

Université Claude Bernard



Lyon 1

Faculté des Sciences et Technologies

Rapport de Stage

Licence 3 Informatique

**Conception et développement d'un site web
professionnel pour SIRA SARL**

Étudiant : Sow Malick

Numéro étudiant : 12312542

Période du stage : 19 Mai – 18 Juillet 2025

Entreprise d'accueil : SIRA SARL, Dakar (Sénégal)

Tuteur pédagogique : Monsieur Alexandre Meyer (Université Lyon 1)

Tuteur professionnel : Monsieur Kabirou SOW (Directeur général,
SIRA SARL)

Année universitaire 2024–2025

Table des matières

1	Introduction	2
2	Présentation de l'entreprise	3
3	Cahier des charges	4
1	Objectifs fonctionnels	4
2	Contraintes techniques	4
3	Périmètre	4
4	Environnement technique et outils	5
5	Méthodologie de travail	6
6	Développement et sécurité	8
7	Résultats, tests et validation	9
8	Conclusion	16
	Annexes	17

Chapitre 1

Introduction

Ce rapport présente le travail effectué dans le cadre d'un stage de Licence 3 Informatique, réalisé à distance au sein de l'entreprise SIRA SARL. Le projet visait la création d'un site web professionnel afin de permettre à l'entreprise de débiter sa digitalisation et de valoriser ses produits et services.

L'objectif était de développer une plateforme vitrine complète, intégrant un module de gestion de produits, un formulaire de contact, ainsi qu'une fonctionnalité de panier. Ce projet m'a permis de travailler en autonomie et de mobiliser un ensemble de compétences acquises au cours de ma formation.

Le présent document est structuré comme suit : une présentation de l'entreprise, un cahier des charges, les outils et technologies utilisés, la méthodologie de travail adoptée, une description détaillée du développement réalisé, une évaluation des résultats, suivie d'une conclusion et d'annexes techniques.

Chapitre 2

Présentation de l'entreprise

SIRA SARL est une entreprise sénégalaise créée en 2019 à Dakar. Elle exerce dans des domaines variés :

- vente de matériel informatique ;
- location et vente de véhicules ;
- gestion immobilière ;
- activités agricoles.

Malgré cette diversité, l'entreprise ne disposait jusqu'à présent d'aucune présence numérique. Le développement d'un site web représentait une première étape stratégique dans sa volonté de modernisation et de communication.

Chapitre 3

Cahier des charges

1. Objectifs fonctionnels

Le site devait présenter les activités de l'entreprise, afficher dynamiquement les produits disponibles, intégrer une interface d'administration sécurisée, un formulaire de contact fonctionnel, ainsi qu'une page panier permettant au visiteur d'envoyer une sélection de produits avec ses coordonnées.

2. Contraintes techniques

Aucune technologie spécifique ne m'a été imposée. L'entreprise attendait un site fiable, moderne et évolutif. Le site devait être responsive, sécurisé, et basé sur une architecture claire.

3. Périmètre

Le projet couvrait le développement du site vitrine avec module d'administration. J'ai proposé d'intégrer un système de paiement en ligne, mais l'entreprise a préféré reporter cette fonctionnalité. En effet, une grande partie de sa clientèle locale préfère encore les paiements physiques.

Chapitre 4

Environnement technique et outils

Le développement a été réalisé avec les langages PHP 8 (backend) et MySQL pour la base de données. Le front-end a été codé en HTML5, CSS3 et JavaScript, sans framework.

L'architecture du projet suit le modèle MVC (Modèle-Vue-Contrôleur), où le fichier `index.php` joue le rôle de routeur frontal.

J'ai utilisé Visual Studio Code comme éditeur principal, avec Git pour le suivi de version, et une instance EC2 d'AWS pour le déploiement et les tests. Les transferts de fichiers s'effectuaient via SCP¹.

1. SCP : Secure Copy Protocol, protocole sécurisé de copie de fichiers via SSH.

Chapitre 5

Méthodologie de travail

Le stage s'est déroulé à distance, ce qui m'a amené à adopter une organisation rigoureuse et structurée. Des points de suivi hebdomadaires étaient organisés avec mon tuteur professionnel via WhatsApp. Chaque échange permettait de valider les étapes précédentes, de recueillir des retours précis et de planifier les tâches suivantes. Cette communication régulière a facilité le bon avancement du projet malgré l'absence de présence physique.

J'ai adopté une méthode de développement incrémentale, basée sur une progression par lots de fonctionnalités. Chaque itération comprenait :

- l'analyse du besoin fonctionnel ;
- le développement technique du module concerné ;
- des tests unitaires et manuels ;
- la validation avec le tuteur ;
- un commit Git documenté avec un message explicite.

