

Université de Rouen Normandie
UFR Sciences et Techniques
Départements Informatique



Mini projet de LW1– M1 Informatique e-bazar– Une plateforme de petites annonces

Application Web E-Bazar

Étudiante :
KHERBACHI Rima
CHERIFI Yasmine

Sous la direction de : M.NICART Florent
M.MALLET Olivier

Année Universitaire : 2025-2026

1 Introduction

Ce rapport présente la mise en œuvre d'un site web nommé EBazar, l'objectif est de déployer une appli web sur un serveur Linux via une Machine Virtuelle.

2 Architecture du projet

Notre projet est composé de 6 sous-dossiers, dont 3 qui permettent d'illustrer la technique MVC.

1. Le dossier config contient la configuration pour faire le lien entre la base de données et le projet.
2. Le dossier bdd contient le fichier init.sql, qui contient le code de la base de données.
3. Dossier Public (/public) : Sert de point d'entrée unique (Root) pour le serveur Web, contenant l'index, le CSS, le JavaScript et les images.

2.1 Architecture Technique (MVC)

L'application suit une architecture Modèle-Vue-Contrôleur (MVC) afin de séparer la logique métier de l'affichage :

1. Modèles (/models) : Gèrent les interactions avec la base de données,
 - **user.php** : Il nous permet de gérer les requêtes SQL pour tout ce qui concerne les données des utilisateurs, notamment pour afficher leur profil, leur permettre de s'inscrire et de se connecter.
 - **annonce.php** : Il nous permet de gérer l'ajout, la suppression et l'affichage des différentes annonces dans les différentes pages de l'application en se basant sur leur statut.
 - **categorie.php** : Il contient les requêtes pour afficher les catégories dans les pages et permet à l'administrateur de faire la gestion des catégories (ajout, modification).
 - **achat.html** : Il permet d'ajouter les achats à la base de données et de pouvoir récupérer les informations des achats.
 - **reserver.php** : Il permet d'ajouter les réservations à la base de données et de pouvoir récupérer leur les informations.
2. Vues (/views) : Contiennent les templates HTML/PHP pour le rendu visuel (accueil, connexion, ajout d'annonce).
3. Contrôleurs (/controllers) : Réceptionnent les requêtes et font le lien entre les modèles et les vues.

3 Mise en œuvre et Déploiement

Le déploiement a été réalisé sur la machine virtuelle Debian fournie, afin de garantir un environnement de production identique à celui du correcteur.

- **Configuration Réseau** : L'accès au serveur s'effectue via une adresse IP fixe (192.168.76.76) configurée sur un adaptateur réseau privé hôte (*Host-only*).
- **Transfert et Intégrité** : L'intégralité des sources a été transférée via le protocole SFTP vers le répertoire cible `/var/www/html/ebazar/`. Nous avons utilisé le logiciel FileZilla.
- **Automatisation de la Base de Données** : Conformément aux exigences, l'initialisation de la base `projet` est facilitée par un script SQL fourni (`init.sql`), incluant la création des tables et des catégories initiales.

- **Validation sur l’environnement cible** : Le bon fonctionnement du site a été validé directement sur la VM, permettant de corriger en amont les problèmes liés à la casse (*case sensitivity*) et aux permissions d’écriture.

4 Notice de Déploiement

Cette section détaille la procédure nécessaire pour installer et tester l’application **E-Bazar** sur le serveur de référence Debian.

4.1 Installation des fichiers et configuration serveur

1. **Transfert** : Déposer le répertoire `ProjetWeb-Kherbachi-Cherifi` dans `/var/www/html/` du serveur.
2. **Point d’entrée** : L’application est configurée pour être accessible via l’URL : `http://192.168.76.76/ProjetWeb-Kherba`
3. **Droits d’accès** : Pour permettre l’upload des photos, les commandes suivantes ont été exécutées pour définir le serveur Apache (`www-data`) comme propriétaire du dossier de médias :
 - `chmod -R 755 /var/www/html`
 - `chmod -R 755 /var/www/html/ebazar`
 - `chmod -R 755 /var/www/html/ebazar/public`
 - `chown -R www-data:www-data /var/www/html/ebazar/public/uploads`
 - `chmod -R 777 /var/www/html/ebazar/public/uploads`

4.2 Base de données et accès Administrateur

L’initialisation de la base de données est semi-automatisée via l’importation du script SQL fourni.

- **Importation** : Charger le fichier `init.sql` dans une base nommée `projet` via `phpMyAdmin` [?]
- **Identifiants de connexion (config.php)** :
 - **Hôte** : `localhost`
 - **Utilisateur** : `projet`
 - **Mot de passe** : `tejorp`
- **Compte Administrateur par défaut** :
 - **Email** : `admin@ebazar.local`
 - **Mot de passe** : `rimarima`

Le hachage utilisé pour ce compte administrateur est : `$2y$12$TGxqDWeiuKxvIwWlE3DTw.kx/89C8ExgJFXNxZUayR6LfQZEhMyA`

5 Sécurité et Ergonomie

5.1 Sécurité

5.1.1 Protection des mots de passe

L’application ne stocke aucun mot de passe ”en clair” dans la base de données afin de garantir la confidentialité des utilisateurs. Lors de l’inscription, nous utilisons la fonction native PHP `password_hash()` avec l’algorithme BCRYPT pour générer une empreinte sécurisée. Lors de la connexion, l’authentification est réalisée via `password_verify()`, qui compare le mot de passe saisi avec le hachage stocké, empêchant toute lecture directe des mots de passe en cas de compromission de la base de données.

