Sécurité de mon application

Pour protéger mon application contre différentes attaques (XSS, injection SQL, etc.), j'ai mis en place plusieurs mesures de sécurité :

1. Protection contre les attaques XSS (Cross-Site Scripting)

- **Validation et assainissement des entrées utilisateur** :

  - Utilisation de **Formik** et **Yup** pour valider et restreindre les données saisies (ex. empêcher les balises <script>).

  - Échappement des données affichées dans l'interface utilisateur pour éviter l'exécution de code malveillant.

2. Protection contre l'injection SQL

- Utilisation d'un ORM sécurisé qui empêche l'exécution de requêtes SQL directes.

3. Sécurisation des Tokens et de l'Authentification

- **Utilisation de JWT (JSON Web Token) sécurisé :**

  - Signature avec un algorithme sécurisé (**HS256 ou RS256**).

  - Expiration et renouvellement des tokens pour limiter leur réutilisation.

- **Gestion des rôles et permissions** :

  - Mise en place d'un **contrôle d'accès basé sur les rôles (RBAC)** pour restreindre l'accès aux fonctionnalités sensibles.

  - Vérification des permissions côté backend et frontend.

4. Une mesure de sécurité contre les requêtes provenant de sources non autorisées :

En utilisant **CORS** (Cross-Origin Resource Sharing). CORS est une politique de sécurité qui contrôle les ressources accessibles par des domaines externes (origines), ce qui est important pour éviter les attaques par **Cross-Site Request Forgery (CSRF)** ou autres accès indésirables.

- **Vérification de l'origine des requêtes** : Le tableau allowedOrigins contient une liste d'origines autorisées. Dans ce cas, c'est https://zoo-rouge.vercel.app. Si une requête provient de cette origine, elle est acceptée. Si l'origine de la requête n'est pas présente dans cette liste, elle sera rejetée.
- **Validation dynamique de l'origine** : La fonction de rappel (callback) permet de vérifier dynamiquement l'origine de chaque requête et de la comparer à la liste d'origines autorisées. Si l'origine est valide (présente dans allowedOrigins), la requête

est autorisée. Sinon, une erreur est renvoyée, ce qui empêche l'accès aux ressources du serveur.

- **Gestion des erreurs CORS** : Si l'origine de la requête n'est pas autorisée, le serveur répond par une erreur "Not allowed by CORS", ce qui empêche l'accès aux ressources sensibles du serveur.