS

## 1. Fonctionnalité principale

Pour ce projet, j'ai choisi de créer un système permettant aux utilisateurs de gérer des **produits** et de les ajouter à un **panier**.

J'ai créé un service pour gérer l'ajout des produits au panier. Ce service prend en entrée l'**ID du produit** et la **quantité souhaitée**, puis retourne un objet avec les informations du produit ajouté et la mise à jour du panier.

Ensuite, j'ai développé une fonction pour traiter l'ajout de ces produits au panier via une requête envoyée au backend. Cette fonction permet de vérifier si le produit est déjà présent dans le panier et met à jour la quantité ou ajoute le produit s'il n'est pas encore dans le panier.

Le code pour augmenter et diminuer la quantité :

```
#[Route('/increase-quantity/{id}', name: 'increase_quantity')]
```

```
public function increaseQuantity(int $id): Response
```

```
{
```

```
    $user = $this->getUser();
```

```
    if (!$user) {
```

```
        return $this->redirectToRoute('login');
```

```
    }
```

```
    $cartItem = $this->entityManager
```

```
        ->getRepository(Cart::class)
```

```
        ->find($id);
```

```
    if ($cartItem && $cartItem->getUser() === $user) {
```

```
        $product = $cartItem->getProduct();
```

```
        // Vérifier le stock disponible
```

```
        if ($product->getStock() > 0) {
```

```
            $cartItem->setQuantity($cartItem->getQuantity() + 1);
```

```
            // Déduire du stock
```

```
            $product->setStock($product->getStock() - 1);
```



```

        $this->entityManager->flush();
    } else {
        $this->addFlash('error', 'Ce produit est en rupture de stock');
    }
}

return $this->redirectToRoute('cart');
}

```

```

#[Route('/decrease-quantity/{id}', name: 'decrease_quantity')]
public function decreaseQuantity(int $id): Response
{
    $user = $this->getUser();
    if (!$user) {
        return $this->redirectToRoute('login');
    }
}

```

```

$cartItem = $this->entityManager
    ->getRepository(Cart::class)
    ->find($id);

```

```

if ($cartItem && $cartItem->getUser() === $user) {
    $product = $cartItem->getProduct();

    // Vérifier si la quantité est supérieure à 1
    if ($cartItem->getQuantity() > 1) {
        $cartItem->setQuantity($cartItem->getQuantity() - 1);
        // Restaurer le stock
        $product->setStock($product->getStock() + 1);
    } else {
        // Supprimer l'élément du panier et restaurer le stock
        $product->setStock($product->getStock() + 1);
        $this->entityManager->remove($cartItem);
    }
}

```

```

    }
    $this->entityManager->flush();
}

return $this->redirectToRoute('cart');
}

```

## 2. Ajout au Panier

Pour l'ajout au panier, j'ai utilisé un middleware de validation pour vérifier que le produit existe bien dans la base de données avant d'être ajouté. Si le produit est valide, il est inséré dans le panier de l'utilisateur.

Voici comment se passe l'ajout au panier au niveau des routes :

- J'ai créé une route pour gérer l'ajout d'un produit au panier.
- Lorsque l'utilisateur soumet l'action d'ajout, cette fonction vérifie si le produit est en stock et si le panier de l'utilisateur n'est pas déjà plein.
- Si tout est correct, le produit est ajouté et une réponse avec le panier mis à jour est retournée.

*Le code pour l'ajout d'un produit au panier :*

```

#[Route('/add-to-cart/{productId}', name: 'add_to_cart')]
public function addToCart(int $productId, ProductRepository $productRepository):
Response
{
    $product = $productRepository->find($productId);

    if (!$product) {
        throw $this->createNotFoundException('Produit non trouvé');
    }

    // Vérifier le stock disponible
    if ($product->getStock() <= 0) {
        $this->addFlash('error', 'Produit en rupture de stock');
        return $this->redirectToRoute('cart');
    }
}

```

```

$user = $this->getUser();
if (!$user) {
    return $this->redirectToRoute('login');
}

// Chercher ou créer un article dans le panier
$cartItem = $this->entityManager
    ->getRepository(Cart::class)
    ->findOneBy(['user' => $user, 'product' => $product]);

if ($cartItem) {
    $cartItem->setQuantity($cartItem->getQuantity() + 1);
} else {
    $cartItem = new Cart();
    $cartItem->setUser($user);
    $cartItem->setProduct($product);
    $cartItem->setQuantity(1);
}

// Déduire du stock
$product->setStock($product->getStock() - 1);

$this->entityManager->persist($cartItem);
$this->entityManager->flush();

return $this->redirectToRoute('cart');
}

```

---

### 3. Affichage du Panier

Sur le front-end, la gestion de l'affichage du panier est assez simple. J'ai utilisé une **condition pour vérifier** si le panier contient des produits. Si oui, je les affiche sous forme de liste avec le nom, la quantité et le prix. Si le panier est vide, un message indique à l'utilisateur qu'il n'y a pas de produits dans le panier.