Le projet a été organisé selon une structure de répertoires claire, assurant une séparation des responsabilités conforme aux bonnes pratiques du développement web. Cette structuration facilite la maintenance, l'évolution du code, ainsi que la lisibilité pour tout développeur amené à reprendre le projet. Les principaux dossiers sont les suivants :

- **modele/** : contient les fichiers de requêtes SQL et les fonctions d'accès à la base de données ;
- **controleurs/** : centralise la logique métier et les traitements des actions utilisateur ;
- **vues/** : regroupe les fichiers HTML/PHP générant l'interface visible du site ;
- **inc/** : rassemble les fichiers partagés, tels que la configuration de la base de données ou les constantes globales ;
- **css/** : contient les feuilles de style CSS, organisées par page ou composant du site ;

- `img/` : regroupe les sous-dossiers d'illustrations et d'images (logo, images produits, visuels d'accueil, etc.).

J'ai adopté une convention de nommage stricte :

- les noms de dossiers sont en minuscules ;
- les fichiers et fonctions PHP utilisent le style *camelCase* (ex. `getProduitById.php`, `verifieConnexion()`), ce qui améliore la lisibilité et la cohérence ;
- les noms de variables sont choisis pour être explicites et représentatifs.

Cette organisation m'a permis de garantir un code lisible, structuré et facilement maintenable, tout en facilitant la collaboration et les futures évolutions du projet.

Chapitre 6

Développement et sécurité

Le projet a été structuré en plusieurs modules. La page d'accueil introduit l'entreprise et ses domaines d'activité. Le catalogue produits permet un affichage dynamique avec filtres, pagination et tri.

Une interface d'administration protégée permet d'ajouter, modifier ou supprimer des produits. Les formulaires sont sécurisés contre les injections et validés côté serveur.

Un formulaire de contact permet à un utilisateur de laisser un message. J'ai également intégré une page panier : le visiteur peut sélectionner des produits et soumettre ses coordonnées (téléphone, e-mail). Le contenu du panier est stocké dans une session valable une heure, évitant la perte d'informations. Côté sécurité, plusieurs mécanismes ont été mis en place afin de garantir l'intégrité des données et de se prémunir contre les attaques courantes :

- le hachage des mots de passe est réalisé via la fonction native `password_hash()` de PHP, garantissant un stockage sécurisé des identifiants ;
- des requêtes SQL préparées sont utilisées systématiquement afin d'éviter toute injection de code malveillant (voir Annexe C.1) ;
- les données affichées dans les pages sont échappées avec `htmlspecialchars()` pour prévenir les attaques XSS¹ (voir Annexe C.2) ;
- un jeton CSRF² est intégré aux formulaires sensibles pour valider l'authenticité des requêtes (voir Annexe C.3) ;
- un fichier `.htaccess` a été configuré pour bloquer l'accès aux fichiers critiques, restreindre les répertoires sensibles, et rediriger les requêtes vers le contrôleur principal (voir Annexe C.4).

1. XSS (Cross-Site Scripting) : faille permettant à un utilisateur malveillant d'injecter du code dans une page web

2. CSRF : Cross-Site Request Forgery, attaque qui exploite une session utilisateur active pour exécuter une action sans son consentement

Chapitre 7

Résultats, tests et validation

Le site a été déployé sur un serveur Ubuntu 22.04 à l'aide d'une instance EC2 d'AWS. Il est accessible en ligne à l'adresse suivante :

<http://16.170.201.194/>

Le code source est versionné sur GitHub dans une version publique allégée, excluant toute donnée sensible :

<https://github.com/MSow17/sira-public>

Tests fonctionnels

Des tests unitaires ont été réalisés sur les fonctions critiques du site, notamment la connexion à la base de données, l'authentification et le routage. À cela s'ajoutent des tests manuels sur la navigation, les formulaires, la gestion du panier, ainsi que la protection contre les accès non autorisés à l'espace d'administration.

Le formulaire de contact a été vérifié pour plusieurs cas de figure : champs vides, formats incorrects (e-mail, téléphone) et tentatives d'injection. Le panier a également été testé, notamment la persistance des données via les sessions pendant environ une heure, conformément aux spécifications attendues.

