



# 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

# Table des Matières



1. CSS et Media Queries : Design Responsive

2. JavaScript et Interactivité

3. PHP et Gestion du Site avec Symfony

4. Entity Produit avec Gestion du Stock

5. Migration : Ajout de la Colonne Stock

6. PanierController : Gestion du Panier

7. Screenshots : Panier et Dashboard Admin

8. Screenshots : Authentification et Produits

9. Base de Données : Structure et Relations

10. Flux de Gestion du Stock

11. Sécurité : Points Clés

12-13. Utilisation de Symfony et Structure

14. Pages Additionnelles

Conclusion

# 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

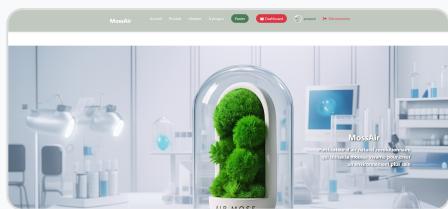
## Tests de Responsive

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



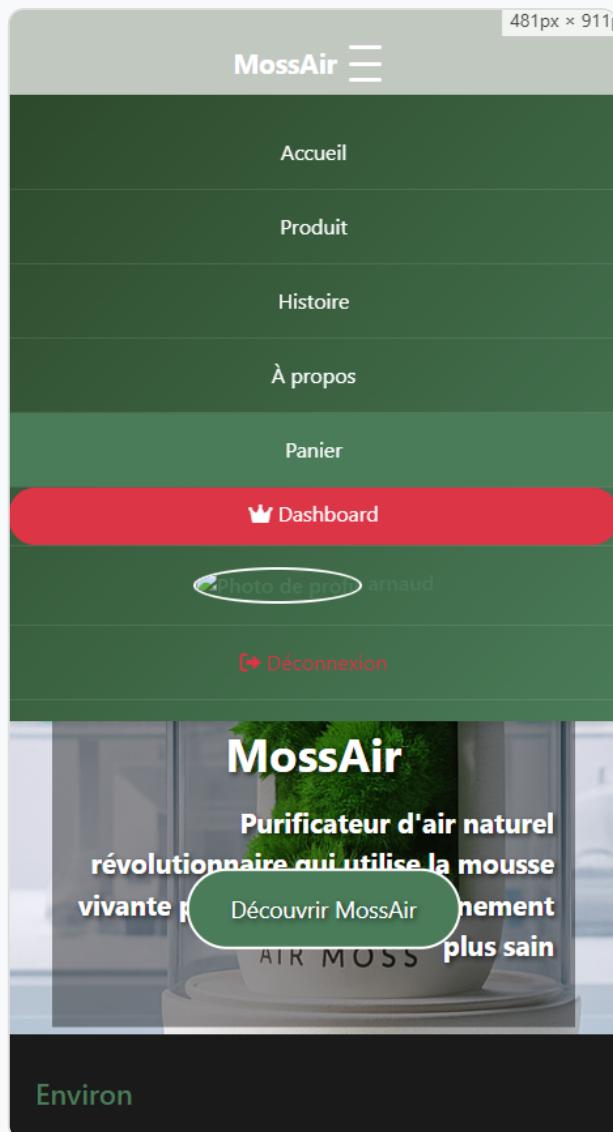
Version Desktop



Version Mobile



## Menu Hamburger Mobile



Menu latéral déployé avec navigation complète

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('adminModal');
    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

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 ===
}
```

```

/**
 * 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

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, ProduitRepository $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

Mon Panier

Moss Air 3  
Prix unitaire: 199,99€  
Quantité: 1  
Stock restant: 1

Total: 199,99€

Ajouter au panier Valider le paiement

Panier avec Moss Air 3

Confirmer la commande

Êtes-vous sûr de vouloir valider cette commande ?

Total: 199,99€

Annuler Valider le paiement

Modal de confirmation

# Dashboard Administrateur

## Gestion des Produits et Stocks

The screenshot shows the 'Dashboard Admin' page. At the top, there's a green notification bar stating 'Produit supprimé avec succès !'. Below it is a table titled 'Gestion des Produits' with columns: ID, Image, Nom, Prix, Stock, Statut, Date création, and Actions. The table contains three rows of data:

ID	Image	Nom	Prix	Stock	Statut	Date création	Actions
4		Moss Air 1	149,99€	9 (Fiable)	Actif	04/12/2025 11:40	
5		Moss Air 2	179,99€	4 (Fiable)	Actif	04/12/2025 11:40	
6		Moss Air 3	199,99€	3 (Fiable)	Actif	04/12/2025 11:40	

Interface d'administration avec badges de stock colorés

## Création de Produit

The screenshot shows the 'Créer un produit' (Create a product) form. It includes fields for Nom du produit (Name), Description, Prix (Price), Stock disponible (Available stock), Image (Image), and a checkbox for Produit actif (Active product). The 'Stock disponible' field has a dropdown menu showing the value '4'.

