🌿 MossAir

Purificateur d'air naturel révolutionnaire

Site E-Commerce Symfony

Projet de formation

RNCP Niveau 5 - BAC +2

Développeur Web et Web Mobile

CAHIER DES CHARGES TECHNIQUE

Pages 9-39 : Frontend & Backend

Année 2024-2025

Introduction

D'aussi loin que je me souvienne, j'ai toujours été curieux de ce qui vit, de ce souffle invisible qui traverse la nature. Grand lecteur de science-fiction et amateur de films comme Sunshine, Matrix ou Interstellar, je suis captivé par ce point de rencontre entre le vivant et la technologie. Pendant longtemps, je me suis retrouvé coincé dans le quotidien exigeant de la restauration, avec en tête ce rêve un peu flou : exercer un métier créatif, connecté, où je pourrais gérer mon temps librement — un monde que je croyais réservé aux as du code ou aux autodidactes précoces.

Le déclic est venu quand j'ai découvert les bienfaits de la respiration consciente, des étirements dynamiques et des mécanismes de la neurochimie positive. Petit à petit, j'ai laissé derrière moi la fatigue chronique et les rythmes hachés, pour construire un mode de vie plus sain, où bien-être, sport, alimentation ciblée et souplesse cognitive sont devenus mes nouveaux fondamentaux. C'est dans cet élan que je me suis formé au développement web, avec une idée claire en tête : créer des interfaces vivantes, qui permettent aux gens de se reconnecter à eux-mêmes.

C'est en mêlant cette façon d'envisager la vie, mon attrait pour le design inspiré de la nature et mes convictions écologiques qu'est né mon projet : un site dédié à **Moss Air**, un purificateur d'air à la fois innovant et organique, basé sur la mousse vivante. Ce produit, qui respire littéralement avec son environnement, représente pour moi le lien parfait entre ville et nature, une manière douce de réintroduire de l'intelligence naturelle dans nos espaces urbains.

Objectif du projet

L'objectif de cette plateforme est simple : proposer une expérience d'achat qui ait du sens, qui invite à ralentir, à respirer, et à repenser la technologie non pas comme un simple

> outil, mais comme une extension apaisante de notre corps et de notre quotidien.

Résumé du Projet

Le Produit : Moss Air

Moss Air est un purificateur d'air révolutionnaire utilisant la mousse vivante stabilisée pour filtrer naturellement l'air intérieur. Contrairement aux purificateurs électriques traditionnels, Moss Air fonctionne sans filtre à changer, sans bruit et sans consommation électrique.

La Gamme de Produits

Produit	Prix	Description
Moss Air 1	149,99€	Modèle d'entrée de gamme, idéal pour les petits espaces
Moss Air 2	179,99€	Version premium avec double filtration naturelle
Moss Air 3	199,99€	Haut de gamme avec technologie avancée de mousse stabilisée

Fonctionnalités du Site E-Commerce

- **Catalogue produits** : Présentation détaillée des 3 modèles Moss Air
- **Panier d'achat** : Gestion dynamique avec vérification du stock en temps réel

- **Authentification** : Inscription, connexion et gestion du profil utilisateur
- **Historique des commandes** : Suivi des achats pour les utilisateurs connectés
- **Dashboard administrateur** : Gestion des produits, stocks et utilisateurs
- **Design responsive** : Interface adaptée mobile, tablette et desktop

Utilisateurs du Système

Rôle	Utilisateur	Accès
Admin	arnaud	Dashboard, gestion produits/stocks/ utilisateurs
User	user	Catalogue, panier, commandes, profil

Technologies Utilisées

Stack Technique

Catégorie	Technologies	Version
Backend	PHP, Symfony	PHP 8.x, Symfony 6
ORM	Doctrine	Dernière version
Base de données	MySQL	8.x
Frontend	HTML5, CSS3, JavaScript	ES6+
Framework CSS	Bootstrap	5.x
Templating	Twig	3.x
Versioning	Git, GitHub	-

Pourquoi Symfony ?

- **Architecture MVC** : Séparation claire entre modèles, vues et contrôleurs
- **Doctrine ORM** : Gestion simplifiée de la base de données avec des entités PHP

- **Sécurité intégrée** : Protection CSRF, hachage des mots de passe, contrôle d'accès
- **Twig** : Moteur de templates sécurisé avec protection XSS automatique
- **Migrations** : Historique et versioning des modifications de base de données
- **Communauté active** : Documentation riche et écosystème de bundles

Base de données : projet-symf-1

```
Tables principales :
├── user          → Gestion des utilisateurs (admin, user)
├── produit       → Catalogue des produits Moss Air
└── commande      → Historique des commandes
```

Table des Matières

1. CSS et Media Queries : Design Responsive

01

Pour garantir que le site soit **responsive** et s'adapte à tous les types d'appareils, j'ai utilisé **CSS3** et les **media queries**. Les media queries permettent d'appliquer des styles spécifiques en fonction de la taille de l'écran.

Exemple : Header Responsive

Le header du site s'adapte automatiquement selon la taille de l'écran. Sur mobile, un menu hamburger apparaît pour une navigation optimale.

```
/* Header pour desktop */
.header-section {
    width: 100%;
    height: auto;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    padding: 15px 0;
    background: linear-gradient(135deg, #4a7c59, #6b8e7a);
}

/* Media query pour tablettes (largeur max 990px) */
@media (max-width: 990px) {
    .header-section {
        height: 80px;
        background: #4a7c59;
        padding: 10px;
        align-items: flex-start;
    }

    .list ul {
        flex-direction: column;
        height: 100vh;
```

```
        width: 250px;
        position: fixed;
        top: 0;
        right: -250px; /* Caché par défaut */
        background-color: #333;
        transition: all 0.4s linear;
        z-index: 999;
    }

    /* Quand le menu est actif (ouvert) */
    .list ul.active {
        right: 0; /* Menu visible */
    }
}
```

Explication

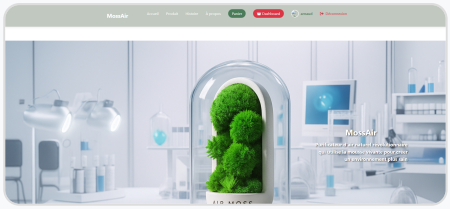
- Sur desktop, le header affiche tous les liens de navigation horizontalement
- Sur tablette/mobile (max-width: 990px), le menu devient un panneau latéral qui glisse depuis la droite
- La propriété `transition` rend l'animation fluide

Tests de Responsive

J'ai testé le site sur plusieurs tailles d'écran en utilisant les outils de développement du navigateur (F12 > Mode responsive) pour m'assurer qu'il fonctionne bien sur :

