Documentation du Projet de Gestion d'Événements Sportifs JO 2024

Table des matières

1. Introduction
2. Architecture du projet
3. Fonctionnalités principales
4. Points de terminaison API
5. Installation et configuration
6. Dépannage

Introduction

Le projet JO Tickets est une solution complète de gestion de billets pour les Jeux Olympiques de Paris 2024. Il comprend :

- Un projet Django avec une API backend
- Une application mobile pour les supporters
- Une application de scan de billets pour les stadiers

Architecture du projet

Structure des dossiers

```
ProjetJoTicket/
├── admin/     # Projet Django principal
├── mobile/    # Application mobile pour les supporters
├── scanner/   # Application de scan de billets
```

Composants principaux

- **Backend Django** :
  - Gestion de la base de données
  - API RESTful
  - Interface d'administration
- **Application mobile** : Frontend pour les supporters
- **Application scanner** : Vérification des billets

Fonctionnalités principales

Pour les utilisateurs

- Inscription et connexion
- Consultation des événements sportifs

- Achat de billets
- Consultation et téléchargement des billets
- Génération de QR codes pour chaque billet

Pour les administrateurs

- Gestion complète des événements
- Mise à jour des informations des événements
- Interface d'administration sécurisée

Points de terminaison API

Authentification

URL	Méthode	Description	Authentification requise
`/api/register`	POST	Inscription d'un nouvel utilisateur	Non
`/api/login`	POST	Connexion utilisateur	Non
`/api/logout`	GET	Déconnexion utilisateur	Oui
`/api/loginAdmin`	POST	Connexion administrateur	Non
`/logout/`	GET	Déconnexion administrateur	Oui

Ressources

URL	Méthode	Description	Authentification requise
`/api/stadiums`	GET	Liste de tous les stades	Non
`/api/teams`	GET	Liste de toutes les équipes	Non
`/api/events`	GET	Liste de tous les événements	Non
`/api/events/<eventId>`	GET	Détails d'un événement spécifique	Non

Billets

URL	Méthode	Description	Authentification requise
`/api/buyTicket`	POST	Achat de billets	Oui
`/api/tickets/<userId>`	GET	Liste des billets d'un utilisateur	Oui*
`/api/ticket/<ticketid>`	GET	Détails d'un billet spécifique	Oui*

Administration

URL	Méthode	Description	Authentification requise

URL	Méthode	Description	Authentification requise
`/api/manageEvent`	GET	Gestion des événements	Oui (admin)
`/api/updateField/<event_id>/<field_name>`	POST	Mise à jour d'un champ d'événement	Oui (admin)

*Bien que l'authentification ne soit pas explicitement requise, les données sont spécifiques à un utilisateur.

Installation et configuration

Prérequis

- Python
- Django
- django-cors-headers

Étapes d'installation

1. Cloner le dépôt du projet
2. Créer un environnement virtuel
3. Installer les dépendances : `pip install -r requirements.txt`
4. Installer django-cors-headers : `pip install django-cors-headers`
5. Configurer la base de données
6. Importer les données SQL : `data_jo.sql`
7. Lancer le serveur : `python manage.py runserver`

Configuration CORS

Pour permettre les requêtes cross-origin, configurer `settings.py` :

```
INSTALLED_APPS = [
    'corsheaders',
    # autres applications
]

MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
    # autres middlewares
]

CORS_ALLOW_ALL_ORIGINS = True
CORS_ALLOW_CREDENTIALS = True
CSRF_TRUSTED_ORIGINS = [
    "http://127.0.0.1:5500"
]
ALLOWED_HOSTS = ['127.0.0.1']
```

Authentification et API

Types de requêtes

- Inscription : POST `/api/register`
- Connexion : POST `/api/login`
- Déconnexion : GET `/api/logout`
- Achat de billets : POST `/api/buyTicket`

Gestion des requêtes POST

Utiliser `fetch()` avec les en-têtes appropriés :

```
fetch(apiPath, {
    method: "POST",
    credentials: "include",
    headers: {
        "Content-type": "application/json",
        "X-CSRFToken": getCookie("csrftoken"),
    },
    body: JSON.stringify(body),
});
```

Catégories de billets

- Silver : 100€
- Gold : 200€
- Platinium : 300€

Dépannage

Problèmes courants

- Vérifier les identifiants de connexion
- S'assurer que le serveur Django tourne
- Vérifier les configurations CORS
- Examiner les logs du serveur

Sécurité

- Seuls les superutilisateurs peuvent accéder à l'interface admin
- Authentification requise pour certaines actions
- Gestion des tokens CSRF

Notes supplémentaires

- Utiliser la police Paris2024.ttf pour le design

- Tester l'application sur un serveur HTTPS pour l'accès caméra