NOM	Marie-sainte
Prénom	Terry
Date de naissance	27/12/1988

Copie à rendre

Bloc 3 – Développement d'une solution digitale avec Java

Documents à compléter et à rendre

Lien git Frontend: https://github.com/thugcoder972/FrontOlympique
Lien git Backend: https://github.com/thugcoder972/backendJolypique
gestion de projet : https://trello.com/b/2iYCYyWS/bloc-3-developpement-dune-solution-digital-java

application deployée : en cours
Lien documentation :https://github.com/thugcoder972/FrontOlympique/tree/master/Documentation

Dossier 1 : Spécifier une solution data/ia

1. toutes les fonctionnalités attendues par le client sous forme d'User Story (agilité)

En tant que visiteur, je veux pouvoir accéder à la page d'accueil pour découvrir les Jeux olympiques et consulter les principales épreuves.

En tant que visiteur, je veux pouvoir consulter les différentes offres (solo, duo, familiale) afin de choisir celle qui correspond à mes besoins.

En tant que visiteur, je veux pouvoir ajouter une offre à mon panier pour procéder à l'achat.

En tant qu'utilisateur authentifié, je veux pouvoir créer un compte en fournissant mon nom, prénom, adresse e-mail, et un mot de passe sécurisé.

En tant qu'utilisateur, je veux pouvoir m'authentifier sur le site pour accéder à mon compte et acheter des billets.

En tant qu'utilisateur, je veux pouvoir finaliser mon achat et payer mes billets (mock de paiement à intégrer).

En tant qu'utilisateur, je veux recevoir un e-ticket sécurisé sous forme de QR code après l'achat, comprenant une clé de sécurité pour garantir son authenticité.

En tant qu'employé des Jeux olympiques, je veux pouvoir scanner les e-tickets des visiteurs le jour de l'événement pour vérifier leur authenticité grâce à la combinaison des clés.

En tant qu'administrateur, je veux pouvoir visualiser, modifier ou ajouter de nouvelles offres via un espace administrateur dédié.

En tant qu'administrateur, je veux pouvoir consulter le nombre de ventes par offre pour analyser les statistiques de vente.

2. Quels sont les éléments que vous allez sécuriser ? Comment allez-vous procéder ?

Authentification utilisateur :

Protection des mots de passe et de l'authentification via Spring Boot Security.

JSON Web Tokens (JWT) :

- Sécurisation de l'authentification et des communications avec les JWT.

Communication client-serveur :

- Sécurisation des échanges de données avec HTTPS et les JWT.

Protection CSRF :

- Prévention des attaques CSRF pour les requêtes à haut risque.

Clés de sécurité et QR codes :

- Sécurisation des clés de sécurité générées et du QR code par chiffrement.

Déploiement serveur :

- Sécurisation du serveur via un pare-feu (UFW) et une protection contre les attaques par force brute (Fail2ban).

Comment allez-vous procéder ?

Authentification utilisateur avec Spring Boot Security :

Spring Boot Security sera configuré pour gérer l'authentification des utilisateurs. Lors de la connexion, un token JWT sera généré après la validation des informations d'identification. Les utilisateurs non authentifiés n'auront pas accès aux parties sécurisées de l'application, garantissant ainsi la protection des données sensibles.

Gestion des JWT (JSON Web Tokens) :

Nous utiliserons les JWT pour gérer les sessions utilisateur et l'authentification sans avoir à stocker les sessions sur le serveur. Les tokens JWT seront générés lors de la connexion de l'utilisateur. Chaque token sera signé pour éviter toute falsification. Les JWT seront utilisés dans l'en-tête Authorization pour les requêtes futures, permettant ainsi une autorisation sécurisée. Nous configurerons également une expiration des tokens JWT afin de limiter leur durée de validité et forcer la réauthentification.

Sécurisation des communications client-serveur :

HTTPS sera utilisé pour toutes les communications entre le client et le serveur, garantissant que les données échangées sont chiffrées en transit. Les requêtes HTTP authentifiées utiliseront un JWT pour assurer l'intégrité des sessions utilisateur et éviter les interceptions de données.

Protection CSRF (Cross-Site Request Forgery) :

Spring Security CSRF sera activé pour protéger les requêtes POST, PUT, DELETE contre les attaques CSRF. Un token CSRF sera généré et vérifié sur chaque requête mutative.