- **Desktop** : 1920x1080px
- **Tablette** : 768x1024px
- **Mobile** : 375x667px

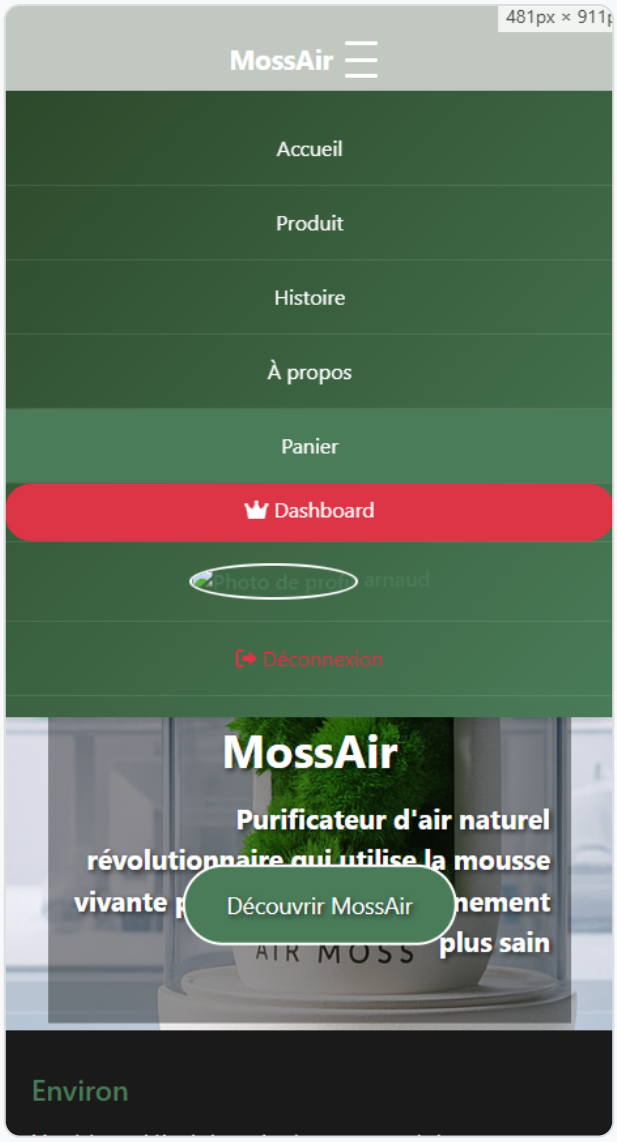
📸 Design Responsive : Desktop vs Mobile



Version Desktop

Version Mobile

📸 Menu Hamburger Mobile



Menu latéral déployé avec navigation complète

2. JavaScript et Interactivité 02

JavaScript est utilisé pour ajouter des fonctionnalités interactives et améliorer l'expérience utilisateur.

Exemple 1 : Menu Hamburger Responsive

Le menu hamburger permet d'ouvrir/fermer le menu de navigation sur mobile.

```
// Menu hamburger pour mobile/tablette
(function() {
    function initBurger() {
        const hamburger =
document.getElementById('hamburger');
        const navMenu = document.getElementById('navMenu');

        if (!hamburger || !navMenu) return;

        // Toggle menu au clic sur le bouton hamburger
        hamburger.addEventListener('click', function(e) {
            e.preventDefault();
            hamburger.classList.toggle('active');
            navMenu.classList.toggle('active');
            hamburger.setAttribute('aria-expanded',
                navMenu.classList.contains('active'));
        });

        // Fermer le menu au clic sur un lien
        navMenu.querySelectorAll('a').forEach(link => {
            link.addEventListener('click', () => {
                hamburger.classList.remove('active');
                navMenu.classList.remove('active');
            });
        });
    }

    // Initialiser au chargement du DOM
    if (document.readyState === 'loading') {
```

```
        document.addEventListener('DOMContentLoaded',
initBurger);
    } else {
        initBurger();
    }
})();
```

Explication

- **IIFE** (fonction auto-exécutée) pour isoler le code
- `toggle('active')` ajoute/retire la classe CSS qui montre/cache le menu
- Le menu se ferme automatiquement après avoir cliqué sur un lien

Exemple 2 : Accès Admin Secret

Un système d'accès admin caché via 3 clics sur le logo ou Ctrl+A.

```
// Accès admin (Ctrl+A ou 3 clics sur logo)
document.addEventListener('DOMContentLoaded', function() {
    let clickCount = 0;
    let clickTimer = null;

    // Raccourci clavier Ctrl+A
    document.addEventListener('keydown', function(e) {
        if (e.ctrlKey && e.key === 'a') {
            e.preventDefault();
            window.location.href = '/admin';
        }
    });

    // 3 clics sur le logo
    const siteLogo = document.getElementById('siteLogo');
    if (siteLogo) {
        siteLogo.addEventListener('click', function(e) {
            e.preventDefault();
            clickCount++;
```

```
            if (clickTimer) clearTimeout(clickTimer);

            // Si 3 clics, ouvrir modale admin
            if (clickCount >= 3) {
                const modal = document.getElementById('adminM
odal');
                if (modal && typeof bootstrap !==
'undefined') {
                    new bootstrap.Modal(modal).show();
                }
                clickCount = 0;
                return;
            }

            // Timer: navigation normale après 600ms
            clickTimer = setTimeout(() => {
                window.location.href = siteLogo.href;
                clickCount = 0;
            }, 600);
        });
    }
});
```

3. PHP et Gestion du Site avec Symfony

03

Le projet utilise le framework **Symfony** pour gérer les interactions côté serveur, traiter les données et garantir la sécurité.

Architecture Symfony

```
src/
├── Controller/
│   ├── PanierController.php      # Gestion du panier
│   ├── AdminController.php       # Dashboard admin
│   ├── ProduitController.php     # Liste des produits
│   └── SecurityController.php    # Authentification
├── Entity/
│   ├── Produit.php               # Entité Produit (avec stock)
│   ├── User.php                  # Entité Utilisateur
│   └── Commande.php              # Entité Commande
├── Repository/
│   ├── ProduitRepository.php
│   └── UserRepository.php
migrations/
└── Version20251203150000.php     # Migration ajout stock
```

4. Entity Produit avec Gestion du Stock

04

L'entité `Produit` représente un produit en base de données avec Doctrine ORM.

Code de l'Entity Produit

```php
<?php

namespace App\Entity;

use App\Repository\ProduitRepository;
use Doctrine\ORM\Mapping as ORM;

#[ORM\Entity(repositoryClass: ProduitRepository::class)]
class Produit
{
    // Identifiant unique auto-incrémenté
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    // Nom du produit (obligatoire, max 255 caractères)
    #[ORM\Column(length: 255)]
    private ?string $nom = null;

    // Prix du produit (nombre décimal)
    #[ORM\Column]
    private ?float $prix = null;

    // === CHAMP STOCK ===
    // Quantité disponible en stock
    #[ORM\Column]
    private ?int $stock = 0;

    // === MÉTHODES DE GESTION DU STOCK ===
```

```php
    /**
     * Récupérer le stock disponible
     */
    public function getStock(): ?int
    {
        return $this->stock;
    }

    /**
     * Définir le stock disponible
     */
    public function setStock(int $stock): static
    {
        $this->stock = $stock;
        return $this;
    }

    /**
     * Vérifier s'il reste du stock
     */
    public function hasStock(): bool
    {
        return $this->stock > 0;
    }

    /**
     * Décrémenter le stock (enlever une quantité)
     * La fonction max(0, ...) empêche le stock de devenir négatif
     */
    public function decrementStock(int $quantity): static
    {
        $this->stock = max(0, $this->stock - $quantity);
        return $this;
    }
}
```

Points clés

- Le champ `stock` est de type `INT` avec une valeur par défaut à 0
- La méthode `hasStock()` vérifie rapidement si le produit est disponible

- La méthode `decrementStock()` diminue le stock de façon sécurisée (jamais négatif)

5. Migration : Ajout de la Colonne Stock

05

Pour ajouter la colonne `stock` à la table `produit`, j'ai créé une migration Doctrine.

