# Documentation Technique - Gestion des Clients

# 1. Introduction

Le projet **Gestion des Clients** est une application web développée en PHP permettant la gestion des utilisateurs et de leurs sessions. Il inclut un système d'authentification sécurisé et une interface d'administration pour la gestion des utilisateurs et des logs.

# 2. Architecture et Technologies

## 2.1 Technologies utilisées

- **Langage principal** : PHP (version 8.2.0)
- **Base de données** : MySQL (version 5.7)
- **Front-end** : HTML5, TailwindCSS (version 4.0)
- **Gestion des dépendances** : Composer
- **Serveur local** : Serveur PHP intégré

## 2.2 Architecture du projet

L'application suit une architecture **MVC (Modèle - Vue - Contrôleur)** organisée comme suit :

```
PHP_MVC/
├── app/
│   ├── Controllers/      # Gestion des requêtes et de la logique métier
│   ├── Models/           # Interaction avec la base de données
│   ├── Views/            # Fichiers de vue (HTML, PHP)
│   ├── Core/             # Classes parents et configuration
│   ├── Config/           # Fichiers de configuration
├── public/               # Fichiers accessibles publiquement (CSS, JS,
images)
├── vendor/               # Dépendances Composer
├── composer.json         # Fichier de configuration Composer
├── composer.lock         # Fichier des dépendances installées
└── README.md             # Documentation du projet
```

# 3. Base de données

## 3.1 Schéma de la base de données

Le projet repose sur une base de données MySQL avec trois tables principales :

### Table `users` (Gestion des utilisateurs)

| Champ | Type | Description |
|---|---|---|
| **id** | INT (PK, AI) | Identifiant unique de l'utilisateur |

| | | |
|---|---|---|
| **name** | VARCHAR(100) | Nom complet de l'utilisateur |
| **email** | VARCHAR(255) | Adresse e-mail unique |
| **password** | VARCHAR(255) | Mot de passe haché (bcrypt) |
| **role** | ENUM('admin', 'client') | Rôle de l'utilisateur |
| **status** | BOOLEAN | Activation ou désactivation du compte |
| **created_at** | TIMESTAMP | Date de création |

**Table `sessions` (Suivi des connexions)**

| Champ | Type | Description |
|---|---|---|
| **id** | INT (PK, AI) | Identifiant unique de la session |
| **user_id** | INT (FK) | Identifiant de l'utilisateur |
| **login_time** | TIMESTAMP | Date et heure de connexion |
| **logout_time** | TIMESTAMP | Date et heure de deconnexion |

**Table `roles` (Gestion des roles)**

| Champ | Type | Description |
|---|---|---|
| **id** | INT (PK, AI) | Identifiant unique du log |
| **name** | VARCHAR(50) | Nom du role |

# 4. Fonctionnalités détaillées

## 4.1 Authentification et Sécurité

- **Inscription et connexion sécurisée** avec hachage de mot de passe (bcrypt).
- **Gestion des sessions** via `$_SESSION`.
- **Restrictions d'accès** :
  - Un **client** ne peut accéder qu'à son profil et à son historique.
  - Un **administrateur** peut gérer les comptes utilisateurs et consulter les logs.

## 4.2 Espace Administrateur

- Tableau de bord avec un aperçu des utilisateurs.
- Gestion des utilisateurs (création, modification, suppression, activation/désactivation).
- Attribution et gestion des rôles (`admin`, `client`).
- Consultation de l'historique des connexions.

## 4.3 Espace Client

- Accès au profil avec possibilité de mise à jour des informations.
- Consultation de l'historique des connexions.

# 5. Déploiement et Configuration

### 5.1 Installation locale

1. **Cloner le dépôt** :

```
git clone https://github.com/marc575/PHP_MVC.git
cd PHP_MVC
```

2. **Configurer la base de données** :

   Créer une base de données MySQL.

   Importer le fichier SQL :

   ```
   mysql -u utilisateur -p nom_de_la_base < fichier.sql
   ```

3. **Installer les dépendances avec Composer** :

```
composer install
```

4. **Démarrer le serveur** :

```
php -S localhost:8000 -t public
```

5. **Accéder à l'application** via :
   - **Page d'accueil** : http://localhost:8000
   - **Inscription** : http://localhost:8000/auth/register
   - **Connexion** : http://localhost:8000/auth/login

# 6. API interne et Points d'accès

| Route | Méthode | Rôle | Description |
|---|---|---|---|
| `/auth/register` | GET | Authentifié | Affiche le formulaire d'inscription |
| `/auth/register` | POST | Authentifié | Traite l'inscription |
| `/auth/login` | GET | Public | Affiche le formulaire de connexion |
| `/auth/login` | POST | Public | Authentifie un utilisateur |
| `/logout` | GET | Authentifié | Déconnecte l'utilisateur |
| `/users/show` | GET | Admin | Liste tous les utilisateurs |
| `/user/update?id=:id` | POST | Authentifié | Met à jour un utilisateur |
| `/user/delete?id=:id` | GET | Authentifié | Supprime un utilisateur |
| `/logs` | GET | Authentifié | Affiche les logs des connexions |

# 7. Sécurité et Bonnes Pratiques

- **Hashage des mots de passe** : Utilisation de `password_hash()` avec `bcrypt`.
- **Validation des entrées utilisateurs** : Filtrage des entrées pour éviter les injections SQL.
- **Protection contre le Cross-Site Scripting (XSS)** : Encodage des sorties avec `htmlspecialchars()`.
- **Protection CSRF** : Ajout de jetons CSRF sur les formulaires sensibles.
- **Sécurisation des sessions** : Régénération de session (`session_regenerate_id()`) après connexion.

# 8. Évolutions et Améliorations possibles

- **Intégration d'un système de récupération de mot de passe**.
- **Ajout de la pagination pour la gestion des utilisateurs**.
- **Amélioration de l'interface utilisateur avec des animations TailwindCSS**.
- **Mise en place d'une API RESTful pour faciliter l'intégration avec d'autres services**.
- **Ajout d'un système de notifications** (par email ou en temps réel).

# 9. Conclusion

Le projet **Gestion des Clients** est une application robuste pour l'administration des utilisateurs avec un système sécurisé d'authentification et de suivi des sessions. Cette documentation technique sert de guide pour les développeurs souhaitant contribuer ou améliorer le projet.

Pour toute question ou amélioration, merci de contacter l'auteur :

- **GitHub** : https://github.com/marc575