La figure ci-dessous illustre la page d'accueil, qui présente les pôles d'activité de SIRA SARL dans une mise en page responsive et claire :

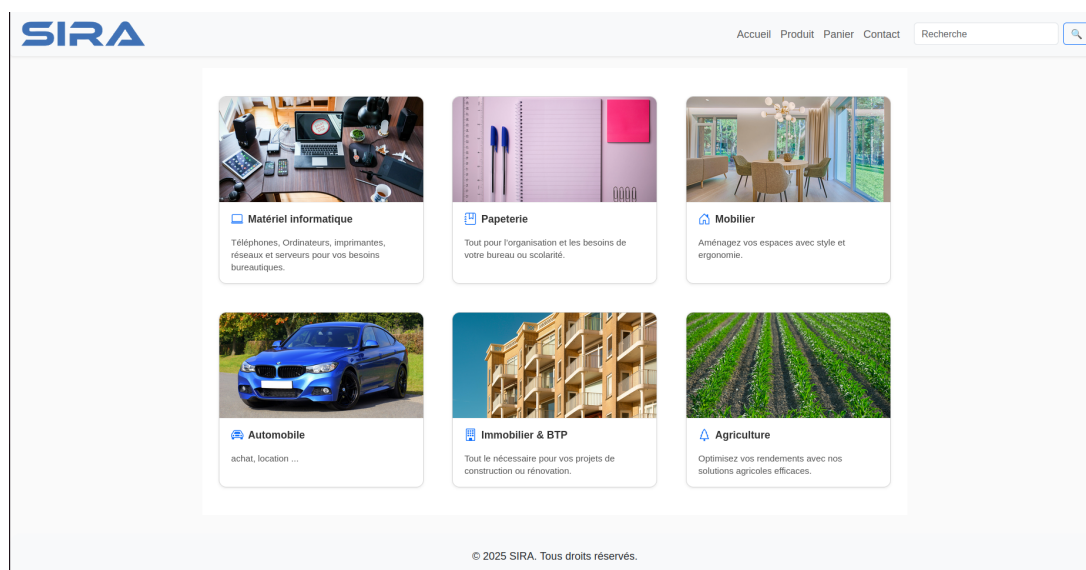


FIGURE 7.1 – Page d'accueil du site présentant les activités principales

L'interface d'administration permet la gestion des produits informatiques proposés par l'entreprise (ajout, modification, suppression). L'accès est protégé par authentification :

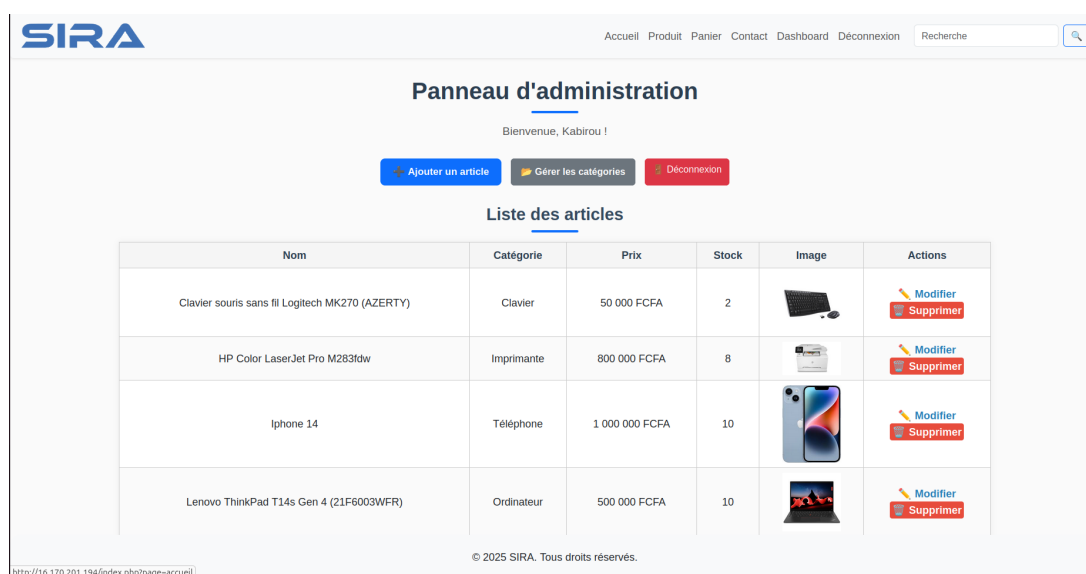


FIGURE 7.2 – Interface d'administration pour la gestion des produits

Tests de sécurité

Afin de s'assurer de la robustesse du site face aux attaques courantes, plusieurs vérifications ont été menées :

1. Analyse dynamique avec Burp Suite

Une analyse approfondie a été réalisée à l'aide de l'outil Burp Suite¹. Les tests suivants ont été menés pour évaluer la robustesse du site face aux attaques courantes :

- vérification de la présence de l'attribut `HttpOnly` sur les cookies de session ;
- injection de scripts malveillants dans les champs utilisateurs (XSS) ;
- modification des champs de formulaires pour détecter des failles CSRF ;
- simulation de requêtes POST falsifiées ou incomplètes ;
- injection SQL dans le formulaire de connexion ;
- vérification de la non-révélation d'informations sensibles dans les en-têtes HTTP ou messages d'erreur.

