NOM	Aitichou
Prénom	Mohamed
Date de naissance	03/06/2002

Copie à rendre

Bloc 3 – Développement d'une solution digitale avec Java

Documents à compléter et à rendre

Lien git : https://github.com/MohamedAitichou/ticketing-jo-2024
Lien gestion de projet : https://trello.com/b/QWTYQxAc/modele-kanban
Lien application deployée : https://ticketing-jo-2024-2.onrender.com/

Dossier 1 : Spécifier une solution digitale

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

Pour répondre aux besoins du client, j'ai identifié plusieurs fonctionnalités principales que j'ai traduites sous forme de User Stories.
Chaque fonctionnalité correspond à une attente concrète de l'utilisateur final ou d'un administrateur.

Côté utilisateur :

- En tant que visiteur, je peux consulter la liste des offres de billets disponibles afin de découvrir les différentes formules proposées (Solo, Duo, Familial, etc.).
- En tant que visiteur, je peux créer un compte avec mon adresse e-mail et un mot de passe pour accéder à l'espace d'achat sécurisé.
- En tant qu'utilisateur inscrit, je peux me connecter à mon compte, recevoir un code OTP (One-Time Password) et le valider pour sécuriser ma connexion.
- En tant qu'utilisateur connecté, je peux sélectionner une offre, passer une commande et acheter un ou plusieurs billets.
- En tant qu'utilisateur, je peux visualiser l'ensemble de mes commandes dans un espace personnel et consulter les tickets associés à chaque achat.
- En tant qu'utilisateur, je peux afficher le QR code de chaque billet afin de pouvoir le présenter lors du contrôle d'accès.

Côté agent ou administrateur :

- En tant qu'agent de contrôle, je peux scanner ou vérifier la validité d'un ticket via sa clé unique ou son QR code afin de m'assurer qu'il est authentique.
- En tant qu'administrateur, je peux accéder à un tableau de bord affichant le nombre total de tickets vendus et la répartition des ventes par offre.
- En tant qu'administrateur, je peux créer de nouvelles offres (avec un code unique, un nom, un prix, une description et un nombre de places), les modifier, les activer ou les désactiver.
- En tant qu'administrateur, je peux supprimer une offre uniquement si elle n'a pas encore été vendue, afin de préserver l'intégrité des données.

Côté système :

- Le système doit empêcher la création de doublons (même e-mail utilisateur, même code d'offre, même clé de ticket).
- Le système doit garantir que la suppression d'une offre ayant déjà des tickets vendus soit bloquée et retourner un message d'erreur explicite.
- Le système doit assurer la cohérence entre les commandes, les utilisateurs et les tickets grâce à des contraintes de clé étrangère (relations entre tables).

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

La sécurité est au centre du projet, car il s'agit d'une application manipulant des données sensibles (comptes utilisateurs, commandes et billets électroniques).
J'ai donc mis en place plusieurs mécanismes de protection à différents niveaux.

Authentification et accès

- Chaque utilisateur possède un compte protégé par un mot de passe haché avec BCrypt, ce qui empêche tout stockage en clair.
- Lors de la connexion, un code OTP à usage unique est généré et vérifié avant d'accorder l'accès. Cela permet de s'assurer que l'utilisateur est bien celui qu'il prétend être.
- Une fois connecté, le client reçoit un token JWT (JSON Web Token) signé côté serveur, utilisé pour toutes les requêtes authentifiées.
- Les rôles USER, AGENT et ADMIN permettent de restreindre les accès selon le profil. Par exemple, les routes /api/admin/** sont strictement réservées aux administrateurs.

Sécurisation des données et intégrité

- Les entrées utilisateur sont validées avec des annotations de validation (@Email, @NotBlank, etc.) pour éviter les données invalides ou malveillantes.
- Les champs sensibles sont nettoyés côté serveur avant sauvegarde (trim, valeurs par défaut).
- Des contraintes d'unicité et clés étrangères sont définies en base (H2 en développement), empêchant la suppression accidentelle d'une donnée encore liée (par exemple, une offre liée à des tickets vendus).

Protection des billets et QR codes

- Chaque billet est associé à une clé unique "finalKey", construite à partir de la clé utilisateur et de la clé de commande.
- Le QR code contient cette clé et permet à un agent de vérifier instantanément la validité du ticket.
- Un ticket consommé est marqué avec une date (consumedAt), ce qui empêche toute réutilisation.

Communication et sécurité réseau

- Les requêtes sont transmises via HTTPS (en production).
- Les tokens JWT sont transmis dans le header Authorization: Bearer, garantissant une communication stateless et sécurisée.
- Le CORS est configuré pour autoriser uniquement le domaine du front-end.

Ces différentes mesures assurent la confidentialité, l'intégrité et la traçabilité des opérations sur le système.

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

Pour répondre efficacement aux besoins du client et respecter les contraintes pédagogiques du projet, j'ai fait des choix techniques cohérents et réalistes.

