

DOCUMENT TECHNIQUE FINAL LOOTOPIA

**Réalisé par :**

**Amina LOZI**

**Mariya ABAAKIL**

**Année 2024/2025**

[***loziamina/lootopia\_v1***](https://github.com/loziamina/lootopia_v1)

Table des matières

[**1.** **Introduction générale du projet Lootopia** 3](#_Toc203844819)

[1.1 Contexte du projet 3](#_Toc203844820)

[1.2 Objectifs stratégiques et fonctionnels du projet 3](#_Toc203844821)

[**2.** **Architecture technique** 4](#_Toc203844822)

[2.1 Modélisation des entités et structure de la base de données 4](#_Toc203844823)

[2.2 Description des entités 5](#_Toc203844824)

[2.3 Synthèse des relations entre entités 6](#_Toc203844825)

[2.4 Choix techniques et architecture logicielle 6](#_Toc203844826)

[**3. Mise en place et configuration du projet** 9](#_Toc203844827)

[3.1 Définition des variables d’environnement 9](#_Toc203844828)

[3.2 Procédure d’installation locale 9](#_Toc203844829)

[**4. Fonctionnalités de l’application** 10](#_Toc203844830)

[4.1 Authentification et gestion des comptes utilisateurs 10](#_Toc203844831)

[4.2 Accueil administrateur : suivi des chasses crées 12](#_Toc203844832)

[4.3 Navigation vers les détails d’une chasse et la gestion des avis 13](#_Toc203844833)

[4.4 Carte interactive et expérience de jeu 14](#_Toc203844834)

[4.5 Gestion des utilisateurs 15](#_Toc203844835)

[4.6 Gestion des chasses au trésor 16](#_Toc203844836)

[4.7 Gestion du profil administrateur 18](#_Toc203844837)

[4.8 Visualisation des chasses côté utilisateur : 19](#_Toc203844838)

[4.9 Visualisation des chasses côté utilisateur : 19](#_Toc203844839)

[4.10 Système d’avis utilisateur 20](#_Toc203844840)

[4.11 Rejoindre la chasse et carte interactive 21](#_Toc203844841)

[**5. Documentation API :** 22](#_Toc203844842)

[**6. Annexe :** 23](#_Toc203844843)

# **Introduction générale du projet Lootopia**

### Contexte du projet

Le projet **Lootopia** s’inscrit dans un contexte de transformation numérique croissante, où les utilisateurs sont en quête d’activités innovantes mêlant technologie et divertissement. Face à l’émergence d’une demande pour des expériences immersives qui transcendent les frontières du virtuel et du réel, Lootopia propose une réponse originale : une plateforme interactive de chasses au trésor géolocalisées.

Conçue pour enrichir l’univers du loisir numérique, Lootopia offre aux utilisateurs la possibilité de vivre des aventures personnalisées, tout en exploitant les apports technologiques contemporains tels que la **géolocalisation** et la **réalité augmentée**. Cette double approche vise à stimuler l’engagement des participants en les plongeant dans des scénarios de jeu immersifs, ancrés dans leur environnement réel.

Parallèlement, la plateforme s’adresse également aux **organisateurs d’événements**, leur fournissant une solution digitale intuitive pour concevoir, publier et suivre des chasses au trésor adaptées à leurs besoins. Elle permet ainsi de renforcer leur visibilité, de proposer des parcours interactifs sur mesure, et d’enrichir l’expérience utilisateur à travers des mécaniques de jeu modernes.

### Objectifs stratégiques et fonctionnels du projet

L’ambition de Lootopia est double : d’une part, proposer aux utilisateurs une expérience ludique innovante à travers des chasses au trésor interactives ; d’autre part, fournir aux organisateurs une plateforme complète de conception et de gestion d’événements numériques.

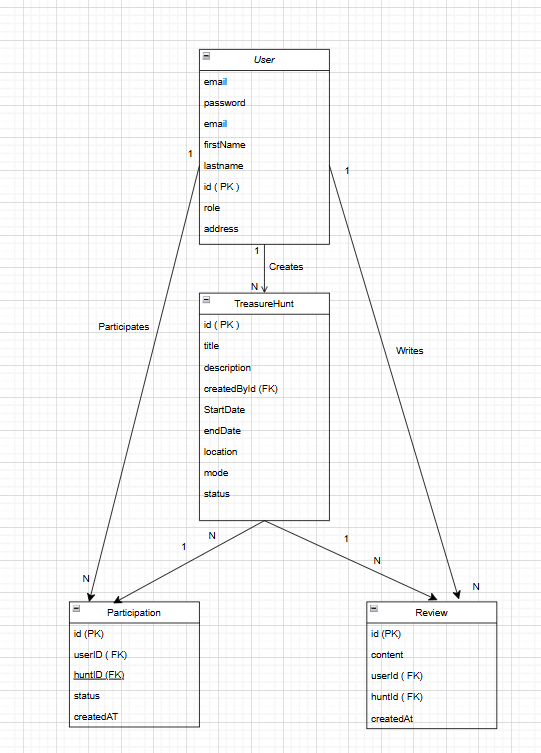
Les objectifs principaux se déclinent comme suit :

* **Proposer une plateforme ludique et interactive** : offrir une interface intuitive permettant aux utilisateurs de participer à des chasses au trésor où l’objectif principal est de trouver un trésor caché sur une carte interactive. À chaque découverte, le trésor change d’emplacement, offrant une dynamique de jeu renouvelée et stimulant la motivation du joueur à continuer sa quête.
* **Mettre à disposition des outils de création personnalisée :** permettre aux administrateurs et organisateurs d’événements de concevoir des parcours de chasse adaptés à divers contextes avec des fonctionnalités de personnalisation avancées.
* **Garantir un environnement sécurisé :** intégrer un système d’authentification robuste basé sur les technologies JWT/JWK, gérer les droits d’accès par rôles (utilisateur, administrateur) et assurer la protection des données personnelles tout au long de l’expérience.

# **Architecture technique**

### Modélisation des entités et structure de la base de données

La conception du modèle de données repose sur quatre entités principales : ***User***, ***TreasureHunt***, ***Participation*** et ***Review***. Le diagramme ci-dessous illustre les relations entre ces entités ainsi que les principaux attributs associés.

****

### Description des entités

1. **Utilisateur (*User*) :**

L'entité ***User*** représente les utilisateurs inscrits sur la plateforme. Elle contient les attributs essentiels à l'identification et à la gestion des comptes :

* **email, password, firstName, lastName, address**
* Le champ **role** détermine le type d'utilisateur : **ADMIN** ou **USER**.

Relations :

* Un utilisateur peut **créer plusieurs chasses au trésor** ***(TreasureHunt).***
* Il peut **participer à plusieurs chasses** via l’entité ***Participation.***
* Il peut **laisser plusieurs avis** ***(Review)*** sur les chasses terminées.

1. **Chasse au trésor** *(****TreasureHunt****)*

Cette entité représente une chasse créée par un administrateur. Elle comprend les champs :

* **title**, **description**, **startDate**, **endDate**, **location**, **mode**, **status**

Le champ **createdById** est une clé étrangère pointant vers l'utilisateur créateur.

Relations :

* Une chasse est **liée à un créateur unique** ***(User).***
* Elle peut **accueillir plusieurs participations** ***(Participation).***
* Elle peut **recevoir plusieurs avis** ***(Review).***

1. **Particitpation**

L'entité ***Participation*** fait le lien entre un utilisateur et une chasse donnée. Elle permet de suivre la progression de l’utilisateur à travers les statuts suivants :

* ***PENDING :*** l'utilisateur a consulté le détail d’une chasse.
* ***IN\_PROGRESS :*** l'utilisateur a commencé la chasse sans trouver le trésor.
* ***COMPLETED :*** En moins un trésor a été trouvé.