J'ai également intégré un bouton "Passer la commande" qui n'est affiché que si le panier contient des produits. Ce bouton appelle la fonction de commande lorsque l'utilisateur est prêt à valider son achat.

```
#[Route('/remove-from-cart/{id}', name: 'remove_from_cart')]

public function removeFromCart(int $id): Response
{
    $user = $this->getUser();
    if (!$user) {
        return $this->redirectToRoute('login');
    }

    $cartItem = $this->entityManager
        ->getRepository(Cart::class)
        ->find($id);

    if ($cartItem && $cartItem->getUser() === $user) {
        // Récupérer le produit et restaurer la quantité
        $product = $cartItem->getProduct();
        $product->setStock($product->getStock() + $cartItem->getQuantity());

        $this->entityManager->remove($cartItem);
        $this->entityManager->flush();
    }

    return $this->redirectToRoute('cart');
}
```

---

## 4. Passer une Commande

Pour passer une commande, l'utilisateur clique sur le bouton "Passer la commande". Une fois cliqué, le backend reçoit une requête qui contient les informations du panier actuel.

Je vérifie que l'utilisateur a bien des produits dans son panier. Ensuite, je crée une commande en base de données et je marque les produits du panier comme commandés (le stock diminue).

---

## 5. Gestion des rôles

Le système de rôles dans cette application est simple. L'utilisateur peut être un **client** ou un **administrateur**.

- Un **client** peut ajouter des produits au panier et passer des commandes.
- Un **administrateur** a accès à une interface de gestion pour ajouter, modifier ou supprimer des produits, mais n'a pas la possibilité de passer des commandes.

Au niveau des pages, j'ai créé une condition pour afficher les fonctionnalités réservées aux administrateurs, comme la gestion des produits, tandis que les clients ont accès uniquement à la section "Panier" et "Commandes".

```
#[Route('/order/history', name: 'order_history')]
public function history(): Response
{
    $user = $this->getUser();

    if (!$user) {
        return $this->redirectToRoute('login');
    }

    // Vérifie si l'utilisateur est un administrateur
    if ($this->isGranted('ROLE_ADMIN')) {
        // Si l'utilisateur est un admin, récupérer toutes les commandes
        $orders = $this->entityManager->getRepository(Order::class)->findAll();
    } else {
        // Sinon, récupérer seulement les commandes de l'utilisateur connecté
        $orders = $this->entityManager->getRepository(Order::class)->findBy(['user' =>
$user]);
    }

    return $this->render('order/history.html.twig', [
        'orders' => $orders,
    ]);
}
```

# SPÉCIFICATIONS TECHNIQUES

## *A. Architecture du projet :*

- Framework utilisé : Symfony 5.x pour construire une application web robuste, performante et bien structurée, avec l'utilisation de MVC (Modèle-Vue-Contrôleur) pour séparer clairement les responsabilités.
- Base de données : Utilisation de MySQL comme système de gestion de base de données relationnelle, avec des entités et des relations gérées via Doctrine ORM.

## *B. Gestion des utilisateurs et de l'authentification :*

- Inscription et authentification : Mise en place de l'authentification via des formulaires sécurisés, en utilisant le système de sécurité natif de Symfony pour gérer les connexions des utilisateurs (login, logout).
- Rôles et autorisations : Les rôles des utilisateurs sont stockés dans la base de données et permettent de gérer l'accès aux différentes parties du site, avec des rôles comme utilisateur et administrateur.

## *C. Gestion du panier :*

- Entités liées au panier : Création d'une entité Cart pour gérer les articles dans le panier des utilisateurs. Chaque article est lié à un produit spécifique et peut contenir plusieurs quantités.
- Ajout au panier : Le processus d'ajout au panier vérifie la disponibilité du produit, ajuste la quantité en fonction des actions de l'utilisateur (ajout ou modification), et valide que le produit est en stock.
- Gestion des stocks : À chaque ajout d'article au panier, une vérification est effectuée pour s'assurer qu'il y a suffisamment de stock disponible pour le produit.

