Université des Sciences et de la Technologie Houari Boumediene

Faculté d'Informatique

3ème année Ingénieur – Sécurité Informatique

Mini-Projet : Plateforme E-commerce Web

Module : Développement Web

Année universitaire : 2025–2026

Équipe : pravan_2025_team_17

Réalisé par :

Bentafat Wail – 232331419809
Benbraham Sara – 232331708115
Louail Wissam – 232331500218
Saiki Meriem – 232331454607

Enseignant : Monsieur Boubenia

8 janvier 2026

Table des matières

Chapitre 1

Introduction

Le développement des applications Web occupe aujourd'hui une place essentielle dans le domaine de l'informatique. Parmi ces applications, les plateformes de commerce électronique représentent des systèmes complexes intégrant plusieurs notions fondamentales telles que la gestion des utilisateurs, la sécurité, la persistance des données et l'interaction client–serveur.

Dans le cadre du module de Développement Web, ce mini-projet a pour objectif de concevoir et de développer une plateforme e-commerce Web fonctionnelle nommée **RAM-CORE**, spécialisée dans la vente de modules de mémoire RAM haute performance.

Cette plateforme implémente une architecture **MVC** (Model-View-Controller) côté serveur, utilisant le pattern **REST API** pour les interactions asynchrones, et respecte les principes de **séparation des préoccupations** (Separation of Concerns).

Ce projet a également pour but de familiariser les étudiants avec une organisation de projet proche du monde professionnel, en mettant l'accent sur :

— La collaboration en équipe via Git/GitHub
— L'utilisation de conteneurs Docker pour l'environnement de développement
— Le respect des bonnes pratiques en sécurité (OWASP Top 10)
— L'architecture orientée services (SOA)

Chapitre 2

Cahier des charges

2.1 Objectifs du projet

L'objectif principal est de mettre en pratique les concepts théoriques abordés durant le module de Développement Web à travers la réalisation d'une application e-commerce respectant les standards actuels.

2.1.1 Objectifs pédagogiques

1. Concevoir une architecture Web **three-tier** (Présentation, Logique Métier, Données)
2. Développer une application dynamique utilisant le pattern **MVC**
3. Implémenter un système d'authentification basé sur **sessions PHP** avec tokens CSRF
4. Manipuler une base de données relationnelle via **PDO** (PHP Data Objects)
5. Utiliser **AJAX** pour les interactions asynchrones
6. Appliquer les principes **RESTful** pour les Web Services
7. Gérer les versions avec **Git** et le workflow GitFlow

2.2 Fonctionnalités attendues

2.2.1 Fonctionnalités utilisateur

— Affichage dynamique des produits avec pagination côté serveur
— Moteur de recherche full-text avec filtrage multi-critères
— Système de panier avec persistance côté client (sessionStorage)
— Authentification sécurisée avec validation des entrées
— Page détaillée pour chaque produit (routing dynamique)

2.2.2 Fonctionnalités administrateur

— Interface CRUD complète pour la gestion des produits
— Tableau de bord avec statistiques
— Gestion des catégories
— Contrôle d'accès basé sur les rôles (RBAC)

Chapitre 3

Technologies utilisées

3.1 Stack technique

Technologie	Version	Rôle
HTML5	–	Structure sémantique des pages
CSS3	–	Mise en forme avec Flexbox/Grid, glassmorphism design
JavaScript (ES6+)	–	Interactivité client, fetch API, manipulation DOM
PHP	8.1+	Logique serveur, sessions, API REST
MySQL	8.0	SGBDR, stockage persistant
PDO	–	Couche d'abstraction base de données, requêtes préparées
Git/GitHub	–	Gestion de versions, collaboration
Docker	20.10+	Conteneurisation de l'environnement

Table 3.1 – Stack technique du projet RAMCORE

3.2 Patterns et principes appliqués

— **MVC** : Séparation Modèle-Vue-Contrôleur
— **REST** : Architecture pour les Web Services
— **DRY** : Don't Repeat Yourself
— **SOLID** : Principes de conception orientée objet
— **Progressive Enhancement** : Amélioration progressive

