

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*



Paris - 10/02/2025

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.....	6
A. <i>Créer une base de données</i>	6
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>	23
E. <i>Gestion des images et des fichiers</i>	23
F. <i>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>	24
<i>J. Gestion du code avec Git et GitHub</i>	24
Routes front-end et back-end.....	24
A. Back-end	24
B. Front-end	26
Création de la base de données.....	27
A.MCD.....	27
B.MLD	27
VULNÉRABILITÉS DE SÉCURITÉ ET VEILLE.....	29
CONCLUSION	30
ANNEXE.....	31

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", 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.

Wireframes :

Pour réaliser la maquette de mon site, j'ai utilisé Figma, un outil de design collaboratif très intuitif, qui permet de concevoir des interfaces précises tout en intégrant des éléments interactifs. Mon objectif était de créer une maquette fidèle à l'apparence et aux fonctionnalités du site final, tout en pensant à la navigation sur différents supports.

J'ai donc conçu deux versions distinctes de la maquette : une pour un affichage desktop, et une autre optimisée pour l'iPhone 16 Pro Max. Cela m'a permis d'adapter le design et les éléments interactifs en fonction des dimensions de l'écran, afin d'assurer une expérience utilisateur fluide et cohérente, quel que soit le type d'appareil utilisé.

La maquette inclut des carrousels interactifs pour faire défiler les images, des boutons fonctionnels simulant les actions principales, ainsi qu'une structure fidèle à celle du site en ligne. Les couleurs, typographies, espacements et alignements ont été soigneusement respectés pour garantir une cohérence visuelle entre la maquette et le rendu final. Grâce à Figma, j'ai pu visualiser, tester et affiner le design avant le développement, ce qui m'a permis d'optimiser l'ergonomie du site sur tous les supports.

Desktop :

Connexion

Connexion
Nom d'utilisateur
Mot de passe
Mot de passe oublié ?
S'inscrire

© 2024 JOE-EPICE. Tous droits réservés.

accueil client

Nos produits
TSE - Mélange épices
AC - Piment rouge
Mélange épices

© 2024 JOE-EPICE. Tous droits réservés.
Génération de code QR

accueil admin

Nos produits
TSE - Mélange épices
AC - Piment rouge
Mélange épices

© 2024 JOE-EPICE. Tous droits réservés.

Inscription

S'inscrire
Nom d'utilisateur
Adresse mail
Mot de passe
Confirmer mot de passe
S'inscrire

© 2024 JOE-EPICE. Tous droits réservés.

Déconnexion

Votre connexion est en tant que ... Déconnexion
Connexion
Nom d'utilisateur
Mot de passe
Mot de passe oublié ?
S'inscrire

© 2024 JOE-EPICE. Tous droits réservés.

Admin

Ajouter un nouveau produit	Image	Nom	Prix	
		Nom: Mélange épices	Prix: 10€	Ajouter Modifier
		Nom: Piment rouge	Prix: 4€	Ajouter Modifier
		Nom: Mélange épices	Prix: 14€	Ajouter Modifier

Panier

Mélange d'épice 5€ Quantité : 1
Retirer de mon panier

Total : 5€ **Continuer à commander** **Continuer mes achats**

© 2024 JOE-EPICE. Tous droits réservés.

Produit

Fakou 4€
Le fakou est une épice traditionnelle utilisée principalement au Nigéria, en Afrique centrale, où elle est utilisée dans les plats locaux tels que le jambon et le riz. Cet épice est composée de diverses graines et feuilles qui donnent une saveur unique et complexe. Elle est souvent utilisée pour assaisonner les plats locaux tels que le fufu et le gari.

Ajouter au panier

© 2024 JOE-EPICE. Tous droits réservés.

Paiement

Checkout forms
Numéro de carte MM/AA
PAYER

Historique Admin

Historique des commandes

ID de commande	Date de commande	Nom d'utilisateur	Articles	Total
----------------	------------------	-------------------	----------	-------

Résumer à l'accueil

© 2024 JOE-EPICE. Tous droits réservés.

Historique client

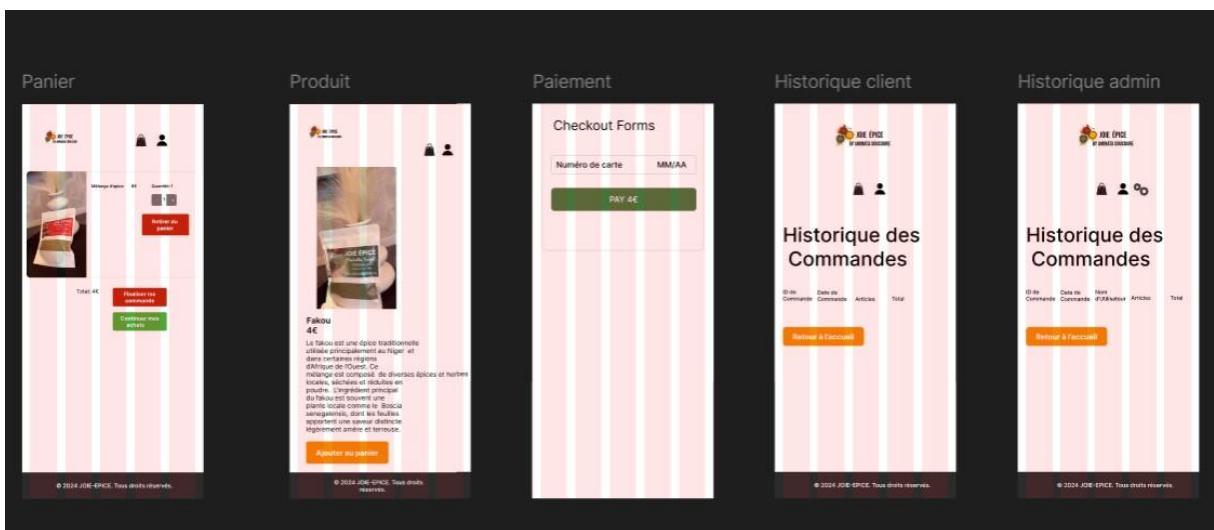
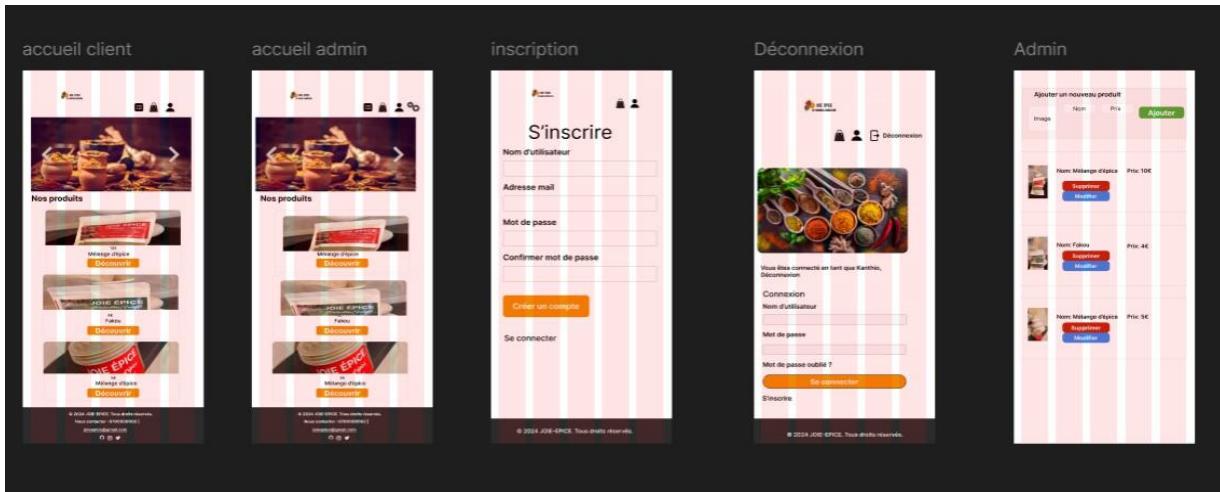
Historique des commandes

