Documentation Technique et fonctionnelle de l’application

Introduction

L'application de gestion pour le parc d'attraction vise à offrir une plateforme robuste permettant aux administrateurs de contrôler efficacement les attractions disponibles et aux visiteurs de profiter d'une expérience immersive et interactive. Avec un système de gestion avancé, l'application répond aux besoins de suivi des attractions, à la collecte des critiques des visiteurs et à l'amélioration continue de l'expérience utilisateur. Destinée à un large public, l'application doit être intuitive et accessible, fournissant une interface conviviale tant pour les administrateurs que pour les visiteurs.

Architecture de l'application

L'architecture de l'application repose sur une approche client-serveur, avec Angular utilisé pour le développement du frontend et Python, avec Flask pour le backend. Cette structure permet une séparation claire des responsabilités entre le frontend et le backend, facilitant la maintenance et l'évolutivité de l'application. En utilisant Docker, l'application peut être facilement déployée sur différentes plateformes, offrant ainsi une flexibilité et une portabilité maximales. Une base de données relationnelle, telle que MySQL, est intégrée pour stocker les données critiques des attractions, des critiques et des utilisateurs, assurant ainsi une gestion efficace des données.

Fonctionnalités de l'application

- Gestion de l'état des attractions : Les administrateurs peuvent désormais contrôler l'état de visibilité des attractions, offrant ainsi une expérience plus ciblée et personnalisée aux visiteurs.

- Système de critiques pour les attractions : Les visiteurs peuvent soumettre des critiques pour les attractions, comprenant un texte descriptif, une note et la possibilité de rester anonyme. Cela permet une collecte précieuse de feedbacks, améliorant ainsi la qualité globale de l'expérience dans le parc.

- Amélioration de l'interface utilisateur : L'interface utilisateur a été retravaillée pour une navigation plus fluide et une expérience plus immersive. Avec une attention particulière portée à l'ergonomie, l'interface offre désormais une expérience utilisateur optimisée, améliorant ainsi la satisfaction globale.

- Support multi-langue : En intégrant la langue anglaise, l'application devient accessible à un public plus large. De plus, en offrant la possibilité de basculer entre les langues sur le même site, les utilisateurs peuvent choisir leur préférence linguistique en toute simplicité.

- Tests unitaires côté Angular : Des tests unitaires ont été mis en place pour garantir la stabilité et la fiabilité du frontend, assurant ainsi une meilleure qualité du code et une maintenance plus facile à long terme.

Schéma de base de données

1. Table "Attraction" :

ID_Attraction (clé primaire)

Nom_Attraction

Description_Attraction

Etat_Visibilite (booléen : visible / pas visible)

2. Table "Critique" :

ID_Critique (clé primaire)

ID_Attraction (clé étrangère référençant la table "Attraction")

ID_Utilisateur (clé étrangère référençant la table "Utilisateur")

Texte_Critique

ID_Note (clé étrangère référençant la table "Note")

Anonyme (booléen : true / false)

3. Table "Note" :

ID_Note (clé primaire)

Valeur_Note (entier de 1 à 5)

4. Table "Utilisateur" :

ID_Utilisateur (clé primaire)

Nom_Utilisateur

Prenom_Utilisateur

Email_Utilisateur

Mot_De_Passe (stocké de manière sécurisée, par exemple en utilisant une fonction de hachage)

Avec ces liaisons :

Chaque critique est liée à une attraction spécifique à travers l'ID_Attraction, et à un utilisateur spécifique à travers l'ID_Utilisateur.

Chaque critique est également liée à une note spécifique à travers l'ID_Note, qui fait référence à la valeur de note dans la table "Note".

Les tables "Attraction" et "Utilisateur" ne sont pas directement liées entre elles dans ce schéma, car elles ont des relations indépendantes avec la table "Critique".

Documentation du code

- Expliquez la structure du code source, les conventions de nommage, les modèles de conception utilisés, etc.

- Documentez les parties complexes du code pour aider les développeurs à comprendre facilement le fonctionnement de l'application.

Améliorations possibles

1. **Rendre l'écriture du mot de passe sous forme de points pour améliorer la sécurité d'un compte :**

Avantages :

- En rendant l'écriture du mot de passe sous forme de points ou d'étoiles lors de la saisie, vous améliorez la sécurité de compte en masquant le mot de passe à la vue des tiers.

- Cela réduit le risque d'exposition du mot de passe si quelqu'un regarde par-dessus l'épaule de l'utilisateur pendant qu'il saisit son mot de passe.

Inconvénients :

- Il peut être légèrement moins pratique pour les utilisateurs de voir le mot de passe pendant la saisie, ce qui peut entraîner des erreurs de frappe ou une saisie plus lente.

- Certains utilisateurs peuvent préférer voir le mot de passe pendant la saisie pour s'assurer qu'ils l'ont correctement entré.

2. **Vérifier qui est l'utilisateur connecté :**

Avantages :

- En vérifiant l'identité de l'utilisateur connecté, vous renforcez la sécurité de l'application en empêchant l'accès non autorisé à certaines fonctionnalités ou données sensibles.

- Cela permet également de personnaliser l'expérience de l'utilisateur en fonction de son profil, en lui montrant uniquement les informations et les fonctionnalités auxquelles il est autorisé à accéder.

Inconvénients :

- La vérification de l'identité de l'utilisateur peut ajouter une complexité supplémentaire au processus de connexion et à la gestion des sessions utilisateur.

- Si la vérification n'est pas correctement mise en œuvre, cela peut entraîner des problèmes d'accès pour les utilisateurs légitimes ou des failles de sécurité si les contrôles ne sont pas suffisamment rigoureux.

3. Sécurité côté admin :

Avantages :

- En renforçant la sécurité du côté administrateur, vous protégez les fonctionnalités et les données critiques de l'application contre les accès non autorisés ou les attaques potentielles.

- Cela peut inclure des mesures telles que l'authentification à deux facteurs, le chiffrement des données sensibles, la journalisation des activités des administrateurs, etc.

Inconvénients :

- La mise en place de mesures de sécurité supplémentaires peut nécessiter des ressources supplémentaires en termes de développement et de maintenance.

- Il peut y avoir une courbe d'apprentissage pour les administrateurs lors de l'utilisation de nouvelles fonctionnalités de sécurité, ce qui peut entraîner une résistance au changement ou une utilisation incorrecte.

Conclusion

Cette documentation détaille l'architecture, les fonctionnalités et le schéma de base de données de notre application de gestion pour le parc d'attraction. Notre objectif est de fournir une expérience immersive et interactive aux administrateurs et aux visiteurs.

L'architecture client-serveur avec Angular et Python (Flask), ainsi que la base de données MySQL, assurent une gestion efficace des données et une maintenance simplifiée.

Les principales fonctionnalités incluent la gestion des attractions, les critiques, l'amélioration de l'interface utilisateur, le support multi-langue et les tests unitaires.

Des améliorations sont suggérées pour renforcer la sécurité, notamment la sécurisation des mots de passe, la vérification de l'utilisateur connecté et la sécurité côté administrateur.