Chapitre 4

Architecture du projet

4.1 Architecture three-tier

Le projet RAMCORE adopte une architecture en trois couches distinctes :

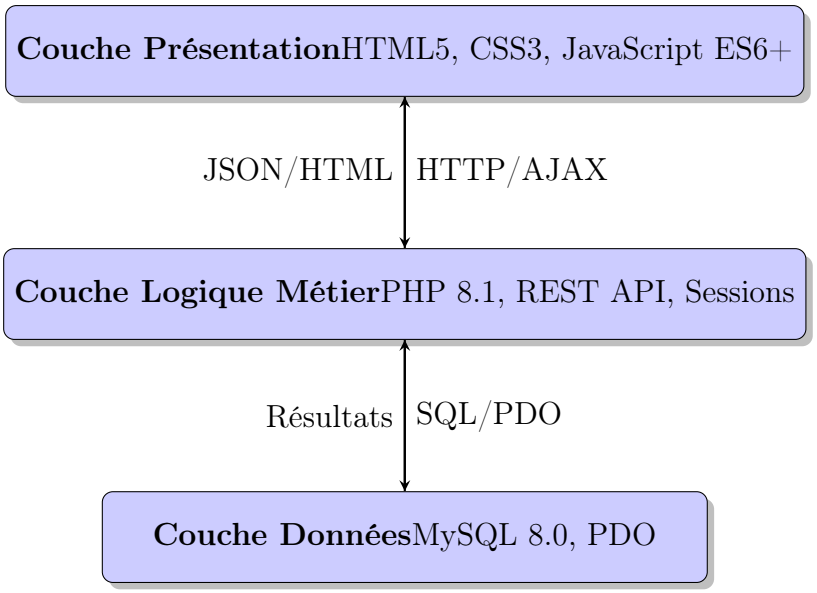

FIGURE 4.1 – Architecture three-tier de RAMCORE

4.2 Organisation des dossiers

```
ramcore/
          admin/                    # Interface d'administration
                dashboard.php
                manage_products.php
                check_admin.php     # Middleware de protection
          api/                      # Web Services REST
                products.php        # CRUD produits
                auth.php            # Authentification
                cart.php            # Gestion panier
                categories.php      # Gestion cat gories
          assets/
```

```
                css/                # Feuilles de style
                      global.css      # Styles globaux
                      cart.css
                      login.css
                js/                 # Scripts client
                      app.js          # Fonctions globales
                      cart.js         # Logique panier
                      login.js        # Authentification AJAX
                hero.png            # Assets visuels
          config/
                db.php              # Configuration PDO
          sql/
                migration.sql       # Scripts de migration
          index.php               # Page d'accueil
          login.php               # Authentification
          register.php            # Inscription
          cart.php                # Panier
          product.php             # D tail produit
          docker-compose.yml      # Configuration Docker
```