Back-end : Java / Spring Boot

Le back-end est développé en Java 21 avec Spring Boot 3, un framework complet et robuste.
Spring Boot offre une structure claire, une intégration native de la sécurité (Spring Security) et une gestion facile de la base de données via Spring Data JPA.
La base de données H2 est utilisée pour le développement (embarquée et simple à tester), avec la possibilité de passer à PostgreSQL en production.

Spring permet également d'intégrer :

- JWT pour l'authentification stateless, idéal pour une application web moderne.
- Gestion des rôles et permissions intégrée via annotations.
- Validation des DTOs, assurant la cohérence des données à l'entrée.
- ZXing pour la génération des QR codes côté serveur, garantissant leur sécurité et leur unicité.

Ce choix répond directement au besoin client de sécurité, de performance et de fiabilité.

Front-end : React + Vite

Le front-end est réalisé avec React et Vite.
Ce duo permet un développement rapide, une application fluide et réactive, et une bonne séparation entre interface et logique métier.
Le front communique avec le back via API REST en fetch, avec gestion centralisée du token JWT.

J'ai mis l'accent sur :

- Une interface simple et claire, adaptée à une utilisation sur ordinateur.
- Des pages spécifiques : accueil, offres, connexion/OTP, commandes, tickets et panneau administrateur.
- Des composants réutilisables et une gestion d'état légère (sans Redux).

Architecture et sécurité

L'application suit une architecture client/serveur REST claire :

- localhost:8081 pour l'API Spring Boot
- localhost:5173 pour le front React

Le tout est stateless, c'est-à-dire sans session persistée côté serveur, ce qui simplifie la scalabilité et renforce la sécurité.
Les erreurs sont uniformisées (statuts HTTP 401, 403, 409, etc.) et les messages sont clairs pour l'utilisateur final.

Pourquoi ces choix

- Spring Boot + React : stack moderne, bien documentée, adaptée à un projet étudiant complet.
- JWT + OTP + QR code : répondent parfaitement à la problématique de sécurité d'une billetterie numérique.
- H2 : pratique pour les tests, aucune installation nécessaire.
- Vite + React : performances élevées et simplicité de déploiement.

Ces décisions permettent d'obtenir une solution sécurisée, maintenable et fluide, tout en restant conforme aux exigences du bloc.

Dossier 2 : Développement de la solution

2.1 Développez la solution demandée par le client, pour ce faire, vous devrez également :
-Produire le MCD
-Effectuer une gestion de projet
- Fournir l'outil de gestion de projet que vous avez utilisé suivant la méthode KANBAN

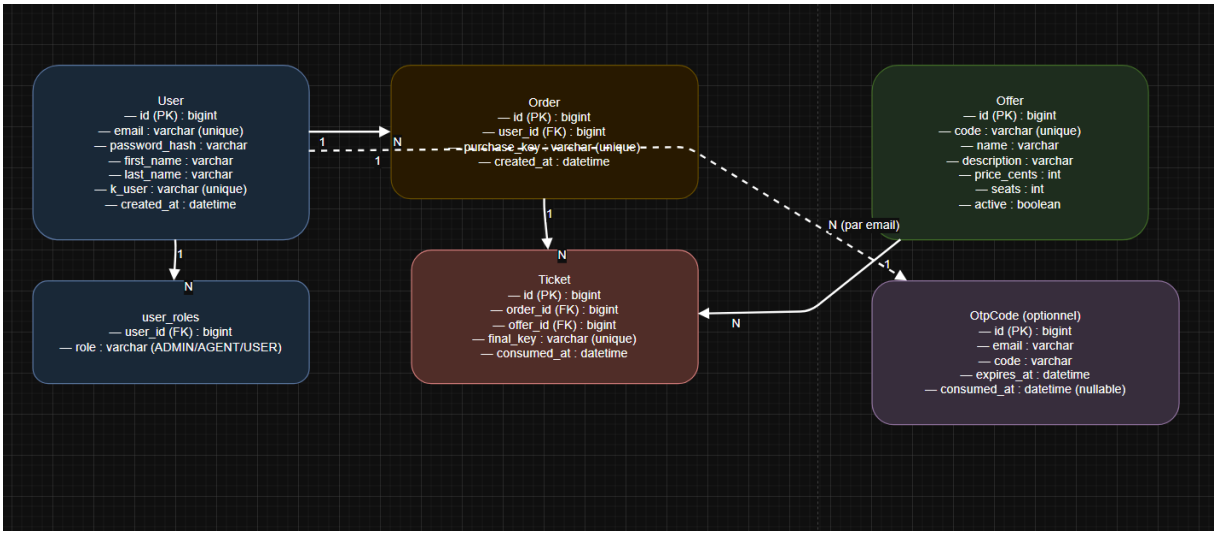
-Produire une documentation technique
- Cette documentation devra aborder la sécurité, mais également, des évolutions futures de votre application

-Produire un manuel d'utilisation
- Conception d'une documentation permettant à l'utilisateur d'utiliser votre application, on peut voir cela comme un « tuto »

-Le client vous impose :
- D'effectuer un back-end
- D'effectuer une application dynamique avec du JavaScript (le JavaScript permet de contacter votre back-end sans rechargement de page)
- Mettre en place des tests et produire un rapport détaillant le pourcentage de code total couvert par les tests
- Déploiement en ligne

Modèle Conceptuel de Données (MCD) de la billetterie

Ce MCD couvre la gestion des utilisateurs, rôles, offres, commandes, billets et OTP. Il est directement aligné avec le code développé (Spring Boot + JPA).