ID de commande	Date de commande	Nom d'utilisateur	Articles	Total
----------------	------------------	-------------------	----------	-------

Résumer à l'accueil

© 2024 JOE-EPICE. Tous droits réservés.

□ Mobile :



Pour plus de lisibilité, l'intégralité des wireframes est disponible à l'adresse suivante:

<https://www.figma.com/design/FnszthSrKOAwccq6l9yKug/Untitled?t=CmyEDfpHiK0GBzRz-0>

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=ni6Ufe9s8Ig>

Page produit :

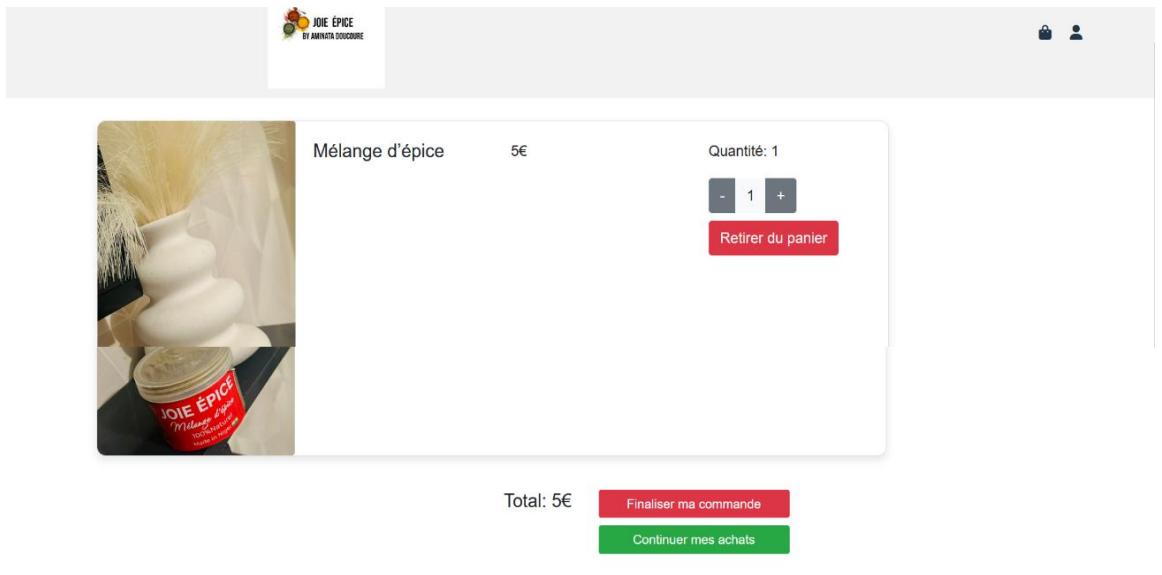
- Détails du produit sélectionné : nom, prix, image, description.
- Bouton "Ajouter au panier".



The screenshot shows a product page for a spice blend. At the top, there's a logo for 'JOIE ÉPICE BY ANNITA DOUCOURÉ'. Below the logo is a large image of a white ceramic vessel containing a yellow, fibrous substance, identified as the spice mix. To the right of the image, the product name 'Mélange d'épice' is displayed in bold black text, followed by the price '5€'. A detailed description in French follows: 'Un mélange d'épices unique et aromatique, combinant la douceur de la cannelle et de la vanille, les notes épicées des poivres, et la fraîcheur du curcuma et du gingembre, rehaussé par une sélection d'autres épices fines pour des saveurs riches et équilibrées.' At the bottom right is an orange button labeled '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".



The screenshot shows the shopping cart page. It features the same product image and details as the previous screenshot: 'Mélange d'épice' at 5€, quantity 1, with a 'Retirer du panier' button. Below this, the total 'Total: 5€' is shown, along with two buttons: a red 'Finaliser ma commande' button and a green 'Continuer mes achats' button. At the very bottom of the page is a dark footer bar with the text '© 2024 JOIE-EPICE. Tous droits réservés.' and a small logo.