Listing 4.1 – Arborescence du projet

4.3 Flux de données

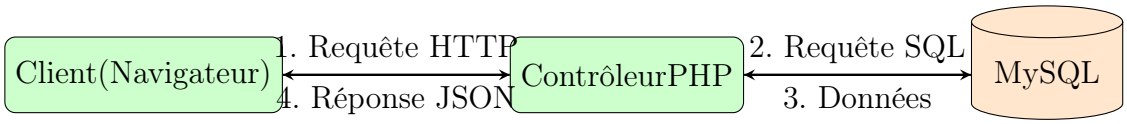

FIGURE 4.2 – Flux de données dans l'architecture

Chapitre 5

Base de données

5.1 Diagramme de classes (BDD)

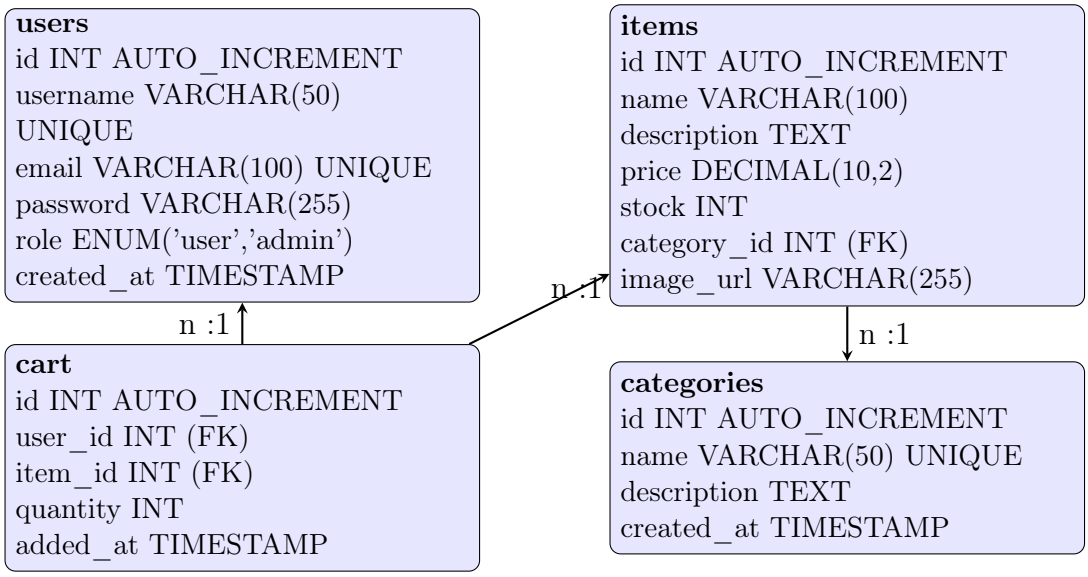

FIGURE 5.1 – Diagramme de classes de la base de données (format compact)

5.2 Contraintes d'intégrité

5.2.1 Contraintes référentielles

- **cart.user_id** → users.id (ON DELETE CASCADE)
- **cart.item_id** → items.id (ON DELETE CASCADE)
- **items.category_id** → categories.id (ON DELETE SET NULL)

5.2.2 Contraintes de domaine

- price ≥ 0
- stock ≥ 0
- quantity > 0
- role $\in$ {'user', 'admin'}

5.3 Optimisations

```sql
-- Index composites pour recherche avanc e
CREATE INDEX idx_items_search
ON items(category_id, price);

-- Index fulltext pour recherche textuelle
CREATE FULLTEXT INDEX idx_items_fulltext
ON items(name, description);

-- Partitionnement par date pour cart (optionnel)
ALTER TABLE cart PARTITION BY RANGE (YEAR(added_at));
```

Listing 5.1 – Index et optimisations

Chapitre 6

Fonctionnalités développées

6.1 Recherche et filtrage

6.1.1 Architecture technique

Le moteur de recherche implémente une approche **client-serveur hybride** :

1. Filtrage côté client (JavaScript) pour les interactions immédiates
2. Requêtes AJAX vers l'API REST pour la recherche serveur
3. Pagination côté serveur avec LIMIT/OFFSET

```
async function searchProducts() {
    const query = document.getElementById('product-search').value;
    const category = document.getElementById('category-filter').
       value;
    const minPrice = document.getElementById('min-price').value;
    const maxPrice = document.getElementById('max-price').value;

    const params = new URLSearchParams({
        search: query,
        category: category !== 'all' ? category : '',
        min_price: minPrice || 0,
        max_price: maxPrice || 999999
    });

    const response = await fetch(`api/products.php?${params}`);
    const data = await response.json();
    renderProducts(data.products);
}
```

Listing 6.1 – Implémentation recherche AJAX

6.1.2 Requête SQL optimisée

```
SELECT i.*, c.name as category_name
FROM items i
LEFT JOIN categories c ON i.category_id = c.id
WHERE
```

```
    (i.name LIKE :search OR i.description LIKE :search)
    AND (:category = '' OR c.name = :category)
    AND i.price BETWEEN :min_price AND :max_price
    AND i.stock > 0
ORDER BY i.created_at DESC
LIMIT :offset, :limit;
```

Listing 6.2 – Requête de recherche avec filtres

6.2 Gestion du panier

6.2.1 Architecture de persistance

Le panier utilise une approche **dual-storage** :