Génération et sécurisation des clés de sécurité :

Lors de la création d'un compte ou de l'achat d'un billet, des clés de sécurité seront générées. Ces clés seront stockées de manière sécurisée et jamais exposées directement à l'utilisateur. Les clés seront concaténées et chiffrées avant d'être intégrées dans un QR code, garantissant ainsi que seul le serveur peut les décrypter.

Sécurité du déploiement :

Fail2ban sera configuré pour surveiller les tentatives de connexion sur le serveur. Il bloquera automatiquement les adresses IP après un nombre excessif d'échecs, protégeant ainsi contre les attaques par force brute. UFW (Uncomplicated Firewall) sera utilisé pour configurer le pare-feu et limiter les connexions aux seuls ports nécessaires, tels que ceux utilisés pour HTTPS.

Conclusion

En sécurisant ces éléments, nous garantissons la protection des données utilisateur, la sécurité des communications et la robustesse de l'application contre les attaques courantes. L'utilisation de technologies modernes comme JWT, bcrypt, et des outils de sécurité comme Fail2ban et UFW, combinée à des pratiques de développement sécurisées, permet de créer une application

3. Évoquez les choix techniques que vous avez choisis concernant votre application et justifiez-les (tout en faisant référence au besoin client) ?

   Les éléments sécurisés dans cette application sont les suivants :

1. **Connexion au site via un token d'authentification** : Nous sécuriserons la connexion des utilisateurs en générant un **token JWT** lors de l'authentification, ce qui garantira une session sans stockage d'état (stateless) côté serveur.

2. **Vérification de l'authenticité des billets** : Une combinaison de deux clés, **token_user** et **token_billet**, sera utilisée pour vérifier l'authenticité de chaque billet acheté, garantissant ainsi qu'il n'y a aucune falsification ou fraude.

3. **Vérification de connexion en deux étapes (2FA)** : Pour renforcer la sécurité des comptes, une authentification à deux facteurs (2FA) sera mise en place, ce qui nécessite une validation supplémentaire lors de la connexion des utilisateurs.

Choix Techniques et Justifications

Backend - Java Spring Boot

1. **Choix de Spring Boot et Spring Security**
    - **Justification** : Spring Boot est un framework Java mature, reconnu pour sa capacité à construire des applications robustes, évolutives et sécurisées. Avec **Spring Security**, nous disposons d'outils puissants pour implémenter des mécanismes d'authentification et d'autorisation, en particulier dans les environnements nécessitant une grande sécurité, comme les systèmes de billetterie pour les Jeux Olympiques.

2. **Modèle de données personnalisé**
    - **Justification** : En définissant un modèle utilisateur personnalisé, nous pouvons facilement adapter la gestion des rôles et des permissions (comme **utilisateurs, administrateurs**), ainsi que capturer des informations supplémentaires comme les **numéros de téléphone** ou des types de comptes spécifiques. Cela répond directement aux besoins du client en termes de flexibilité dans la gestion des utilisateurs.
3. **JWT pour l'authentification**
    - **Justification** : Les **JSON Web Tokens (JWT)** fournissent une solution sécurisée et sans état pour l'authentification. Cela permet aux utilisateurs de s'authentifier et de réaliser des actions sur le site de manière efficace, tout en protégeant la session utilisateur. Cette approche est idéale pour une **Single Page Application (SPA)** comme notre frontend basé sur **React**, facilitant ainsi la gestion des sessions côté client.

Frontend - React

1. **Choix de React**
    - **Justification** : **React** est un outil puissant et flexible pour le développement d'interfaces utilisateurs dynamiques. Il permet de créer une expérience utilisateur fluide et réactive grâce à sa gestion performante du **DOM virtuel**, ce qui est essentiel pour une application nécessitant de nombreuses interactions comme celle liée aux événements sportifs des Jeux Olympiques.
2. **Redux pour la gestion de l'état**
    - **Justification** : **Redux** sera utilisé pour centraliser la gestion de l'état de l'application de manière prévisible. Dans un projet où les interactions utilisateurs sont nombreuses et où l'état doit être partagé entre plusieurs composants (comme la gestion des profils utilisateurs ou le panier d'achat des billets), Redux offre une solution robuste et fiable pour synchroniser les données à travers l'application.
3. **Styled-components pour le styling**
    - **Justification** : L'utilisation de **Styled-components** permet d'intégrer le CSS directement dans les composants JavaScript, favorisant une modularité et une maintenabilité accrues du code. Cette approche est particulièrement

avantageuse dans les projets de grande envergure, car elle permet de concevoir des interfaces réutilisables et faciles à maintenir.

