Documentation Technique - Projet « Bloc 3 » exercice de réalisation d'un site de réservation de tickets pour les évènements des JO 2024

Surligné en jaune : dans le cadre de ce premier dépôt, je note ici ce que je compte faire mais non réalisé à cette date.

1. Introduction

Cette application réalisée dans le cadre d'un exercice doit permettre l'achat et la réservation de tickets pour les événements des Jeux Olympiques de Paris 2024.

Elle est développée en Python avec Django pour le backend, et utilise PostgreSQL pour la base de données.

2. Architecture de l'Application

- Backend : Framework Django, structuré en une application principale pour gérer les billets, les réservations et les utilisateurs.
- Base de données : PostgreSQL, utilisée pour stocker les informations liées aux utilisateurs, billets, événements et transactions.
- Frontend : Templates HTML avec Django Template Language (DTL), CSS pour le style et JavaScript pour les interactions dynamiques.

3. Modèles de Données

Les modèles suivants sont utilisés au sein de l'application Django et correspond aux données présentes sur la base de données.

- Profile : Contient les informations utilisateur (nom, prénom, adresse, etc…).
- TypeBillet : Représente les différents types d'offre de billets (solo, duo, famille), avec les champs pour le nom, prix et quantité. Il est possible pour l'administrateur de créer d'autre types d'offre. Chaque offre permet de créer des packs plus ou moins économiques comprenant un nombre variable de billets débloqué après l'achat.
- Billet : Représente chaque billet acheté, incluant les liens vers l'utilisateur, le type de billet et une clé de sécurité unique. Lors de l'achat d'une offre de type famille par exemple, 2 billets sont générés sur le compte utilisateur.

- Evenement : Contient les événements disponibles pour réservation. Contient notamment un champ pour préciser la date de l'évènement ainsi qu'un champ date limite de réservation, se remplissant automatiquement si laissé vide par l'administrateur et permettant d'interdire la réservation ou l'annulation de l'évènement à partir d'une certaine date (la veille de l'évènement pas défaut). Contient également un booléen « A la une » permettant la gestion des évènements qui seront affichés sur la page d'accueil du site.

- Reservation : Gère les réservations des billets pour les événements, ainsi que les QR code des tickets d'accès.

Un diagramme UML montrant les relations entre les différents modèles accompagne le projet dans son dépôt git.

4. Sécurité

Mesures de sécurité implémentées :

- Authentification des utilisateurs : Utilisation de Django User Model avec des mots de passe hashés.

- Clés de sécurité pour les billets : Chaque billet est associé à une clé unique, générée lors de l'achat.

- Protection des formulaires : Utilisation de tokens CSRF pour prévenir les attaques de type Cross-Site Request Forgery.

- Validation des entrées : Les données utilisateur sont strictement validées pour éviter les injections SQL.

5. Fonctionnalités Actuelles

- Achat de billets :
    - Les utilisateurs peuvent sélectionner et acheter des billets via une interface conviviale.
    - Calcul automatique du nombre total de billets et du montant à payer en fonction du type de billet sélectionné.

- Réservation d'événements :
    - Une fois les billets achetés, les utilisateurs peuvent réserver des événements, sous réserve de leur disponibilité.

- Gestion du profil utilisateur :

    - Les utilisateurs peuvent consulter et modifier leurs informations personnelles via la page « Mon compte ».
- Tests :
    - Développement de tests automatisés pour garantir la qualité du code.

6. Évolutions Futures

- Protocole de gestion des erreurs de sécurité :
    - Mise en place d'une stratégie de gestion des erreurs personnalisé.
    - Journalisation et la notification en cas de détection d'activités suspectes.

- Implémentation d'un système de paiement en ligne :
    - Intégration avec une API de paiement pour gérer les transactions financières en toute sécurité.
    - Authentification à deux facteurs (2FA) :

- Optimisation de la performance :
    - Amélioration de la performance de l'application, notamment en optimisant le code et les requêtes à la base de données.

- Traduction en plusieurs langues :
    - Ajout de la prise en charge de plusieurs langue étrangère pour répondre aux besoins de clients non francophones.

- Stratégie de sauvegarde :
    - Mise en place de solution de sauvegarde automatisée pour la base de données et les fichiers critiques.

7. Déploiement

- Hébergement sur un serveur sécurisé, avec HTTPS.

8. Documentation Utilisateur

Le fichier readme.md du projet permet la mise en route du projet et explique les différentes fonctionnalités de l'application.

9. Annexes utiles

Références : Documentation officielles (Django, PostgreSQL, …)

Journal des modifications : Historique des principales mises à jour du projet. (changelog.txt)