Code de la Migration

```php
<?php

declare(strict_types=1);

namespace DoctrineMigrations;

use Doctrine\DBAL\Schema\Schema;
use Doctrine\Migrations\AbstractMigration;

/**
 * Migration pour ajouter la colonne 'stock' à la table
produit
 *
 * PRINCIPE :
 * - up() = Ajoute la colonne stock
 * - down() = Supprime la colonne stock (pour annuler)
 */
final class Version20251203150000 extends AbstractMigration
{
    public function getDescription(): string
    {
        return 'Ajouter la colonne stock à la table produit';
    }

    // Méthode exécutée pour appliquer la migration
    public function up(Schema $schema): void
    {
        // Ajouter la colonne 'stock' de type INT
        // NOT NULL = obligatoire
        // DEFAULT 0 = valeur par défaut
        $this->addSql('ALTER TABLE produit ADD COLUMN stock
```

```
INT NOT NULL DEFAULT 0');
    }

    // Méthode pour annuler la migration
    public function down(Schema $schema): void
    {
        $this->addSql('ALTER TABLE produit DROP COLUMN
stock');
    }
}
```

Commandes pour exécuter la migration

```
# Créer une nouvelle migration
php bin/console make:migration

# Appliquer la migration
php bin/console doctrine:migrations:migrate

# Vérifier que la colonne a été ajoutée
php bin/console doctrine:schema:validate
```

Résultat en base de données

```
-- Table produit après migration
CREATE TABLE produit (
    id INT AUTO_INCREMENT NOT NULL,
    nom VARCHAR(255) NOT NULL,
    description LONGTEXT DEFAULT NULL,
    prix DOUBLE PRECISION NOT NULL,
    image VARCHAR(255) DEFAULT NULL,
    actif TINYINT(1) NOT NULL,
    stock INT NOT NULL DEFAULT 0,  -- NOUVELLE COLONNE
    created_at DATETIME NOT NULL,
    updated_at DATETIME NOT NULL,
    PRIMARY KEY(id)
);
```

6. PanierController : Gestion Complète du Panier

06

Le `PanierController` gère toutes les opérations liées au panier : ajout, suppression, modification de quantité et validation de commande.

Structure du Panier en Session

Le panier est stocké dans la **session PHP** :

```
// Structure d'un article dans le panier
$panier = [
    [
        'productId' => 1,    // ID du produit
        'quantity' => 2      // Quantité demandée
    ],
    [
        'productId' => 5,
        'quantity' => 1
    ]
];
```

Méthode : Afficher le Panier

```
/**
 * Affiche la page panier avec tous les produits
 */
#[Route('/panier', name: 'app_panier')]
public function index(SessionInterface $session, ProduitRepos
itory $produitRepo): Response
{
    // Récupérer le panier depuis la session
    $panier = $session->get('panier', []);

    $panierComplet = [];
```

```
    $total = 0;

    // Pour chaque article du panier
    foreach ($panier as $item) {
        $produit = $produitRepo->find($item['productId']);

        if ($produit) {
            $sousTotal = $produit->getPrix() * $item['quantit
y'];

            $panierComplet[] = [
                'productId' => $produit->getId(),
                'name' => $produit->getNom(),
                'price' => $produit->getPrix(),
                'quantity' => $item['quantity'],
                'total' => $sousTotal,
                'stock' => $produit->getStock() // Stock
disponible
            ];

            $total += $sousTotal;
        }
    }

    return $this->render('panier/index.html.twig', [
        'panier' => $panierComplet,
        'total' => $total
    ]);
}
```

Méthode : Ajouter un Produit au Panier

```
/**
 * Ajoute un produit au panier avec vérification du stock
 */
#[Route('/panier/ajouter', name: 'app_panier_ajouter',
methods: ['POST'])]
public function ajouter(
    Request $request,
    SessionInterface $session,
    ProduitRepository $produitRepo
): Response {
    $productId = (int) $request->request->get('product_id');
    $quantite = (int) $request->request->get('quantite', 1);
```

```
    $produit = $produitRepo->find($productId);

    if (!$produit) {
        $this->addFlash('error', 'Produit introuvable !');
        return $this->redirectToRoute('app_panier');
    }

    $panier = $session->get('panier', []);

    // Chercher si le produit est déjà dans le panier
    foreach ($panier as $key => $item) {
        if ($item['productId'] === $productId) {
            $nouvelleQuantite = $item['quantity'] +
$quantite;

            // === VÉRIFICATION DU STOCK ===
            if ($nouvelleQuantite > $produit->getStock()) {
                $this->addFlash('error',
                    "Stock insuffisant ! Seulement {$produit-
>getStock()} disponible(s).");
                return $this->redirectToRoute('app_panier');
            }

            $panier[$key]['quantity'] = $nouvelleQuantite;
            $session->set('panier', $panier);
            return $this->redirectToRoute('app_produit');
        }
    }

    // Nouveau produit - vérifier le stock
    if ($quantite > $produit->getStock()) {
        $this->addFlash('error', "Stock insuffisant !");
        return $this->redirectToRoute('app_produit');
    }

    $panier[] = ['productId' => $productId, 'quantity' => $qu
antite];
    $session->set('panier', $panier);

    $this->addFlash('success', "{$produit->getNom()} ajouté
au panier !");
    return $this->redirectToRoute('app_produit');
}
```

Points clés

- Vérification du stock **avant** d'ajouter au panier
- Si le produit est déjà dans le panier, on met à jour la quantité
- Message d'erreur explicite si stock insuffisant