Attributs :

* ***userId :*** référence à l’utilisateur
* ***huntId :*** référence à la chasse
* ***status :*** état de progression
* ***createdAt :*** date de participation

1. **Avis (Review)**

Un Review est un retour laissé par un utilisateur après avoir terminé une chasse. Il contient :

* ***content :*** le texte de l’avis
* ***createdAt :*** date de publication
* ***userId, huntId :*** références croisées vers l’utilisateur et la chasse

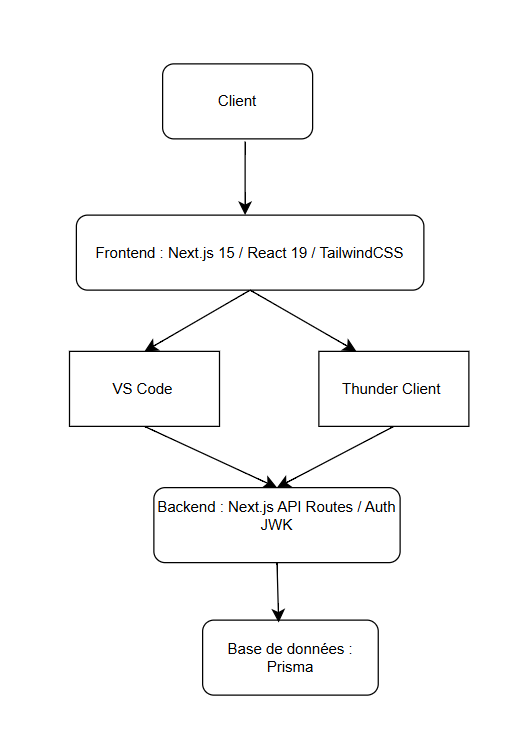
### 2.3 Synthèse des relations entre entités

* **User ↔ TreasureHunt : 1-N** → Un utilisateur peut créer plusieurs chasses.
* **User ↔ Participation : 1-N** → Un utilisateur peut rejoindre plusieurs chasses.
* **User ↔ Review : 1-N** → Un utilisateur peut soumettre plusieurs avis.
* **TreasureHunt ↔ Participation : 1-N** → Une chasse peut avoir plusieurs participants.
* **TreasureHunt ↔ Review : 1-N** → Une chasse peut recevoir plusieurs avis.

### 2.4 Choix techniques et architecture logicielle

2.4.1 Vue d’ensemble de l’architecture

L’architecture de l’application Lootopia repose sur une stack moderne orientée JavaScript, combinant une interface web réactive, une couche API légère et une base de données relationnelle optimisée par Prisma.



Le parcours utilisateur débute côté **client**, via une interface web développée avec **Next.js 15**, **React 19** et **TailwindCSS**. Cette interface interagit avec un backend intégré dans **Next.js** (via les API Routes), sécurisé par un système d’authentification basé sur **JWK**. Les données sont centralisées et gérées à travers **Prisma ORM** connecté à une base **PostgreSQL**.

2.4.2 Outils et environnement de développement

1. **Visual Studio Code**

L’éditeur de code principal utilisé est **VS Code**, choisi pour sa légèreté, ses nombreuses extensions dédiées à **JavaScript/React**, et ses fonctionnalités intégrées de débogage et de contrôle de version. Il permet une productivité accrue grâce à son intégration fluide avec **GitHub, Prisma et TailwindCSS**.

1. **Thunder Client**

Thunder Client est un outil léger mais puissant utilisé pour tester les **API REST** de manière rapide et visuelle. Il a permis de vérifier les endpoints backend sans quitter l’environnement de développement. Simple d’utilisation, il remplit les mêmes fonctions que Postman, tout en s’intégrant parfaitement à **VS Code**.

1. **Github**

La gestion du code source est assurée via GitHub, permettant le versionnage, la collaboration en équipe, l’organisation des tâches via les issues, et la revue de code à travers les pull requests. Cela assure une traçabilité complète des évolutions du projet.

2.4.2 Choix du frontend

Le développement de l’interface utilisateur repose sur les outils suivants :

* **Next.js 15** : framework React moderne intégrant le rendu côté serveur (SSR) et la génération statique (SSG), assurant une excellente performance et un bon référencement.
* **React 19** : bibliothèque de composants réactifs permettant une construction fluide d’interfaces interactives.
* **TailwindCSS** : framework CSS *utility-first*, facilitant un design rapide, propre et cohérent sans surcharge de styles inutiles.

2.4.3 Choix du backend

Le développement de l’interface utilisateur repose sur les outils suivants :

* **Next.js 15 :** framework React moderne intégrant le rendu côté serveur (SSR) et la génération statique (SSG), assurant une excellente performance et un bon référencement.
* **React 19 :** bibliothèque de composants réactifs permettant une construction fluide d’interfaces interactives.
* **TailwindCSS :** framework CSS utility-first, facilitant un design rapide, propre et cohérent sans surcharge de styles inutiles.

2.4.4 Choix du backend

Le backend repose sur les **API Routes de Next.js**, offrant une architecture unifiée dans un seul projet. Cette approche permet une séparation claire entre logique métier et interface, tout en simplifiant le déploiement.

2.4.5 Gestion de l’authentification avec JWK

Pour sécuriser l’accès aux données, l’application utilise une authentification par JSON Web Key (JWK) associée aux JWT. Ce mécanisme permet une validation fiable des sessions sans stockage côté serveur.

Les avantages de JWK dans ce projet sont multiples :

* **Sécurité renforcée :** les clés sont centralisées et protégées, garantissant que seuls les JWT valides sont acceptés.
* **Gestion flexible des clés :** les clés publiques peuvent être mises à jour sans affecter les clients, facilitant la rotation.
* **Interopérabilité :** supporté par de nombreuses bibliothèques, ce format s’intègre facilement à divers systèmes.
* **Flexibilité étendue :** prise en charge de plusieurs clés simultanées, utile pour la montée en charge ou la gestion multi-services.

Grâce à ce système, les données des utilisateurs sont protégées tout en maintenant une expérience fluide et sans friction.

2.4.6 Base de données et ORM

Le choix s’est porté sur Prisma ORM, un outil moderne pour interagir avec une base PostgreSQL. Ce choix repose sur les critères suivants :

* **Facilité d’utilisation :** syntaxe intuitive, génération automatique du client Prisma.
* **Gestion simplifiée des migrations :** outils CLI pour appliquer et suivre les évolutions du schéma de base.
* **Haute performance :** requêtes SQL optimisées, adaptées aux applications à fort trafic.
* **Intégration JavaScript/TypeScript :** meilleure cohérence avec Next.js, réduction des erreurs de type.

## **3. Mise en place et configuration du projet**

### 3.1 Définition des variables d’environnement

Afin d’isoler les informations sensibles et de personnaliser la configuration en fonction de l’environnement, le projet Lootopia utilise un fichier **.env**. Ce fichier stocke les constantes nécessaires à l'exécution de l’application en local.

Les variables indispensables sont :

|  |  |  |
| --- | --- | --- |
| **Clé** | **Rôle** | **Variables à définir** |
| ***DATABASE\_URL*** | Contient la chaîne de connexion à la base de données PostgreSQL utilisée par Prisma. | *postgresql://user:password@localhost:5432/lootopia* |
| ***JWT\_SECRET*** | Clé secrète utilisée pour signer et vérifier les JSON Web Tokens | *your\_jwt\_secret\_key* |
| ***NEXT\_PUBLIC\_BASE\_URL*** | Indique l’URL publique de l'application, utilisée notamment dans les appels API côté client. | *http://localhost:3000* |

### 3.2 Procédure d’installation locale

1. **Cloner le dépôt Git**

*git clone https://github.com/loziamina/lootopia\_v1.git*

*cd lootopia\_v1*

1. **Installer les dépendances**

*npm install*

1. **Créer le fichier .env**

Copiez les variables ci-dessous et adaptez **DATABASE\_URL** à vos identifiants locaux :

*DATABASE\_URL=postgresql://<user>:<password>@localhost:5432/lootopia*

*JWT\_SECRET=<votre\_clé\_secrète>*

*NEXT\_PUBLIC\_BASE\_URL=http://localhost:3000*

1. **Initialiser la base de données et appliquer les migrations Prisma**

*npx prisma migrate dev*

Cette commande :

* Crée la base selon le fichier schema.prisma,
* Applique les migrations initiales,
* Génère le client Prisma.

1. **Démarrer le serveur de développement**

*npm run dev*

L’application Next.js est alors accessible sur [**http://localhost:3000**](http://localhost:3000).