Architecture et Communication

1. **Architecture Client-Serveur**
    - **Justification** : La séparation claire entre le **backend (Spring Boot)** et le **frontend (React)** permet une meilleure organisation du code, une évolutivité facilitée et une maintenance simplifiée. Cette architecture modulaire permet également de répartir les responsabilités entre la gestion des données, la logique métier et l'interface utilisateur.
2. **API RESTful**
    - **Justification** : La mise en place d'**API RESTful** standardisées facilite la communication entre le client et le serveur. Ces API permettent d'exécuter les opérations **CRUD** (Create, Read, Update, Delete) de manière efficace, répondant ainsi aux besoins en gestion de billets, utilisateurs et événements sportifs.

Conclusion

Les choix techniques réalisés pour ce projet ont été faits dans l'objectif de répondre aux besoins spécifiques du client tout en assurant des performances optimales et une sécurité renforcée. L'utilisation de **Spring Boot** pour le backend et de **React** pour le frontend, associée à des technologies comme **JWT** pour l'authentification et **Redux** pour la gestion de l'état, garantit une base solide pour une application web moderne, évolutive et sécurisée. Ces choix permettent de créer une application performante, capable de gérer des flux de données complexes et des interactions utilisateurs nombreuses, tout en assurant la protection des informations sensibles.

Dossier 2 : Développement de la solution

2.1 Présentez la veille technique concernant la sécurité que vous avez effectuée.
Lien trello:
https://trello.com/b/2iYCYyWS/bloc-3-developpement-dune-solution-digital-java

Veille technique concernant la sécurité

La sécurité d'un site Web

Assurer la sécurité d'un site web exige une vigilance constante dans tous les aspects de sa conception et de son utilisation. Cet article ne vous transformera pas en expert en sécurité, mais il vous aidera à comprendre les menaces auxquelles vous pouvez être confronté et les mesures à prendre pour renforcer la sécurité de votre application web contre les attaques les plus courantes.

Pré-requis :

Connaissances de base en informatique.

Objectif :

Comprendre les menaces les plus fréquentes pour la sécurité des applications web et les actions à entreprendre pour réduire les risques de piratage.

Qu'est-ce que la sécurité d'un site web ?

Internet est un environnement risqué ! Nous entendons souvent parler de sites web devenus inaccessibles à cause d'attaques par déni de service, ou affichant des informations falsifiées sur leur page d'accueil. Dans certains cas, des millions de mots de passe, d'adresses électroniques et de détails de cartes de crédit sont divulgués, exposant ainsi les utilisateurs à des embarras personnels et à des pertes financières.

L'objectif de la sécurité des sites web est de prévenir ces types d'attaques. Plus formellement, cela implique de protéger les sites web contre tout accès, utilisation, modification, destruction ou perturbation non autorisés.

Assurer la sécurité d'un site web nécessite un effort concerté tout au long de sa conception : au niveau de l'application web, de la configuration du serveur web, des politiques de création et de renouvellement des mots de passe, ainsi que du code côté client.

Bien que ces menaces puissent sembler préoccupantes, la bonne nouvelle est que si vous utilisez un framework web côté serveur comme **Spring Boot**, celui-ci intègre par défaut des mécanismes de défense robustes contre plusieurs des attaques les plus courantes. D'autres menaces peuvent être atténuées par une configuration adéquate de votre serveur web, notamment en activant HTTPS.

Enfin, des outils d'analyse de vulnérabilités accessibles au public peuvent vous aider à identifier d'éventuelles erreurs dans votre conception.

Menaces visant la sécurité des sites web

Cette section présente quelques-unes des menaces les plus courantes auxquelles les sites web sont confrontés et les moyens de les contrer. Notez comment ces menaces surviennent lorsque l'application web accorde trop de confiance aux données provenant du navigateur !

Cross-Site Scripting (XSS)

Le XSS désigne une classe d'attaques permettant à un attaquant d'injecter des scripts exécutés côté client, en exploitant le site web pour cibler le navigateur des autres utilisateurs. Comme le code injecté provient du site lui-même, le navigateur le considère comme sûr, ce qui peut permettre à l'attaquant de récupérer des informations sensibles, telles que le cookie d'authentification. Une fois en possession de ce cookie, l'attaquant peut accéder au site comme s'il était l'utilisateur visé et réaliser toutes les actions autorisées.