- **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 :**

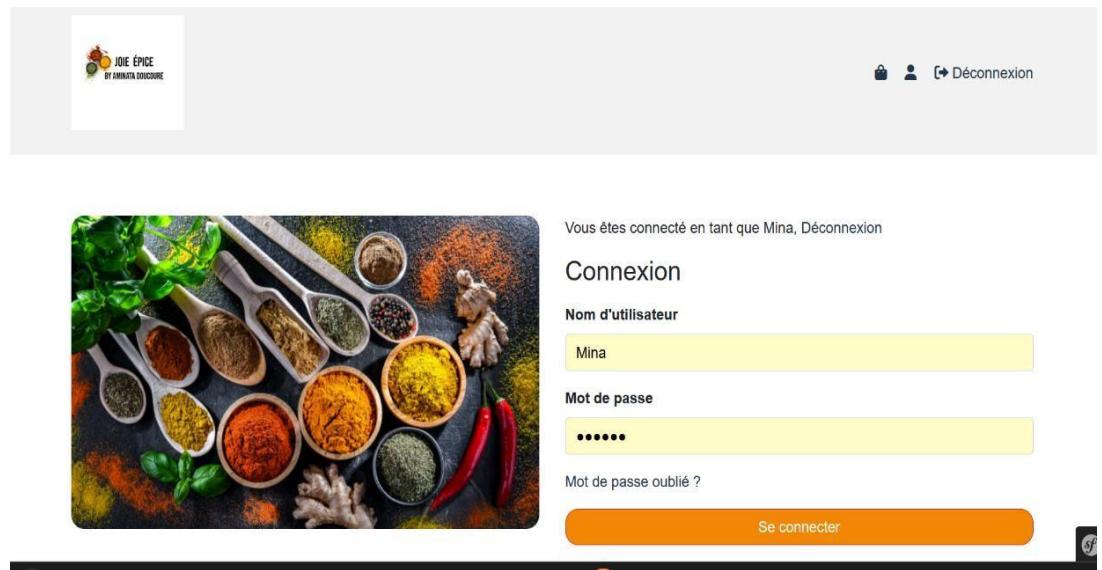
- **Processus de demande :**

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.

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



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

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".

The screenshot shows the 'Historique des Commandes' (Order History) page. At the top, there is a logo for 'JOIE ÉPICE BY ANNATH DISCOURS' and a user icon. Below the header is a table with two rows of order data:

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 €

Below the table is an orange button labeled 'Retour à l'accueil' (Return to Home). At the bottom of the page is a dark footer bar with the text '© 2024 JOIE-EPICE. Tous droits réservés.'

III. Fonctionnalités

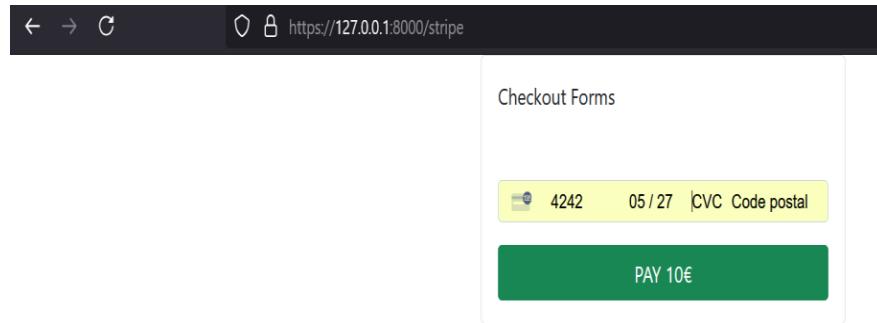
- Gestion des utilisateurs :**
 - Création de compte, connexion, réinitialisation de mot de passe.
 - Gestion du profil utilisateur.
- Gestion des produits :**
 - Ajout, modification, suppression de produits (réservé à l'administrateur).
 - Gestion des catégories de produits.

The screenshot shows a product management interface with three entries:

- Mélange d'épice**: Nom : Mélange d'épice, Prix : 10€. Actions: Supprimer (red), Modifier (blue).
- Fakou**: Nom : Fakou, Prix : 4€. Actions: Supprimer (red), Modifier (blue).
- Mélange d'épice**: Nom : Mélange d'épice, Prix : 5€. Actions: Supprimer (red), Modifier (blue).

At the bottom right of the interface is a small logo for Stripe.

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



IV. Base de données

Tables principales

user

- Cette table contient les informations des utilisateurs inscrits.
- Attributs principaux : username, email, password, roles, et un champ is_verified pour savoir si le compte est activé.
- Un utilisateur peut avoir plusieurs rôles (stockés au format JSON).
- Il peut passer des commandes ou remplir un panier.

cart

- Représente le panier temporaire d'un utilisateur.
- Il contient les produits que l'utilisateur souhaite acheter.
- Chaque entrée dans cette table correspond à un produit sélectionné, avec une quantité.
- Il est lié à un user (l'auteur du panier) et à un product.

Relation :

- Un utilisateur peut avoir plusieurs produits dans son panier.
- Un produit peut apparaître dans plusieurs paniers différents.

product

- Stocke les produits disponibles à la vente.
- Attributs : name, price, image, description, stock.
- Ces produits sont utilisés dans les paniers et dans les commandes.

Relations :

- Un produit peut être dans plusieurs paniers (cart) ou dans plusieurs commandes (order_item).

order

- Représente une commande réelle passée par un utilisateur.
- Attributs : order_date, user_id.
- Chaque commande est liée à un utilisateur unique.

Relation :

- Un utilisateur peut passer plusieurs commandes.