Les tests ci-dessous illustrent deux cas concrets de failles testées.

1.1 Injections XSS dans le champ de recherche

Des tentatives d'injection de scripts JavaScript ont été effectuées dans le champ de recherche pour tester la prévention des failles XSS. Les payloads suivants ont été utilisés :

- `<svg onload=alert(1)>`
- `<img src=x onerror=alert(1)>`
- `<script>prompt(1)</script>`

Tous ces scripts ont été neutralisés côté serveur par un encodage HTML (`<`, `>`), confirmant l'usage correct de la fonction `htmlspecialchars()` lors de l'affichage.

1. Burp Suite : logiciel d'analyse de sécurité utilisé pour tester les vulnérabilités des applications web

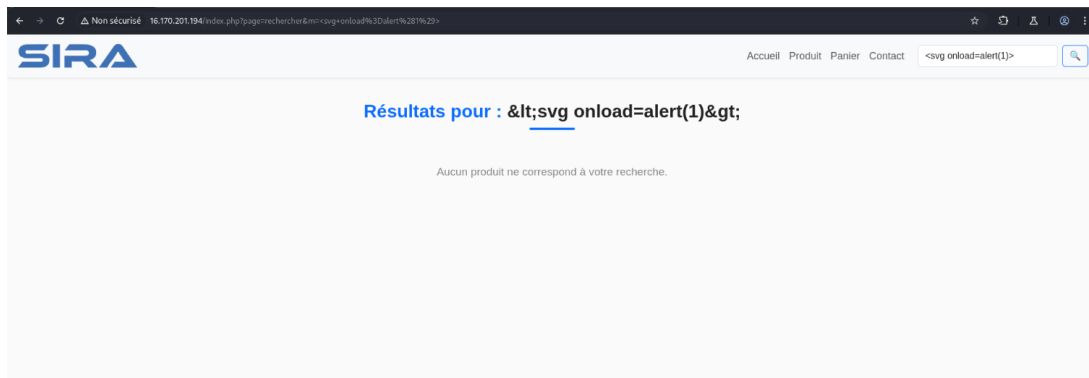
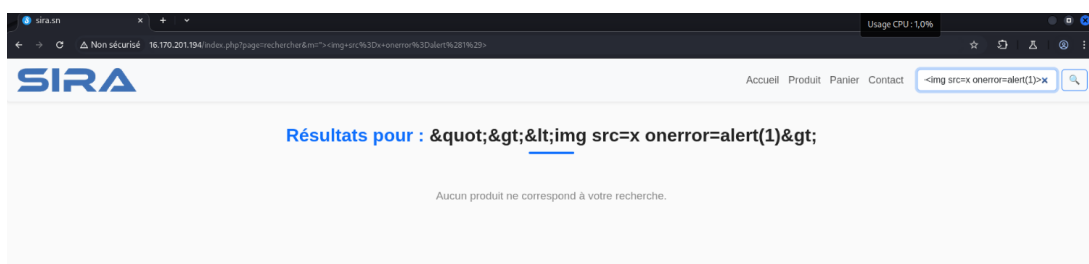
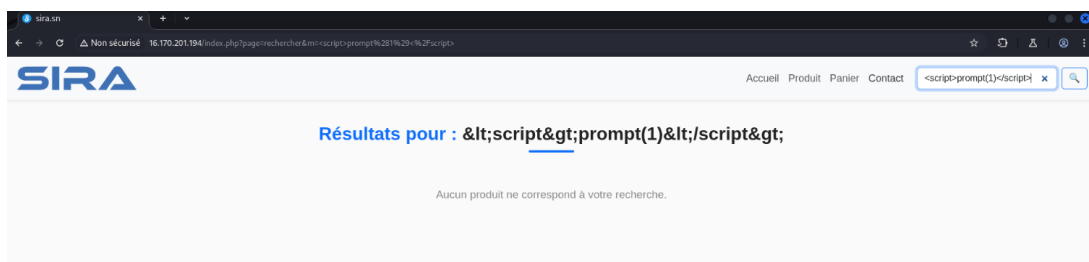


FIGURE 7.3 – Injection SVG encodée automatiquement dans le champ de recherche

FIGURE 7.4 – Test d’une balise image avec événement `onerror`, sans exécutionFIGURE 7.5 – Injection de balise `<script>` neutralisée côté serveur

1.2 Injection SQL via Burp Suite

Une requête POST modifiée a été envoyée via Burp Suite pour tester la résistance du formulaire de connexion à une injection SQL :