6. PanierController : Validation et Stock (suite)

06

Méthode : Valider le Paiement et Décrémenter le Stock

```
/**
 * Valide le paiement, enregistre la commande et décrémente
le stock
 */
#[Route('/panier/paiement-effectue', name: 'app_panier_paieme
nt_effectue', methods: ['POST'])]
public function paiementEffectue(
    SessionInterface $session,
    EntityManagerInterface $em,
    ProduitRepository $produitRepo
): Response {
    $panier = $session->get('panier', []);
    $user = $session->get('user');

    if (empty($panier)) {
        $this->addFlash('error', 'Votre panier est vide !');
        return $this->redirectToRoute('app_panier');
    }

    try {
        foreach ($panier as $item) {
            $produit = $produitRepo->find($item['productId'])
;

            if (!$produit) continue;

            // === VÉRIFICATION FINALE DU STOCK ===
            if ($item['quantity'] > $produit->getStock()) {
                $this->addFlash('error', "Stock insuffisant
pour {$produit->getNom()}");
                return $this->redirectToRoute('app_panier');
            }

            // Créer une nouvelle commande
```

```
            $commande = new Commande();
            $commande->setNomClient($user ? $user['nom'] : 'C
lient anonyme');
            $commande->setProduit($produit->getNom());
            $commande->setQuantite($item['quantity']);
            $commande->setPrix($produit->getPrix());
            $commande->setImage($produit->getImage());
            $commande->setDateCommande(new \DateTime());

            // === DÉCRÉMENTATION DU STOCK ===
            $produit->decrementStock($item['quantity']);
            $produit->setUpdatedAt(new \DateTimeImmutable());

            $em->persist($commande);
            $em->persist($produit);
        }

        $em->flush();
        $session->remove('panier');

        $this->addFlash('success', 'Paiement effectué et
commande enregistrée !');
    } catch (\Exception $e) {
        $this->addFlash('error', 'Erreur : ' . $e-