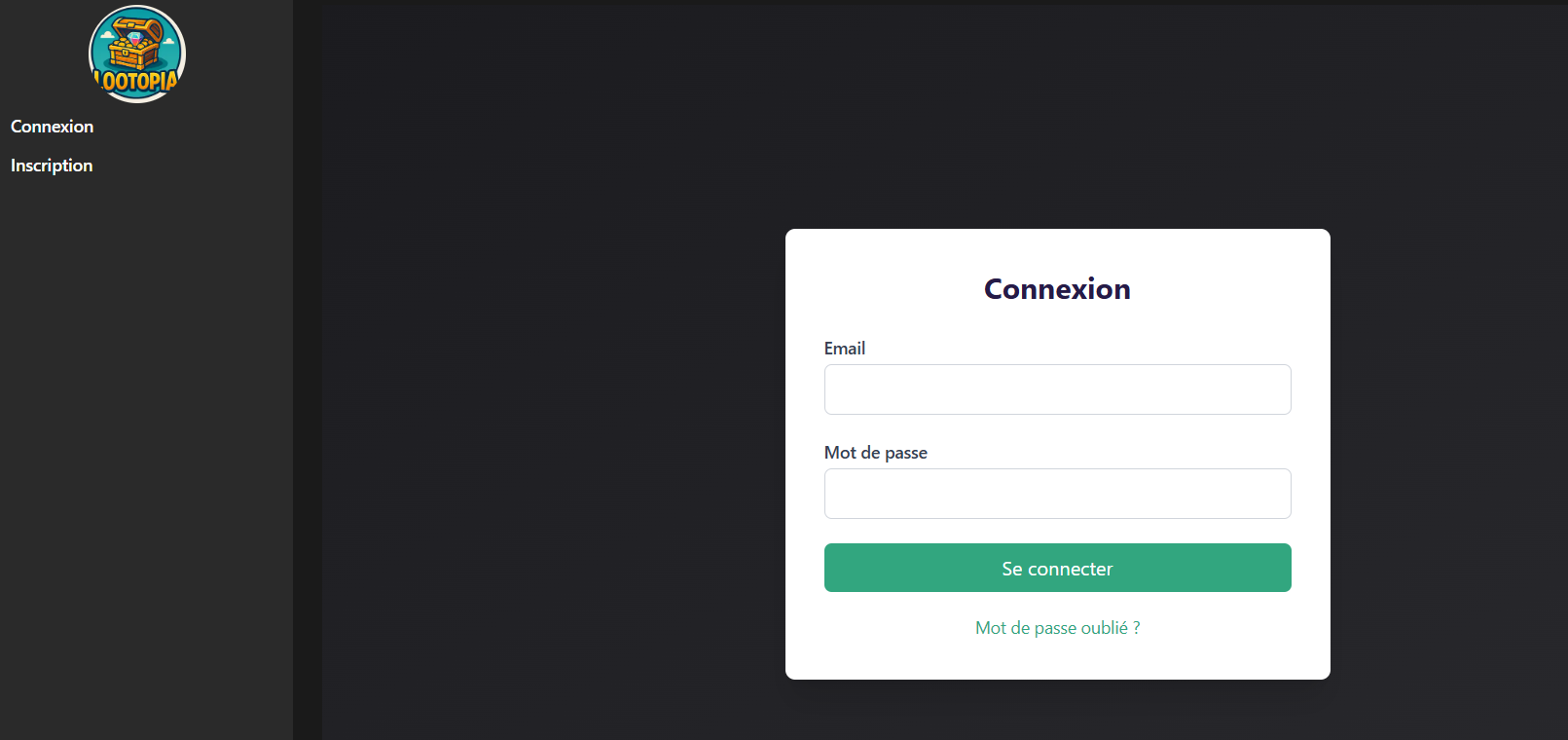
## **4. Fonctionnalités de l’application**

### 4.1 Authentification et gestion des comptes utilisateurs

L’accès à la plateforme Lootopia est protégé par un système d’authentification sécurisé, garantissant à chaque utilisateur une gestion privée de son compte et de ses données.

1. **Interface de connexion**

L’utilisateur accède à son espace personnel via une interface de connexion claire et minimaliste, où il doit renseigner son adresse email et son mot de passe.



Un lien « Mot de passe oublié ? » est également disponible pour déclencher une procédure de réinitialisation.

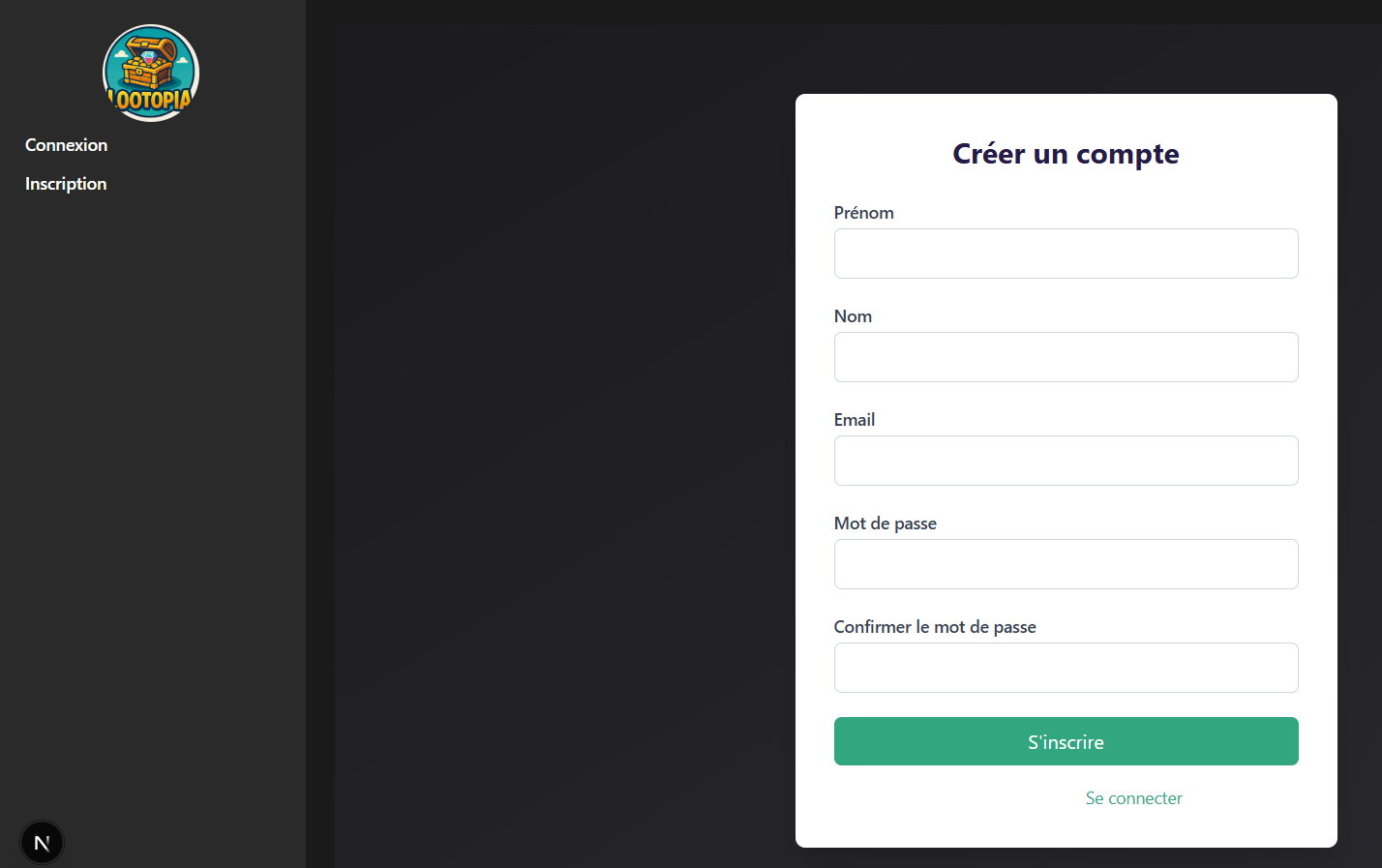
Le mécanisme d’authentification repose sur la technologie des **JSON Web Tokens (JWT)**, sécurisés par des **JSON Web Keys (JWK)**. Cette architecture permet :

* Une **vérification d'identité sans stockage côté serveur**,
* Une **sécurisation des sessions** à chaque requête,
* Une gestion légère et performante des accès utilisateurs.