## *D. Sécurisation des formulaires et prévention des attaques :*

- Protection CSRF : Utilisation de tokens CSRF dans les formulaires pour protéger l'application contre les attaques de type Cross-Site Request Forgery.
- Validation des entrées utilisateurs : Mise en place de contrôles côté serveur pour valider et assainir les données entrées par l'utilisateur (par exemple, en utilisant des contraintes de validation Symfony pour les formulaires).
- Sanitisation des données : Les données envoyées par l'utilisateur sont soigneusement échappées pour prévenir les attaques XSS.

## *E. Gestion des images et des fichiers :*

- Téléchargement d'images : Intégration d'un système d'upload d'images via un service S3 (ou équivalent) pour la gestion des photos des produits, avec une vérification des formats et tailles de fichier acceptables.
- Gestion des erreurs de téléchargement : Mise en place de contrôles pour vérifier la validité des fichiers et éviter les erreurs d'upload.

### *F. Performance et optimisations :*

- Mise en cache : Mise en place d'un système de mise en cache via Symfony pour optimiser les requêtes de la base de données et améliorer la rapidité du rendu des pages.
- Optimisation des requêtes SQL : Utilisation de Doctrine DQL pour générer des requêtes SQL optimisées, avec un suivi de la performance de ces requêtes pour éviter des appels redondants ou inutiles.

### *G. Gestion des erreurs et des logs :*

- Gestion des exceptions : Mise en place de mécanismes de gestion des erreurs dans les contrôleurs pour capturer les erreurs système, avec des messages d'erreur personnalisés en fonction des exceptions rencontrées.

### *H. Tests et déploiement :*

- Tests fonctionnels et d'intégration : Mise en place de tests unitaires et fonctionnels avec PHPUnit pour valider la logique de l'application et la stabilité de ses fonctionnalités principales.
- CI/CD : Intégration continue et déploiement automatisé via des outils comme GitHub , afin d'assurer que le code est toujours dans un état déployable.

### *I. Interface utilisateur (UI) et expérience utilisateur (UX) :*

- Responsive Design : Conception de l'interface utilisateur en utilisant des pratiques de responsive design, assurant une expérience optimale sur desktop, mobile et tablette.

## Routes front-end et back-end

### A. Back-end

ENDPOINT	MÉTHODE	OUTPUT / DESCRIPTION	CONTROLLER (Méthode)	AUTH
/verify/email	GET	Traite la confirmation d'email (via URL signée) et valide l'adresse de l'utilisateur, redirige vers la page de connexion en cas de succès.	verifyUserEmail()	Public
/add-to-cart/{productId}	GET	Ajoute un produit au panier si le stock est disponible, décrémente le stock, et redirige vers le panier.	addToCart()	Connecté (client)
/remove-from-cart/{id}	GET	Supprime un article du panier, rétablit la quantité en stock et redirige vers le panier.	removeFromCart()	Connecté
/increase-quantity/{id}	GET	Incrémente la quantité d'un article dans le panier (si le stock le permet) et déduit la quantité en stock.	increaseQuantity()	Connecté
/decrease-quantity/{id}	GET	Diminue la quantité d'un article dans le panier ou supprime l'article si la quantité devient inférieure à 1 (en restaurant le stock).	decreaseQuantity()	Public (ou Connecté selon votre logique)
/product/{id}	GET	Affiche les détails d'un produit en fonction de son identifiant.	productDetails()	Connecté
/stripe/create-charge	POST	Traite le paiement Stripe (création de charge), crée la	createCharge()	Connecté



		commande correspondante, vide le panier et redirige avec un message de confirmation.		
--	--	--	--	--

