

Rapport de Projet

Par : Ali Ismail

Destinataire : Madame Marie Amine

Date : 12/5/2025

Objet : Rapport de l'avancement du projet de formulaire sécurisé

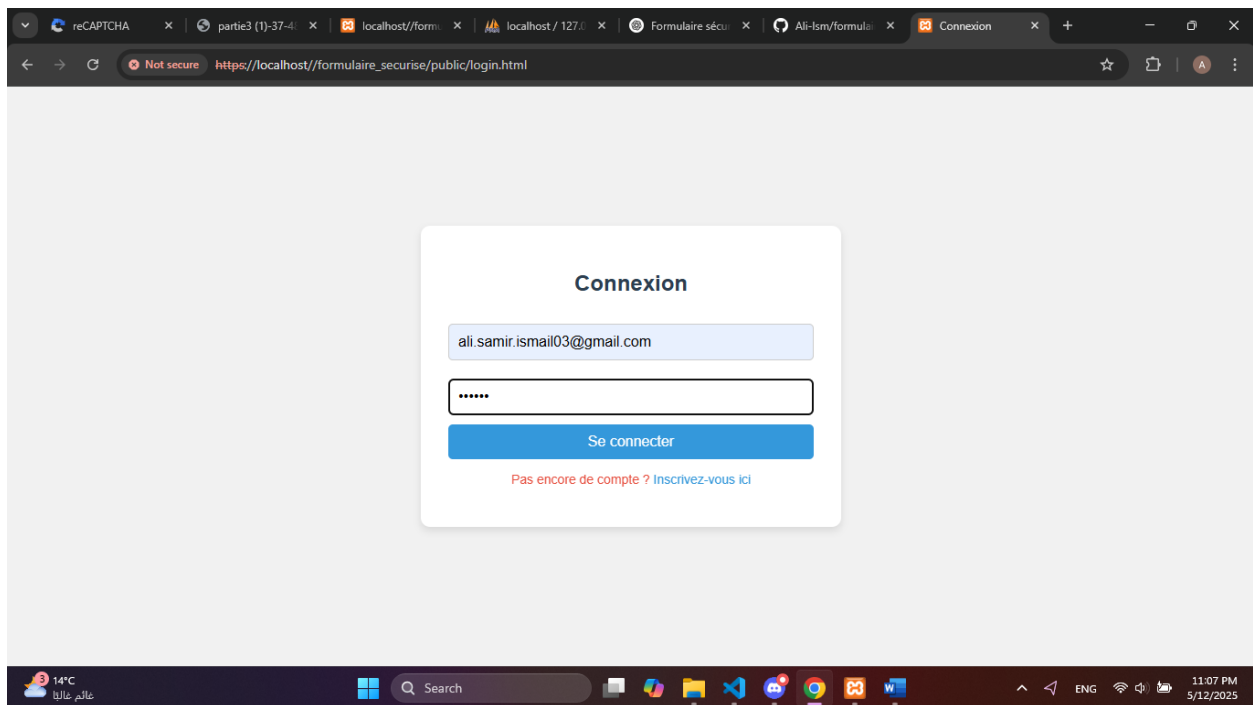
Introduction

Madame,

Je vous présente ce rapport concernant l'avancement de mon projet de **formulaire sécurisé**. Ce projet a pour but de sécuriser les entrées utilisateurs via un formulaire web en appliquant des mécanismes de protection contre les attaques courantes telles que l'injection SQL et les attaques XSS.

Je vais détailler les étapes réalisées ainsi que les tests effectués sur le formulaire sécurisé.

LOGIN :



Le formulaire de connexion présenté dans ce projet est une partie essentielle de l'application permettant aux utilisateurs de se connecter de manière sécurisée à leur compte. Le formulaire est conçu avec HTML et CSS pour une interface simple et intuitive. Le backend est implémenté en PHP et assure une validation sécurisée des utilisateurs à

l'aide de mécanismes de protection contre les injections SQL et la gestion des mots de passe hachés.

Structure de la Page

La page de connexion est constituée de plusieurs éléments :

- **HTML :** La page HTML comprend un formulaire avec deux champs principaux : l'email et le mot de passe. Ces champs sont validés avant soumission et sont marqués comme required pour éviter les soumissions vides.
- **CSS :** Le style est simple, moderne et assure une bonne expérience utilisateur. Le formulaire est centré et dispose d'un bouton de soumission visuellement attrayant.
- **Formulaire de soumission :** Le formulaire envoie les données via la méthode POST à un script PHP (login_process.php) qui va traiter la connexion.