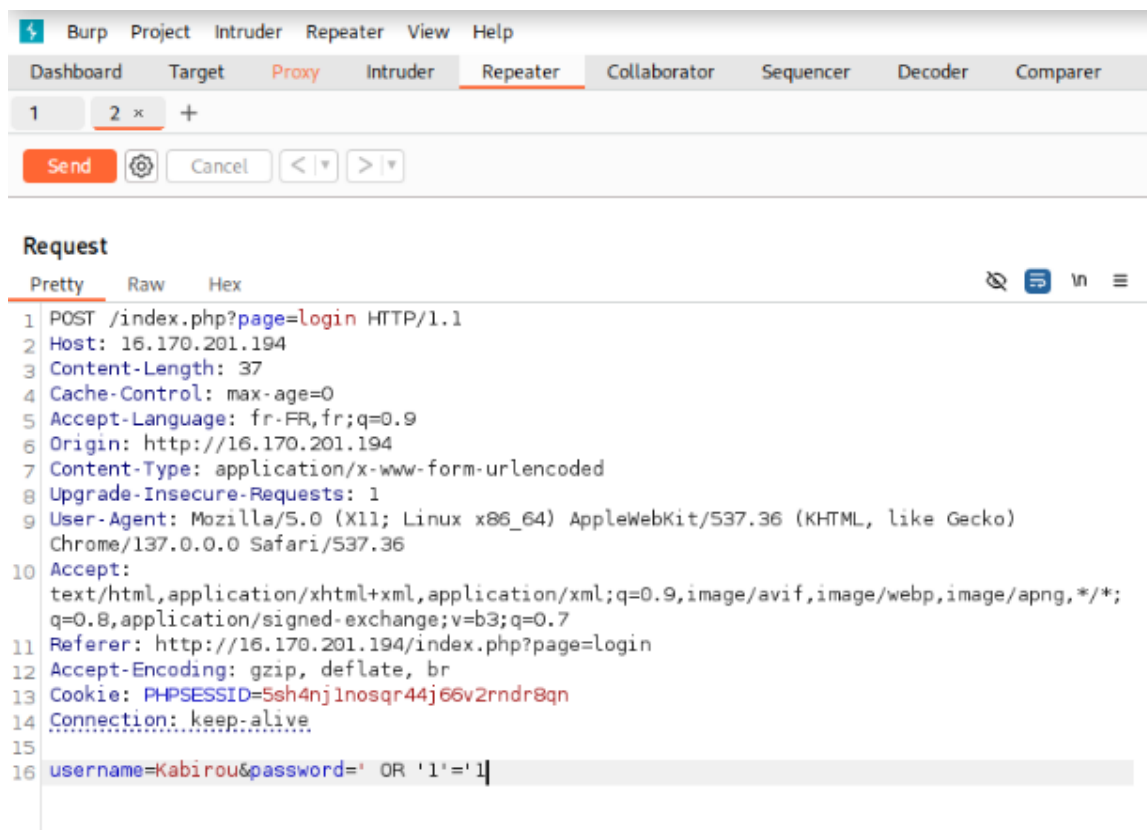
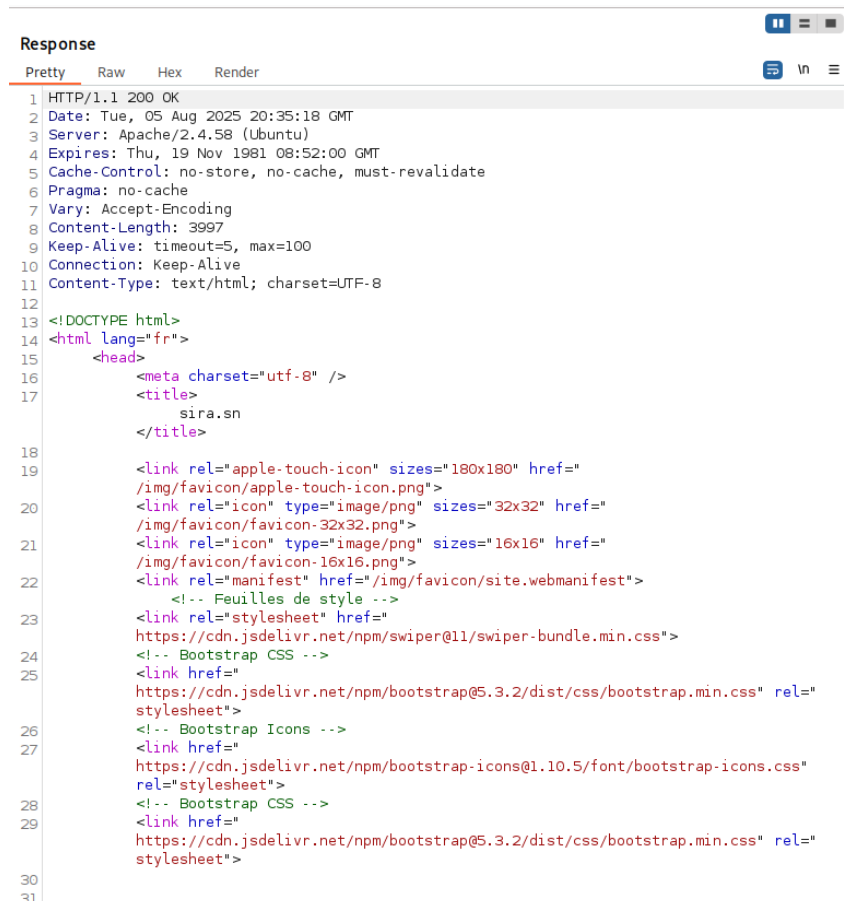


