Fonctionnalités de sécurité mis en place

1- Authentification – Permissions - Chiffrement

- **De quoi et comment la fonctionnalité protège notre code ?**

Pour avoir accès à notre application il faut s'authentifier pour garantir la légitimité d'accès au système. L'authentification est donc mise en place pour empêcher ceux qui ne sont pas autorisés à avoir accès au système. Un nom d'utilisateur et un mot de passe sont requis à cet effet.

Les mots de passe sont stockés après être passés par une fonction de Hashage gérée par Django permettant le chiffrement des mots de passe.

Nous avons par la même sécurisé nos urls pour empêcher le contournement du système d'authentification et de permissions

L'administration de Django nous a permis d'implémenter ces fonctionnalités !

L'implémentation de cette sécurité nous a permis d'établir les **principes de moindre confiance** et les **principes de moindre privilège**

- **Test**

Au lancement de l'application on peut tester la fonctionnalité en essayant de contourner l'authentification en saisissant directement les urls.

On peut essayer également de passer le système d'authentification en fournissant de mauvais utilisateurs et ou mot de passe, ce qui ramènera sur la page d'authentification

2- ORM

- **De quoi et comment la fonctionnalité protège notre code ?**

Pour la gestion des données de notre application, nous n'avons pas utilisé les requêtes SQL mais l'**ORM** fourni par Django.

Ceci nous permet d'éviter les attaques de types **injections SQL** étant donné que nous n'exécutons pas directement du SQL qui peut laisser une faille exploitable pour modifier les données dans notre base de données.

L'utilisation des ORM nous a permis de minimiser notre surface d'attaque au niveau de la base de données !

- **Test**

Pour tester il suffit d'injecter du code sql dans les champs de saisie ce qui n'aura pas d'impact sur notre base de données.

3- Le langage de gabarit

- **De quoi et comment la fonctionnalité protège notre code ?**

Les gabarits nous permettent de nous protéger des attaques **XSS.** Les gabarits permettent d'éviter l'injection de scripts javascripts dans le client.

Les structures telles que les variables, les balises, les filtres et les commentaires qu'offrent les gabarits permettent soit d'afficher une variable, d'appliquer une logique dans le processus de rendu etc.

Les gabarits permettent donc **d'échapper les caractères spécifiques** qui sont dangereux en HTML. Les gabarits permettent donc de minimiser la **surface d'attaque au niveau du client.**

4- La protection CSRF

- **De quoi et comment la fonctionnalité protège notre code ?**

Le **CSRF middleware** et le **template tag CSRF** nous ont permis de mettre en place une protection contre les attaques de type Cross Site.

Le template tag **csrf_token** nous permet de protéger nos formulaires.

La protection **CSRF** fonctionne en contrôlant un jeton dan chaque requête POST. Cela garantit qu'un utilisateur ne peut « rejouer » un envoi de formulaire POST tout en faisant soumettre ce formulaire de manière involontaire par un autre utilisateur connecté !

Cette protection implémente comme principe de sécurité **la minimisation de la surface d'attaque.**

5- Protection de PostgreSQL

L'accès a notre base de données est conditionnée par une authentification.

Un login et un mot de passe sont requis comme pour l'accès à notre application.

Ce mécanisme de sécurité met en exergue les principes de sécurité de **moindre confiance**, de **moindre privilège.**