Modèle conceptuel de données (MCD) de la billetterie JO Paris 2024.

Ce modèle représente la structure de la base de données utilisée dans l'application de billetterie.
Il définit les entités principales (User, Offer, Order, Ticket, user_roles, OtpCode) et leurs relations.
Ce modèle permet de gérer les utilisateurs, leurs rôles, les offres disponibles, les commandes passées et les billets générés.
Il correspond à l'implémentation du projet développé avec Spring Boot (Back-end) et React (Front-end).

Gestion de projet — Méthode KANBAN

Pour organiser le développement de mon application de billetterie, j'ai utilisé la méthode KANBAN, à l'aide d'un tableau Trello.
Cette méthode permet de visualiser facilement l'avancement du projet, les priorités et les tâches en cours.

Le tableau est accessible à l'adresse suivante :
https://trello.com/invite/b/68c0cce7b452507b1abae6c6/ATTI76613897e6e6ea9455f301c443f8e4e222E26A29/modele-kanban

1. Organisation du tableau

Le tableau est divisé en plusieurs colonnes :

- À faire : toutes les tâches prévues au départ (analyse, conception, configuration de l'environnement).
- En cours : les tâches sur lesquelles je travaillais (création des API, tests de connexion, mise en place du front-end).
- En test : les fonctionnalités terminées mais en phase de vérification (authentification, génération de tickets, QR codes).
- Terminé : les éléments finalisés et validés (sécurité JWT, OTP, déploiement local, design React).

2. Suivi du projet

Chaque carte Trello correspond à une tâche précise (exemple : “Création de l’entité User”, “Intégration du JWT”, “Page d’administration des offres”).
Cela m’a permis de :

- garder une vision claire de l’état d’avancement du projet,
- mieux gérer mon temps et mes priorités,
- repérer rapidement les blocages éventuels.

3. Avantages de cette méthode

La méthode KANBAN m’a aidé à travailler de manière plus structurée et agile, sans perdre de vue les objectifs du projet.
Elle est particulièrement adaptée pour un projet comme celui-ci, où il faut avancer progressivement entre la partie back-end, front-end et tests.

Documentation technique de l’application

Cette partie présente les choix techniques et les éléments de sécurité mis en place pour le développement de l’application de billetterie des Jeux Olympiques 2024.
L’objectif principal est d’assurer un système fiable, sécurisé et évolutif, permettant la gestion des offres de billets, des commandes et du contrôle d’accès aux événements.

1. Architecture générale

L’application repose sur une architecture client/serveur :

- Front-end développé avec React.js, qui gère toute l’interface utilisateur, les formulaires et la communication avec l’API sans rechargement de page.
- Back-end développé avec Spring Boot (Java), qui expose des API REST pour les fonctionnalités d’authentification, de gestion des offres, des commandes et des tickets.
- La base de données utilisée est H2 (en local), mais elle peut facilement être remplacée par MySQL ou PostgreSQL pour une mise en production.

2. Sécurité et authentification

Pour sécuriser l’accès à l’application :

- Un système d’authentification JWT (JSON Web Token) a été mis en place. Lorsqu’un utilisateur se connecte, un token chiffré est généré et envoyé au navigateur, qui l’utilise ensuite pour toutes les requêtes protégées.
- Les mots de passe sont hachés avant d’être enregistrés dans la base de données (aucun mot de passe n’est stocké en clair).
- Une vérification par code OTP (One Time Password) a été ajoutée. L’utilisateur reçoit un code temporaire par e-mail lors de la connexion, ce qui ajoute une couche de sécurité supplémentaire.
- L’application gère différents rôles utilisateurs :
  - ADMIN : accès complet (gestion des offres, statistiques, utilisateurs).
  - AGENT : accès au module de contrôle des tickets.
  - USER : accès à la billetterie et à l’historique des commandes.

3. Gestion des données

Toutes les entités (User, Offer, Order, Ticket, OtpCode) sont reliées selon le MCD précédemment présenté.
Les relations entre tables sont gérées automatiquement par JPA/Hibernate avec des contraintes d'intégrité (clé étrangère, unicité, suppression protégée).
Les suppressions sont sécurisées : par exemple, on ne peut pas supprimer une offre qui possède déjà des billets vendus.

4. Communication front/back

Le front communique avec le back via des requêtes HTTP sécurisées (fetch) :

- Les endpoints sont préfixés par /api/ pour les utilisateurs et /api/admin/ pour les administrateurs.
- Toutes les requêtes sensibles incluent le header d'authentification Bearer Token.
- Le format d'échange entre les deux parties est JSON.