FIGURE 7.6 – Requête POST modifiée injectant le payload ' OR '1'='1

Le serveur a répondu sans accorder l'accès, affichant un simple message d'erreur d'authentification. Cela montre que la requête SQL est sécurisée, probablement via l'utilisation de requêtes préparées.

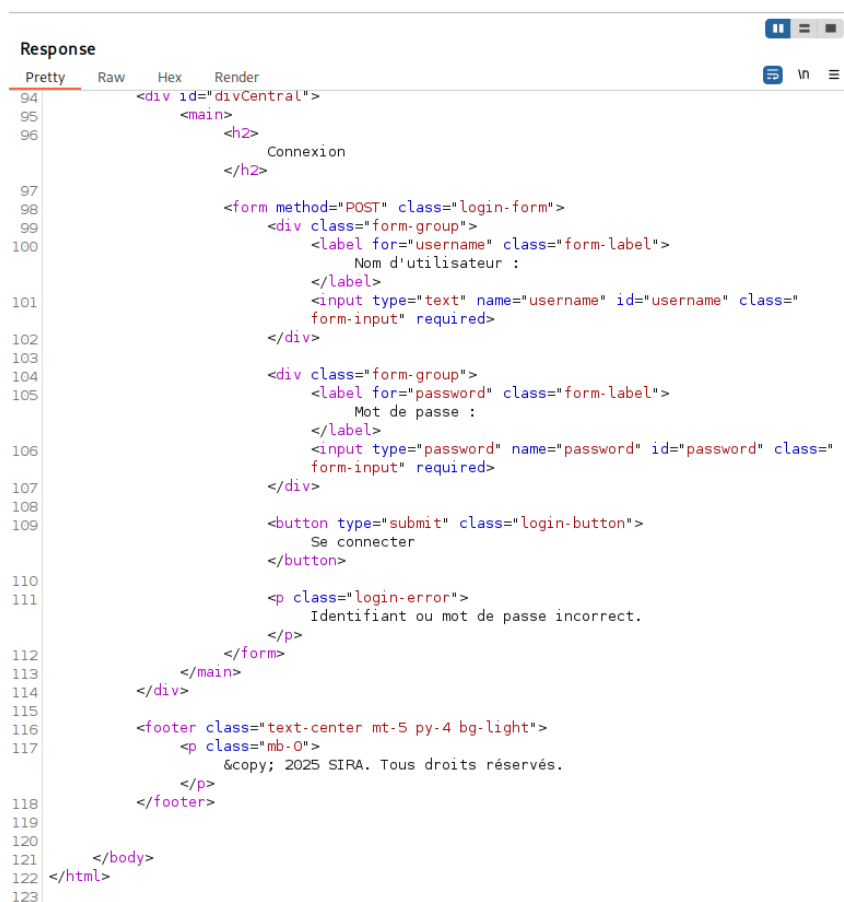


```

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Date: Tue, 05 Aug 2025 20:35:18 GMT
3 Server: Apache/2.4.58 (Ubuntu)
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Cache-Control: no-store, no-cache, must-revalidate
6 Pragma: no-cache
7 Vary: Accept-Encoding
8 Content-Length: 3997
9 Keep-Alive: timeout=5, max=100
10 Connection: Keep-Alive
11 Content-Type: text/html; charset=UTF-8
12
13 <!DOCTYPE html>
14 <html lang="fr">
15   <head>
16     <meta charset="utf-8" />
17     <title>
18       sira.sn
19     </title>
20
21     <link rel="apple-touch-icon" sizes="180x180" href="
22       /img/favicons/apple-touch-icon.png">
23     <link rel="icon" type="image/png" sizes="32x32" href="
24       /img/favicons/favicon-32x32.png">
25     <link rel="icon" type="image/png" sizes="16x16" href="
26       /img/favicons/favicon-16x16.png">
27     <link rel="manifest" href="/img/favicons/site.webmanifest">
28     <!-- Feuilles de style -->
29     <link rel="stylesheet" href="
30       https://cdn.jsdelivr.net/npm/swiper@11/swiper-bundle.min.css">
31     <!-- Bootstrap CSS -->
32     <link href="
33       https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="
34       stylesheet">
35     <!-- Bootstrap Icons -->
36     <link href="
37       https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-icons.css"
38       rel="stylesheet">
39     <!-- Bootstrap CSS -->
40     <link href="
41       https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="
42       stylesheet">
43   </head>
44   <body>
45     <div id="divCentral">
46       <main>
47         <h2>
48           Connexion
49         </h2>
50         <form method="POST" class="login-form">
51           <div class="form-group">
52             <label for="username" class="form-label">
53               Nom d'utilisateur :
54             </label>
55             <input type="text" name="username" id="username" class="
56               form-input" required>
57           </div>
58           <div class="form-group">
59             <label for="password" class="form-label">
60               Mot de passe :
61             </label>
62             <input type="password" name="password" id="password" class="
63               form-input" required>
64           </div>
65           <button type="submit" class="login-button">
66             Se connecter
67           </button>
68           <p class="login-error">
69             Identifiant ou mot de passe incorrect.
70           </p>
71         </form>
72       </main>
73     </div>
74     <div id="divFooter">
75       <div class="text-center mt-5 py-4 bg-light">
76         <p class="mb-0">
77           &copy; 2025 SIRA. Tous droits réservés.
78         </p>
79       </div>
80     </div>
81   </body>
82 </html>