1. **Création de compte (inscription)**

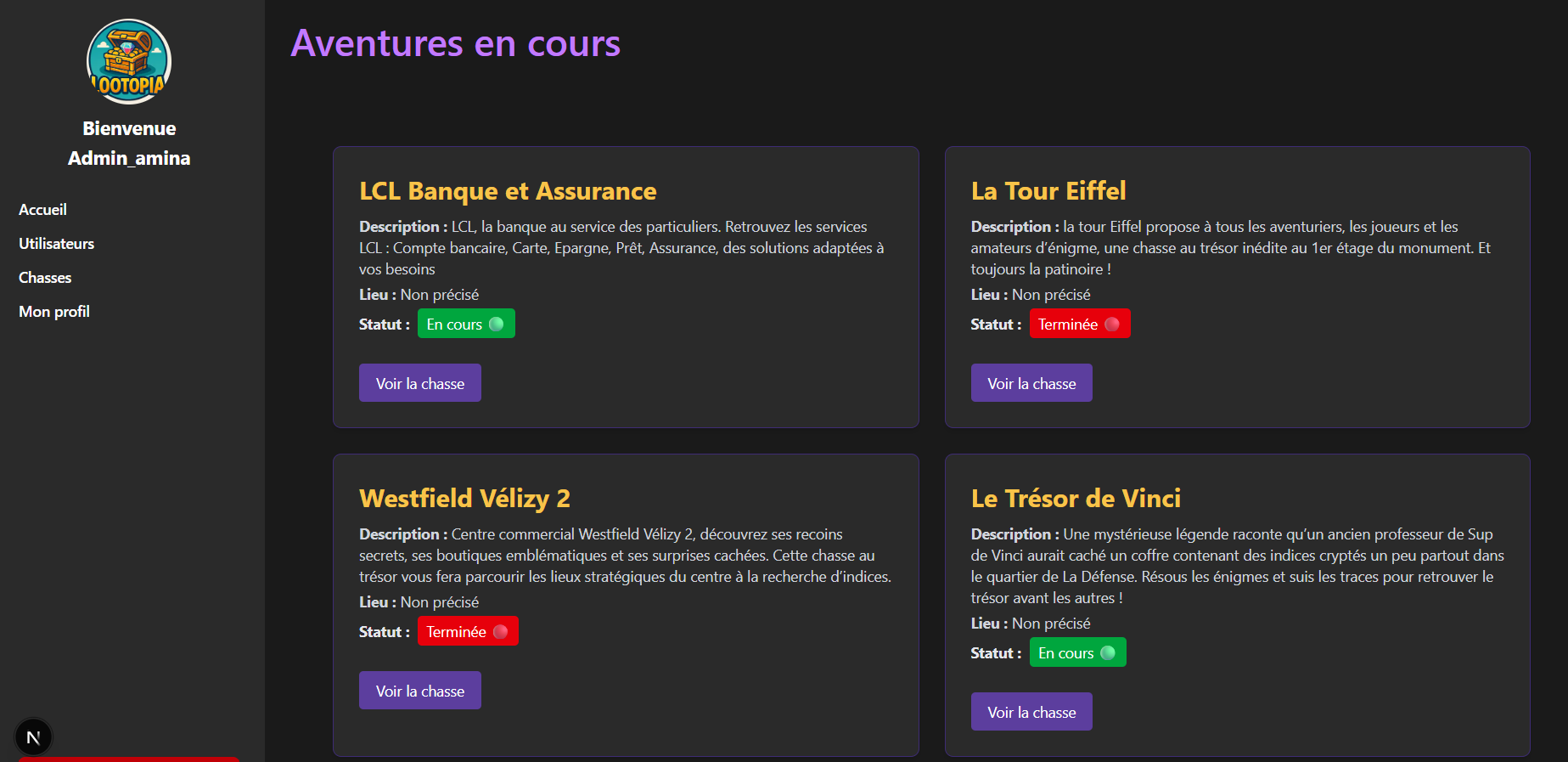
Les nouveaux utilisateurs peuvent créer leur compte via un formulaire d’inscription, en renseignant :

* Leur prénom,
* Leur nom,
* Leur adresse email,
* Un mot de passe sécurisé,



### 4.2 Accueil administrateur : suivi des chasses crées

Après la connexion, l’administrateur est automatiquement redirigé vers une page d’accueil dédiée, qui lui offre une vue centralisée de toutes les chasses au trésor qu’il a créées sur la plateforme.



* ***Affichage des chasses et suivi par statut :***

L’interface regroupe l’ensemble des chasses publiées sous forme de cartes individuelles. Chacune contient :

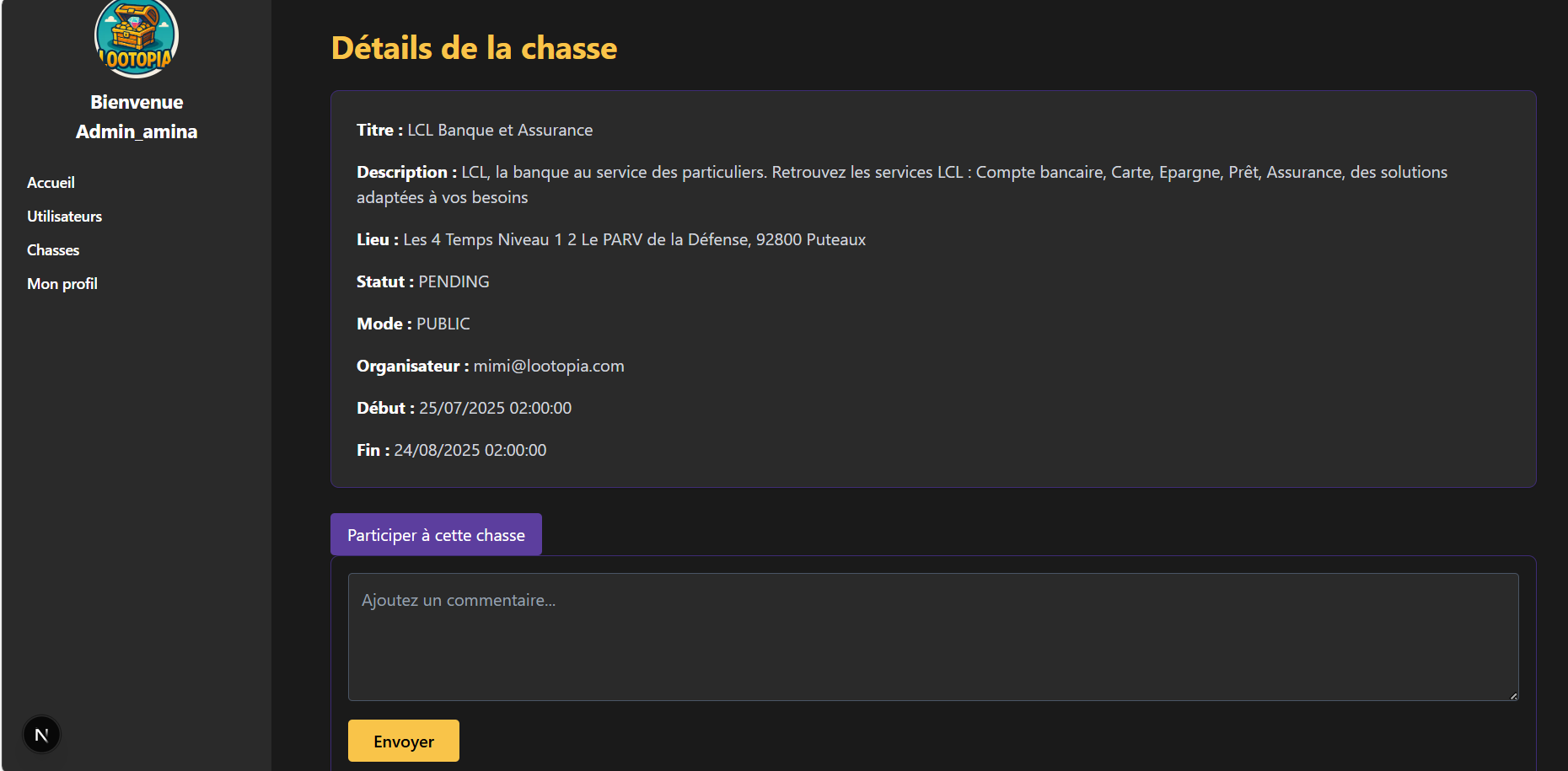
* Le titre de la chasse
* Une description concise
* Le lieu
* Le statut actuel de la chasse (affiché par code couleur) :
  + En attente : aucun participant n’a encore commencé la chasse
  + En cours : un ou plusieurs participants sont actuellement actifs
  + Terminée : la chasse a été complétée par au moins un utilisateur
* Un bouton **«Voir la chasse »** est affiché sous chaque carte. Lors d’un clic, l’administrateur est redirigé vers une page de **détails de la chasse**