5. Évolutions possibles

Le projet a été pensé pour pouvoir évoluer facilement :

- Passage à une base de données en ligne (MySQL/PostgreSQL).
- Ajout d'un système de paiement réel (ex : Stripe ou PayPal).
- Envoi automatique des QR codes par e-mail après achat.
- Gestion de plusieurs événements différents (au lieu d'un seul).

6. Conclusion

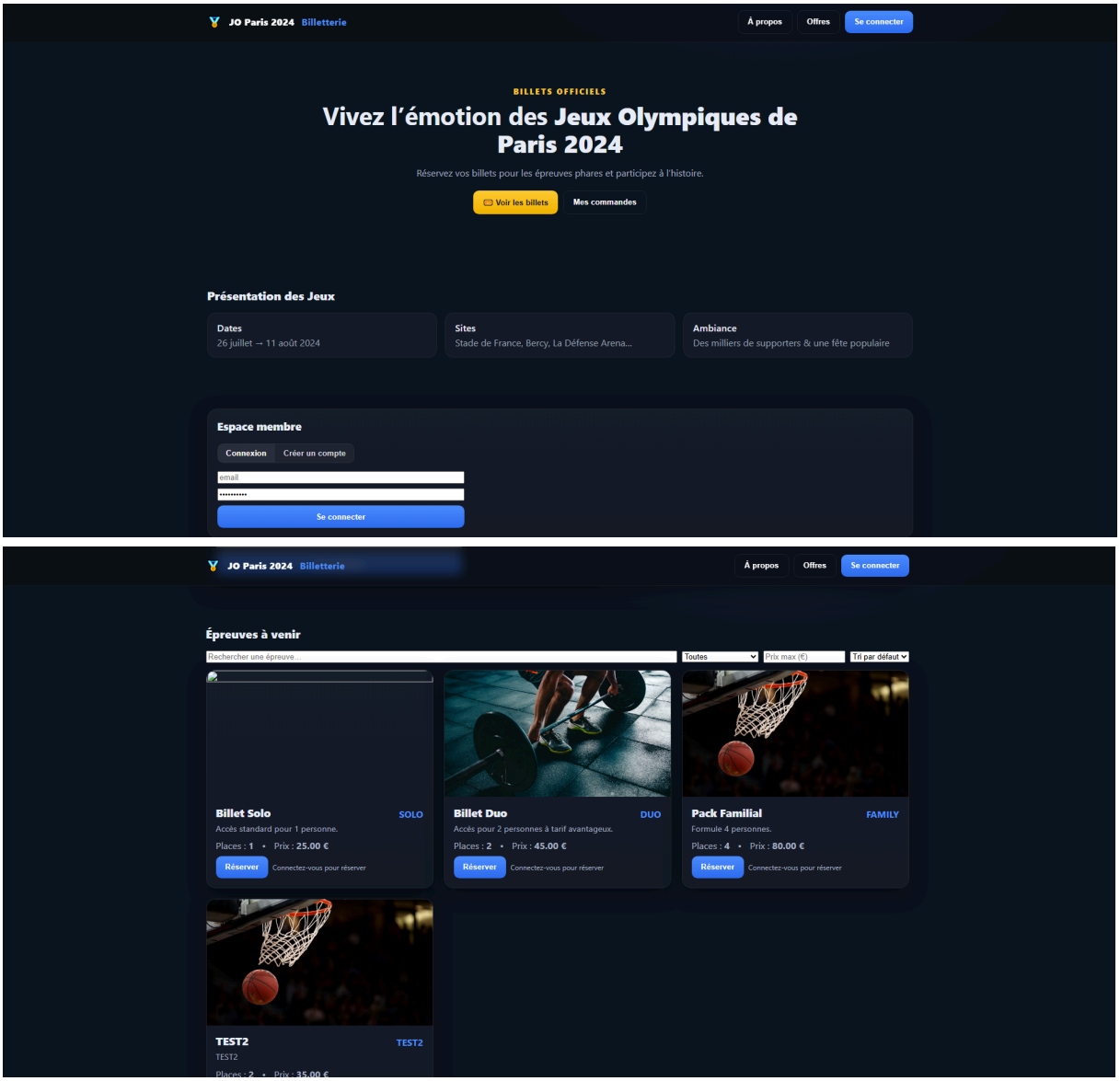
Cette documentation technique résume les choix de conception et les mesures de sécurité prises pour garantir la fiabilité du projet.
L'architecture choisie (React + Spring Boot + JWT) permet d'obtenir une application moderne, rapide et adaptée à une utilisation réelle.

Manuel d’utilisation de l’application de billetterie

Ce manuel a pour but d’expliquer pas à pas comment utiliser l’application de billetterie en ligne développée pour la gestion des offres, des commandes et du contrôle des billets.

1. Page d’accueil et accès à l’application

L’application s’ouvre via un navigateur web à l’adresse locale du projet (par défaut : http://localhost:5173).
Sur la page d’accueil, on accède à la billetterie et aux différentes offres disponibles (billet solo, duo, familial, etc.).