order_item

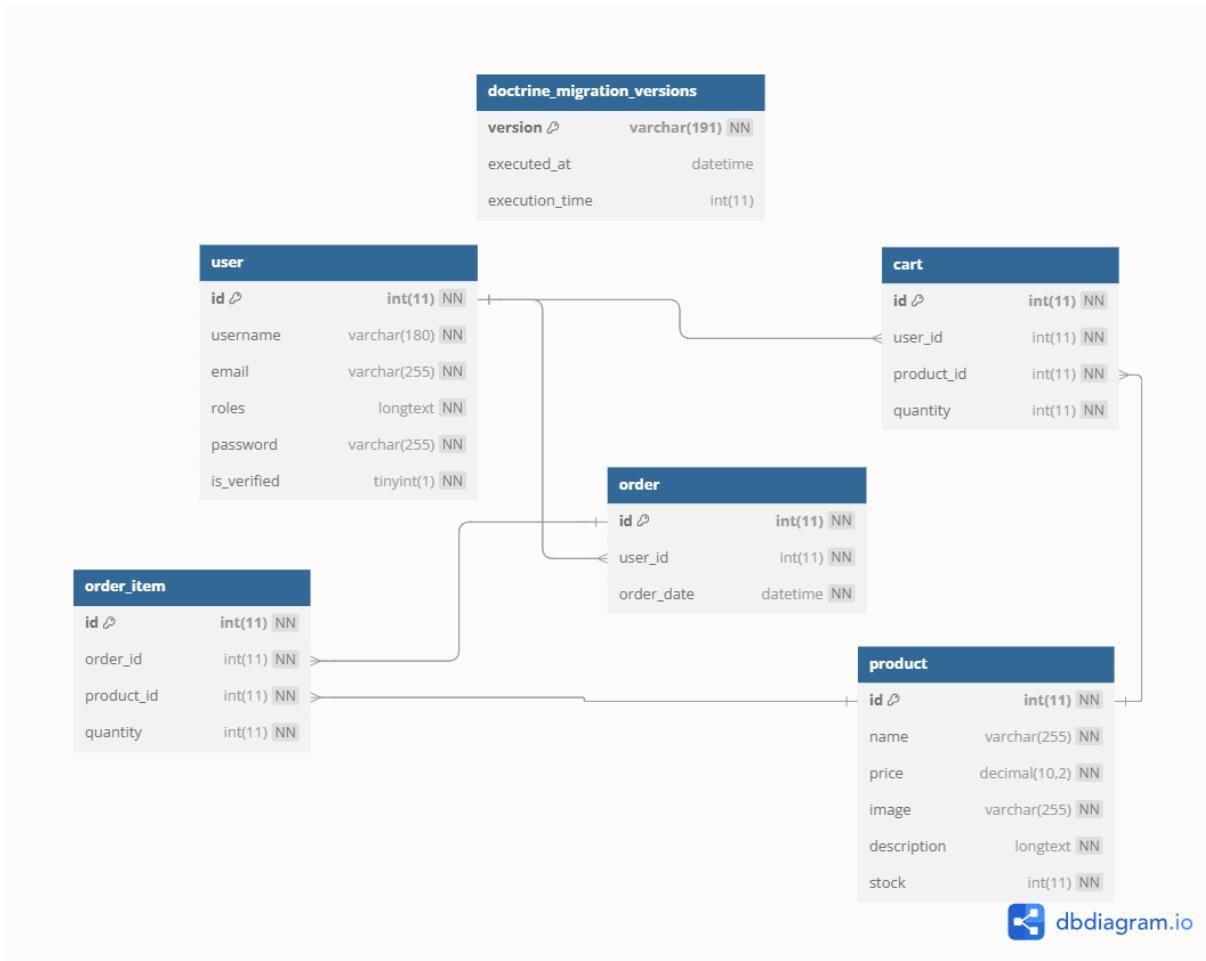
- Table de détail des lignes de commande.
- Chaque ligne contient un product_id, un order_id et une quantity.
- Cela permet de gérer les commandes multi-produits.

Relations :

- Chaque commande peut avoir plusieurs lignes de commande.
- Chaque ligne de commande concerne un seul produit.

doctrine_migration_versions

- Table technique utilisée par Doctrine (framework PHP) pour gérer l'historique des migrations de base de données.
- Elle ne concerne pas directement le métier, mais garantit le bon suivi de l'évolution du schéma.



Pour plus de détails :<https://dbdiagram.io/d/681bc6065b2fc4582fa97d52>

Affichage des lignes 0 - 2 (total de 3, traitement en 0,0002 seconde(s)).

SELECT * FROM `user`

	id	username	email	roles (JSON)	password
<input type="checkbox"/>	15	Kanthio	kanthiodoucoure5@gmail.com	["ROLE_ADMIN"]	\$2y\$13\$4dZDI2org4XgLoK0VUAbO0ce69oD8CTZgrnd4log
<input type="checkbox"/>	16	Mina	doucouremina013@gmail.com	["ROLE_CLIENT"]	\$2y\$13\$.Bp9QadgkVUcd87QHbL38ecNwjnfsxOhXEC1W23
<input type="checkbox"/>	29	Mathis	kanthiod35@gmail.com	["ROLE_CLIENT"]	\$2y\$13\$\$WCZxDsogiTe4mRizdehz.zM8s.S2dfd48brvE3LH

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

- **Police d'écriture :** Bebas Neue

Typography

Bebas Neue

Regular

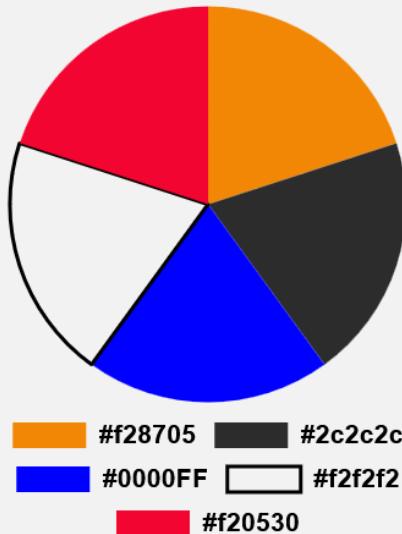
Lorem ipsum dolor sit amet, consectetur
adipiscing elit.

Bold

Lorem ipsum dolor sit amet, consectetur
adipiscing elit.

- Palette de couleurs

Répartition des Couleurs



- **Logo :** Crée 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

Dans mon projet, le back transmet les infos au front via Twig, le moteur de template de Symfony. Le contrôleur récupère les données utiles, comme les produits à afficher ou les formulaires, puis les passe au template Twig sous forme de variables. Twig injecte ces variables directement dans le HTML généré côté serveur. Du coup, quand la page arrive sur le navigateur, toutes les données sont déjà intégrées dans le code, prêtes à être affichées à l'utilisateur.

A. Architecture du projet :

L'interface utilisateur du site e-commerce **JOIE-EPICE** a été conçue en utilisant **Twig** (**moteur de template Symfony**), **Bootstrap 4.5**, **HTML5**, et **CSS personnalisé** pour garantir une **expérience utilisateur fluide et responsive**.