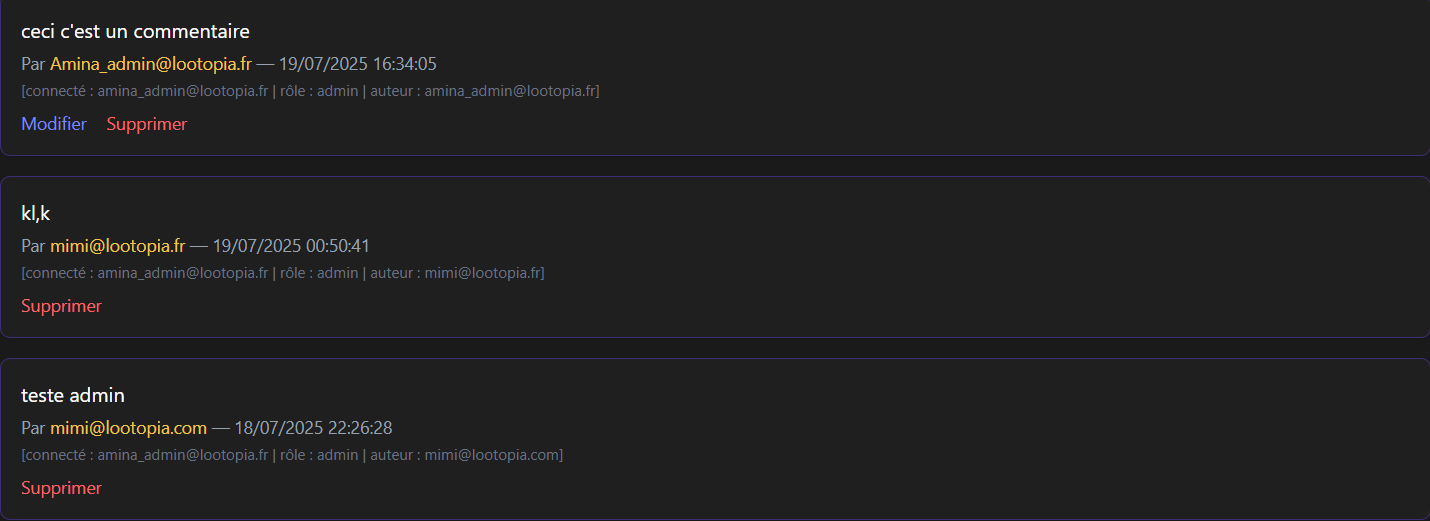
Cette visualisation permet à l’administrateur de suivre en un coup d’œil l’état de progression de ses événements.

### 4.3 Navigation vers les détails d’une chasse et la gestion des avis

Dans la page de détails d’une chasse, l’administrateur peut :

* **Consulter les informations complètes de la chasse :** titre, description, lieu, organisateur, statut, etc.
* Cliquer sur le **bouton « Participer à cette chasse »** pour accéder à la carte interactive et tenter lui-même de trouver le trésor, comme un joueur.
* Gérer les avis :
  + Rédiger un commentaire personnel sur la chasse.
  + Modifier ou supprimer son propre commentaire.
  + Supprimer les commentaires rédigés par d’autres utilisateurs.





### 4.4 Carte interactive et expérience de jeu

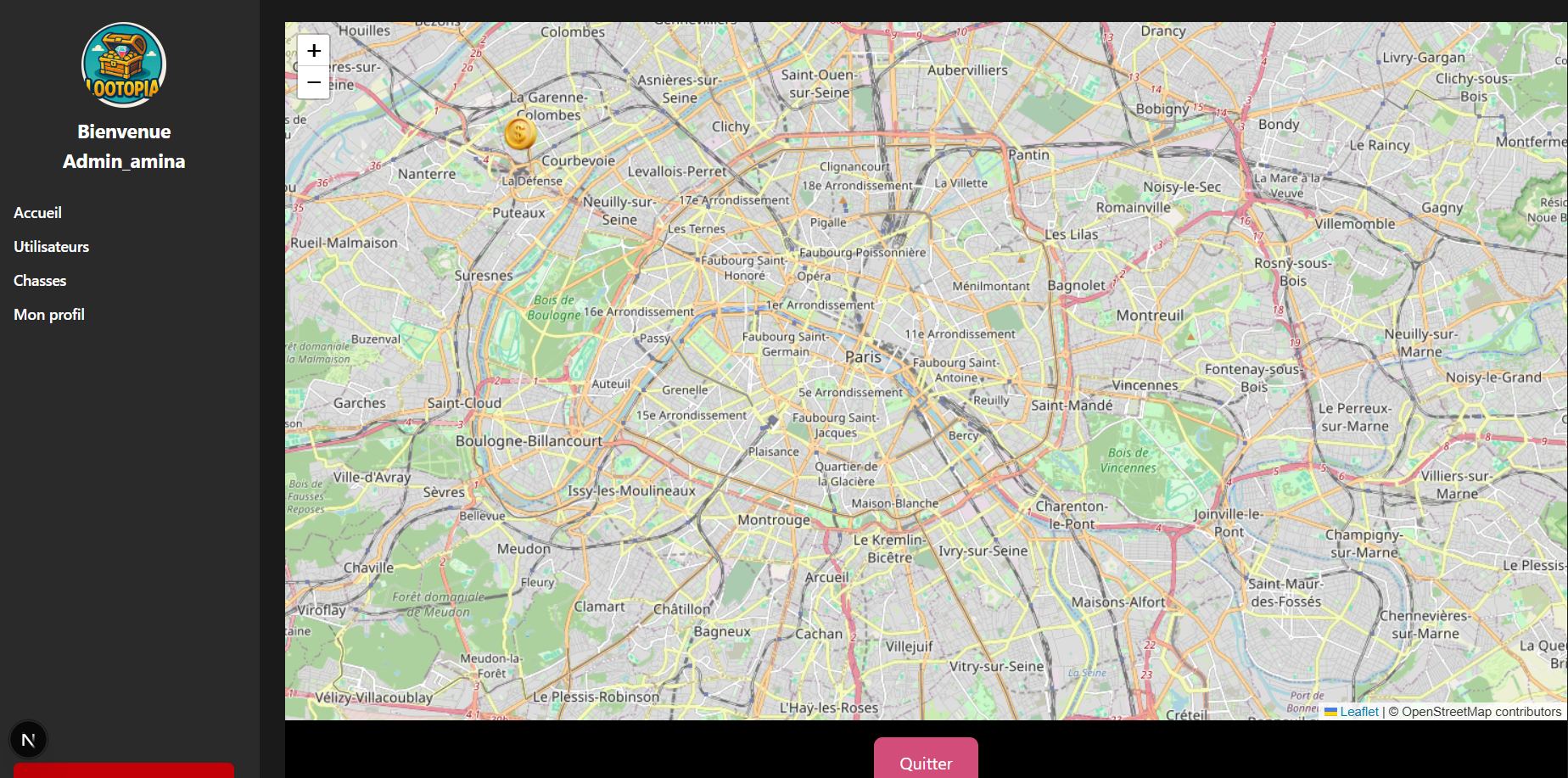
Lootopia intègre une carte interactive basée sur OpenStreetMap, rendue dynamique grâce à la bibliothèque React Leaflet. Cette carte offre une immersion totale dans une véritable chasse au trésor numérique.

Lorsque l’utilisateur – y compris l’administrateur – accède à la carte, il peut cliquer sur différentes zones pour tenter de découvrir l’emplacement exact du trésor. Un système calcule en arrière-plan la distance entre la position cliquée et le lieu réel du trésor.

* Si le clic est éloigné de la cible, un message d’encouragement s’affiche, accompagné d’un son stimulant pour inciter à continuer.
* Si l’utilisateur clique à proximité immédiate ou directement sur l’emplacement du trésor, une notification de réussite apparaît avec un effet sonore de victoire.

Une fois le trésor trouvé, celui-ci est automatiquement déplacé vers un nouvel emplacement aléatoire, permettant ainsi de renouveler la partie et de prolonger l'expérience ludique.

Enfin, un bouton **« Quitter »** est disponible pour mettre fin à la session de jeu à tout moment. Ce système rend la chasse à la fois interactive, motivante et durable.