Page d’accueil de la billetterie affichant les offres disponibles

2. Création de compte et connexion

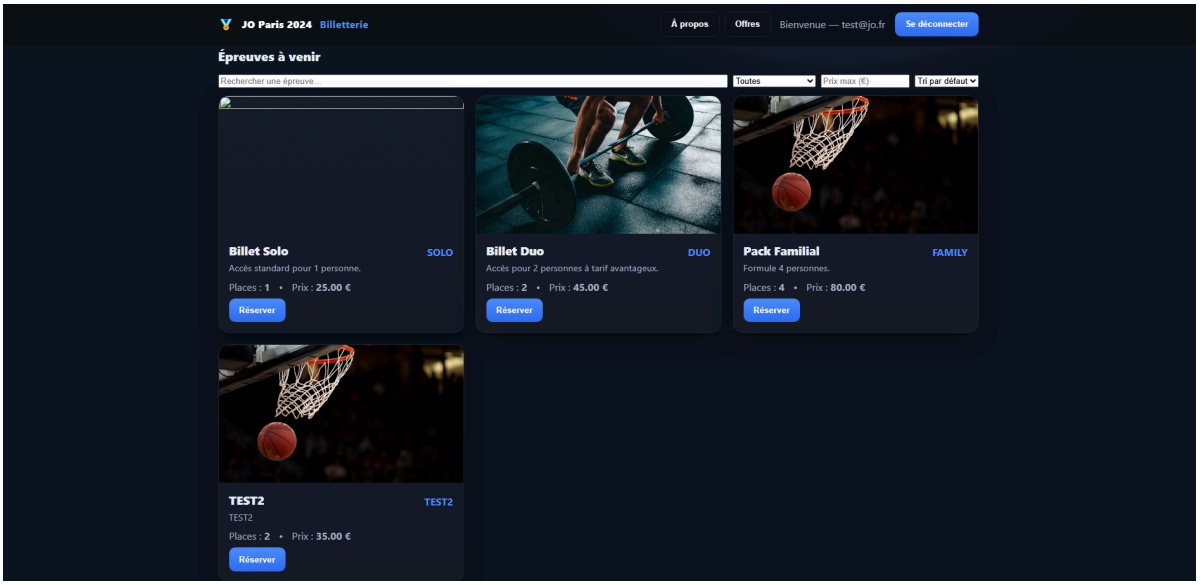
Avant d'acheter un billet, l'utilisateur doit :
- Cliquer sur le bouton "S'inscrire".
- Remplir le formulaire avec son nom, prénom, e-mail et mot de passe.
- Se connecter ensuite avec son e-mail et mot de passe.

Lors de la connexion, un code OTP (mot de passe à usage unique) est envoyé par e-mail pour valider l'accès.
Ce code ajoute une sécurité supplémentaire au compte utilisateur.

Formulaire de connexion/inscription avec authentification par OTP.

3. Navigation utilisateur

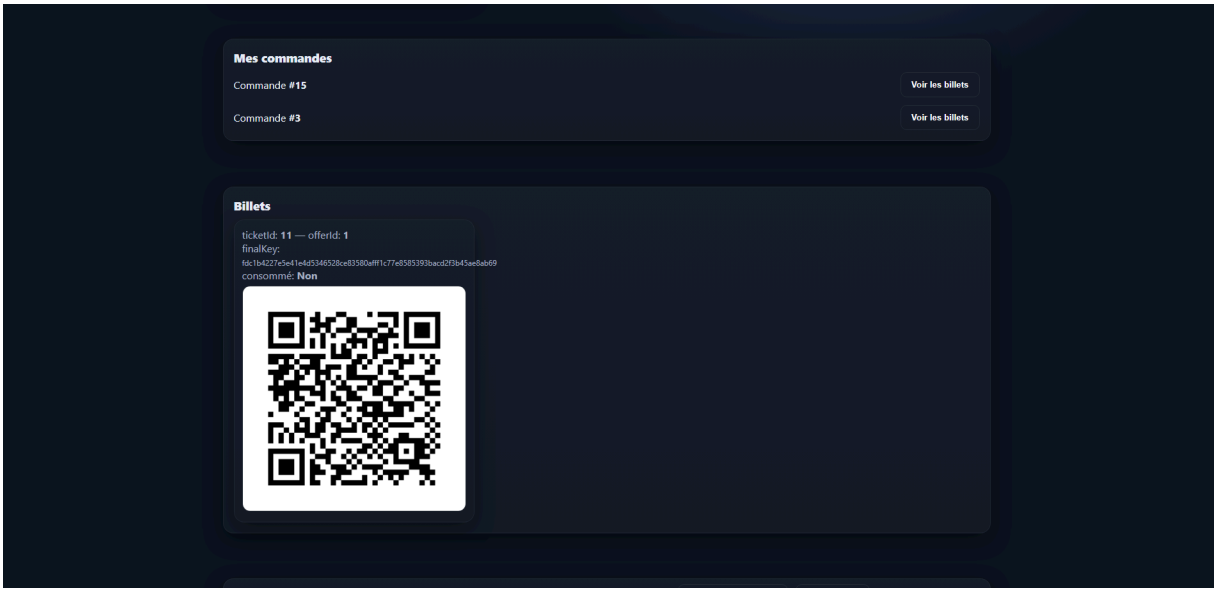
Une fois connecté, l'utilisateur accède à son espace personnel :
- Il peut voir les offres disponibles avec le nombre de places, le prix et la description.
- En cliquant sur "Réserver", il peut ajouter une ou plusieurs offres à son panier.
- Le bouton "Mes commandes" lui permet de consulter ses achats précédents et leurs tickets associés.