Créer un produit

Nom du produit \*

Description

Prix \*

 €

Stock disponible \*

4 unité(s)

Nombre de produits disponibles à la vente

Image

Choisir un fichier Aucun fichier choisi

Produit actif

Formulaire de création avec gestion du stock

# 8. Screenshots : Authentification et Produits

08

## Authentification Utilisateur

### Connexion et Inscription

The screenshot shows the MossAir login page. At the top, there are two tabs: "Connexion" (highlighted in green) and "Inscription". Below the tabs are input fields for "Prénom" (First name) and "Mot de passe" (Password). There is also a checkbox for "Se souvenir de moi" (Remember me) and a "Se connecter" (Log in) button.

Formulaire de connexion

The screenshot shows the MossAir registration page. It features two tabs at the top: "Connexion" and "Inscription" (highlighted in green). The registration form includes fields for "Prénom" (First name), "Nom" (Last name), "Email", and "Mot de passe" (Password). It also includes a password confirmation field ("Confirmer le mot de passe"), a checkbox for accepting "conditions d'utilisation" (Terms of service), and a "S'inscrire" (Sign up) button.

Formulaire d'inscription

# Catalogue Produits

## Fiche Produit Détailée

MossAir

Accueil   Produit   Histoire   À propos   Panier   Dashboard   arnaud   Déconnexion

Moss Air 3 ajouté au panier !



**Moss Air 3**

**199.99€**

✓ En stock (3 disponible(s))

Purificateur d'air haut de gamme avec technologie avancée de mousse stabilisée. Le meilleur de notre gamme.

Quantité :

1

Ajouter au panier

**Caractéristiques :**

- ✓ Filtration naturelle par mousse vivante
- ✓ Réduction de 90% des particules fines
- ✓ Diminution de 60% des COV
- ✓ Réduction de 40% du CO2
- ✓ Design minimaliste et élégant
- ✓ Entretien minimal

Page produit avec caractéristiques et gestion du stock



## Liste des Produits - Vue Mobile/Tablet

MossAir ☰

**Moss Air 3**  
199.99€

✓ En stock (3 disponible(s))

Purificateur d'air haut de gamme avec technologie avancée de mousse stabilisée. Le meilleur de notre gamme.

Quantité :

Ajouter au panier

**Caractéristiques :**

- ✓ Filtration naturelle par mousse vivante
- ✓ Réduction de 90% des particules fines
- ✓ Diminution de 60% des COV
- ✓ Réduction de 40% du CO2
- ✓ Design minimaliste et élégant
- ✓ Entretien minimal

**Moss Air 2**  
179.99€

✓ En stock (4 disponible(s))

Purificateur d'air premium avec double filtration naturelle. Idéal pour les grandes pièces.

Quantité :

Ajouter au panier

**Caractéristiques :**

- ✓ Filtration naturelle par mousse vivante
- ✓ Réduction de 90% des particules fines
- ✓ Diminution de 60% des COV

MossAir ☰

**Moss Air 3**  
199.99€

✓ En stock (3 disponible(s))

Purificateur d'air haut de gamme avec technologie avancée de mousse stabilisée. Le meilleur de notre gamme.

Quantité :

Ajouter au panier

**Caractéristiques :**

- ✓ Filtration naturelle par mousse vivante
- ✓ Réduction de 90% des particules fines
- ✓ Diminution de 60% des COV
- ✓ Réduction de 40% du CO2
- ✓ Design minimaliste et élégant
- ✓ Entretien minimal

**Moss Air 2**  
179.99€

✓ En stock (4 disponible(s))

Purificateur d'air premium avec double filtration naturelle. Idéal pour les grandes pièces.