- **SessionStorage** pour les utilisateurs non authentifiés
- **Base de données** pour les utilisateurs authentifiés

```
function addToCart(productId, name, price) {
    let cart = JSON.parse(sessionStorage.getItem('cart')) || [];

    const existingItem = cart.find(item => item.id === productId);

    if (existingItem) {
        existingItem.quantity++;
    } else {
        cart.push({ id: productId, name, price, quantity: 1 });
    }

    sessionStorage.setItem('cart', JSON.stringify(cart));
    updateCartUI();

    // Synchronisation serveur si authentifi
    if (isLoggedIn()) {
        syncCartToServer(cart);
    }
}
```

Listing 6.3 – Gestion du panier côté client

6.3 Authentification

6.3.1 Processus d'authentification

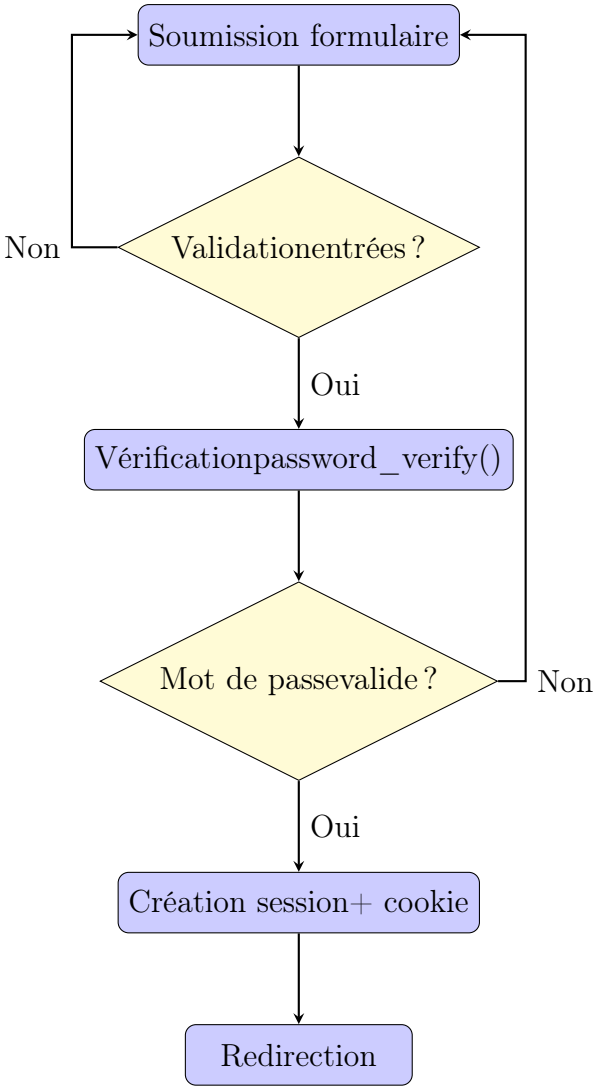

FIGURE 6.1 – Flux d'authentification

6.3.2 Implémentation sécurisée

```php
<?php
session_start();

// Validation des entr es
$username = filter_input(INPUT_POST, 'username',
                         FILTER_SANITIZE_STRING);
$password = $_POST['password'];

// Requ te pr par e PDO
$stmt = $pdo->prepare(
    "SELECT id, username, password, role
```

```
␣␣␣␣␣FROM␣users␣WHERE␣username␣=␣:username"
);
$stmt->execute(['username' => $username]);
$user = $stmt->fetch(PDO::FETCH_ASSOC);

// V rification du hash
if ($user && password_verify($password, $user['password'])) {
    // R g n ration de l'ID de session (protection session
       fixation)
    session_regenerate_id(true);

    // Stockage des donn es utilisateur
    $_SESSION['user_id'] = $user['id'];
    $_SESSION['username'] = $user['username'];
    $_SESSION['role'] = $user['role'];

    // R ponse JSON
    echo json_encode(['success' => true]);
} else {
    echo json_encode(['success' => false,
                      'message' => 'Identifiants␣invalides']);
}
?>
```