>getMessage());
    }

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

Points clés

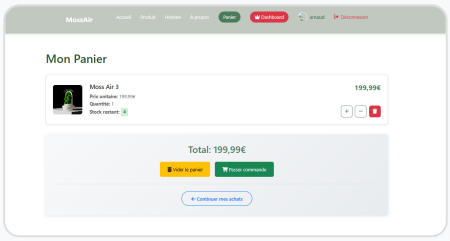
- Double vérification du stock (avant ajout panier + avant validation)
- Utilisation de `decrementStock()` pour diminuer le stock de façon sécurisée
- Transaction atomique : tout est enregistré ensemble avec `flush()`
- En cas d'erreur, rien n'est modifié (rollback automatique)

7. Screenshots : Panier et Dashboard Admin

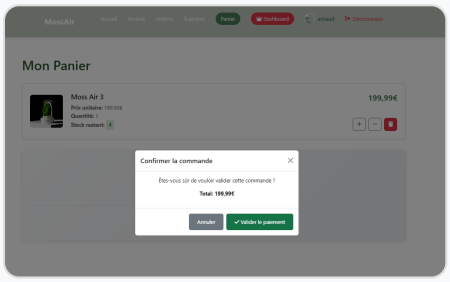
07

Gestion du Panier

📸 Panier avec Produit et Confirmation



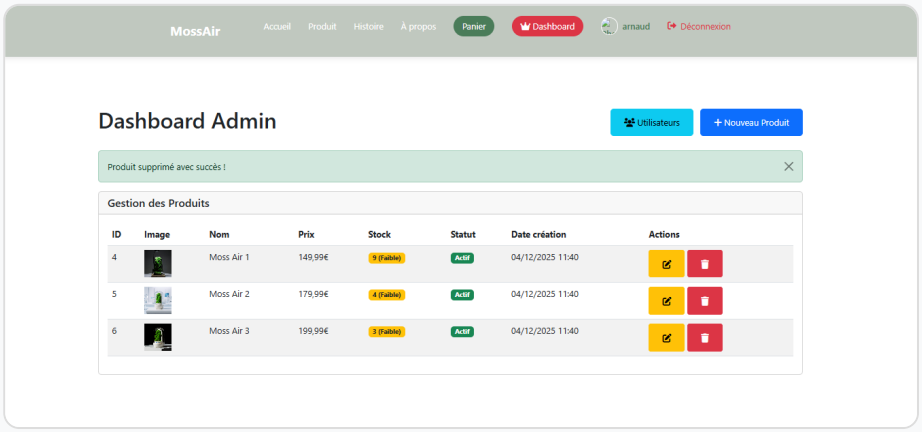
Panier avec Moss Air 3



Modal de confirmation

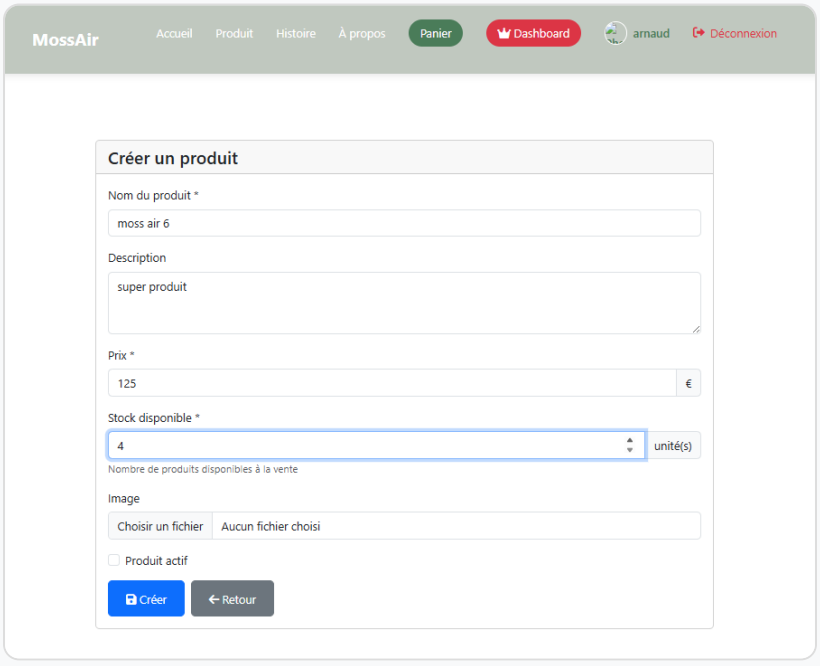
Dashboard Administrateur

📸 Gestion des Produits et Stocks



Interface d'administration avec badges de stock colorés

📸 Création de Produit



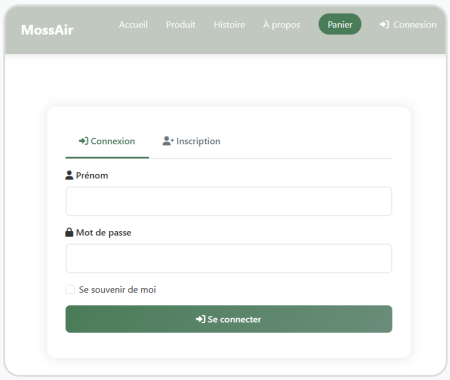
Formulaire de création avec gestion du stock

8. Screenshots : Authentification et Produits

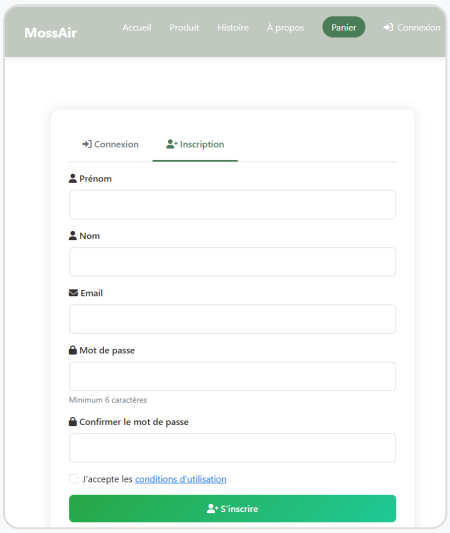
08

Authentification Utilisateur

📸 Connexion et Inscription



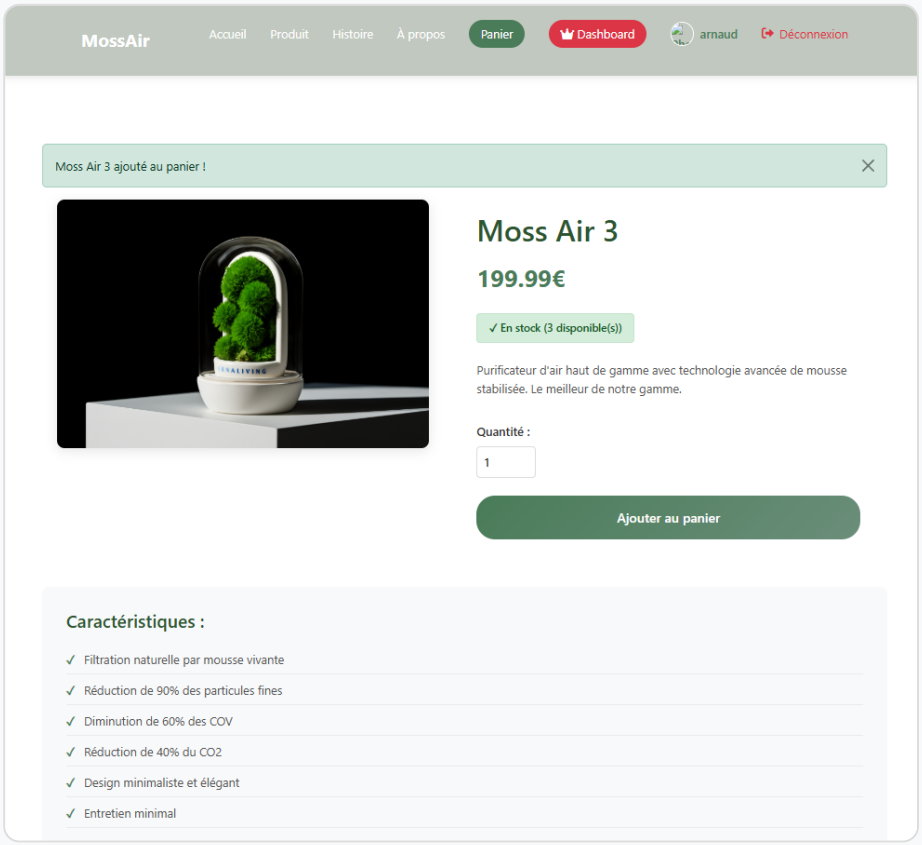
Formulaire de connexion



Formulaire d'inscription

Catalogue Produits

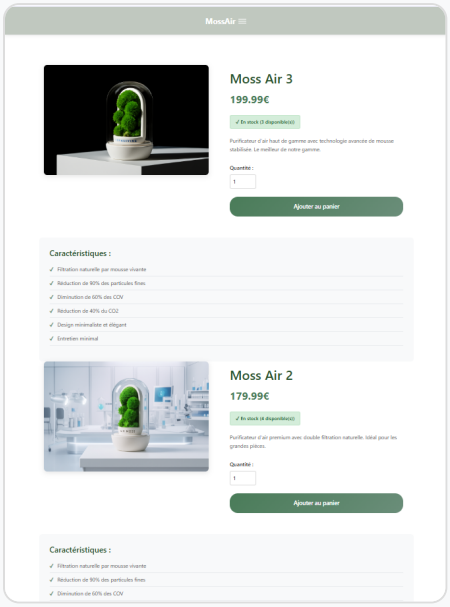
📸 Fiche Produit Détaillée



Page produit avec caractéristiques et gestion du stock

📸 Liste des Produits - Vue Mobile/Tablet

Vue Mobile



Vue Tablet

9. Base de Données : Structure et Relations

09

Modèle Conceptuel de Données (MCD)

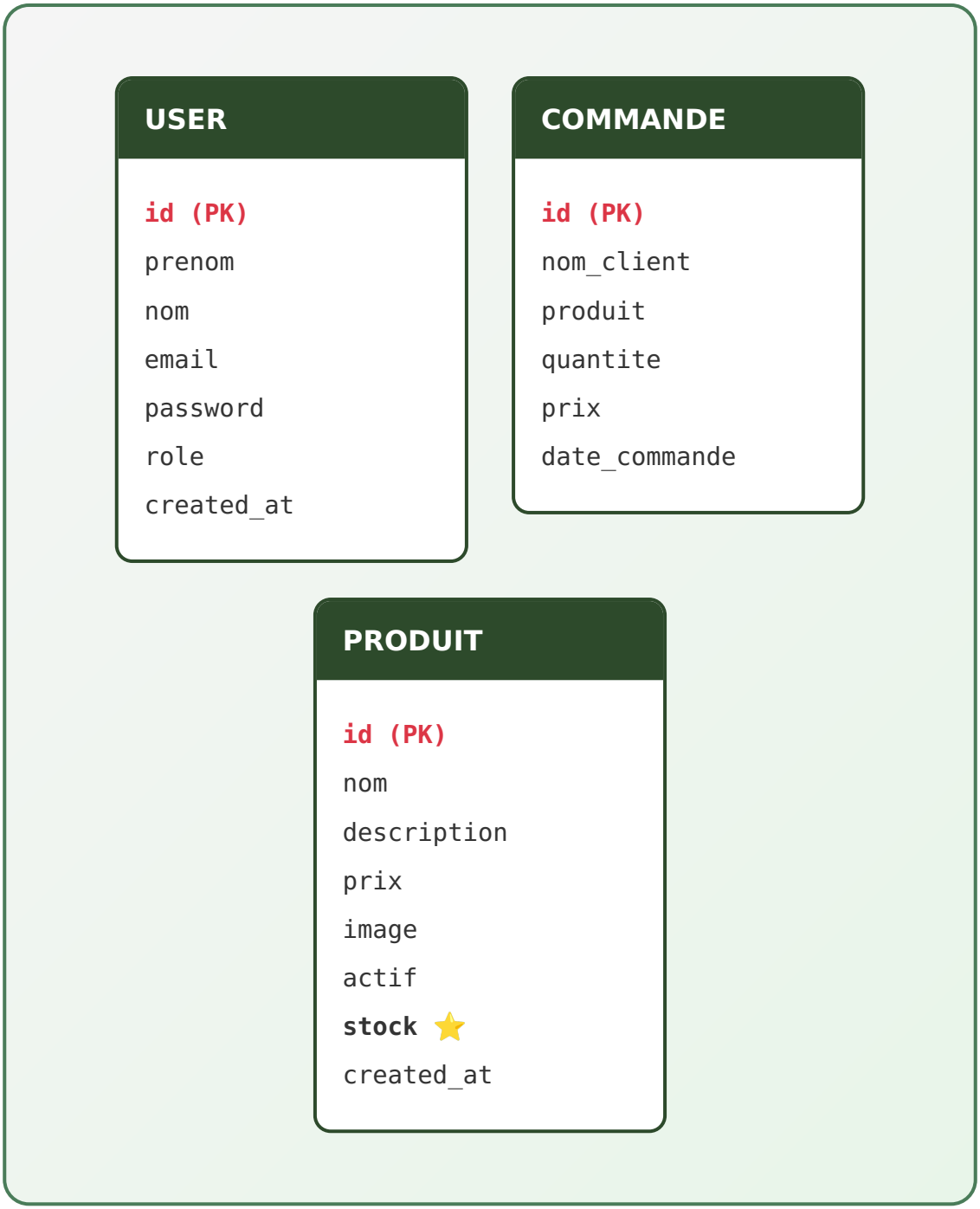


Table : produit

#	Nom	Type	Null	Défaut
1	**id**	int	Non	AUTO_INCREMENT

2	**nom**	varchar(255)	Non	-
3	**description**	longtext	Oui	NULL
4	**prix**	double	Non	-
5	**image**	varchar(255)	Oui	NULL
6	**actif**	tinyint(1)	Non	1
7	**stock** ⭐	int	Non	**0**
8	**created_at**	datetime	Non	-

Screenshots phpMyAdmin

📸 Structure des Tables

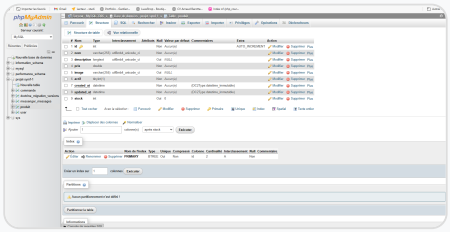


Table Produit

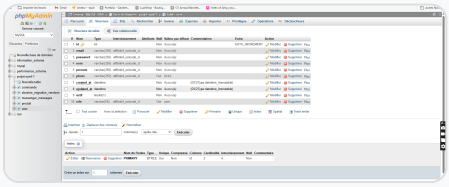
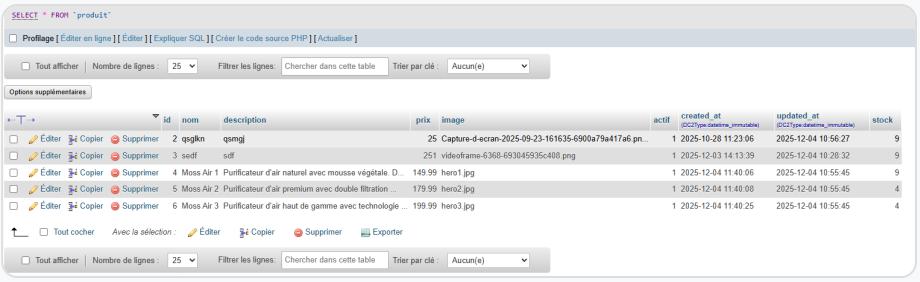


Table User

📸 Données en Base