Quantité :

Ajouter au panier

**Caractéristiques :**

- ✓ Filtration naturelle par mousse vivante
- ✓ Réduction de 90% des particules fines
- ✓ Diminution de 60% des COV

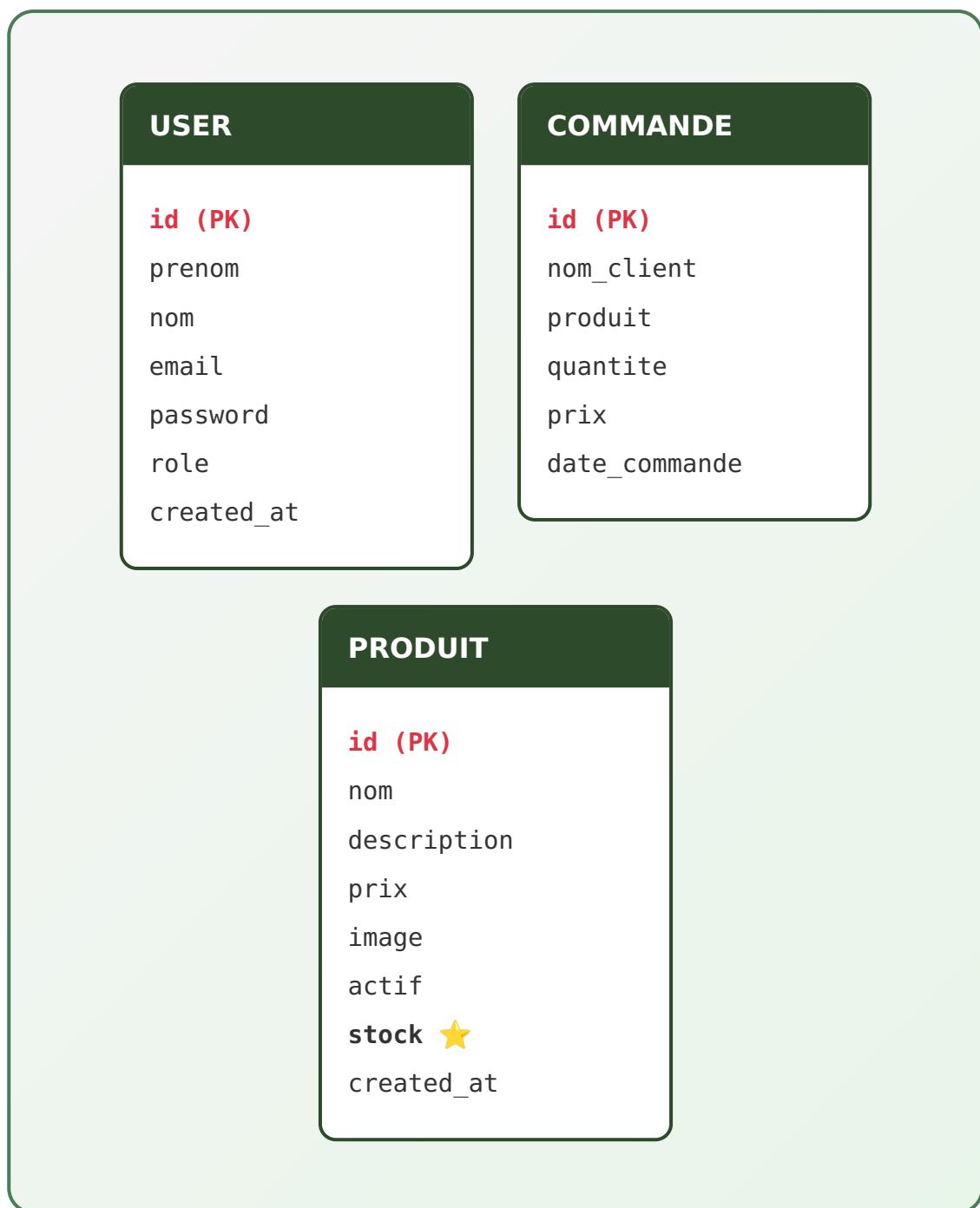
Vue Tablet

Vue Mobile

## **9. Base de Données : Structure et Relations**

09

# Modèle Conceptuel de Données (MCD)



## Table : produit

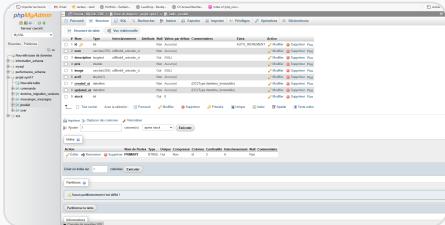
#	Nom	Type	Null	Défaut
1	<b>id</b>	int	Non	AUTO_INCREMENT