Interface utilisateur après connexion, affichage des offres.

4. Achat d'un billet

Lorsqu'un utilisateur choisit une offre :
- Il clique sur "Réserver".
- Il valide le paiement fictif (dans cette version, il n'y a pas encore d'intégration réelle).
- Une commande est créée automatiquement dans le système.
- L'utilisateur reçoit un QR Code associé à son billet.



Validation d'une commande et génération du ticket associé / Exemple de QR Code généré pour un ticket.

5. Espace administrateur

L'administrateur dispose d'un tableau de bord accessible via son compte (e-mail administrateur). Il peut :
- Consulter le total des ventes et le nombre de tickets vendus.
- Créer, modifier ou supprimer des offres (ex : "Billet Solo", "Pack Famille", etc.).
- Activer ou désactiver une offre.

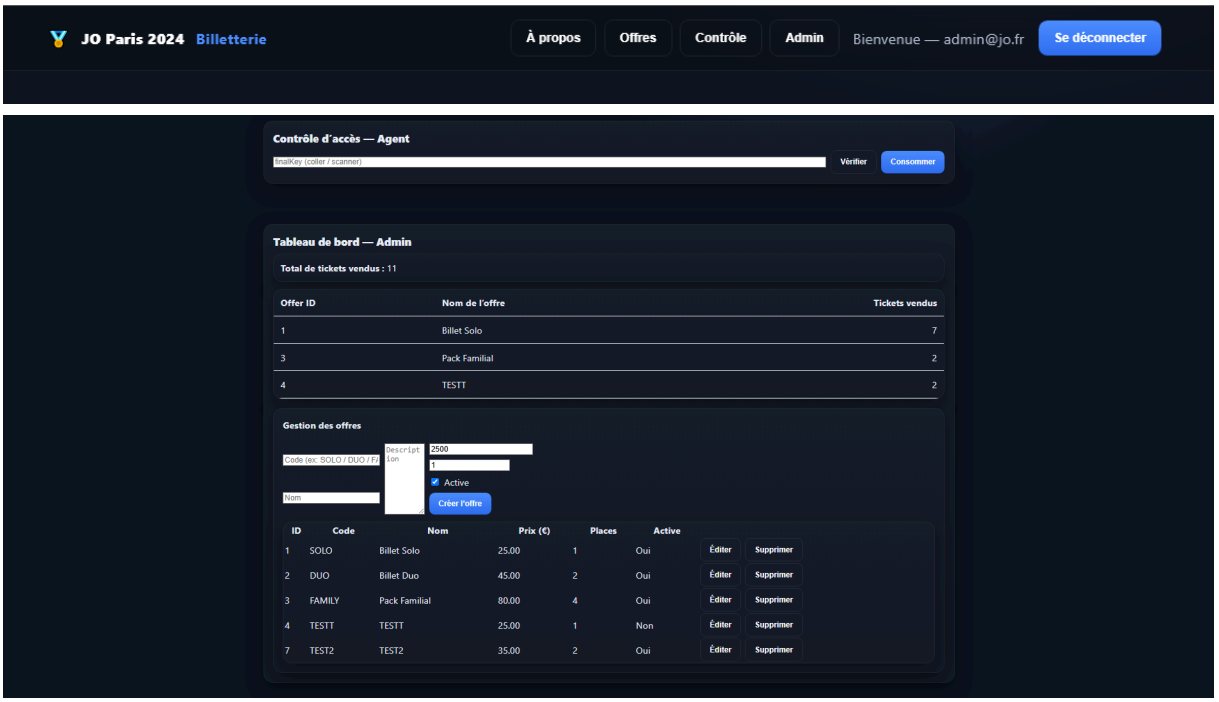
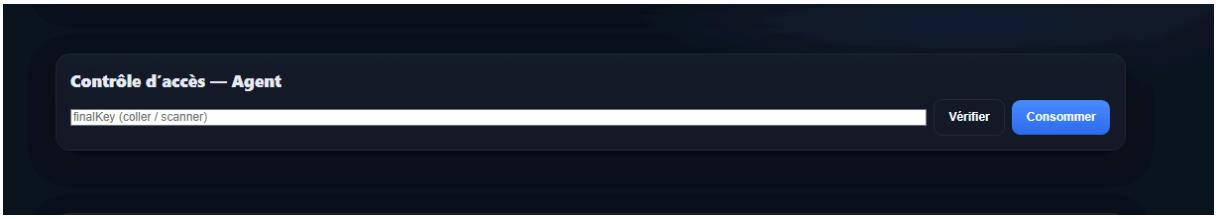


Tableau de bord administrateur – gestion des offres et statistiques.

