Explications des améliorations :

1. Protection CSRF : Ajout d'un token CSRF pour prévenir les attaques de type Cross-Site Request Forgery.
2. Affichage des messages : Gestion des messages d'erreur et de succès via les paramètres GET, permettant un retour utilisateur clair.
3. Sécurité des mots de passe : Il est recommandé d'utiliser les fonctions password\_hash() pour le hachage des mots de passe lors de l'enregistrement, et password\_verify() pour la vérification lors de la connexion.
4. Sécurisation des sessions : Assure-toi que les pages sensibles vérifient la présence d'une session valide pour restreindre l'accès aux utilisateurs non authentifiés.
5. Mise à jour de config\_connexion.php : Assure-toi que ce fichier vérifie le token CSRF et utilise password\_verify() pour comparer les mots de passe.
6. Sécurisation des pages administratives : Ajoute une vérification de session en haut de chaque page réservée aux administrateurs pour garantir que seul un utilisateur authentifié peut y accéder.
7. Gestion des erreurs : Implémente une redirection vers connexion.php avec des messages d'erreur appropriés en cas d'échec de la connexion.

Oui, l'ajout d'un token CSRF (Cross-Site Request Forgery) est fortement recommandé, même pour un projet en développement local. Il s'agit d'une mesure de sécurité essentielle pour protéger ton application contre les attaques malveillantes.

Pourquoi ajouter un token CSRF ?

Un token CSRF est une valeur secrète et unique générée pour chaque session utilisateur. Il est inclus dans les formulaires HTML et vérifié lors de la soumission pour s'assurer que la requête provient bien de l'utilisateur authentifié et non d'un attaquant. Cela empêche des actions non autorisées, comme la soumission de formulaires malveillants, d'être exécutées à l'insu de l'utilisateur.

Même si tu travailles en local, il est important de simuler des conditions de production sécurisées. Les attaques CSRF peuvent survenir même sur des environnements locaux si des vulnérabilités existent. De plus, l'intégration de cette sécurité dès le début du développement permet de l'appliquer de manière cohérente tout au long du projet.

* Vérification du token CSRF : Avant de traiter les données du formulaire, on s'assure que le token envoyé correspond à celui stocké en session.
* Sécurisation de la connexion : La connexion à la base de données est effectuée avec les identifiants appropriés. Les requêtes sont préparées pour éviter les injections SQL.
* Gestion des erreurs : Des messages d'erreur clairs sont affichés en cas de problème, comme un mot de passe incorrect ou un token CSRF invalide.

En intégrant cette vérification, tu renforces la sécurité de ton application dès le début du développement. Si tu as besoin d'aide pour d'autres parties du projet, n'hésite pas à demander.

Bonnes pratiques pour la gestion des clés API

* Ne partagez jamais votre clé API publiquement : Évitez de la publier dans des dépôts publics ou de l'inclure dans du code côté client.
* Utilisez des restrictions : Limitez l'utilisation de la clé à des adresses IP spécifiques, des référents HTTP ou des applications particulières.
* Surveillez l'utilisation : Gardez un œil sur l'activité liée à votre clé pour détecter toute utilisation abusive.
* Renouvelez régulièrement vos clés : Changez vos clés périodiquement pour renforcer la sécurité.