```
SELECT * FROM `produit`
```

Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

Tout afficher | Nombre de lignes : 25 | Filtrer les lignes: Chercher dans cette table | Trier par clé : Aucun(e)

Options supplémentaires

	id	nom	description	prix	image	actif	created_at (DC2Type:datetime_immutable)	updated_at (DC2Type:datetime_immutable)	stock
Éditer Copier Supprimer	2	qsglkn	qsmgj	25	Capture-d-ecran-2025-09-23-161635-6900a79a417a6.pn...	1	2025-10-28 11:23:06	2025-12-04 10:56:27	9
Éditer Copier Supprimer	3	sedf	sdf	251	videoframe-6368-693045935c408.png	1	2025-12-03 14:13:39	2025-12-04 10:28:32	9
Éditer Copier Supprimer	4	Moss Air 1	Purificateur d'air naturel avec mousse végétale. D...	149.99	hero1.jpg	1	2025-12-04 11:40:06	2025-12-04 10:55:45	9
Éditer Copier Supprimer	5	Moss Air 2	Purificateur d'air premium avec double filtration ...	179.99	hero2.jpg	1	2025-12-04 11:40:08	2025-12-04 10:55:45	4
Éditer Copier Supprimer	6	Moss Air 3	Purificateur d'air haut de gamme avec technologie ...	199.99	hero3.jpg	1	2025-12-04 11:40:25	2025-12-04 10:55:45	4

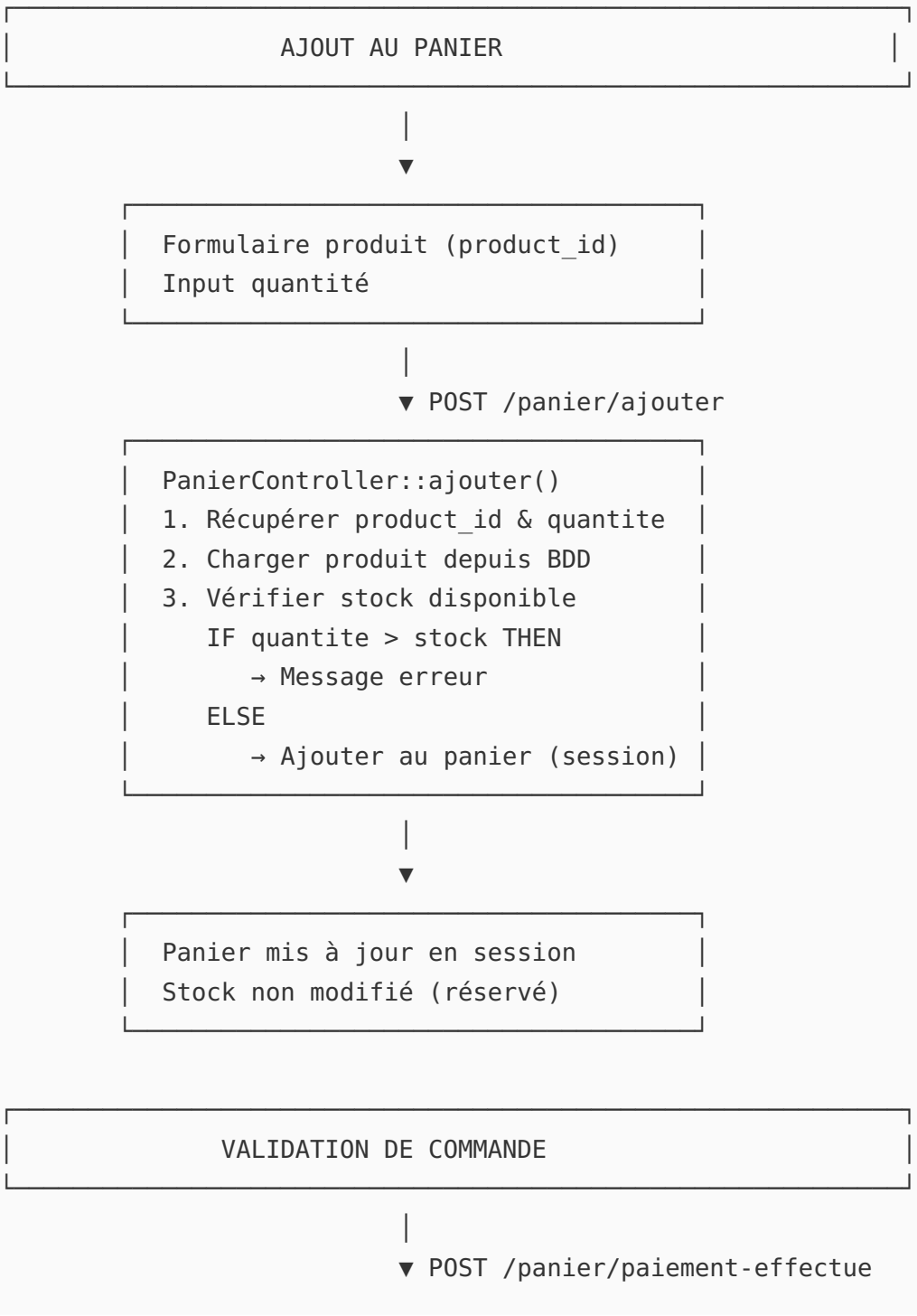
Tout cocher Avec la sélection : Éditer Copier Supprimer Exporter

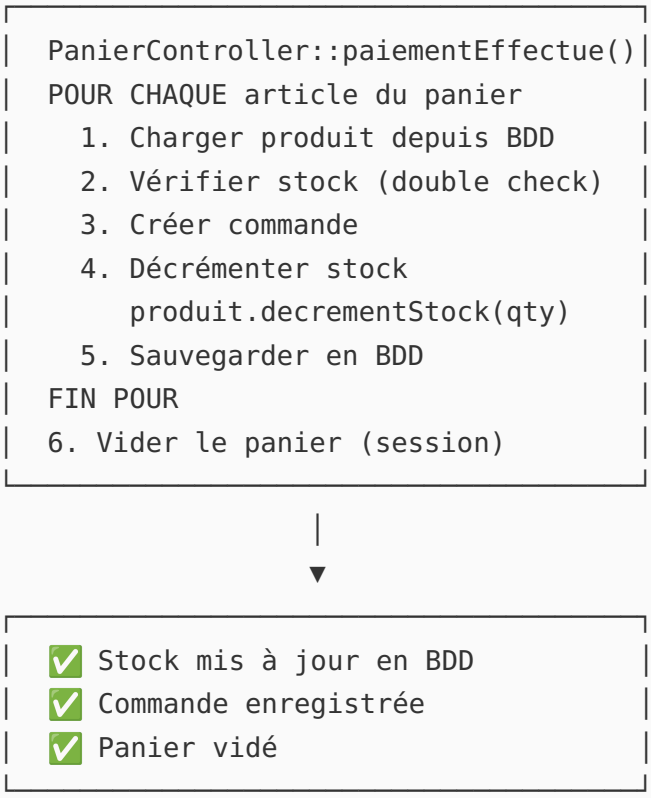
Tout afficher | Nombre de lignes : 25 | Filtrer les lignes: Chercher dans cette table | Trier par clé : Aucun(e)

Produits Moss Air 1, 2, 3 avec leurs stocks

10. Flux de Gestion du Stock 10

Diagramme du Flux





Exemple Concret

Scénario : Un client commande 3 "Moss Air"

1. **État initial** : Produit "Moss Air" a 10 en stock
2. **Ajout au panier** :
   - Client clique "Ajouter au panier" avec quantité = 3
   - Vérification : `3 <= 10` ✅ OK
   - Stock BDD reste à **10** (non modifié)
3. **Validation commande** :
   - Client clique "Valider le paiement"
   - Double vérification : `3 <= 10` ✅ OK
   - **Décrémentation** : `10 - 3 = 7`
   - Stock BDD mis à jour à **7**

Si un autre client tente de commander 8 unités

- Vérification : `8 > 7` ❌ ERREUR
- Message : "Stock insuffisant ! Seulement 7 disponible(s)."
- Commande bloquée

11. Sécurité : Points Clés 11

1. Hachage des Mots de Passe