Page d'accueil / Liste des produits

Chaque produit est affiché dans une **carte Bootstrap**, avec un bouton d'action vers sa fiche :

```
<div class="col-md-3">
    <div class="card mb-3">
        
            <p class="card-text">{{ product.price }}€</p>
            <h4 class="card-title">{{ product.name }}</h4>
            <a href="{{ path('product_details', { 'id': product.id }) }}" class="btn btn-primary">Découvrir</a>
        </div>
    </div>
</div>
```

Design responsive avec la grille `col-md-3`, hover animations, et intégration Twig dynamique.

.

Page produit

Chaque fiche produit permet de **consulter les détails** et d'ajouter au panier :

```
<div class="row">
    <div class="col-md-6 d-flex justify-content-center">
        
    <div class="col-md-6 d-flex align-items-center">
        <div class="product-details">
            <h2>{{ product.name }}</h2>
```

```

<h3>{{ product.price }}€</h3>
<p>{{ product.description }}</p>
<a href="{{ path('add_to_cart', { 'productId': product.id }) }}" class="btn btn-primary">Ajouter au panier</a>
</div>
</div>
</div>

```

✿ Présentation claire du produit, bouton bien visible, design fluide sur mobile et desktop.

Panier

Affichage des articles ajoutés, gestion des quantités et lien vers le paiement Stripe :

```

<div class="card mb-3">
  <div class="row no-gutters">
    <div class="col-md-3">
      
    </div>
    <div class="col-md-3">
      <h5>{{ item.product.name }}</h5>
      <p>{{ item.product.price }}€</p>
    </div>
    <div class="col-md-3">
      Quantité : {{ item.quantity }}
      <div class="btn-group">
        <a href="{{ path('decrease_quantity', { 'id': item.id }) }}" class="btn btn-secondary">-</a>
        <a href="{{ path('increase_quantity', { 'id': item.id }) }}" class="btn btn-secondary">+</a>
      </div>
    </div>
    <div class="col-md-3">
      <a href="{{ path('remove_from_cart', { 'id': item.id }) }}" class="btn btn-danger">Retirer</a>
    </div>
  </div>
</div>

```

```
</div>
</div>



 L'utilisateur peut modifier en temps réel la quantité ou supprimer un article.


```

Historique des commandes

Affichage conditionnel (admin ou non), détails par commande :

```
<table class="table">
  <thead>
    <tr>
      <th>ID</th>
      <th>Date</th>
      { % if is_granted('ROLE_ADMIN') % }<th>Utilisateur</th>{ % endif % }
      <th>Articles</th>
      <th>Total</th>
    </tr>
  </thead>
  <tbody>
    { % for order in orders % }
    <tr>
      <td>{ { order.id } }</td>
      <td>{ { order.orderDate|date('Y-m-d H:i') } }</td>
      { % if is_granted('ROLE_ADMIN') % }<td>{ { order.user.username } }</td>{ % endif %
      }
      <td>
        <ul>
          { % for item in order.orderItems % }
          <li>{ { item.product.name } } ({ { item.quantity } })</li>
          { % endfor %}
        </ul>
      </td>
      <td>{ { order.total } }€</td>
    </tr>
    { % endfor %}
```

```
</tbody>
```

```
</table>
```

 Affichage intelligent avec Twig : l'admin voit les utilisateurs, les clients non.

Page de paiement – Stripe Checkout

Cette page permet à l'utilisateur d'effectuer son paiement en toute sécurité via l'API Stripe. Elle combine Bootstrap pour la mise en page responsive et JavaScript pour l'intégration Stripe.

```
<div class="card-body">  
    {% for message in app.flashes('success') %}  
        <div style="color: green; border: 2px solid green; text-align: center; padding: 5px; margin-bottom: 10px;">  
            {{ message }}  
        </div>  
    {% endfor %}  
  
<form id='checkout-form' method='post' action="{{ path('app_stripe_charge') }}">  
    <input type='hidden' name='stripeToken' id='stripe-token-id'>  
    <label for="card-element" class="mb-5">Checkout Forms</label><br>  
    <div id="card-element" class="form-control"></div>  
    <button  
        id='pay-btn'  
        class="btn btn-success mt-3"  
        type="button"  
        style="margin-top: 20px; width: 100%; padding: 7px;"  
        onclick="createToken()">  
        >  
        PAY {{ total }}€  
    </button>  
</form>  
</div>  


 Fonctionnalités clés :



- Intégration de Stripe via JavaScript (stripe.js)

```

- Tokenisation des cartes pour une sécurité maximale : aucune donnée bancaire n'est stockée sur le serveur
- Message flash de confirmation affiché après le paiement
- Formulaire stylisé avec Bootstrap et interface responsive

❖ Sécurité et logique métier :

- Clé Stripe dynamique injectée depuis le contrôleur : {{ stripe_key }}
- Utilisation de createToken() pour générer un token unique
- Envoi du formulaire après validation par Stripe

⌚ Page de connexion

Page élégante et fonctionnelle pour permettre aux utilisateurs de se connecter à leur compte.

```
<form method="post" action="{{ path('app_login') }}">
    {% if error %}
        <div class="alert alert-danger">{{ error.messageKey|trans(error.messageData, 'security') }}</div>
    {% endif %}

    {% if app.user %}
        <div class="mb-3">
            Vous êtes connecté en tant que {{ app.user.userIdentifier }},
            <a href="{{ path('app_logout') }}">Déconnexion</a>
        </div>
    {% endif %}

    <h1 class="h3 mb-3 font-weight-normal">Connexion</h1>

    <div class="form-group">
        <label for="inputUsername">Nom d'utilisateur</label>
        <input type="text" value="{{ last_username }}" name="username" id="inputUsername" class="form-control" autocomplete="username" required autofocus>
    </div>

    <div class="form-group">
        <label for="inputPassword">Mot de passe</label>
```

```

<input type="password" name="password" id="inputPassword" class="form-control"
autocomplete="current-password" required>

</div>

<div class="form-group mb-3">
  <a href="{{ path('forgot_password') }}>Mot de passe oublié ?</a>
</div>

<input type="hidden" name="_csrf_token" value="{{ csrf_token('authenticate') }}>

<button class="btn btn-primary" type="submit">Se connecter</button>

<p class="mt-3">
  <a href="{{ path('app_register') }}>S'inscrire</a>
</p>
</form>