```

(a) Première partie de la réponse du serveur



```

Response
Pretty Raw Hex Render
94 <div id="divCentral">
95   <main>
96     <h2>
97       Connexion
98     </h2>
99     <form method="POST" class="login-form">
100       <div class="form-group">
101         <label for="username" class="form-label">
102           Nom d'utilisateur :
103         </label>
104         <input type="text" name="username" id="username" class="
105           form-input" required>
106       </div>
107       <div class="form-group">
108         <label for="password" class="form-label">
109           Mot de passe :
110         </label>
111         <input type="password" name="password" id="password" class="
112           form-input" required>
113       </div>
114       <button type="submit" class="login-button">
115         Se connecter
116       </button>
117       <p class="login-error">
118         Identifiant ou mot de passe incorrect.
119       </p>
120     </form>
121   </main>
122 </div>
123   <div id="divFooter">
124     <div class="text-center mt-5 py-4 bg-light">
125       <p class="mb-0">
126         &copy; 2025 SIRA. Tous droits réservés.
127       </p>
128     </div>
129   </div>
130 </body>
131 </html>

```

(b) Deuxième partie de la réponse du serveur

FIGURE 7.7 – Réponse complète du serveur après tentative d'injection SQL avec Burp Suite

Les détails de l'implémentation des requêtes SQL préparées sont fournis en **Annexe C.1** (extrait de code sécurisé en PHP).

Validation finale

L'ensemble des fonctionnalités a été validé de manière itérative lors des réunions hebdomadaires avec le tuteur professionnel. Ce dernier a salué :

- la fluidité de navigation ;
- la couverture fonctionnelle du site ;
- et le respect strict du cahier des charges initial.

Chapitre 8

Conclusion

Ce stage m'a offert l'opportunité de mener un projet concret, complet et professionnalisant. En partant d'un besoin réel exprimé par l'entreprise, j'ai conçu, développé et mis en ligne un site vitrine moderne, structuré et sécurisé, sans recourir à un framework.

L'environnement en autonomie m'a permis de prendre des décisions techniques éclairées, de planifier mon travail de manière rigoureuse et de répondre efficacement aux retours hebdomadaires du tuteur professionnel. J'ai également proposé, de ma propre initiative, d'héberger le site sur une instance EC2 d'AWS afin de faciliter les phases de test et de validation à distance. Cette expérience m'a permis d'aborder des notions concrètes d'administration système, de déploiement cloud et de gestion des droits sous Linux.

Le développement du projet a renforcé mes compétences en PHP, MySQL, architecture MVC, sécurité web (XSS, CSRF, gestion des sessions, mot de passe haché), mais également en structuration de projet, en documentation, et en versionnement via Git.

À l'avenir, plusieurs pistes d'évolution peuvent être envisagées :

- intégrer un certificat SSL et un nom de domaine propre à l'entreprise ;
- adopter un framework CSS (comme Bootstrap ou Tailwind) pour faciliter l'extension du site ;
- ajouter une API REST afin d'ouvrir les données à des partenaires externes ;
- proposer un tableau de bord d'administration avec statistiques et vues consolidées.