```
// Lors de l'inscription
$hashedPassword = password_hash($plainPassword,
PASSWORD_DEFAULT);

// Lors de la connexion
if (password_verify($plainPassword, $hashedPassword)) {
    // Mot de passe correct
}
```

Algorithme utilisé : bcrypt (via `PASSWORD_DEFAULT`)

2. Validation des Entrées Utilisateur

```
// Conversion sécurisée en entier
$productId = (int) $request->request->get('product_id');

// Vérification d'existence
if (!$produit) {
    throw $this->createNotFoundException('Produit
introuvable');
}
```

3. Protection CSRF

```
{# Token CSRF dans les formulaires de suppression #}
<input type="hidden" name="_token"
value="{{ csrf_token('delete' ~ produit.id) }}">

// Vérification côté serveur
if ($this->isCsrfTokenValid('delete' . $produit->getId(), $to
```

```
ken)) {
    // Suppression autorisée
}
```

4. Contrôle d'Accès Admin

```
private function checkAdmin(SessionInterface $session): bool
{
    $user = $session->get('user');
    return $user && isset($user['role']) && $user['role']
=== 'admin';
}

// Dans chaque méthode admin
if (!$this->checkAdmin($session)) {
    $this->addFlash('error', 'Accès refusé');
    return $this->redirectToRoute('app_home');
}
```

5. Prévention des Stocks Négatifs

```
// Dans Produit::decrementStock()
public function decrementStock(int $quantity): static
{
    // max(0, ...) garantit que le stock ne peut pas être
négatif
    $this->stock = max(0, $this->stock - $quantity);
    return $this;
}
```

12-13. Utilisation de Symfony & Structure

12

Pourquoi Symfony ?

1. **Doctrine ORM** : Gestion simplifiée de la base de données
    - Entities = objets PHP ↔ tables SQL
    - Pas besoin d'écrire du SQL brut
    - Migrations automatiques
2. **Routing** : URLs propres et sémantiques
3. **Twig** : Moteur de templates sécurisé avec protection XSS automatique
4. **Sessions** : Gestion simplifiée avec `$session->get()` et `$session->set()`

Structure du Projet Symfony