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

## Screenshots phpMyAdmin



### Structure des Tables



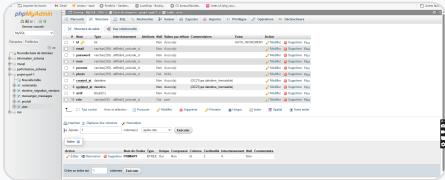


Table User

Table Produit



## 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 ([C2]type_datetime_immutable)	updated_at ([C2]type_datetime_immutable)	stock
2	qsdkm	qsml	25	Capture-d'écran-2025-09-23-161635-6900a79a417a6.png	1	2025-10-28 11:23:06	2025-12-04 10:56:27	9
3	sedf	sdf	251	videoframe-6360-693045935c408.png	1	2025-12-03 14:13:39	2025-12-04 10:28:32	9
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
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
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

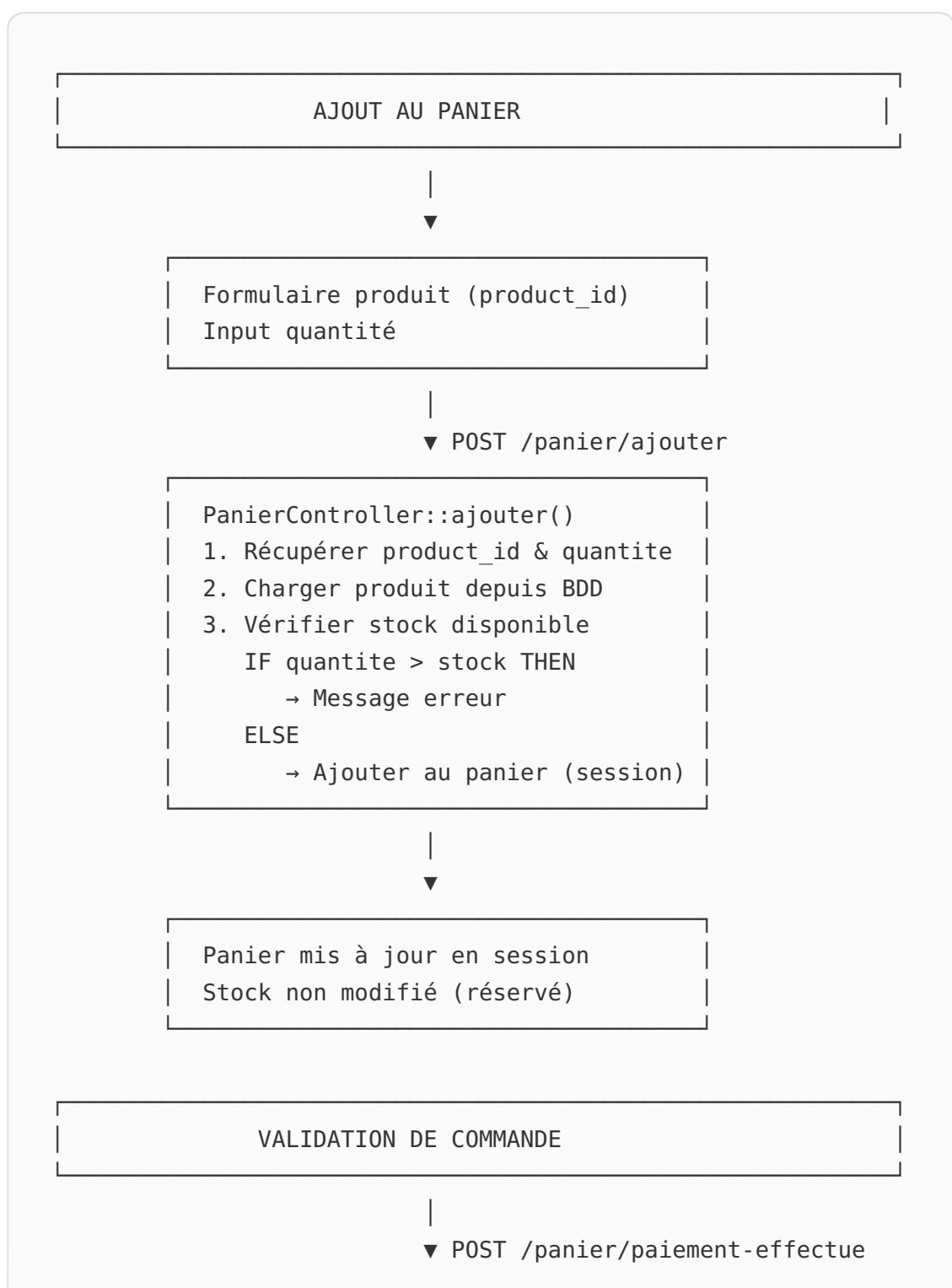
Tout afficher | Nombre de lignes : 25  Filtrer les lignes | Chercher dans cette table | Trier par clé : Aucun(e) ▾

