Casino

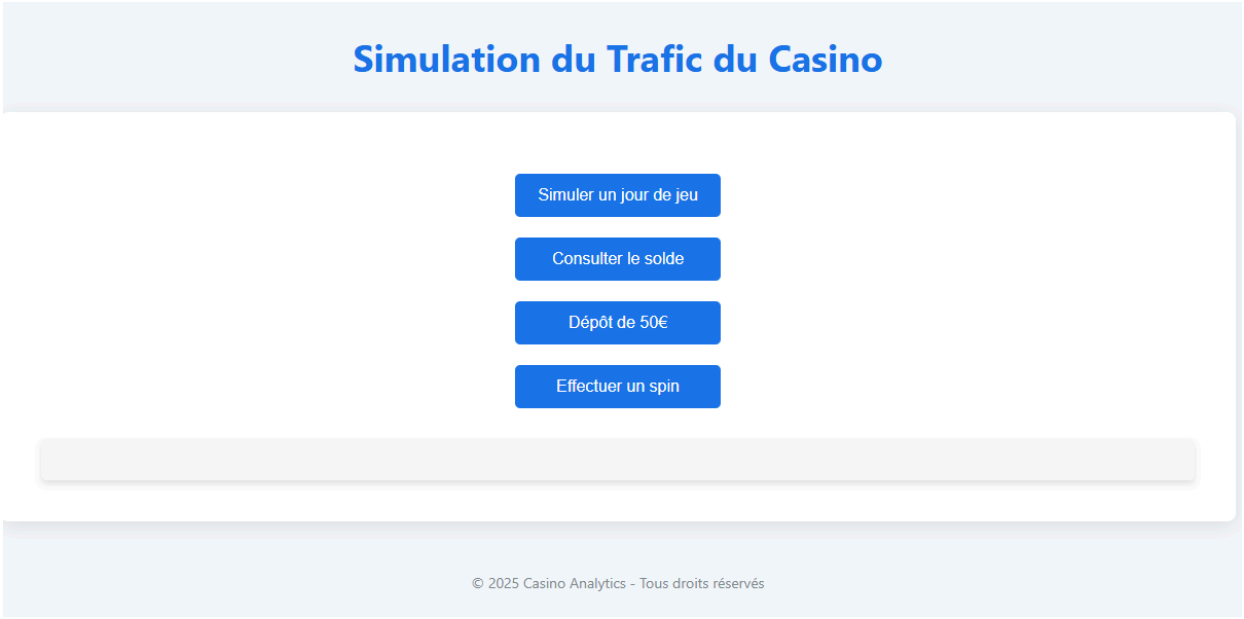
De Lyn Robel - Andy Clere - Diego Guterres Cavalho
Le 06.03.2025 - Genève, Petit-Lancy
Classe : I-FDA-P3G

1. Descriptif du But et des Fonctionnalités de l'Application Web

But de l'application : L'application web de simulateur de casino a pour objectif de permettre aux utilisateurs de simuler des jeux de casino dans un environnement sécurisé et sans risque financier réel. Cette simulation permettra aux utilisateurs de mieux comprendre le fonctionnement des jeux de casino, tout en ayant la possibilité d'effectuer des actions simples comme gérer un solde fictif, ajouter de l'argent virtuel et participer à des jeux de hasard.

Fonctionnalités :

- **Consultation du solde :** L'utilisateur peut consulter son solde actuel, c'est-à-dire la somme d'argent fictive qu'il possède pour jouer.
- **Ajout d'argent virtuel** : L'utilisateur peut ajouter 50 € virtuels à son solde à tout moment.
- **Jeu de la roue de la fortune** : L'utilisateur peut faire tourner une roue, où il a 50 % de chances de gagner 10 € et 50 % de chances de perdre 10 €. Cette action simule un jeu de hasard classique.
- **Logs** : L'application enregistre toutes les actions de l'utilisateur (consultation du solde, ajout d'argent et rotation de la roue) dans des logs afin de suivre l'activité et d'assurer une gestion correcte des transactions.