```
src/
├── Controller/                    # Contrôleurs (logique des routes)
│   ├── AdminController.php        # Gestion admin (CRUD produits,
stocks)
│   ├── AuthController.php         # Authentification (login, register)
│   ├── CommandeController.php     # Gestion des commandes
│   ├── HistoireController.php     # Page "Histoire"
│   ├── PageController.php         # Pages statiques
│   ├── PanierController.php       # Gestion du panier et stock
│   ├── ProduitController.php      # Affichage des produits
│   ├── ProfileController.php      # Profil utilisateur
│   └── SitemapController.php      # Sitemap XML
│
├── Entity/                        # Entités Doctrine (modèles de
données)
│   ├── Commande.php              # Entité Commande (historique)
│   ├── Produit.php               # Entité Produit avec champ stock
│   └── User.php                  # Entité User (avec rôles)
```

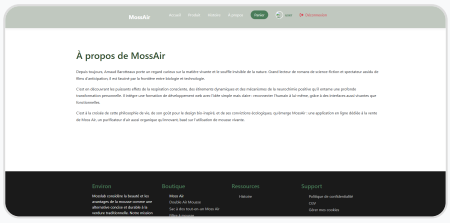
```
│
├── Repository/                    # Repositories (requêtes BDD)
│   ├── CommandeRepository.php
│   ├── ProduitRepository.php
│   └── UserRepository.php
│
└── Service/                       # Services (logique métier)
    ├── FirebaseAuthService.php
    └── FirebaseStorageService.php

migrations/
├── Version20251028105658.php      # Migration initiale
├── Version20251121000000.php      # Modifications intermédiaires
├── Version20251128090112.php      # Ajustements structure
└── Version20251203150000.php      # ⭐ Migration ajout colonne STOCK
```

Avantages des migrations Doctrine

- ✅ Historique complet des modifications de la BDD
- ✅ Versioning : chaque migration a un numéro unique
- ✅ Réversibilité : possibilité de revenir en arrière
- ✅ Travail en équipe : les migrations se partagent via Git
- ✅ Automatisation : pas besoin d'écrire le SQL manuellement

14. Pages Additionnelles 14

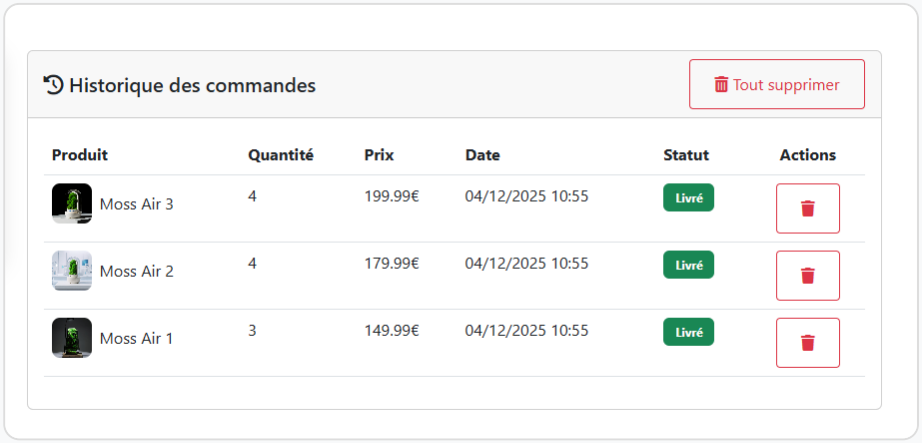
Page À Propos et Footer



Page À propos

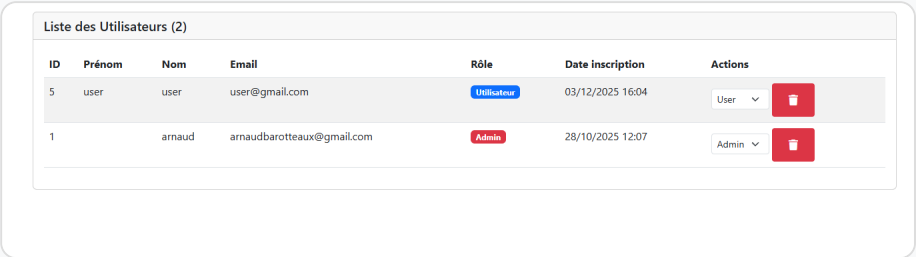
Footer du site

Historique des Commandes



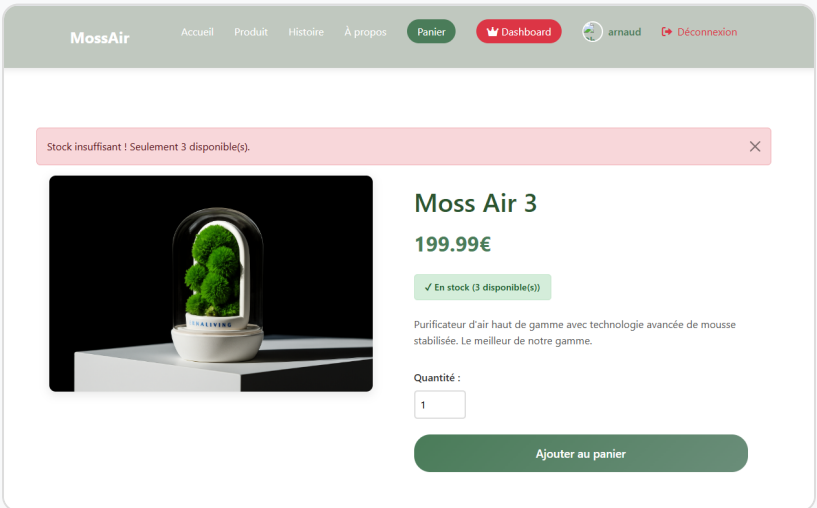
Liste des commandes passées par l'utilisateur

Gestion des Utilisateurs (Admin)



Interface d'administration des utilisateurs

Alerte Stock Insuffisant



Message d'alerte lorsque le stock est insuffisant

Conclusion ✅

Ce cahier des charges technique démontre la mise en place d'un **système e-commerce complet** avec :

Fonctionnalités Implémentées

- ✅ **Gestion dynamique du stock** : Vérification avant ajout, décrémentation après commande
- ✅ **Interface responsive** : CSS media queries pour mobile/tablette/desktop
- ✅ **Interactivité JavaScript** : Menu hamburger, accès admin, animations
- ✅ **Architecture Symfony** : MVC, Entities, Controllers, Templates
- ✅ **Sécurité renforcée** : Hachage bcrypt, CSRF, validation, contrôle d'accès
- ✅ **Base de données relationnelle** : MySQL avec Doctrine ORM
- ✅ **Dashboard admin** : Gestion complète des produits et du stock

Technologies Utilisées

Catégorie	Technologies
Frontend	HTML5, CSS3, JavaScript, Bootstrap 5

Backend	PHP 8, Symfony 6, Doctrine ORM
Base de données	MySQL, phpMyAdmin
Templating	Twig
Versionning	Git, GitHub

🌿 MossAir

Purifier l'air, naturellement.

Projet réalisé dans le cadre de la formation
Développeur Web et Web Mobile - RNCP Niveau 5

Document généré le 05/12/2025
Développé avec Symfony 6