Enfin, cette expérience a consolidé mon intérêt pour les domaines des systèmes, des réseaux, du cloud computing et de la cybersécurité, que je souhaite approfondir au cours de mon futur cursus en Master.

Annexes

Annexe A : Schéma SQL

```
CREATE TABLE categories (  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    nom VARCHAR(100),  
    description TEXT  
);  
  
CREATE TABLE produit (  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    nameP VARCHAR(100),  
    prix DECIMAL(10,2),  
    descriptionP TEXT,  
    descriptionT TEXT,  
    imageP VARCHAR(255),  
    stock INT DEFAULT 0,  
    categorie_id INT,  
    FOREIGN KEY (categorie_id) REFERENCES categories(id)  
);
```

Annexe B : extrait du code index.php

```
<?php
require('inc/routes.php');           // Définition des routes du site
require('inc/includes.php');         // Constantes globales : nom du site, slogan, etc
require('inc/config-bd.php');        // Paramètres de connexion à la base de données
require('inc/config-stmp.php');      // Configuration pour l'envoi d'e-mails SMTP
require('inc/session_config.php');   // Sécurité, configuration session
require('modele/modele.php');        // Fonctions du modèle
$pdo = getConnexionBD(); // Connexion à la base de données
?>

<!DOCTYPE html>
<html lang="fr">
<head>...</head>
<body>
<?php include($pathHeader); ?>
<div id="divCentral">
    <main>
        <?php
            // Valeurs par défaut
            $controleur = $controleurAccueil;
            $vue = $vueAccueil;

            if (isset($_GET['page'])) {
                $nomPage = $_GET['page'];

                if (isset($routes[$nomPage])) {
                    // Vérification des droits d'accès
                    $adminPages = ['admin', 'add_article', 'edit_article', 'delete_article'];
                    if (in_array($nomPage, $adminPages) && !isset($_SESSION['user_id'])) {
                        header('Location: index.php?page=login');
                        exit;
                    }
                }

                // Affectation des bons fichiers
                $controleur = $routes[$nomPage]['controleurs'];
                $vue = $routes[$nomPage]['vues'];
            }
        }
    </div>
    // Inclusion du contrôleur et de la vue correspondants
```

```
        include('controleurs/' . $controleur . '.php');
        include('vues/' . $vue . '.php');
    ?>
</main>
</div>
<?php include($pathFooter); ?>
</body>
</html>
```

Annexe C : Extraits de code liés à la sécurité web

C.1 – Requête SQL préparée avec PDO

```
<?php
$req = $pdo->prepare("SELECT * FROM produit WHERE id = ?");
$req->execute([$idProduit]);
$produit = $req->fetch();
?>
```

C.2 – Échappement des données (prévention XSS)

```
<?php
// Lors de l'affichage dans une page HTML
echo htmlspecialchars($produit['descriptionP'], ENT_QUOTES, 'UTF-8');
?>
```

C.3 – Protection CSRF avec jeton

```
<?php
// Génération du token (formulaire)
session_start();
if (empty($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
?>

<form method="post" action="traitement.php">
    <input type="hidden" name="csrf_token" value="<?php echo $_SESSION['csrf_token']; ?>" />
    <!-- autres champs -->
</form>

<?php
// Vérification du token (traitement)
if ($_POST['csrf_token'] !== $_SESSION['csrf_token']) {
    die('Erreur : jeton CSRF invalide.');
```

C.4 – Configuration .htaccess

```
# -----
# Sécurité : bloquer l'accès aux fichiers sensibles
# -----
<FilesMatch "\.(env|ini|log|sql|sh|bak|config|json|md|htaccess)$">
    Require all denied
</FilesMatch>

# -----
# Sécurité : restreindre les dossiers sensibles
# -----
RedirectMatch 403 ^/(\.git|backup|config|inc|modele|controleurs)(/.*)?$

# -----
# Réécriture d'URL pour router vers index.php
# -----
RewriteEngine On

# Redirection HTTPS (si tu as un certificat SSL)
#RewriteCond %{HTTPS} off
#RewriteRule ^ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]

# Si la requête n'est pas un fichier ou un dossier existant
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d

# Rediriger vers index.php
RewriteRule ^ index.php [QSA,L]

# -----
# Définir le charset par défaut
# -----
AddDefaultCharset UTF-8
```

Annexe D : Références

- <https://www.php.net/manual/fr/>
- <https://dev.mysql.com/doc/>
- <https://docs.aws.amazon.com/ec2/>
- <https://owasp.org/www-project-top-ten/>
- <https://developer.mozilla.org/fr/>