5.1.2 Gestion des accès et des sessions

L'accès aux fonctionnalités sensibles est protégé par un système de sessions PHP :

- **Contrôle des rôles** : Le système distingue les utilisateurs "Inscrits" des "Administrateurs" via une variable de session `$_SESSION['role']`.
- **Redirection automatique** : Conformément au sujet, si un utilisateur non authentifié tente d'acheter un bien, il est automatiquement redirigé vers le formulaire de connexion. Une variable de redirection permet de reprendre l'opération après authentification.

5.1.3 Protection contre les failles Web

- **XSS** : Toutes les données provenant de la base de données (titres d'annonces, descriptions) sont échappées avec la fonction `htmlspecialchars()` avant d'être affichées dans les vues, empêchant l'exécution de scripts malveillants.
- **Injections SQL** : L'utilisation de requêtes préparées avec PDO (ex : `$pdo->prepare()`) dans les modèles garantit que les données saisies par les utilisateurs ne peuvent pas modifier la structure de nos requêtes SQL.

5.1.4 Sécurité du déploiement

- **Racine du site** : Seul le dossier `/public` est accessible depuis le navigateur. Les fichiers sensibles comme les configurations (`config.php`), les modèles et les contrôleurs sont placés en dehors de la racine publique pour éviter tout accès direct aux sources.
- **Validation des fichiers** : Lors de l'ajout d'une annonce, le script vérifie que les fichiers sont bien des images JPEG et limite leur poids à 200 kio, évitant ainsi l'envoi de fichiers dangereux ou trop volumineux sur le serveur.

5.2 Ergonomie et Expérience Utilisateur

L'interface de **E-Bazar** a été optimisée pour garantir une navigation intuitive et répondre aux exigences du cahier des charges :

- **Navigation facilitée** : La page d'accueil affiche les catégories avec le décompte des annonces et un aperçu des quatre derniers biens mis en vente.
- **Lisibilité des listes** : Utilisation d'une pagination (10 biens par page) et de vignettes visuelles pour chaque annonce afin d'alléger l'affichage.
- **Parcours utilisateur fluide** : Redirection automatique vers le formulaire de connexion lors d'une action réservée, avec retour au contexte initial après authentification.
- **Fiches produits structurées** : Regroupement cohérent des informations clés (prix, livraison, photos) pour faciliter la prise de décision de l'acheteur.
- **Gestion centralisée** : Un espace membre organisé pour un suivi efficace de l'activité.

6 Difficultés rencontrées et résolutions techniques

Le passage d'un environnement de développement local (Windows) à l'environnement de production imposé (Serveur Debian) a constitué une étape majeure du projet, révélant plusieurs défis techniques :

6.1 Sensibilité à la casse et portabilité du code

Conformément aux avertissements du sujet concernant les environnements Windows, la difficulté principale fut la gestion de la casse.

- **Problème** : Des erreurs *HTTP 500* ou des échecs d'inclusion de classes apparaissaient sur la VM car les appels `require_once` ne respectaient pas exactement la casse des fichiers (ex : `user.php` vs `User.php`).
- **Résolution** : Une normalisation stricte des noms de fichiers et des appels système a été effectuée pour garantir la compatibilité sous Linux.

6.2 Gestion des permissions et propriété du système de fichiers

Le déploiement sur le serveur Apache a nécessité une configuration précise de l'accès aux répertoires, notamment pour la gestion des médias.

- **Problème** : L'upload des images pour les annonces échouait systématiquement avec une erreur *"Permission denied"* dans les logs Apache. Le répertoire `public/uploads` n'autorisait pas l'écriture par le serveur web.
- **Résolution** : Pour résoudre ce blocage, nous avons appliqué une double configuration en ligne de commande sur la VM :
 1. `chmod -R 755 public/uploads` : pour définir les droits de lecture et d'exécution nécessaires.
 2. `chown -R www-data:www-data public/uploads` : pour transférer la propriété du dossier à l'utilisateur système d'Apache (*www-data*).

Cette approche a permis de garantir que l'application puisse enregistrer les photos des biens tout en respectant les contraintes de sécurité du système Linux.

6.3 Débogage via les logs serveur

Contrairement au développement local, les erreurs PHP n'étaient pas affichées directement dans le navigateur pour des raisons de sécurité.

- **Problème** : Identification difficile de l'origine des pages blanches lors de la configuration initiale.
- **Résolution** : L'analyse en temps réel des journaux d'erreurs via la commande `tail -f /var/log/apache2/error.log` dans la console Debian a été indispensable pour identifier les erreurs de chemins et de syntaxe.

7 Conclusion

La réalisation de ce projet **E-Bazar** a permis de mettre en oeuvre l'ensemble des concepts vus en cours, depuis la conception d'une architecture logicielle structurée jusqu'au déploiement sur un serveur de production réel.

L'adoption du modèle **MVC** (Modèle-Vue-Contrôleur) a été déterminante pour garantir la clarté du code et faciliter la maintenance des différentes fonctionnalités, telles que la gestion des annonces, le processus d'achat et l'interface d'administration [?]. Par ailleurs, les contraintes liées au déploiement sur la machine virtuelle Debian ont souligné l'importance de la portabilité du code et de la maîtrise de l'environnement serveur (gestion des permissions, configuration Apache et logs d'erreurs).

Enfin, l'accent mis sur la sécurité via le hachage des mots de passe avec *BCRYPT* et l'utilisation de requêtes préparées assure une plateforme fiable pour les utilisateurs. Ce projet constitue ainsi une base

solide pour comprendre les enjeux du développement web *full-stack* et les rigueurs nécessaires à la mise en ligne d'une application professionnelle.