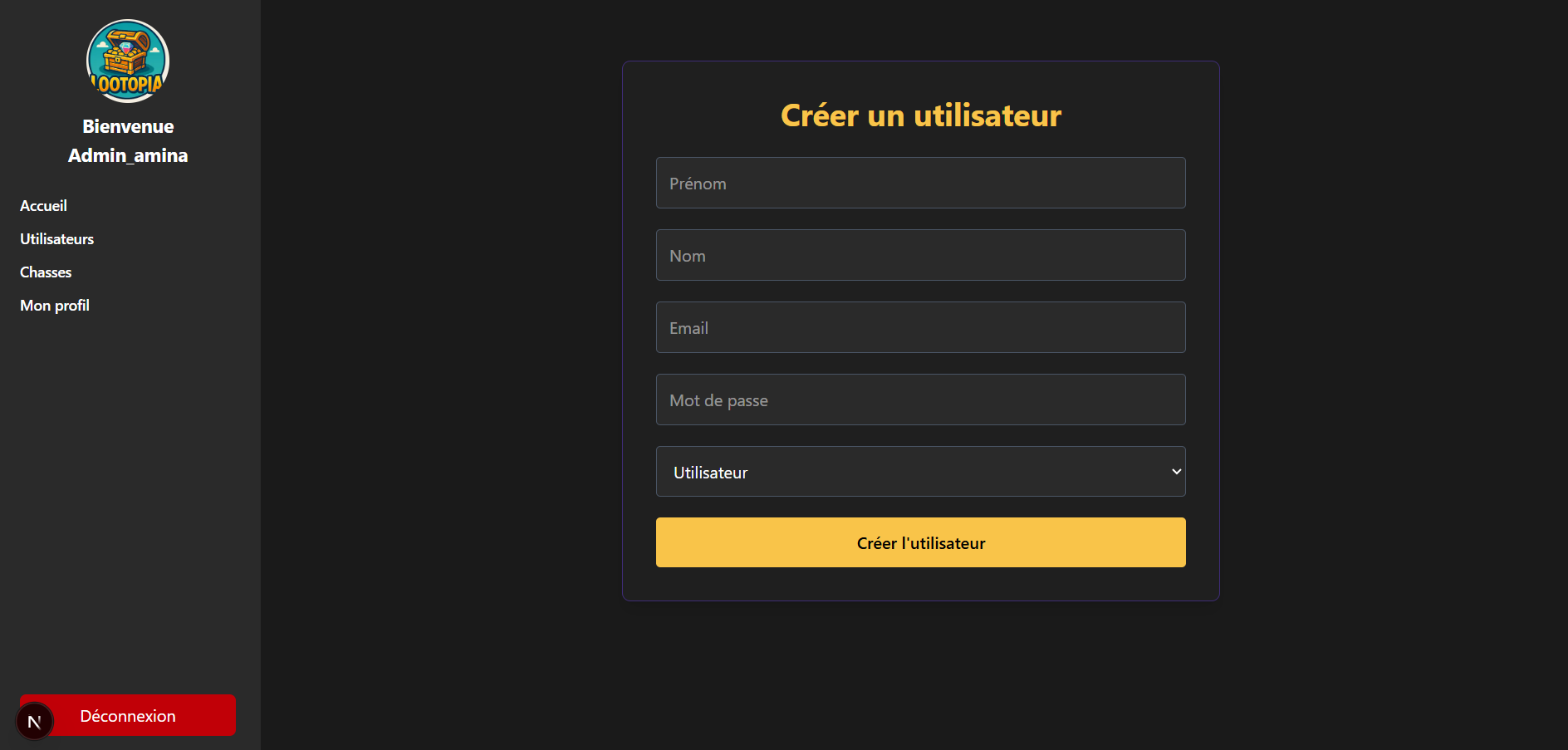


### 4.5 Gestion des utilisateurs

Les administrateurs de Lootopia bénéficient d’un espace de gestion dédié leur offrant un contrôle complet sur la plateforme :

* **Modification des rôles :** Ils peuvent changer le rôle d’un utilisateur à tout moment (de simple utilisateur à administrateur ou inversement) grâce à une liste déroulante intuitive.
* **Suppression de comptes :** Chaque utilisateur peut être supprimé en un clic, permettant un nettoyage rapide en cas d’abus ou de comptes inactifs.
* **Création de nouveaux utilisateurs :** Un formulaire est mis à disposition pour ajouter manuellement un utilisateur. Celui-ci comprend les champs : prénom, nom, email, mot de passe, ainsi qu’un menu déroulant pour définir le rôle (Utilisateur ou Admin).



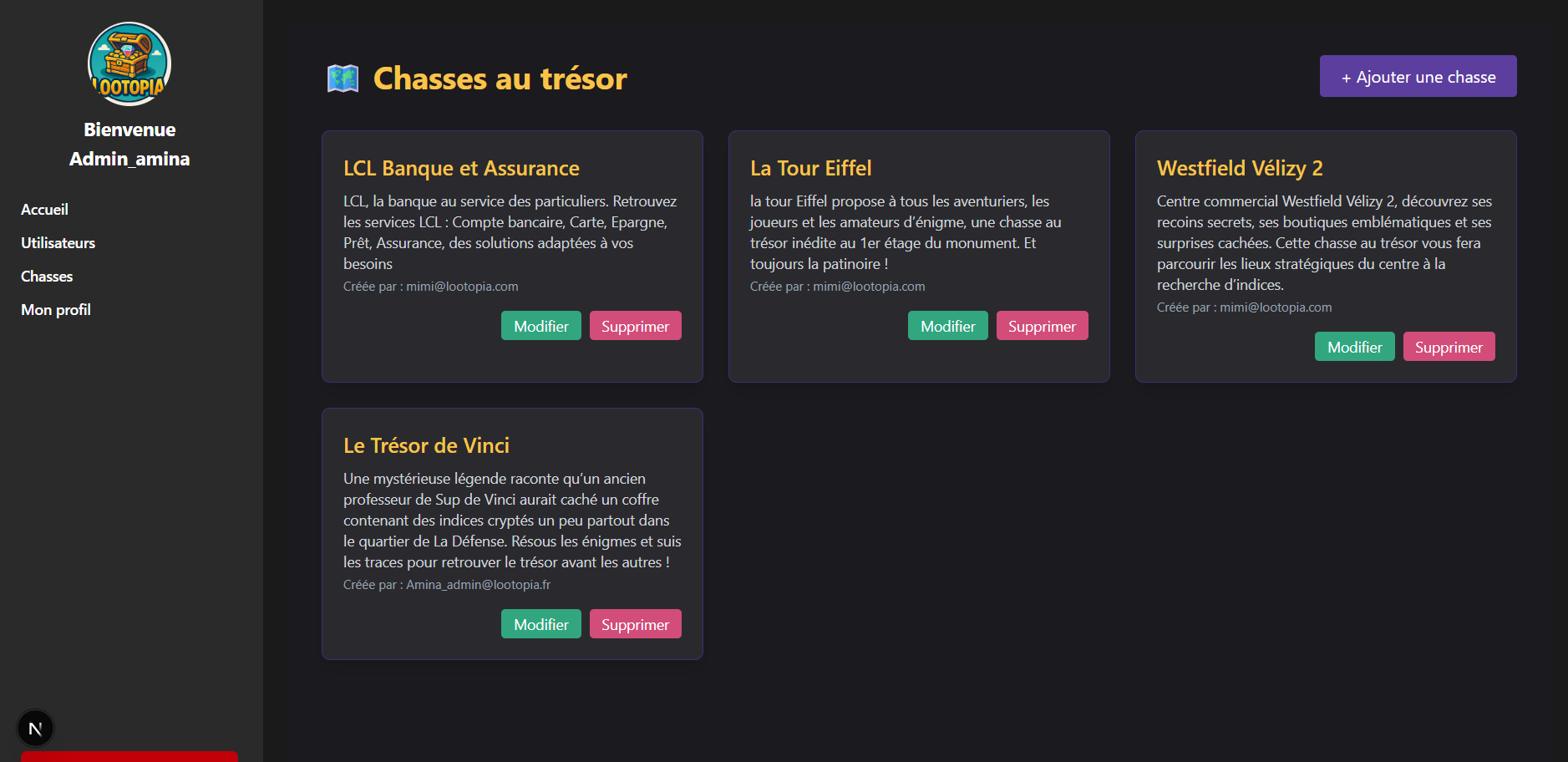


### 4.6 Gestion des chasses au trésor

Dans la rubrique ‘Chasses’, les chasses au trésor déjà créées s’affichent sous forme de cartes récapitulatives. Chaque carte présente :

* Le titre de la chasse ;
* Une description courte ;
* Le nom de l’administrateur ayant publié la chasse ;
* Deux boutons d’action : Modifier et Supprimer, permettant à l’administrateur de gérer dynamiquement le contenu.