```

✓ Design & ergonomie :

- Layout en deux colonnes : image à gauche, formulaire à droite
- Responsive et moderne grâce à Bootstrap 4
- Image de fond personnalisée pour renforcer l'identité visuelle

⚙ Fonctionnalités de sécurité :

- Protection CSRF avec {{ csrf_token('authenticate') }}
- Affichage dynamique des erreurs de connexion
- Redirection conditionnelle si l'utilisateur est déjà connecté

☰ Détails supplémentaires :

- Liens intégrés vers l'inscription et le mot de passe oublié
- Header avec logo, liens vers le panier, compte et déconnexion
- Footer responsive pour la cohérence visuelle du site

Page d'inscription

Structure globale :

- **HTML de base** avec intégration de Bootstrap, FontAwesome, et un CSS personnalisé (inscription.css).
- Utilise Twig et Symfony Forms (form_start, form_row, form_end).

Détail du contenu :

Header :

```
<header>
```

```

    <div class="container">
        <div class="row align-items-center">
            <div class="col-md-6 logo">
                <a href="{{ path('home') }}">
                    
                </a>
            </div>
            <div class="col-md-6 text-right">
                <nav>
                    <a href="#" class="text ml-3"><i class="fas fa-shopping-bag"></i></a>
                    <a href="{{ path('login') }}" class="text ml-3"><i class="fas fa-user"></i></a>
                </nav>
            </div>
        </div>
    </div>
</header>
```

- Logo placé à gauche avec lien vers la route home.

- Icônes à droite (sac pour le panier, utilisateur pour la connexion).

Formulaire d'inscription :

- Affiché au centre de la page (col-md-6).
- Titre principal : S'inscrire.
- Affiche les erreurs globales du formulaire avec {{ form_errors(registrationForm) }}.

Champs de formulaire :

Chaque champ est intégré avec la classe Bootstrap form-control :

```
{{ form_row(registrationForm.username, {'attr': {'class': 'form-control'}}) }}
```

```
{ { form_row(registrationForm.email, {'attr': {'class': 'form-control'}}) } }  
{ { form_row(registrationForm.plainPassword, {'attr': {'class': 'form-control'}}) } }  
{ { form_row(registrationForm.confirmPassword, {'attr': {'class': 'form-control'}}) } }
```

- Ces champs viennent directement du FormType Symfony.
- confirmPassword est probablement ajouté manuellement dans ce form.

Bouton et lien :

- Bouton principal : Créer un compte.
- Lien en bas vers la route app_login : Se connecter.

Footer :

```
<footer class="bg-dark text-white p-3 text-center mt-4">  
    &copy; 2024 JOIE-EPICE. Tous droits réservés.  
</footer>
```

- Pied de page simple, centré : "© 2024 JOIE-EPICE. Tous droits réservés.

Page de confirmation d'email

❖ Structure du contenu :

```
<h1>Bonjour {{ username }}, et bienvenue sur notre site.</h1>  
• Accueil personnalisé de l'utilisateur.  
<a href="{{ signedUrl|raw }}>Confirmer mon Email</a>  
• Lien sécurisé de confirmation généré par Symfony (EmailVerifierInterface  
probablement).  
{ { expiresAtMessage } }  
• Affiche le message d'expiration du lien (par ex. "Ce lien expire dans 1 heure").  
<p>JOIE-EPICE!</p>  
• Signature de l'email.
```

Page de réinitialisation du mot de passe

❖ Structure globale :

- HTML complet avec Bootstrap, FontAwesome, style personnalisé intégré directement dans <style>.

☰ Détail du contenu :

Header :

- Logo JOIE-EPICE à gauche (avec route home).

- Icônes à droite : historique des commandes, panier, utilisateur.
- Affichage conditionnel du dashboard admin si ROLE_ADMIN.

Contenu principal (main) :

- Carte centrée (card) avec titre Confirmer la réinitialisation du mot de passe.

Bloc d'erreur :

```
{% if app.flashes('error') %}

<div class="alert alert-danger">{ { app.flashes('error')[0] } }</div>

{% endif %}
```

- Affiche un message d'erreur (ex : si les mots de passe ne correspondent pas).

Formulaire :

- POST, champ caché email.
- Deux champs password et confirm_password :

```
<input type="password" id="password" name="password" class="form-control" required>

<input type="password" id="confirm_password" name="confirm_password" class="form-control" required>
```

- Bouton : Confirmer.

Footer :

- Identique à celui des autres pages : mention légale simple.

Page : Mot de passe oublié

Structure générale :

- HTML complet avec Bootstrap 4, FontAwesome et styles CSS personnalisés directement intégrés dans la balise <style>.
- Typographie Bebas Neue pour une apparence unique.

Détails importants :

Header :

```
<header>

<div class="container">

  <div class="row align-items-center">

    <div class="col-md-6 logo">
      <a href="{{ path('home') }}">
        
      </a>
    </div>
```

```

</div>
<div class="col-md-6 text-right">
    <nav>
        <a href="{{ path('order_history') }}" class="text ml-3"><i class="fas fa-list-alt"></i></a>
        <a href="{{ path('cart') }}" class="text ml-3"><i class="fas fa-shopping-bag"></i></a>
        <a href="{{ path('login') }}" class="text ml-3"><i class="fas fa-user"></i></a>
        {% if is_granted('ROLE_ADMIN') %}
            <a href="{{ path('admin_dashboard') }}" class="text ml-3"><i class="fas fa-cogs"></i></a>
        {% endif %}
    </nav>
</div>
</div>
</div>
</header>

```

- Présente le logo de JOIE-EPICE cliquable, renvoyant vers home.
- Icônes de navigation en haut à droite :
 - Historique des commandes
 - Panier
 - Connexion
 - Dashboard admin (conditionné à ROLE_ADMIN)

Contenu principal (main) :

- **Formulaire dans une carte (card) centrée :**
 - Titre principal : Mot de passe oublié
 - Si un message d'erreur est présent (ex. e-mail non reconnu), il est affiché via :
- ```

{% if app.flashes('error') %}

<div class="alert alert-danger">
 {{ app.flashes('error')[0] }}
</div>

{% endif %}

```

## **Formulaire HTML (méthode POST) :**

- Champ unique : email, requis.
- Bouton : Envoyer (soumet le formulaire pour que le backend déclenche l'envoi d'un email de réinitialisation).