Processus de Connexion (PHP)

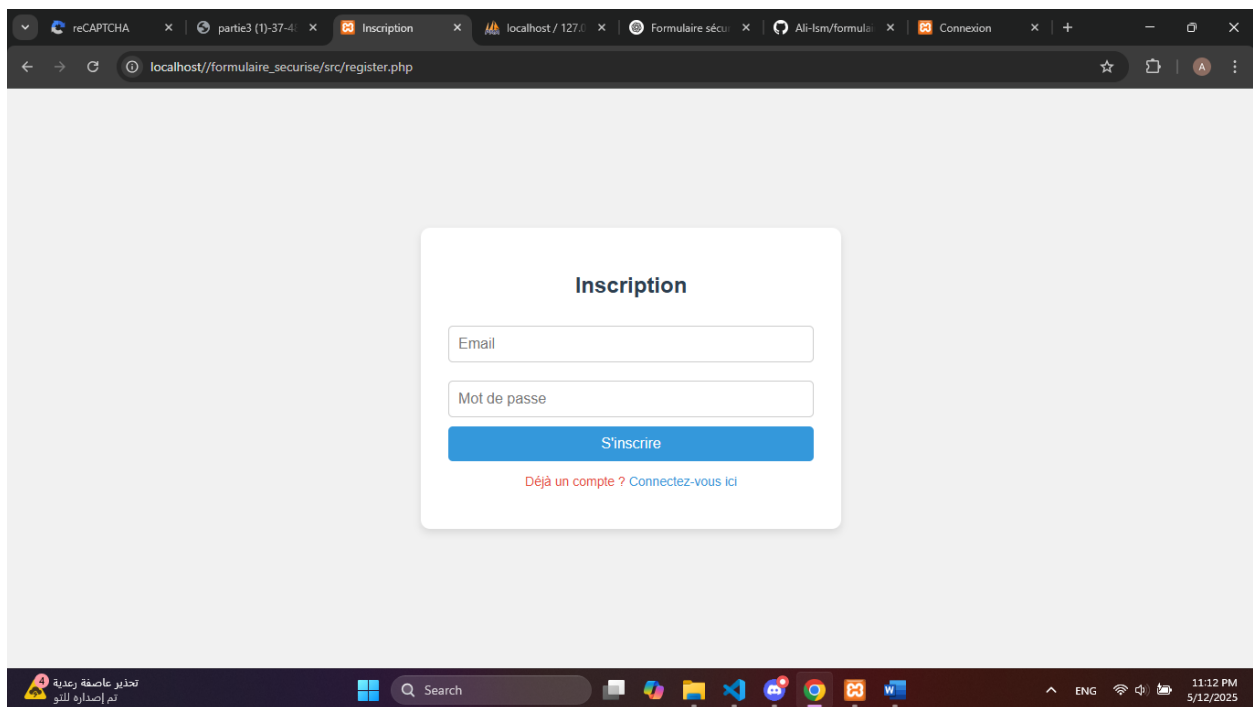
Le fichier login_process.php gère la logique de la connexion en plusieurs étapes :

- **Validation de la méthode HTTP :** Le script commence par vérifier que la méthode de la requête est POST, garantissant que le formulaire est bien soumis.
- **Sanitisation des Entrées :** L'email est nettoyé de tout espace inutile avec trim(). Cependant, une meilleure sécurisation pourrait être ajoutée pour échapper les caractères spéciaux.
- **Requête SQL :** Une requête préparée est utilisée pour récupérer les données de l'utilisateur depuis la base de données, en cherchant un utilisateur avec l'email fourni. Cette approche protège contre les attaques par injection SQL.
- **Vérification du Mot de Passe :** Le mot de passe saisi est comparé à celui stocké en base de données à l'aide de password_verify(), qui garantit une comparaison sécurisée des mots de passe hachés.
- **Gestion des Logs :** Chaque tentative de connexion (réussie ou échouée) est enregistrée dans un fichier log (login_attempts.log). Cela permet de surveiller et de détecter des activités suspectes telles que des tentatives de connexion par force brute.

Sécurisation et Bonnes Pratiques

- **Protection contre les Injections SQL :** En utilisant des requêtes préparées et des paramètres liés, le code empêche les injections SQL, ce qui est essentiel pour garantir la sécurité de la base de données.
- **Gestion des Mots de Passe Hachés :** Les mots de passe sont stockés sous forme de hachage (avec password_verify), ce qui évite de stocker les mots de passe en clair et assure une sécurité renforcée.
- **Enregistrement des Tentatives de Connexion :** Le log des tentatives de connexion permet une surveillance continue du système, aidant à identifier rapidement les attaques potentielles.