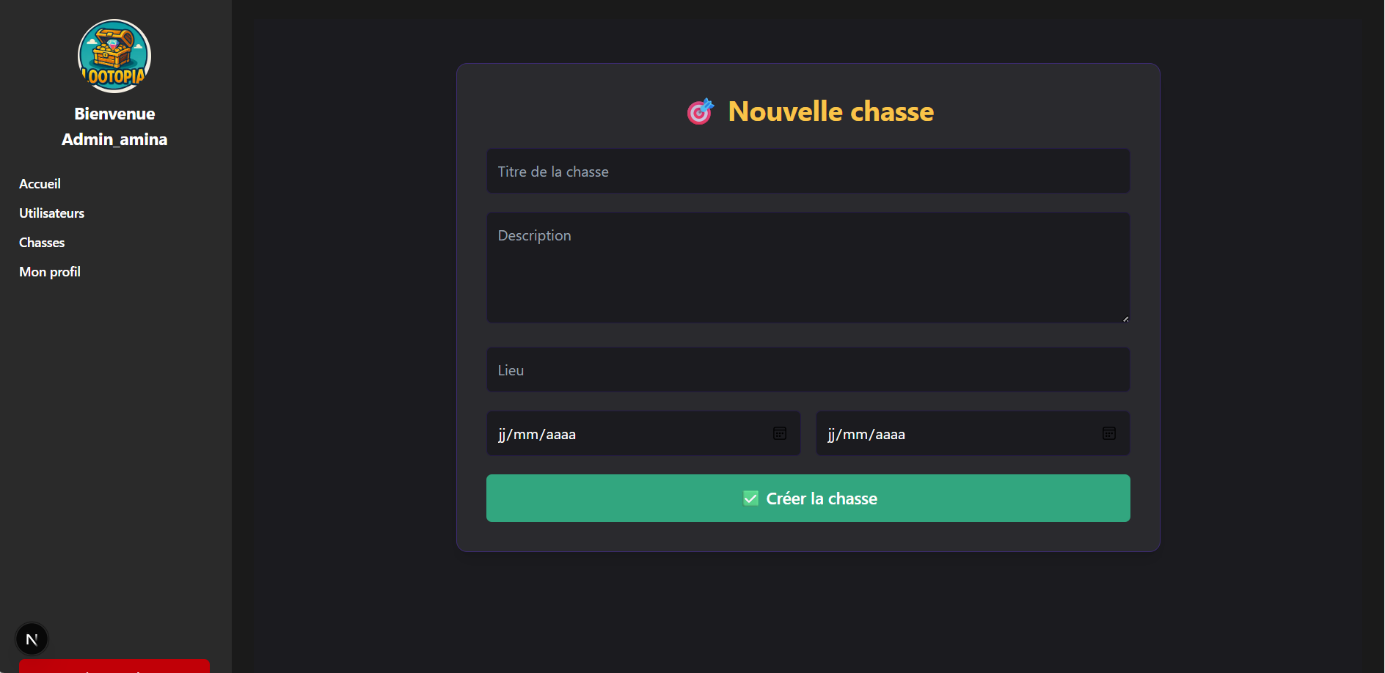
En cliquant sur le titre d’une chasse, l’administrateur est redirigé vers sa page de détails, où il peut consulter l’ensemble des informations liées à cette chasse, accéder à la carte interactive, gérer les commentaires, et bien plus encore.



Un bouton « Ajouter une chasse », situé en haut à droite, permet de créer une nouvelle chasse. Lorsqu’on clique dessus, on accède à un formulaire de création, dans lequel l’administrateur peut saisir :

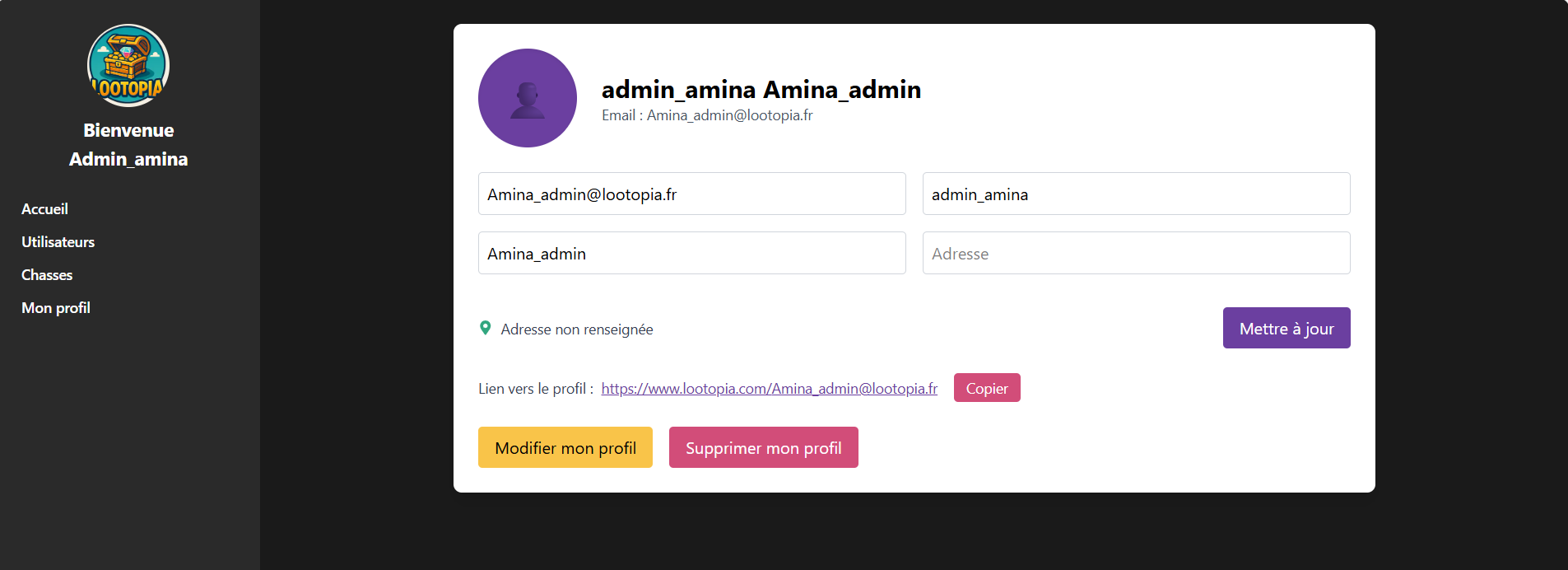
* Le titre de la chasse ;
* Une description détaillée ;
* Le lieu géographique ;
* La date de début et la date de fin.

Une fois le formulaire soumis, la chasse apparaît automatiquement dans la liste, prête à être modifiée ou supprimée si nécessaire.



### 4.7 Gestion du profil administrateur

Après authentification, chaque administrateur est redirigé vers son espace personnel depuis lequel il peut consulter, modifier ou supprimer ses informations de profil en toute autonomie.



* **Informations affichées et modifiables**

L’interface présente un formulaire pré-rempli affichant les données personnelles de l’administrateur :

* Adresse e-mail
* Nom
* Prénom
* Adresse postale (facultative)

Ces champs peuvent être mis à jour directement via le bouton « Mettre à jour ». L’interface, claire et épurée, garantit une expérience utilisateur fluide.

* **Lien de profil public**

Chaque compte administrateur possède un lien public de profil, généré automatiquement. Ce lien peut être copié en un clic pour être partagé avec d’autres membres.  
Il renvoie vers une version publique et restreinte du profil, assurant la confidentialité des données sensibles.

* **Suppression du compte**

L’administrateur dispose également d’un bouton « Supprimer mon profil », permettant la suppression définitive de son compte. Cette action efface l’ensemble des données associées à l’utilisateur :