## **Footer :**

```
<footer class="bg-dark text-white p-3 text-center mt-4">
```

```
 © 2024 JOIE-EPICE. Tous droits réservés.
```

```
</footer>
```

- Bandeau sombre centré avec droits d'auteur : © 2024 JOIE-EPICE. Tous droits réservés.

### **Page : Dashboard Admin - Gestion des produits**

#### **✳ Structure globale :**

- HTML pur avec Bootstrap 4 pour la mise en page.
- Aucun template Symfony (pas de extends, block), mais intégré dans un environnement Twig.

#### **Contenu détaillé :**

##### **1. Formulaire d'ajout de produit :**

- Champ image : chemin de l'image (texte brut).
- Champ name : nom du produit.
- Champ price : prix du produit.
- Bouton : Ajouter — envoie un POST vers admin\_dashboard.

##### **2. Liste des produits existants (boucle for) :**

Pour chaque produit :

- Image affichée avec :

```
, <footer>, <section>) permet de mieux organiser le contenu pour l'indexation. De plus, les images sont optimisées pour le web afin de réduire le temps de chargement, et un sitemap XML est généré pour faciliter l'indexation par Google.

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.

```
PS C:\Users\kanth> cd joie-epice
PS C:\Users\kanth\joie-epice> php bin/phpunit tests\Controller\AdminControllerTest.php
PHPUnit 9.6.20 by Sebastian Bergmann and contributors.

Testing App\Tests\Controller\AdminControllerTest
....                                         4 / 4 (100%)

Time: 00:00.608, Memory: 30.00 MB

OK (4 tests, 12 assertions)
PS C:\Users\kanth\joie-epice>

PS C:\Users\kanth> cd joie-epice
PS C:\Users\kanth\joie-epice> php bin/phpunit tests\Controller\AuthControllerTest.php
PHPUnit 9.6.20 by Sebastian Bergmann and contributors.

Testing App\Tests\Controller\AuthControllerTest
...                                         3 / 3 (100%)

Time: 00:00.529, Memory: 36.00 MB

OK (3 tests, 8 assertions)
PS C:\Users\kanth\joie-epice>

PS C:\Users\kanth\joie-epice> php bin/phpunit tests\Controller\CartControllerTest.php
PHPUnit 9.6.20 by Sebastian Bergmann and contributors.

Testing App\Tests\Controller\CartControllerTest
..                                         2 / 2 (100%)

Time: 00:00.614, Memory: 30.00 MB

OK (2 tests, 6 assertions)
PS C:\Users\kanth\joie-epice>
```

```
PS C:\Users\kanth\joie-epice> php bin/phpunit tests\Controller/HomeControllerTest.php
PHPUnit 9.6.20 by Sebastian Bergmann and contributors.
```

```
Testing App\Tests\Controller\HomeControllerTest
```

```
1 / 1 (100%)
```

```
.
```

```
Time: 00:00.279, Memory: 26.00 MB
```

```
OK (1 test, 2 assertions)
```

```
PS C:\Users\kanth\joie-epice> |
```

```
PS C:\Users\kanth\joie-epice> php bin/phpunit tests\Controller\OrderControllerTest.php
PHPUnit 9.6.20 by Sebastian Bergmann and contributors.
```

```
Testing App\Tests\Controller\OrderControllerTest
```

```
...
```

```
3 / 3 (100%)
```

```
.
```

```
Time: 00:00.594, Memory: 28.00 MB
```

```
OK (3 tests, 9 assertions)
```

```
PS C:\Users\kanth\joie-epice>
```

```
PS C:\Users\kanth\joie-epice> php bin/phpunit tests\Controller\ProductControllerTest.php
PHPUnit 9.6.20 by Sebastian Bergmann and contributors.
```

```
Testing App\Tests\Controller\ProductControllerTest
```

```
..
```

```
2 / 2 (100%)
```

```
.
```

```
Time: 00:00.854, Memory: 34.00 MB
```

```
OK (2 tests, 8 assertions)
```

```
PS C:\Users\kanth\joie-epice>
```

```
PS C:\Users\kanth\joie-epice> php bin/phpunit tests\Controller/RegisterControllerTest.php
PHPUnit 9.6.20 by Sebastian Bergmann and contributors.
```

```
Cannot open file "tests\Controller\RegisterControllerTest.php".
```

```
PS C:\Users\kanth\joie-epice> php bin/phpunit tests\Controller\RegistrationControllerTest.php
PHPUnit 9.6.20 by Sebastian Bergmann and contributors.
```

```
Testing App\Tests\Controller\RegistrationControllerTest
```

```
..
```

```
2 / 2 (100%)
```

```
.
```

```
Time: 00:24.094, Memory: 46.00 MB
```

```
OK (2 tests, 8 assertions)
```

```
PS C:\Users\kanth\joie-epice> php bin/phpunit tests\Controller\SecurityControllerTest.php
PHPUnit 9.6.20 by Sebastian Bergmann and contributors.
```

```
Testing App\Tests\Controller\SecurityControllerTest
```

```
..
```

```
1 / 1 (100%)
```

```
.
```

```
Time: 00:00.371, Memory: 26.00 MB
```

```
OK (1 test, 5 assertions)
```

```
PS C:\Users\kanth\joie-epice>
```

```
PS C:\Users\kanth\joie-epice> php bin/phpunit tests\Controller\StripeControllerTest.php
PHPUnit 9.6.20 by Sebastian Bergmann and contributors.