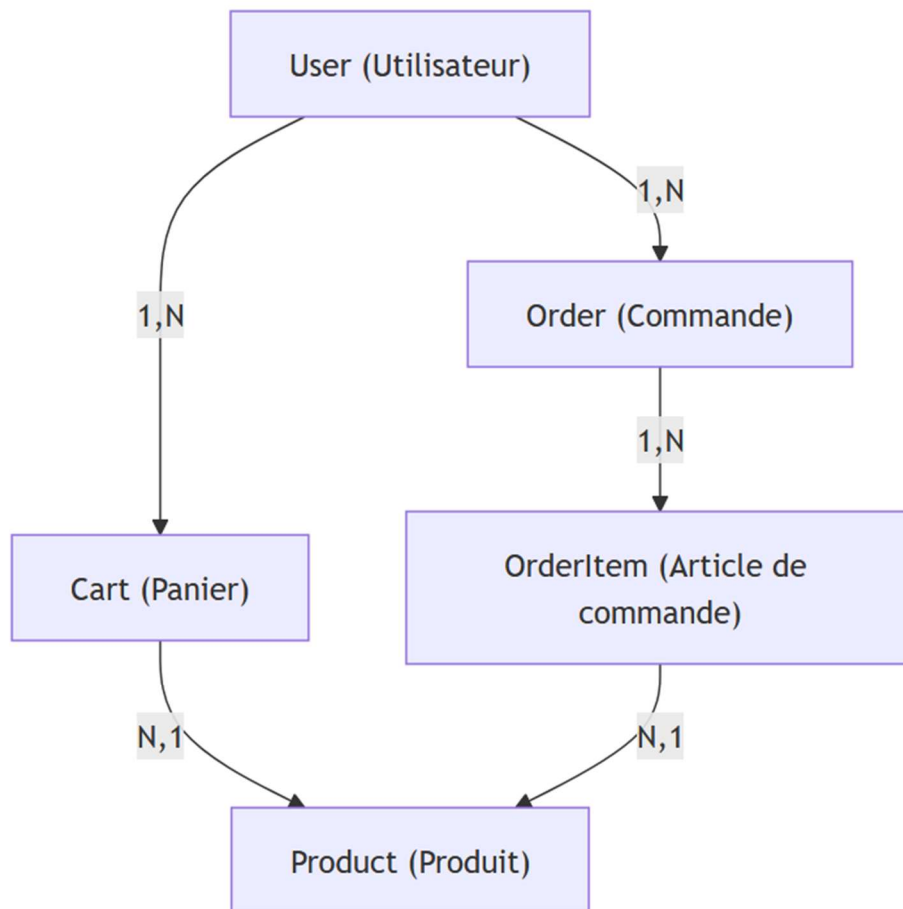
## B. Front-end

ROUTE	PAGE / CONTENU	UTILISATEUR
/admin	Dashboard administrateur (gestion des produits : ajout, modification, suppression)	Administrateur
/mot-de-passe-oublie	Page "Mot de passe oublié" (formulaire de saisie de l'email)	Non-connecté
/confirmer-reset-mot-de-passe/{email}	Page de confirmation de réinitialisation du mot de passe (saisie du nouveau mot de passe)	Non-connecté
/home	Page d'accueil affichant la liste des produits	Public
/cart	Page du panier affichant les articles ajoutés et le total	Connecté
/order/history	Page de l'historique des commandes (soit toutes les commandes pour admin, soit ses propres commandes pour client)	Connecté
/product/{id}	Page de détails d'un produit	Public (ou Connecté)
/register	Page d'inscription (formulaire de création de compte)	Non-connecté
/login	Page de connexion (formulaire d'authentification)	Non-connecté
/stripe	Page de paiement (intégration Stripe)	Connecté

# Création de la base de données

## A. MCD

### Modèle Conceptuel de Données (MCD)



## B. MLD

**User** (id, username, email, roles, password, is\_verified)

**Product** (id, name, price, image, description, stock)

**Cart** (id, user\_id, product\_id, quantity)

**Order** (id, user\_id, order\_date)

**OrderItem** (id, order\_id, product\_id, quantity)

# VULNÉRABILITÉS DE SÉCURITÉ ET VEILLE

Dans le cadre de mon projet, j'ai mis en place plusieurs mesures pour assurer la sécurité de l'application et protéger les utilisateurs contre des attaques potentielles. En particulier, j'ai pris les actions suivantes :

