Documentation technique

Application Jeux Olympiques 2024

Sommaire

➔ Présentation du projet

- **Type de projet :** application web de type “billetterie” pour les jeux olympiques 2024.
- **Liens du projet :**
  - **Front :** https://jeuxolympiques.gregoryfulgueiras.com
    - **GitHub** : https://github.com/Y-rog/jeuxolympiques.git
  - **API :** https:/api-jeuxolympiques.gregoryfulgueiras.com
    - **Swagger :** https://api-jeuxolympiques.gregoryfulgueiras.com/swagger-ui/index.html
    - **GitHub** : https://github.com/Y-rog/api-jeuxolympiques.git
- **Développeur :** Société InfoEvent
- **Nombre de développeur :** 1
- **Description :** Le but de ce projet est de créer une billetterie en ligne pour remplacer les billets physiques pars des billets électronique avec QrCode. L’accent doit être mis sur la sécurité ainsi si que la scalabilité pour gérer un trafic important.

➔ Stack technique

Utilisation de technologies et frameworks fiables et appropriés, ayant fait leur preuves.

- **Front-end :** Angular 19.1.5
- **Back-end :** SpringBoot 3.4.3 avec Java openjdk 21.0.5 LTS
- **BDD :** PostgreSQL 17.4
- **Déploiement :** Heroku avec addon Heroku Postgres et addon CloudAMQP
- **Hébergement du code :** GitHub
- **IDE :**
  - Angular avec VsCode
  - SpringBoot avec IntelliJ

➔ Sécurité

Pour ce projet l'accent est mis sur la sécurité, j'ai utilisé :

- **Choix des frameworks :** Je me suis appuyé sur des frameworks orientés sécurité.
  - **Front :** Angular protège automatiquement contre la majorité les failles XSS
  - **Back :** Spring Boot avec notamment Spring Security et JPA
- **Spring Security :** Permet la configuration de la sécurité de notre API. Sécurisation et restriction des différentes routes de l'API en fonction de rôle. Configuration des CORS pour n'autoriser que notre Front-end à accéder à l'API pour éviter les failles CSRF, XSS ou d'injection.
- **JWT Token :** Il stocke les infos utilisateur "username", "rôle", "date d'expiration", signé avec plusieurs algorithmes de sécurité HS256. Très utile dans notre cas, car il n'a pas besoin d'être stocké côté serveur ou bdd, il améliore la scalabilité (fluidité du trafic).

  J'ai choisi de le stocker en Session storage pour plus de sécurité, celui-ci est effacé lorsque l'onglet est fermé. C'est un peu moins bien pour l'expérience utilisateur mais meilleur pour la sécurité, cela protège des failles CSRF.
- **Spring Data JPA :** Protège des injections SQL
- **BCrypt :** Pour hacher et donc protéger les mots de passe en BDD.

➔ Sécurité (suite)

- **Validation des formulaires :** double validation
  - **Front :** J'ai utilisé le module @angular/forms pour valider les inputs de mes formulaires ex: Validators.required pour les champs requis.
  - **Back** : J'ai utilisé les annotations sur SpringBoot pour valider les formulaire ex: @Notblank, @Email ou @Pattern pour le regex du mot de passe, celui-ci doit contenir au moins 8 caractères, une majuscule, une minuscule et un caractère spécial
- **Certificat SSL/TLS :** pour chiffrer les échanges entre le front et le back.
- **Bonnes pratiques :** bien entendu, j'ai utilisé les bonnes pratiques par ex: pas de mot de passe stocké dans le token , utilisation de variables d'environnement, création d'un gitignore…

➔ Evolutions futures

- **Statistiques :** Différentes stats seraient intéressantes pour le client par ex: les ventes par événementsr.
- **Développement d'applications mobile :**
  - **Client :** Cela permettrait aux utilisateurs d'avoir accès à leurs billets très facilement sur mobile.
  - **Employés JO :** une application mobile pour scanner les QrCode des billets**.**
  - **Notifications :** exemple ajout de notifications à l'approche des événements
- **Gestion des SAV :** Par exemple pour des demandes de remboursements d'événements annulé.
- **Optimiser les performances de l'application :** Implémenter un système de cache pour les offres et événements les plus consultés ou populaires.
- **Authentification 2FA :** cette double authentification permet de sécuriser un compte ou le mot de passe aurait été volé, celle-ci se déroule en 2 étapes, l'utilisateur saisit ses identifiants puis par exemple un code temporaire est envoyé par mail ou généré par SMS et l'utilisateur doit saisir ce code pour s'authentifier. Cette double authentification sera développé dans un second temps.