Testing App\Tests\Controller\StripeControllerTest
..
Time: 00:01.961, Memory: 30.00 MB

OK (2 tests, 8 assertions)
PS C:\Users\kanth\joie-epice> |
```

Le site a été déployé en ligne à l'aide de la plateforme AlwaysData, qui permet d'héberger gratuitement des projets web.

🔗 Lien vers le site en ligne : <https://joie.alwaysdata.net>

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.
- **Accessibilité (RGAA) :**

Dans le cadre de la conception de l'interface, un soin particulier a été apporté à l'accessibilité. Les couleurs des boutons et des liens ont été choisies pour garantir un **contraste suffisant** selon les normes du **RGAA (Référentiel Général d'Accessibilité pour les Administrations)**. Cela assure que l'application soit utilisable par le plus grand nombre d'utilisateurs, y compris ceux ayant des déficiences visuelles. Les éléments interactifs comme les boutons sont clairement identifiables grâce à des contrastes marqués, et des tests ont été effectués pour vérifier leur lisibilité sur tous les supports (desktop, mobile).

J. *Gestion du code avec Git et GitHub :*

Le code du projet est hébergé sur **GitHub**, dans le dépôt suivant :<https://github.com/DKanthio/Joie-epice>.

Le projet utilise la branche **main** pour la version stable du code, et les différentes fonctionnalités sont développées directement sur cette branche.

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 | createCharge() | Connecté |

| | | | | |
|--|--|---|--|--|
| | | charge), crée la commande correspondante, vide le panier et redirige avec un message de confirmation. | | |
|--|--|---|--|--|

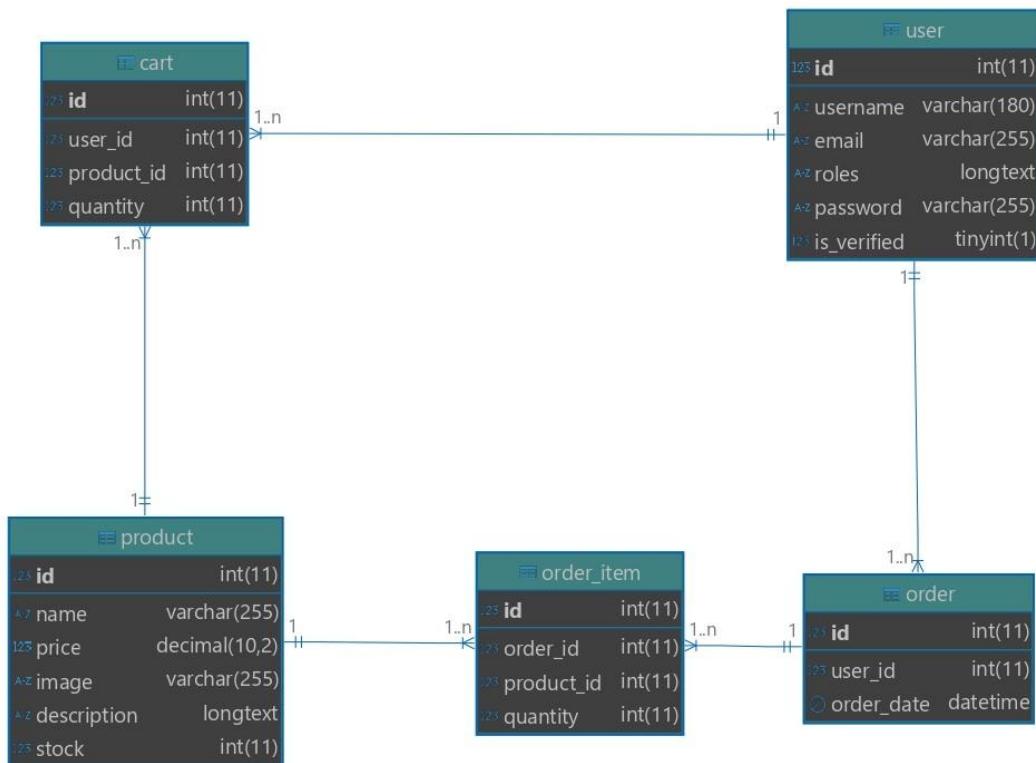
B. Front-end

| 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

Le schéma a été réalisé avec [DBeaver](#)



Pour plus de détails :

👤 user (utilisateur)

→ Un utilisateur peut avoir 0 ou plusieurs éléments dans le panier.

→ Un utilisateur peut passer 0 ou plusieurs commandes.

📦 product (produit)

→ Un produit peut être dans 0 ou plusieurs paniers.

→ Un produit peut apparaître dans 0 ou plusieurs lignes de commande.

cart (panier)

→ Chaque élément du panier appartient à un seul utilisateur.

→ Chaque élément du panier contient un seul produit.

order (commande)

→ Chaque commande appartient à un seul utilisateur.

→ Une commande contient au moins un article.

order_item (détail de commande)

→ Chaque ligne de commande référence un seul produit.

MLD

USER (

id : entier [PK]
username : chaîne [unique]
email : chaîne
roles : JSON
password : chaîne
is_verified : booléen

)

CART (

id : entier [PK, AI]
user_id : entier [FK → USER.id]
product_id : entier [FK → PRODUCT.id]

```
    quantity      : entier  
)  
  
PRODUCT (
```

```
    id          : entier [PK, AI]
```

```
    name        : chaîne
```

```
    price       : décimal
```

```
    image       : chaîne
```

```
    description : texte long
```

```
    stock       : entier
```

```
)
```

```
ORDER (
```

```
    id          : entier [PK, AI]
```

```
    user_id     : entier [FK → USER.id]
```

```
    order_date  : date-heure
```

```
)
```

```
ORDER_ITEM (
```

```
    id          : entier [PK, AI]
```

```
    order_id    : entier [FK → ORDER.id]
```

```
    product_id  : entier [FK → PRODUCT.id]
```

```
    quantity    : entier
```

```
)
```

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

Je tiens à exprimer ma profonde gratitude à toutes les personnes qui m'ont accompagnée tout au long de ce projet.

Un merci particulier à mon mentor Simon pour sa disponibilité, ses conseils avisés et son accompagnement tout au long de la formation.

Je remercie également l'équipe pédagogique du Centre Européen de Formation pour la qualité de l'enseignement, ainsi que le service client, toujours réactif et à l'écoute, qui a grandement facilité mon parcours.

Sur le plan technique, l'un des défis majeurs a été la gestion du stock en temps réel dans le panier. Chaque action (ajout, suppression ou modification de quantité) devait s'accompagner d'une mise à jour cohérente du stock en base de données. Cette logique, complexe à fiabiliser, m'a permis de développer une rigueur précieuse et une meilleure maîtrise de la gestion métier côté back-end.

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 |