Listing 6.4 – Authentification côté serveur

6.4 Administration

6.4.1 Contrôle d'accès (RBAC)

```
<?php
// check_admin.php
session_start();

function requireAdmin() {
    if (!isset($_SESSION['user_id']) ||
        $_SESSION['role'] !== 'admin') {
        header('HTTP/1.1␣403␣Forbidden');
        header('Location:␣/login.php');
        exit('Acc s␣non␣autoris ');
    }
}

requireAdmin();
?>
```

Listing 6.5 – Middleware de protection admin

6.4.2 Interface CRUD

L'interface d'administration implémente les opérations CRUD complètes :

— **Create** : Ajout de produits avec validation
— **Read** : Affichage paginé avec tri
— **Update** : Modification avec gestion des conflits
— **Delete** : Suppression avec confirmation

Chapitre 7

Sécurité

7.1 Mesures de sécurité implémentées

7.1.1 Protection contre les injections SQL

```php
//     Vuln rable
$query = "SELECT␣*␣FROM␣users␣WHERE␣id␣=␣" . $_GET['id'];

//     S curis
$stmt = $pdo->prepare("SELECT␣*␣FROM␣users␣WHERE␣id␣=␣:id");
$stmt->execute(['id' => $_GET['id']]);
```

Listing 7.1 – Requêtes préparées PDO

7.1.2 Protection XSS (Cross-Site Scripting)

```php
//   chappement   HTML
echo htmlspecialchars($user_input, ENT_QUOTES, 'UTF-8');

// Pour JSON
echo json_encode($data, JSON_HEX_TAG | JSON_HEX_AMP);
```

Listing 7.2 – Échappement des sorties

7.1.3 Hashage des mots de passe

```php
// Inscription
$hashed = password_hash($password, PASSWORD_DEFAULT);

// Connexion
if (password_verify($input_password, $stored_hash)) {
    // Authentification r ussie
}
```

Listing 7.3 – Utilisation de password_hash()

7.1.4 Protection CSRF

```php
// G n ration du token
$_SESSION['csrf_token'] = bin2hex(random_bytes(32));

// Validation
if (!hash_equals($_SESSION['csrf_token'],
                 $_POST['csrf_token'])) {
    die('Token␣CSRF␣invalide');
}
```

Listing 7.4 – Tokens CSRF

7.1.5 Configuration sécurisée des sessions

```php
ini_set('session.cookie_httponly', 1);
ini_set('session.cookie_secure', 1); // HTTPS uniquement
ini_set('session.cookie_samesite', 'Strict');
session_start();
```

Listing 7.5 – Configuration session.php

7.2 Checklist OWASP Top 10

Vulnérabilité	**Statut**	**Mesure**
Injection SQL		Requêtes préparées PDO
XSS		htmlspecialchars(), CSP headers
Authentification cassée		password_hash(), sessions sécurisées
Exposition de données		HTTPS, mots de passe hashés
Contrôle d'accès		Middleware RBAC
Configuration		.env, erreurs en production
CSRF		Tokens CSRF

TABLE 7.1 – Conformité OWASP Top 10

Chapitre 8

Tests et validation

8.1 Tests fonctionnels

— Test de recherche avec différents critères
— Test d'ajout/suppression au panier
— Test de l'authentification (succès et échec)
— Test des contrôles d'accès administrateur
— Test CRUD complet sur les produits

8.2 Tests de sécurité

— Test d'injection SQL (sqlmap)
— Test XSS reflected et stored
— Test de session hijacking
— Test de CSRF
— Test de force brute sur login

Chapitre 9

Difficultés rencontrées

9.1 Défis techniques

9.1.1 Gestion des sessions côté serveur

La synchronisation entre le panier sessionStorage et la base de données pour les utilisateurs authentifiés a nécessité une logique de merge complexe pour éviter la perte de données.

9.1.2 Architecture REST

La conception d'une API REST cohérente respectant les conventions HTTP (codes de statut, méthodes) a demandé plusieurs itérations.

9.1.3 Sécurité

L'implémentation correcte des mesures de sécurité (tokens CSRF, requêtes préparées, validation) tout en maintenant une expérience utilisateur fluide.