REGISTER :



The screenshot shows a web browser window with multiple tabs. The active tab is titled 'Inscription' and the address bar shows the URL 'localhost/formulaire_securise/src/register.php'. The page content features a white registration form with the title 'Inscription' in bold. The form contains two input fields: 'Email' and 'Mot de passe'. Below these fields is a blue button labeled 'S'inscrire'. At the bottom of the form, there is a red link that says 'Déjà un compte ? Connectez-vous ici'. The browser's taskbar at the bottom shows various application icons and the system clock indicating 11:12 PM on 5/12/2025.

La page d'inscription fait partie intégrante du projet de formulaire sécurisé et permet aux utilisateurs de créer un nouveau compte pour accéder à l'application. Cette page est conçue en HTML et CSS pour une présentation soignée et accessible. Le formulaire est

destiné à recueillir l'adresse email et le mot de passe de l'utilisateur afin de l'enregistrer dans la base de données.

Structure de la Page

La page d'inscription est constituée des éléments suivants :

- **HTML** : La page présente un formulaire avec deux champs principaux : l'email et le mot de passe. Ces champs sont essentiels pour créer un compte utilisateur. Les champs sont validés côté client avec l'attribut `required`, ce qui garantit que l'utilisateur ne soumettra pas de données vides.
- **CSS** : La page utilise un style simple et épuré pour améliorer l'expérience utilisateur. Le formulaire est centré sur la page et comprend un bouton de soumission visuellement attractif.
- **Formulaire de soumission** : Le formulaire utilise la méthode POST pour envoyer les données à un script PHP pour le traitement des inscriptions (ce code PHP n'est pas inclus dans l'exemple, mais il doit gérer la création de l'utilisateur).

Fonctionnement du Formulaire

Le formulaire est conçu pour collecter deux informations essentielles :

1. **Email** : Ce champ recueille l'adresse email de l'utilisateur. Il est important que l'email soit unique dans la base de données pour éviter les doublons lors de l'inscription.
2. **Mot de Passe** : Le mot de passe est un champ crucial pour la sécurité du compte utilisateur. Ce mot de passe sera haché et enregistré dans la base de données pour garantir que l'utilisateur ne puisse pas le récupérer en clair.

Le bouton "S'inscrire" soumet le formulaire, mais la logique côté serveur n'est pas incluse dans ce code. Le script PHP (qui devrait être créé séparément) devra gérer l'inscription en effectuant les étapes suivantes :

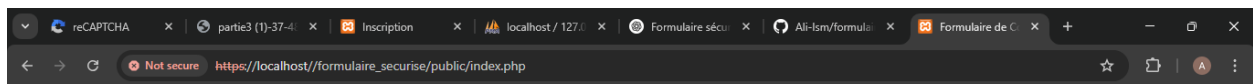
- Vérification que l'email n'est pas déjà utilisé.
- Hachage du mot de passe avec une fonction sécurisée comme `password_hash()`.
- Enregistrement des informations de l'utilisateur dans la base de données.

Sécurisation et Bonnes Pratiques

- **Validation côté client :** Les champs email et password sont obligatoires, ce qui assure une validation de base côté client avant soumission.
- **Gestion sécurisée du mot de passe :** Bien que le formulaire envoie un mot de passe en clair, il est recommandé de le hacher côté serveur avant de l'enregistrer dans la base de données. Le hachage des mots de passe avec des fonctions comme `password_hash()` permet de protéger les données sensibles des utilisateurs.
- **Lien vers la connexion :** La page inclut un lien qui redirige l'utilisateur vers la page de connexion s'il a déjà un compte, ce qui facilite la navigation dans l'application.

Le formulaire d'inscription est fonctionnel et constitue une étape fondamentale pour permettre aux utilisateurs de s'inscrire et de commencer à utiliser l'application.

INDEX.PHP




Contactez-nous

Nom :

Email :

Message :

☐ I'm not a robot 

Envoyer



Résumé du Fonctionnement

Le formulaire permet à un utilisateur de remplir des informations de contact, telles que :

- Nom

- Email
- Message
- reCAPTCHA (pour valider qu'il ne s'agit pas d'un bot)

Les données sont envoyées à un fichier PHP (traitement.php) via la méthode POST lorsque le formulaire est soumis, **après avoir été validées côté client avec JavaScript**. La validation garantit que :

- Le nom a au moins 2 caractères.
- L'email a un format valide.
- Le message contient au moins 10 caractères.
- Le reCAPTCHA est validé.

Si tout est valide, les données sont envoyées au fichier traitement.php. Sinon, des messages d'erreur sont affichés sous les champs concernés.

