Presentation projet 10

Présentation des livrables

a. Points de terminaison (Endpoints)

- **Admin Interface** :
- Méthode : Toutes
- URL : /admin/
- **API Auth** :
- Méthode : Toutes
- URL : /api-auth/
- **Inscription des Utilisateurs** :
- Méthode : GET, POST
- URL : /signup/
- **Login** :
- Méthode : POST
- URL : /login/
- Objectif : Permet à l'utilisateur de s'authentifier et d'obtenir un token d'accès.
- **Rafraîchissement du Token** :
- Méthode : POST
- URL : /api/token/refresh/
- Objectif : Permet de rafraîchir le token d'accès de l'utilisateur.
- **Projets** :
- Méthode : GET, POST, PUT, DELETE
- URL : /projects/
- Objectif : Gérer les projets. Les utilisateurs peuvent créer, récupérer, mettre à jour et supprimer des projets.
- **Problèmes (Issues)** :
- Méthode : GET, POST, PUT, DELETE
- URL : /issues/
- Objectif : Gérer les problèmes relatifs à un projet.
- **Contributeurs** :
- Méthode : GET, POST, PUT, DELETE
- URL : /contributor/
- Objectif : Gérer les contributeurs d'un projet.
- **Commentaires** :
- Méthode : GET, POST, PUT, DELETE
- URL : /comments/
- Objectif : Gérer les commentaires sur les problèmes.
-
- **Accès aux données** :
- Méthode : GET
- URL : /user_data/
- Objectif : Permet à l'utilisateur de visualiser toutes les données que vous détenez à son sujet.

- **Portabilité des données** :
- Méthode : GET
- URL : /user_data/export_data/
- Objectif : Fournir à l'utilisateur un moyen d'exporter ses données dans un format lisible, comme JSON.
- **Droit à l'oubli** :
- Méthode : DELETE
- URL : /user_data/forget_me/
- Objectif : Permettre à l'utilisateur de supprimer son compte et toutes les données associées.
-

Il est important de noter que pour chaque point de terminaison, des permissions appropriées sont appliquées pour garantir que seuls les utilisateurs autorisés peuvent effectuer certaines actions. Par exemple, seuls les administrateurs ou les propriétaires d'un projet peuvent le supprimer, tandis que tous les utilisateurs peuvent le visualiser.

b. Sécurité OWASP
La sécurité est primordiale dans toute application web. Pour garantir une application sécurisée, nous avons suivi les directives de l'OWASP (Open Web Application Security Project) :

b. Sécurité OWASP
L'OWASP (Open Web Application Security Project) identifie les dix principales vulnérabilités des applications web. Voici comment le code du projet traite chacune d'elles :

- **Injection** :
- Grâce à l'utilisation de l'ORM de Django, les requêtes à la base de données sont paramétrées, ce qui protège contre les injections SQL.
- **Authentification brisée** :
- Nous utilisons rest_framework_simplejwt pour gérer l'authentification. Il fournit une authentification sécurisée avec des tokens JWT.
- **Exposition de données sensibles** :
- Les mots de passe sont hashés avant d'être stockés. Django utilise l'algorithme PBKDF2 par défaut pour hasher les mots de passe.
- Les clés secrètes et les tokens sont stockés en toute sécurité.
- **Contrôle d'accès brisé** :
- Les permissions sont définies à l'aide de Django et du DRF (Django Rest Framework). Seuls les utilisateurs ayant les bonnes permissions peuvent effectuer certaines actions.

(À ce stade, j'examinerai le fichier permissions.py pour obtenir des détails sur la manière dont les permissions sont gérées.)

- **Misconfiguration de la sécurité** :
- Le fichier settings.py est conçu pour garantir que seules les configurations sécurisées sont utilisées. Par exemple, le mode DEBUG est désactivé en production, et l'application est configurée pour utiliser HTTPS.
- **Cross-Site Scripting (XSS)** :

- Django échappe automatiquement le contenu pour éviter les attaques XSS. De plus, l'utilisation d'un framework front-end moderne peut également aider à prévenir les attaques XSS.
- **Insecurities in Deserialization** :
- Le DRF utilise des sérialiseurs pour garantir une désérialisation sécurisée des données.