9.2 Solutions apportées

— Utilisation de PDO avec mode d'erreur en exception
— Documentation de l'API avec exemples de requêtes
— Mise en place d'un fichier de configuration centralisé
— Tests unitaires pour les fonctions critiques

Chapitre 10

Perspectives

10.1 Améliorations futures

10.1.1 Fonctionnalités techniques

— **Système de paiement** : Intégration de Stripe/PayPal pour les transactions réelles
— **Gestion des commandes** : Suivi complet du cycle de vie des commandes
— **Notifications en temps réel** : WebSockets pour les mises à jour du stock
— **Cache côté serveur** : Redis pour améliorer les performances
— **API GraphQL** : Alternative à REST pour des requêtes plus flexibles
— **Progressive Web App** : Conversion en PWA avec Service Workers
— **Internationalisation** : Support multi-langues (i18n)

10.1.2 Optimisations

— **CDN** : Distribution des assets statiques via CloudFlare
— **Lazy Loading** : Chargement différé des images produits
— **Code Splitting** : Division du JavaScript en chunks
— **Database Indexing** : Optimisation des requêtes complexes
— **Compression** : Gzip/Brotli pour réduire la bande passante

10.1.3 Sécurité avancée

— **2FA** : Authentification à deux facteurs
— **Rate Limiting** : Protection contre les attaques par force brute
— **WAF** : Web Application Firewall (ModSecurity)
— **CAPTCHA** : Protection anti-bot sur les formulaires
— **Audit Logs** : Traçabilité complète des actions administrateur

10.2 Évolutions de l'architecture

10.2.1 Migration vers une architecture microservices

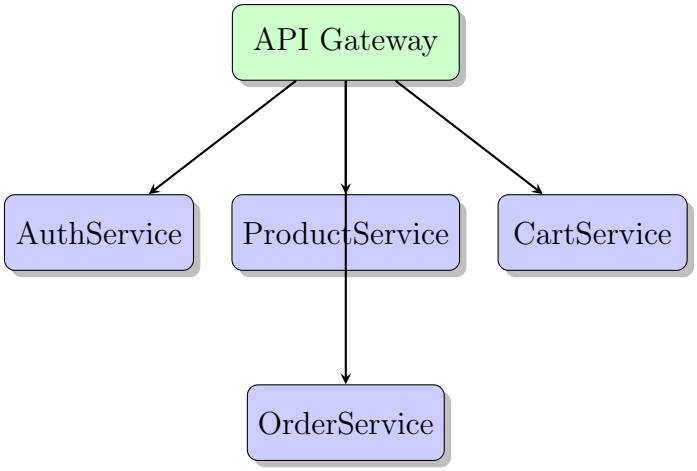

FIGURE 10.1 – Architecture microservices proposée

10.2.2 Stack technologique future

- **Backend** : Laravel 10+ ou Symfony 6+
- **Frontend** : React/Vue.js avec TypeScript
- **Base de données** : PostgreSQL pour les transactions, MongoDB pour le catalogue
- **Cache** : Redis + Varnish
- **Queue** : RabbitMQ pour les tâches asynchrones
- **Monitoring** : Prometheus + Grafana

Chapitre 11

Conclusion

ce projet de plateforme e-commerce a permis d'appliquer les concepts du module de Développement Web, en mobilisant des technologies variées et des bonnes pratiques en architecture, sécurité et conception.

11.1 Objectifs atteints

11.1.1 Compétences techniques

Maîtrise de :

— Architecture three-tier
— Bases de données relationnelles et optimisation SQL
— Sécurité web (OWASP Top 10)
— Développement d'API REST et JavaScript moderne (ES6+, Fetch API)
— PHP avancé (PDO, sessions, validation)
— DevOps : Docker et Git

11.1.2 Compétences transversales

— Travail collaboratif (Git/GitHub)
— Gestion de projet agile et documentation
— Debugging et veille technologique

11.2 Bilan pédagogique

Ce projet a mis en évidence l'importance de la planification, de la sécurité, de la maintenabilité, de la collaboration et de la rigueur dans le développement d'applications Web professionnelles.