Points à Vérifier

1. Validation Côté Client

- Le script JavaScript effectue une validation côté client pour vérifier les entrées de l'utilisateur avant de soumettre le formulaire. Cela améliore l'expérience utilisateur en évitant des requêtes inutiles au serveur lorsque les champs sont incorrects.
- Cependant, la validation côté client peut être contournée si un utilisateur désactive JavaScript dans son navigateur.

2. Protection avec reCAPTCHA

- Le formulaire inclut un reCAPTCHA de Google pour s'assurer que les soumissions proviennent de vrais utilisateurs et non de robots.

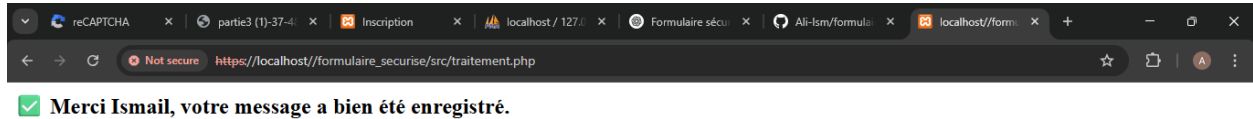
3. Accessibilité

- Il serait utile de s'assurer que chaque champ est bien associé à une étiquette label avec l'attribut for correspondant à l'id de chaque champ de saisie. Cela est déjà pris en compte dans votre code.

.

Conclusion

Ce formulaire de contact avec validation JavaScript et reCAPTCHA offre une bonne expérience utilisateur.



Traitement .php :

Le fichier traitement.php reçoit les données soumises par le formulaire de contact (nom, email, message et reCAPTCHA) via la méthode POST. Il commence par valider la réponse du reCAPTCHA en envoyant une requête à l'API de Google pour vérifier que l'utilisateur est bien humain. Ensuite, il effectue des vérifications supplémentaires sur les données soumises, comme la validité de l'email et la présence d'un message. Si toutes les données sont valides, elles sont donc enregistrées dans une base de données « contact_form » dans le tableau de message. En cas d'erreur, il renvoie un message approprié à l'utilisateur pour l'informer des problèmes rencontrés.

Pour passer de HTTP à HTTPS, j'ai simplement modifié les configurations de mon serveur Apache. J'ai commencé par activer le module SSL dans le fichier de configuration httpd.conf. Ensuite, j'ai configuré un certificat SSL que j'avais déjà pour mon site local. J'ai précisé les chemins d'accès à la clé privée et au certificat SSL dans le fichier de configuration d'Apache. Ensuite, j'ai ajouté une redirection dans le fichier httpd.conf pour forcer toutes les connexions HTTP à être redirigées vers HTTPS. Cela permet de sécuriser les échanges entre le serveur et les utilisateurs en cryptant les données transmises. Après

avoir effectué ces modifications, j'ai redémarré Apache pour appliquer les changements et activer la configuration HTTPS.

Concernant la sécurité contre les attaques XSS et SQL injection, j'ai pris plusieurs mesures pour m'assurer que l'application soit protégée. J'ai testé les vulnérabilités potentielles en utilisant Postman, notamment pour vérifier les injections SQL. Pour cela, j'ai essayé d'injecter des commandes SQL malveillantes via l'API, mais le système a bien réagi et m'a renvoyé un message du type "Email not found in database" lorsque l'email n'existait pas, ce qui montre que l'injection a échoué grâce à l'utilisation de requêtes préparées et d'une validation stricte des données.

Conclusion :

En conclusion, ce projet m'a permis de mieux comprendre l'importance de la sécurité dans les applications web, en particulier en ce qui concerne les vulnérabilités telles que les injections SQL et les attaques XSS. J'ai appris à mettre en place des mesures de protection efficaces pour garantir la sécurité des utilisateurs et des données sensibles.

L'implémentation de la sécurité via HTTPS, la validation et l'assainissement des entrées, ainsi que l'utilisation de techniques de protection comme les cookies sécurisés ont été des étapes clés pour renforcer la fiabilité et la sécurité de l'application. Grâce à ce projet, j'ai pu améliorer mes compétences en développement web et en gestion des vulnérabilités, tout en garantissant une expérience utilisateur optimale et sécurisée.

Je tiens à remercier chaleureusement mon professeur pour son accompagnement et ses conseils tout au long de ce projet. Grâce à son expertise et à ses recommandations, j'ai pu approfondir mes connaissances en développement web sécurisé et je me sens mieux préparé pour aborder des défis similaires à l'avenir. Merci beaucoup pour votre soutien et vos enseignements précieux !