Produits Moss Air 1, 2, 3 avec leurs stocks

# 10. Flux de Gestion du Stock

10

## Diagramme du Flux



```

PanierController::paiementEffectue()
POUR CHAQUE article du panier
    1. Charger produit depuis BDD
    2. Vérifier stock (double check)
    3. Créer commande
    4. Décrémenter stock
        produit.decrementStock(qty)
    5. Sauvegarder en BDD
FIN POUR
6. Vider le panier (session)

```



```

✓ Stock mis à jour en BDD
✓ Commande enregistrée
✓ Panier vidé

```

## 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 :  $3 \leq 10$  ✓ OK
- Stock BDD reste à **10** (non modifié)

**3. Validation commande :**

- Client clique "Valider le paiement"
- Double vérification :  $3 \leq 10$  ✓ 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

## 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/
    ├── AdminController.php
    └── stocks/
        ├── AuthController.php
        ├── CommandeController.php
        ├── HistoireController.php
        ├── PageController.php
        ├── PanierController.php
        ├── ProduitController.php
        ├── ProfileController.php
        └── SitemapController.php

    └── Entity/
        └── données/
            ├── Commande.php
            ├── Produit.php
            └── User.php

# Contrôleurs (logique des routes)
# Gestion admin (CRUD produits,
# Authentification (login, register)
# Gestion des commandes
# Page "Histoire"
# Pages statiques
# Gestion du panier et stock
# Affichage des produits
# Profil utilisateur
# Sitemap XML

# Entités Doctrine (modèles de
# Entité Commande (historique)
# Entité Produit avec champ stock
# 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
├── Version20251121100000.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

The screenshot shows two parts of the MossAir website. On the left is the 'À propos' (About) page, which contains text about the company's mission to protect nature and its products. On the right is the 'Footer du site' (Site footer), which includes links for 'Environnement', 'Boutique', 'Ressources', and 'Support', along with social media icons and a 'Télécharger' button.

**Page À propos**

**Footer du site**

## Historique des Commandes

The screenshot shows the 'Historique des commandes' (Order History) page. It displays a table of past purchases for three different product models: Moss Air 3, Moss Air 2, and Moss Air 1. Each row includes the product image, quantity, price, date, status (all marked as 'Livré' or delivered), and an 'Actions' column with a trash icon.

Produit	Quantité	Prix	Date	Statut	Actions
Moss Air 3	4	199.99€	04/12/2025 10:55	Livré	
Moss Air 2	4	179.99€	04/12/2025 10:55	Livré	
Moss Air 1	3	149.99€	04/12/2025 10:55	Livré	

## Liste des commandes passées par l'utilisateur

# Gestion des Utilisateurs (Admin)

Liste des Utilisateurs (2)						
ID	Prénom	Nom	Email	Rôle	Date inscription	Actions
5	user	user	user@gmail.com	Utilisateur	03/12/2025 16:04	<input type="button" value="User"/> <input type="button" value="Supprimer"/>
1	arnaud	arnaudbarotteaux@gmail.com		Admin	28/10/2025 12:07	<input type="button" value="Admin"/> <input type="button" value="Supprimer"/>

Interface d'administration des utilisateurs

# Alerte Stock Insuffisant

MossAir

Accueil Produit Histoire À propos Panier Dashboard arnaud Déconnexion

Stock insuffisant ! Seulement 3 disponible(s.)

**Moss Air 3**  
199.99€

✓ En stock (3 disponible(s))

Purificateur d'air haut de gamme avec technologie avancée de mousse stabilisée. Le meilleur de notre gamme.

Quantité :

1

Ajouter au panier

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



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