11.3 Apport personnel

Renforcement des capacités à :

— Analyser et décomposer des problèmes complexes
— Rechercher et utiliser des ressources techniques

— Prendre des décisions d'architecture et gérer les priorités
— Apprendre de nouvelles technologies rapidement

11.4 Perspectives professionnelles

Compétences utiles pour :

— Stages en développement Web fullstack
— Contributions open-source et projets personnels
— Préparation aux certifications et maîtrise de frameworks modernes (Laravel, React, Vue.js)

Chapitre 12

Annexes

12.1 Code source principal

12.1.1 Configuration de la base de données

```php
<?php
$host = getenv('DB_HOST') ?: 'localhost';
$db = getenv('DB_NAME') ?: 'db';
$user = getenv('DB_USER') ?: 'user';
$pass = getenv('DB_PASS') ?: 'password';

$dsn = "mysql:host=$host;dbname=$db;charset=utf8mb4";

$options = [
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES => false,
];

try {
    $pdo = new PDO($dsn, $user, $pass, $options);
} catch (PDOException $e) {
    error_log($e->getMessage());
    die('Erreur de connexion  la base de donn es');
}
?>
```

Listing 12.1 – config/db.php

12.1.2 API REST - Produits

```php
<?php
require_once '../config/db.php';
header('Content-Type: application/json');

$method = $_SERVER['REQUEST_METHOD'];

```

```php
switch ($method) {
    case 'GET':
        $search = $_GET['search'] ?? '';
        $category = $_GET['category'] ?? '';
        $minPrice = $_GET['min_price'] ?? 0;
        $maxPrice = $_GET['max_price'] ?? 999999;

        $sql = "SELECT␣i.*,␣c.name␣as␣category_name
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣FROM␣items␣i
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣LEFT␣JOIN␣categories␣c␣ON␣i.category_id␣=␣c.id
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣WHERE␣(i.name␣LIKE␣:search
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣OR␣i.description␣LIKE␣:search)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣AND␣(:category␣=␣''␣OR␣c.name␣=␣:category)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣AND␣i.price␣BETWEEN␣:min_price␣AND␣:max_price
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ORDER␣BY␣i.created_at␣DESC";

        $stmt = $pdo->prepare($sql);
        $stmt->execute([
            'search' => "%$search%",
            'category' => $category,
            'min_price' => $minPrice,
            'max_price' => $maxPrice
        ]);

        echo json_encode([
            'success' => true,
            'products' => $stmt->fetchAll()
        ]);
        break;
}
?>
```

Listing 12.2 – api/products.php (extrait)

12.2 Scripts de migration SQL

```sql
-- Cr ation de la base de donn es
CREATE DATABASE IF NOT EXISTS ramcore_db
CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;

USE ramcore_db;

-- Table categories
CREATE TABLE IF NOT EXISTS categories (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50) UNIQUE NOT NULL,
    description TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    INDEX idx_name(name)
) ENGINE=InnoDB;
```

```
15
16 -- Table users
17 CREATE TABLE IF NOT EXISTS users (
18     id INT PRIMARY KEY AUTO_INCREMENT,
19     username VARCHAR(50) UNIQUE NOT NULL,
20     email VARCHAR(100) UNIQUE NOT NULL,
21     password VARCHAR(255) NOT NULL,
22     role ENUM('user', 'admin') DEFAULT 'user',
23     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
24     INDEX idx_email(email),
25     INDEX idx_username(username)
26 ) ENGINE=InnoDB;
27
28 -- Table items
29 CREATE TABLE IF NOT EXISTS items (
30     id INT PRIMARY KEY AUTO_INCREMENT,
31     name VARCHAR(100) NOT NULL,
32     description TEXT,
33     price DECIMAL(10,2) NOT NULL CHECK (price >= 0),
34     stock INT DEFAULT 0 CHECK (stock >= 0),
35     category_id INT,
36     image_url VARCHAR(255),
37     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
38     FOREIGN KEY (category_id)
39         REFERENCES categories(id)
40         ON DELETE SET NULL,
41     INDEX idx_category(category_id),
42     INDEX idx_price(price),
43     FULLTEXT INDEX idx_search(name, description)
44 ) ENGINE=InnoDB;
45
46 -- Table cart
47 CREATE TABLE IF NOT EXISTS cart (
48     id INT PRIMARY KEY AUTO_INCREMENT,
49     user_id INT NOT NULL,
50     item_id INT NOT NULL,
51     quantity INT DEFAULT 1 CHECK (quantity > 0),
52     added_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
53     FOREIGN KEY (user_id)
54         REFERENCES users(id)
55         ON DELETE CASCADE,
56     FOREIGN KEY (item_id)
57         REFERENCES items(id)
58         ON DELETE CASCADE,
59     UNIQUE KEY unique_cart(user_id, item_id)
60 ) ENGINE=InnoDB;
61
62 -- Insertion de donn es de test
63 INSERT INTO categories (name, description) VALUES
64 ('DDR4', 'Modules␣DDR4␣haute␣performance'),
65 ('DDR5', 'Derni re␣g n ration␣DDR5'),
```

```
('RGB', 'Modules avec  clairage  RGB');

INSERT INTO items (name, description, price, stock, category_id,
   image_url)
VALUES
('RAMCORE Elite 32GB DDR5',
 'Module 32GB DDR5 6400MHz CL32',
 199.99, 50, 2, 'assets/hero.png'),
('RAMCORE Pro 16GB DDR4',
 'Module 16GB DDR4 3200MHz CL16',
 89.99, 100, 1, 'assets/hero.png');
```