(À ce stade, j'examinerai le fichier serializers.py pour obtenir des détails sur la manière dont la sérialisation et la désérialisation sont gérées.)

- **Utilisation de composants avec des vulnérabilités connues** :
- Il est important de maintenir toutes les dépendances à jour pour éviter d'utiliser des composants obsolètes ou vulnérables.
- **Logging insuffisant et surveillance** :
- Django fournit des mécanismes de journalisation qui peuvent être configurés pour capturer et signaler des événements importants.
- **Cross-Site Request Forgery (CSRF)** :
- Django possède une protection intégrée contre les attaques CSRF. Chaque formulaire généré par Django contient un token CSRF pour prévenir ce type d'attaque.

c. Conformité RGPD

Le RGPD (Règlement général sur la protection des données) est une réglementation de l'UE qui vise à donner aux citoyens un meilleur contrôle sur leurs données personnelles. Voici comment le projet s'assure de sa conformité :

- **Consentement de l'utilisateur** :
- Avant de collecter des données, le consentement de l'utilisateur est obtenu. Cela est généralement réalisé via un formulaire d'inscription ou une fenêtre contextuelle lors de la première visite de l'utilisateur.
- **Droit à l'oubli** :
- Les utilisateurs peuvent supprimer leur compte à tout moment, ce qui supprimera également toutes leurs données associées.
- **Portabilité des données** :
- Les utilisateurs peuvent demander une copie de toutes leurs données. Grâce aux sérialiseurs du DRF, il est possible d'exporter ces données dans un format lisible, comme JSON.
- **Minimisation des données** :
- Seules les données nécessaires à la fourniture du service sont collectées. Par exemple, si l'application n'a pas besoin de l'adresse de l'utilisateur, elle ne la collecte pas.
- **Accès aux données** :
- Les utilisateurs peuvent accéder à leurs données à tout moment et les mettre à jour si nécessaire.
- **Sécurité des données** :
- Les données sont stockées en toute sécurité, et des mesures appropriées sont prises pour prévenir les violations de données.

d. Philosophie « Green Code »

Le développement de logiciels éco-responsables ou « Green Code » vise à minimiser l'impact environnemental du code. Voici comment votre projet s'aligne sur cette philosophie :

- **Optimisation des requêtes** :
- En utilisant l'ORM de Django, les requêtes à la base de données sont optimisées pour éviter les appels inutiles. De plus, en utilisant des fonctions comme select_related ou prefetch_related, on peut réduire le nombre de requêtes nécessaires pour récupérer des données liées.
- La pagination est souvent utilisée dans les applications DRF pour limiter le nombre d'objets renvoyés dans une seule requête, réduisant ainsi la charge sur la base de données et le réseau.
- **Écriture de code propre** :
- Un code propre est plus facile à comprendre, à maintenir et à exécuter. En suivant les principes du développement piloté par les tests (TDD) et en s'assurant que le code est bien commenté et structuré, le code peut être exécuté plus efficacement.
- En évitant les calculs redondants et en utilisant des techniques de mise en cache, le code peut être exécuté plus rapidement, économisant ainsi de l'énergie.
- **Tests** :
- Les tests garantissent que le code fonctionne comme prévu. En identifiant et en corrigeant les erreurs tôt dans le processus de développement, on peut éviter les calculs inutiles et les opérations redondantes qui gaspilleraient de l'énergie.
- Les tests automatisés, comme ceux trouvés dans le fichier tests.py, garantissent que le code continue de fonctionner correctement à mesure que des modifications sont apportées.
- **Réduction de la complexité** :
- Un code moins complexe est généralement plus rapide à exécuter. En utilisant des algorithmes efficaces et en évitant la complexité inutile, le code peut être exécuté en utilisant moins de ressources.
- **Utilisation efficace des ressources** :
- En évitant de stocker des données inutiles ou redondantes et en s'assurant que les ressources sont libérées après utilisation (comme la fermeture des connexions à la base de données), le code peut fonctionner de manière plus éco-responsable.

La philosophie du « Green Code » est un aspect essentiel du développement logiciel moderne. Non seulement elle garantit que le code est efficace, mais elle contribue également à réduire l'impact environnemental du secteur technologique