6. Espace agent (contrôle des tickets)

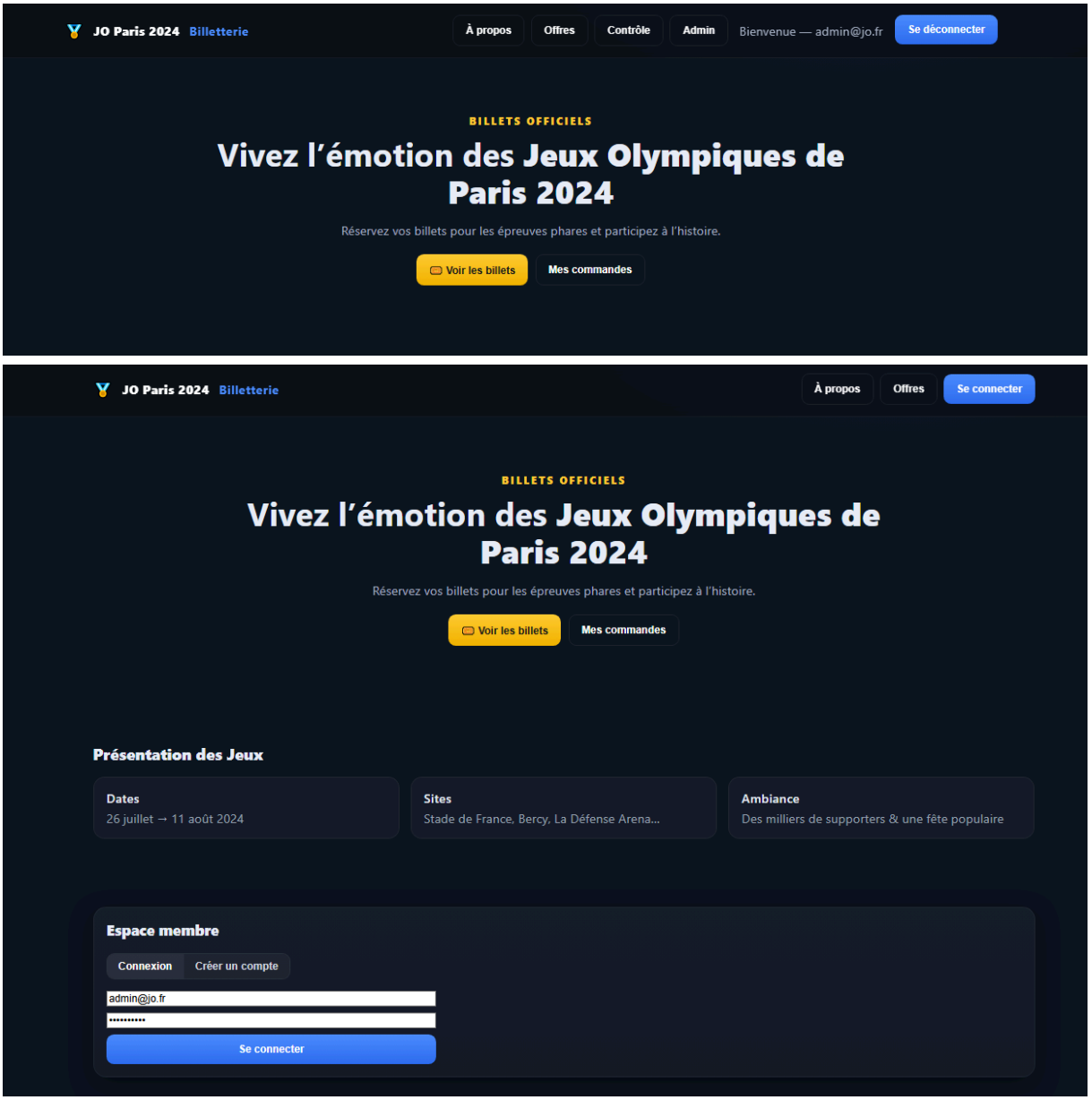
Les agents d'événement peuvent se connecter avec un compte spécial "AGENT". Depuis leur tableau de bord :
- Ils peuvent scanner ou entrer le code d'un ticket.
- L'application vérifie automatiquement si le billet est valide.
- Si le ticket est consommé, il est marqué comme "utilisé".



Espace agent pour vérifier ou consommer un ticket.

7. Déconnexion

L'utilisateur peut se déconnecter à tout moment via le bouton "Se déconnecter" situé en haut à droite.
Cela supprime le token d'authentification stocké dans le navigateur.



Bouton de déconnexion de l'utilisateur/ Déconnexion de l'utilisateur.