Listing 12.3 – sql/migration.sql (extrait)

12.3 Documentation API

12.3.1 Endpoints disponibles

Méthode	Endpoint	Description
GET	/api/products.php	Liste des produits avec filtres
GET	/api/products.php?id={id}	Détails d'un produit
POST	/api/auth.php	Authentification
POST	/api/register.php	Inscription
GET	/api/cart.php	Récupération du panier
POST	/api/cart.php	Ajout au panier
DELETE	/api/cart.php	Suppression du panier
GET	/api/categories.php	Liste des catégories

TABLE 12.1 – Documentation des endpoints API

12.3.2 Exemple de requête

```
# Recherche de produits
curl -X GET "http://localhost/api/products.php?search=DDR5&
   min_price=100&max_price=300"

# Authentification
curl -X POST http://localhost/api/auth.php \
  -H "Content-Type: application/json" \
  -d '{"username":"admin","password":"admin123"}'
```

Listing 12.4 – Exemple cURL

12.4 Guide d'installation

12.4.1 Prérequis

— Docker 20.10+

— Docker Compose 2.0+
— Git

12.4.2 Installation

```
# Cloner le d p t
git clone https://github.com/pravan_2025_team_17/ramcore.git
cd ramcore

# Lancer les conteneurs
docker-compose up -d

# Ex cuter la migration
docker-compose exec web php migrate.php

# Cr er l'utilisateur admin
docker-compose exec web php create_admin.php

# Acc der    l'application
# http://localhost:8080
```

Listing 12.5 – Commandes d'installation

12.5 Glossaire

API REST Architecture pour les services Web utilisant HTTP
CRUD Create, Read, Update, Delete (opérations de base)
CSRF Cross-Site Request Forgery (attaque par requête falsifiée)
MVC Model-View-Controller (pattern architectural)
PDO PHP Data Objects (abstraction base de données)
RBAC Role-Based Access Control (contrôle d'accès par rôles)
XSS Cross-Site Scripting (injection de script)

12.6 Bibliographie

Bibliographie

[1] OWASP Foundation, *OWASP Top Ten Web Application Security Risks*, 2021. https://owasp.org/www-project-top-ten/

[2] PHP Documentation Group, *PHP Manual - PDO*, 2024. https://www.php.net/manual/en/book.pdo.php

[3] Fielding, Roy Thomas, *Architectural Styles and the Design of Network-based Software Architectures*, Doctoral dissertation, University of California, Irvine, 2000.

[4] Mozilla Developer Network, *Web Development Guide*, 2024. https://developer.mozilla.org/

[5] Oracle Corporation, *MySQL 8.0 Reference Manual*, 2024. https://dev.mysql.com/doc/

[6] Docker Inc., *Docker Documentation*, 2024. https://docs.docker.com/

[7] Lien du dépôt GitHub https://github.com/equipe17-WEB/secu-mini-projet-web-2025.git