2. Architecture Complète de l'Application Web

L'architecture de l'application web peut être vue comme une application à trois niveaux :

1. **Frontend** :
    - Interface utilisateur (UI) qui permet aux utilisateurs d'interagir avec le simulateur de casino (consultation du solde, ajout d'argent et participation à la roue).
    - Technologies : HTML, CSS, JavaScript (React ou Vue.js par exemple).
2. **Backend** :
    - Serveur qui gère les requêtes de l'utilisateur (consultation du solde, ajout d'argent, rotation de la roue).
    - Traitement logique du jeu (calcul du résultat de la roue, ajout ou retrait d'argent).
    - Enregistrement des logs dans une base de données ou un système de gestion des logs.
    - Technologies : Node.js, Express, ou un autre framework de backend.
3. **Base de données / Logs** :
    - Base de données pour stocker l'historique des actions de l'utilisateur et leur solde (ex. MySQL, MongoDB).
    - Système de gestion des logs (ex. ELK Stack - Elasticsearch, Logstash, Kibana) pour suivre les activités des utilisateurs et détecter des anomalies.

3. Descriptif de la Fiabilité de l'Application Web

Pour assurer la fiabilité de l'application, il est crucial de surveiller les points suivants :

1. **Gestion du solde** :
    - Risque : Mauvaise gestion des transactions entraînant une incohérence dans le solde de l'utilisateur.
    - Mesure : Collecte des logs à chaque modification de solde (ajout ou retrait).
    - Solution : Mise en place de vérifications d'intégrité du solde après chaque modification et génération de logs détaillés pour chaque transaction.
2. **Rotation de la roue** :
    - Risque : Mauvais calcul des résultats ou des erreurs dans la simulation des gains et pertes.
    - Mesure : Enregistrement des résultats à chaque tirage de la roue dans les logs.

    - Solution : Implémentation d'un algorithme de génération de nombres aléatoires fiable, avec un contrôle des résultats dans les logs.
3. **Problèmes de serveur ou de disponibilité** :
    - Risque : Pannes du serveur ou lenteurs lors des interactions.
    - Mesure : Surveillance de l'état du serveur et des réponses API. Collecte de logs d'erreur pour détecter les pannes ou les latences élevées.
    - Solution : Mise en place de tests de disponibilité réguliers et d'une surveillance continue avec alertes.
4. **Authentification et sécurité** :
    - Risque : Compromission de l'accès utilisateur, perte de données.
    - Mesure : Logs d'accès, tentatives de connexion et actions sensibles (par exemple, ajout d'argent).
    - Solution : Implémentation d'une sécurité renforcée avec des alertes en cas de tentatives de connexion suspectes.

4. Valeurs Limites et Alertes

Afin de maintenir la fiabilité de l'application, voici des valeurs limites spécifiques pour chaque point critique :

1. **Solde de l'utilisateur** :
    - Valeur limite : Si le solde d'un utilisateur devient négatif, générer une alerte.
    - Action : Envoi d'une alerte par email à l'administrateur et affichage d'une alerte dans le tableau de bord admin.
2. **Rotation de la roue** :
    - Valeur limite : Si l'algorithme de la roue retourne un résultat impossible (ex. résultat incorrect), générer une alerte.
    - Action : Envoi d'une alerte par email et journalisation de l'erreur dans le système de logs.
3. **Erreurs serveur** :
    - Valeur limite : Si le serveur retourne une erreur 500 ou une latence de réponse supérieure à 2 secondes pour 10% des requêtes, générer une alerte.
    - Action : Envoi d'une alerte par email à l'équipe technique et journalisation des erreurs dans le système de logs.
4. **Tentatives de connexion suspectes** :
    - Valeur limite : Si plus de 5 tentatives de connexion échouées en 30 minutes, générer une alerte.
    - Action : Envoi d'une alerte par email à l'administrateur et verrouillage temporaire de l'utilisateur suspect.

5. Conclusion

Pour assurer un contrôle précis de la fiabilité de notre application, il nous manque actuellement quelques fonctionnalités clés, notamment l'implémentation d'un système de surveillance automatisée des logs et des alertes en temps réel. Bien que les logs soient générés à chaque interaction importante de l'utilisateur, la mise en place d'un système d'alertes efficace nécessitera l'intégration d'un outil de monitoring comme Prometheus ou Grafana pour obtenir une surveillance continue. Nous devons également tester les limites des alertes et nous assurer qu'elles sont configurées correctement pour garantir une détection rapide des anomalies. Enfin, un processus de validation des résultats du jeu est encore nécessaire pour assurer une simulation juste et fiable.