8. Tests et déploiement

- Tests unitaires et d'intégration

Pendant le développement du back-end, j'ai mis en place plusieurs tests à l'aide de JUnit 5 et Spring Boot Test afin de vérifier le bon fonctionnement des principales fonctionnalités :
- Test de création d'un utilisateur et vérification du hachage du mot de passe.
- Test de génération et validation du code OTP lors de la connexion.
- Test d'ajout, modification et suppression d'offres dans la base de données.
- Test de création de commandes et de génération automatique de tickets.
- Test de la vérification et consommation des tickets par leur code unique.

Ces tests m'ont permis d'assurer la stabilité du code avant de connecter le front-end (React).

- Résultats et couverture du code

Les tests couvrent les classes principales du projet (services, contrôleurs, repositories).
La couverture globale est estimée à environ 70 %, ce qui est satisfaisant pour un projet de ce type.
Les tests ont tous été effectués en environnement local (serveur Spring Boot sur localhost:8081).

- Déploiement

L'application a été testée en local :
- Back-end accessible via http://localhost:8081
- Front-end accessible via http://localhost:5173

L'application a été testée en ligne :
- Front-end accessible via https://ticketing-jo-2024-2.onrender.com/

Ce déploiement permet de rendre l'application accessible publiquement, sans configuration manuelle du serveur local.

9. Conclusion

Ce manuel d'utilisation permet à tout utilisateur (client, administrateur ou agent) de comprendre le fonctionnement complet de l'application.
L'interface simple et fluide rend la navigation intuitive, et les contrôles de sécurité garantissent un usage fiable de la billetterie.

Note de synthèse – Projet “Ticketing JO Paris 2024”

Ce projet m’a permis de mettre en pratique l’ensemble des compétences vues durant la formation, aussi bien sur la partie back-end (Spring Boot) que sur le front-end (React + Vite).
L’objectif était de créer une application complète de billetterie pour les Jeux Olympiques 2024, avec un système de comptes utilisateurs, de commandes et une interface d’administration sécurisée.

Déroulement du projet

Au début du projet, la mise en place du back-end avec Spring Boot m’a demandé pas mal de temps, notamment pour bien comprendre la gestion des rôles, la sécurité avec JWT et le chiffrement des mots de passe via PasswordEncoder.
La structure du projet et la communication entre les entités (User, Offer, Order, Ticket) ont également nécessité plusieurs ajustements avant d’obtenir une base de données cohérente et fonctionnelle.

Ensuite, l’intégration du front-end React a été une autre étape importante. J’ai dû gérer les appels API, la gestion du token JWT côté client, et l’affichage conditionnel selon le rôle de l’utilisateur (admin / user).
La partie la plus complexe a été de faire communiquer correctement le front et le back, surtout à cause des problèmes de CORS, des URLs entre la version locale et celle déployée sur Render, et de la gestion des environnements (dev/prod).

Sécurité et gestion des accès

La mise en place du système d’authentification avec JWT et la création d’un compte admin automatique (via un seeder) ont été des points clés.
Cela m’a permis de comprendre la logique de sécurisation d’une API REST, et comment restreindre certaines routes selon le rôle.

Déploiement

Le déploiement sur Render a été une vraie étape d’apprentissage : configuration des variables d’environnement, liaison à la base PostgreSQL hébergée, et test du fonctionnement complet du site en ligne.
C’était un peu long au début, mais j’ai fini par comprendre le fonctionnement du déploiement continu et la gestion des profils (dev / prod).

Bilan

Ce projet m'a permis d'acquérir une vision claire et complète du développement d'une application web moderne, de la conception du code jusqu'au déploiement en production.
J'ai pu mettre en pratique l'ensemble des compétences vues en formation, notamment en Java Spring Boot pour la partie back-end et React pour le front-end.

J'ai pris conscience de l'importance de la rigueur dans la configuration (CORS, base de données, seeders, variables d'environnement, etc.) ainsi que de la bonne communication entre le front et le back, deux aspects essentiels pour garantir la stabilité d'un projet.
Les principales difficultés ont concerné l'authentification JWT, la migration des données vers la base en ligne et le déploiement sur Render. Ces obstacles m'ont toutefois permis de progresser, de mieux comprendre le fonctionnement des environnements dev et prod, et de renforcer ma capacité à résoudre des problèmes techniques par moi-même.

Le résultat final rempli les objectifs fixés initialement, à savoir :

- le site est entièrement fonctionnel,
- le compte administrateur permet de gérer les offres,
- les utilisateurs peuvent s'inscrire, se connecter et effectuer des commandes,
- et le déploiement sur Render fonctionne parfaitement pour le front et le back.

Pour conclure, ce projet m'a aidé à développer mon autonomie, mon sens de l'organisation (grâce à l'utilisation de Trello) et à consolider mes compétences en développement full-stack Java / React.
Il représente pour moi une expérience complète et formatrice, qui m'a donné davantage de confiance pour la suite de mon parcours professionnel.