Documentation Technique du Projet "Jeux Olympiques"

Introduction

Ce document décrit les choix techniques, les aspects de sécurité et les perspectives d'évolution du projet "Jeux Olympiques". Le projet est basé sur une architecture client-serveur utilisant Django pour le backend et React pour le frontend.

**Maquette/Illustration/Logo/Image**

• Figma

• Diagrams.net

• Canva.com

**Environnement de travail**

• Mac os (Système d’exploitation)

• Datagrip (Ide pour base de données)

• VsCode (éditeur de texte)

**Architecture Générale**

Backend - Django

**Choix de Django et Django REST Framework (DRF)**

* Justification : Django est choisi pour sa robustesse, sa rapidité de développement et ses fonctionnalités de sécurité intégrées. DRF facilite la création d'API RESTful nécessaires pour la communication avec le frontend React.

**Modèle de Données Personnalisé (CustomUser)**

* Justification : Le modèle utilisateur personnalisé permet d'adapter les champs aux besoins spécifiques (rôles des utilisateurs, informations supplémentaires).

**JWT pour l'Authentification**

* Justification : Les JSON Web Tokens (JWT) permettent une authentification stateless et sécurisée, adaptée aux applications SPA comme notre frontend React.

**Frontend - React**

Choix de React

* Justification : React est utilisé pour créer des interfaces utilisateur dynamiques et performantes. Sa gestion efficace du DOM virtuel assure une expérience utilisateur fluide.

**Redux pour la Gestion de l'État**

* Justification : Redux centralise et synchronise l'état global de l'application, essentiel pour gérer les interactions utilisateur et les données (panier d'achat, profils utilisateurs).

**Styled-components pour le Styling**

* Justification : Styled-components permettent une meilleure modularité et maintenabilité du code CSS, favorisant le développement de composants réutilisables et isolés.

**Sécurité**

**Authentification et Autorisation**

* JWT : Utilisation de JWT pour sécuriser les communications entre le client et le serveur.
* Django Permissions : Utilisation des permissions de Django pour contrôler l'accès aux différentes ressources de l'API.

**Protection des Données**

* Hachage des Mots de Passe : Les mots de passe des utilisateurs sont hachés avant d'être stockés dans la base de données.
* HTTPS : Recommandation de l'utilisation de HTTPS pour chiffrer les données en transit.

**Prévention des Vulnérabilités**

* Protection CSRF : Mise en place de protections contre les attaques CSRF.
* Validation des Données : Validation stricte des données entrantes pour prévenir les injections SQL et autres attaques.

**Évolutions Futures**

**Fonctionnalités**

* Notifications en Temps Réel : Utilisation de WebSockets pour les notifications en temps réel concernant les mises à jour des événements sportifs.
* Analyse des Données : Intégration d'outils d'analyse pour fournir des insights sur les préférences des utilisateurs et les tendances des achats de billets.

**Scalabilité**

* Mise en Cache : Utilisation de systèmes de cache (comme Redis) pour améliorer les performances.
* Microservices : Transition vers une architecture de microservices pour faciliter la scalabilité et la maintenance du code.

**Accessibilité et UX**

* Amélioration de l'Accessibilité : Mise en œuvre des meilleures pratiques pour rendre l'application accessible à tous les utilisateurs.
* Optimisation Mobile : Optimisation de l'application pour une meilleure expérience utilisateur sur les appareils mobiles.