1. **Protection contre l'injection SQL** : J'ai utilisé un ORM pour interagir avec la base de données et éviter les requêtes SQL malveillantes. En utilisant Doctrine, j'ai pu garantir que les entrées des utilisateurs sont automatiquement échappées, ce qui empêche les attaques par injection SQL.
2. **Prévention contre les attaques XSS (Cross-Site Scripting)** : J'ai veillé à échapper correctement toutes les sorties HTML provenant des utilisateurs afin d'éviter toute exécution de scripts malveillants dans le navigateur de l'utilisateur. Pour ce faire, j'ai utilisé des fonctions d'échappement dans le code PHP et m'assuré que les données affichées dans le front-end sont sécurisées.
3. **Protection contre le CSRF (Cross-Site Request Forgery)** : J'ai intégré des tokens CSRF dans les formulaires de l'application afin d'empêcher qu'un attaquant ne soumette des requêtes malveillantes en exploitant la session de l'utilisateur. Cela garantit que chaque action sensible dans l'application provient bien de l'utilisateur authentifié.
4. **Gestion sécurisée des sessions et de l'authentification** : J'ai utilisé des cookies sécurisés pour protéger les sessions des utilisateurs. De plus, pour garantir que seules les personnes autorisées ont accès à leurs comptes, j'ai mis en place une gestion rigoureuse des rôles et de l'accès aux différentes fonctionnalités de l'application.
5. **Protection des données sensibles** : J'ai mis en place des pratiques de chiffrement des données sensibles. Les informations personnelles des utilisateurs sont protégées et les transactions sont effectuées via HTTPS pour sécuriser les échanges de données entre le client et le serveur.
6. **Mises à jour des dépendances et surveillance de la sécurité** : J'ai veillé à ce que toutes les bibliothèques et dépendances utilisées dans le projet soient à jour. Grâce à des outils de gestion de dépendances comme Composer, j'ai mis en place une veille de sécurité pour détecter et corriger rapidement toute vulnérabilité connue.
7. **Gestion des erreurs et des logs** : Pour éviter de divulguer des informations sensibles aux attaquants, j'ai configuré un système de gestion des erreurs et des logs qui ne révèle pas de détails techniques ou d'informations confidentielles dans les messages d'erreur visibles par l'utilisateur final.

En appliquant ces mesures, j'ai réussi à garantir un niveau de sécurité satisfaisant pour l'application et ses utilisateurs. Toutefois, la sécurité étant un processus en constante évolution, je m'engage à effectuer une veille régulière et à ajuster les mesures en fonction des nouvelles menaces qui pourraient émerger.

# CONCLUSION

Ce projet a permis de mettre en place une solution complète de gestion de panier pour un site e-commerce. L'objectif principal était de faciliter l'ajout de produits au panier tout en assurant une expérience fluide et sécurisée pour l'utilisateur.

Le processus d'ajout au panier repose sur plusieurs étapes importantes :

Vérification de la disponibilité des produits : Avant d'ajouter un produit au panier, il est essentiel de vérifier que ce dernier est bien en stock. Si le produit est en rupture de stock, un message d'erreur est affiché pour informer l'utilisateur.

Gestion du panier utilisateur : Chaque utilisateur peut ajouter des produits à son panier. Si un produit est déjà présent dans le panier, la quantité est mise à jour. Si le produit est nouveau, il est ajouté comme un nouvel article dans le panier.

Gestion des utilisateurs connectés : Une vérification est effectuée pour s'assurer que l'utilisateur est bien connecté avant de pouvoir ajouter des articles à son panier. En cas d'oubli, l'utilisateur est redirigé vers la page de connexion.

Expérience utilisateur fluide : En cas de succès ou d'échec de l'ajout au panier, des messages appropriés sont affichés, permettant ainsi une navigation claire et réactive.

L'architecture mise en place garantit non seulement la sécurité des données, mais aussi une gestion efficace des produits dans le panier. Ce processus est complété par la possibilité pour l'utilisateur de consulter et modifier son panier avant de passer à l'étape de commande.

Ce projet est une étape clé dans la construction d'une plateforme de commerce en ligne et peut être facilement étendu pour inclure des fonctionnalités supplémentaires, comme la gestion des paiements, le suivi des commandes ou l'intégration de systèmes de recommandations produits.

La mise en place d'une telle fonctionnalité permet d'offrir une expérience client optimale, tout en facilitant la gestion des commandes et des stocks du côté administrateur. Ce projet constitue donc une base solide pour toute application e-commerce.

# ANNEXE

Les repos du projet sont accessibles à cette adresse : <https://github.com/DKanthio/Joie-epice>

RÔLE	NOM D'UTILISATEUR	MOT DE PASSE
Administrateur	Kanthio	maman
Bénévole	Mina	doucoure