La meilleure défense contre les vulnérabilités XSS consiste à désactiver toutes les balises susceptibles de contenir des instructions pour exécuter du code, notamment `<script>`, `<object>`, `<embed>`, et `<link>`. Il est essentiel de nettoyer toutes les données saisies par l'utilisateur (processus connu sous le nom de "sanitization"). La plupart des frameworks, y compris **Spring Boot**, offrent par défaut des mécanismes pour cette vérification.

Injection SQL

L'injection SQL est une vulnérabilité permettant à un attaquant d'exécuter des commandes SQL malveillantes sur une base de données, compromettant ainsi la sécurité des données. Une attaque réussie peut aboutir à l'usurpation de comptes, la création de comptes administratifs, l'accès à toutes les données ou la destruction des informations.

Pour prévenir cette vulnérabilité, il est crucial de s'assurer que toute saisie utilisateur intégrée dans une requête SQL ne peut pas modifier l'intention de cette requête. Échapper tous les caractères spéciaux dans les saisies utilisateur est une méthode efficace.

Falsification de requête inter-sites (CSRF)

Les attaques CSRF permettent à un utilisateur malveillant d'effectuer des actions en utilisant les identifiants d'un autre utilisateur sans son consentement. Par exemple, un utilisateur malveillant pourrait envoyer un formulaire déguisé à un utilisateur authentifié sur un site de transfert d'argent, ce qui entraînerait l'exécution involontaire d'une transaction.

Pour contrer ce type d'attaque, il est recommandé que chaque requête POST comporte un jeton secret, généré par le serveur, pour garantir qu'il provient de l'utilisateur authentifié.

Autres menaces

Les autres menaces et vulnérabilités courantes comprennent :

- **Clickjacking** : Cette attaque détourne les clics d'un utilisateur en les dirigeant vers une page cachée. Pour s'en prémunir, configurez les en-têtes HTTP pour empêcher votre site d'être inclus dans une iframe d'un autre site.
- **Déni de Service (DoS)** : Cette technique consiste à submerger un site de requêtes malveillantes, le rendant inaccessibles aux utilisateurs légitimes. La défense repose sur l'identification et le blocage du trafic indésirable.
- **Navigation dans les répertoires** : Les attaquants essaient d'accéder à des fichiers sensibles sur le serveur en manipulant les noms de fichiers. La désinfection des saisies est cruciale.
- **Inclusion de fichiers** : Cette attaque permet à un utilisateur de spécifier un fichier à exécuter sur le serveur, ce qui peut conduire à des attaques XSS. Il est essentiel de valider les saisies avant leur utilisation.
- **Injection de commandes** : Les attaques par injection de commande permettent à un attaquant d'exécuter des commandes système non autorisées. Vérifiez chaque saisie utilisateur avant de l'utiliser dans des appels système.

Messages clés

La majorité des attaques mentionnées précédemment réussissent lorsque l'application web fait confiance aux données provenant du navigateur. Peu importe les autres mesures que vous prenez pour renforcer la sécurité de votre site, assurez-vous de désinfecter toutes les saisies utilisateur avant de les afficher, de les utiliser dans des requêtes SQL ou de les transmettre dans des appels systèmes.

Autres recommandations :

1. **Politique de gestion des mots de passe** : Encouragez l'utilisation de mots de passe forts et leur renouvellement fréquent. Considérez l'implémentation d'une authentification à deux facteurs.

2. **Configuration du serveur** : Utilisez HTTPS et activez HTTP Strict Transport Security (HSTS) pour chiffrer les données échangées entre le client et le serveur.

3. **Veille sur les menaces** : Restez informé des dernières menaces et concentrez-vous sur les vulnérabilités les plus courantes.

4. **Outils de recherche de vulnérabilités** : Utilisez des outils pour détecter automatiquement les failles de votre site et envisagez de mettre en place un programme de bug bounty.

5. **Stockage des données sensibles** : Limitez l'affichage des données sensibles aux seules informations nécessaires pour l'identification de l'utilisateur.

Les frameworks web comme **Spring Boot** peuvent aider à atténuer de nombreuses vulnérabilités courantes.

2.2 Indiquez les éléments qui vous ont posé difficulté et comment vous les avez franchis

Les éléments qui m'on posé problème était aux niveaux du back-end au niveau mapping , les différentes relations et le déploiement que je n'est pas eu le temps de faire .

2.3 Si vous avez des éléments à ajouter, ajoutez-les ici.