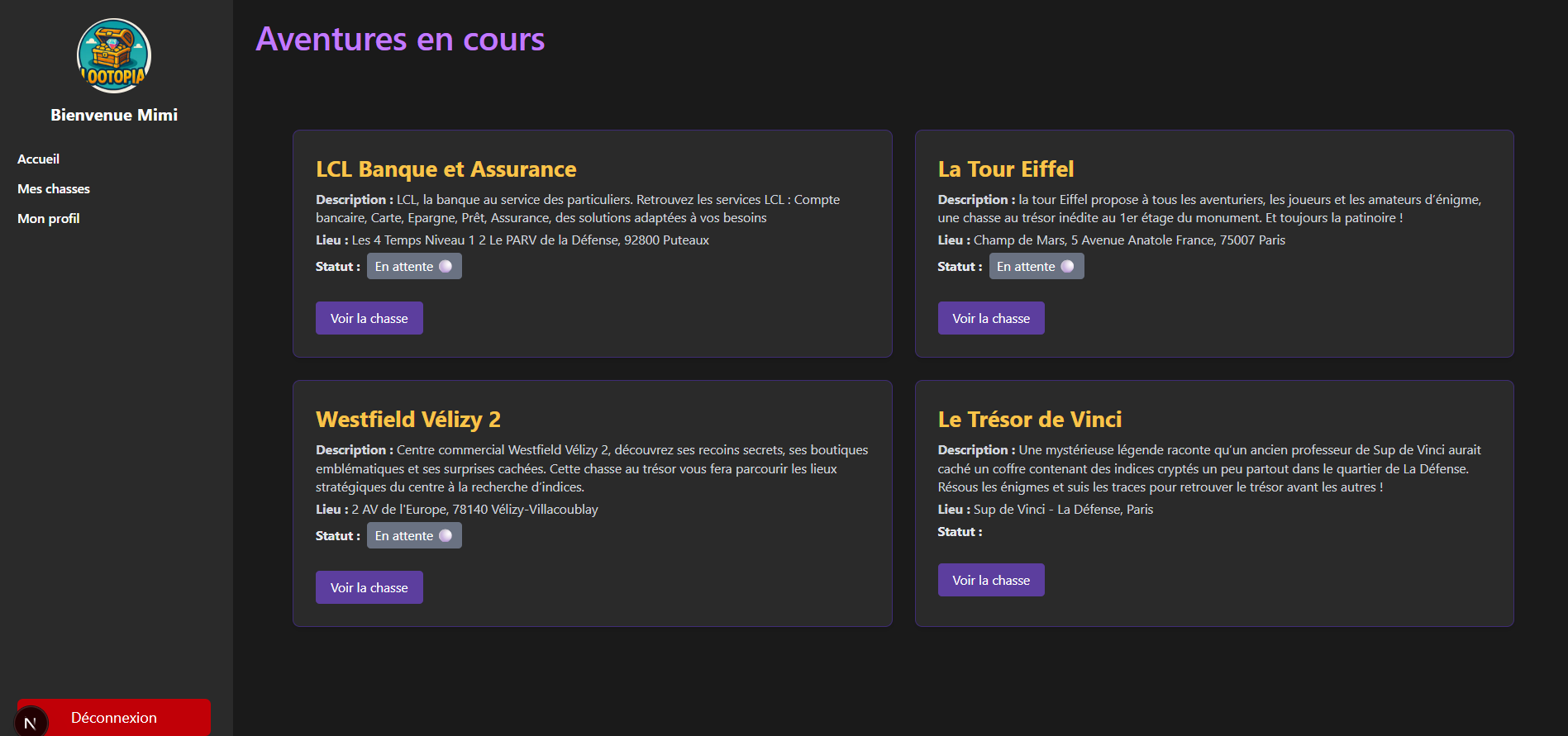
* Informations de profil
* Participations aux chasses
* Avis publiés

Cette fonctionnalité respecte les exigences en matière de protection des données personnelles et permet à l’utilisateur de garder le contrôle total sur ses informations.

### 4.8 Visualisation des chasses côté utilisateur :

Une fois connecté, l’utilisateur accède à une interface listant toutes les **chasses au trésor disponibles**, affichées sous forme de cartes récapitulatives.

Cette page permet à l’utilisateur de choisir facilement la chasse qui l'intéresse, en fonction du lieu ou de son statut. La navigation est fluide et intuitive, orientée vers l’engagement et la participation active aux aventures proposées.



### 4.9 Visualisation des chasses côté utilisateur :

Lorsque l’utilisateur clique sur le bouton **« Voir la chasse »** depuis l’accueil, il est redirigé vers une **page de détails** dédiée à la chasse sélectionnée. Cette page présente de manière complète les informations suivantes :

* **Titre** de la chasse
* **Description** détaillée de l’activité
* **Lieu exact** où se déroule la chasse
* **Statut** actuel de la chasse : *Pending*, *En cours* ou *Terminée*
* **Mode de participation** : *Public* ou *Privé*
* **Organisateur** de la chasse (adresse e-mail)
* **Date de début** et **date de fin** de la chasse

En bas de cette section, l’utilisateur retrouve un bouton **« Rejoindre la chasse »**, lui permettant de s’inscrire à l’aventure et d'accéder à la carte interactive pour tenter de localiser le trésor.



### 4.10 Système d’avis utilisateur

En bas de la page des détails de la chasse, les utilisateurs ont accès à une **zone dédiée aux avis** pour partager leur expérience ou poser des questions. Ce système comprend :

* **Un champ de saisie** permettant à l’utilisateur de rédiger un commentaire librement.
* **Un bouton "Envoyer"** pour publier l’avis.

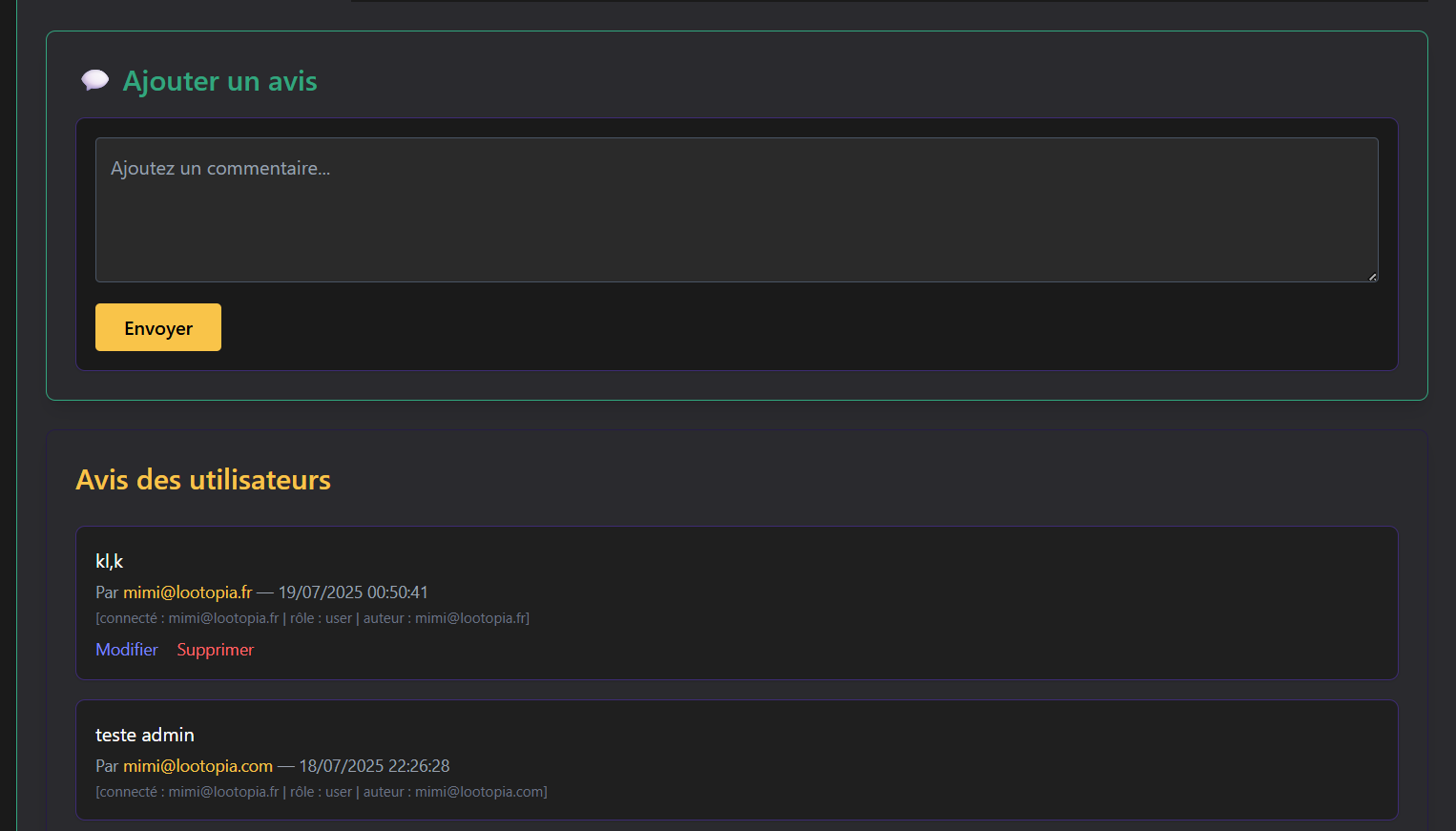
Une fois publié, l’avis apparaît dans la section **« Avis des utilisateurs »**, qui affiche :

* Le **contenu du commentaire**
* L’**auteur** (adresse e-mail)
* La **date et l’heure** de publication

L’utilisateur peut ensuite :

* **Modifier** son propre commentaire
* **Supprimer** ses propres messages

Ce système d’avis favorise l’interaction entre les participants et renforce la communauté autour de chaque chasse.



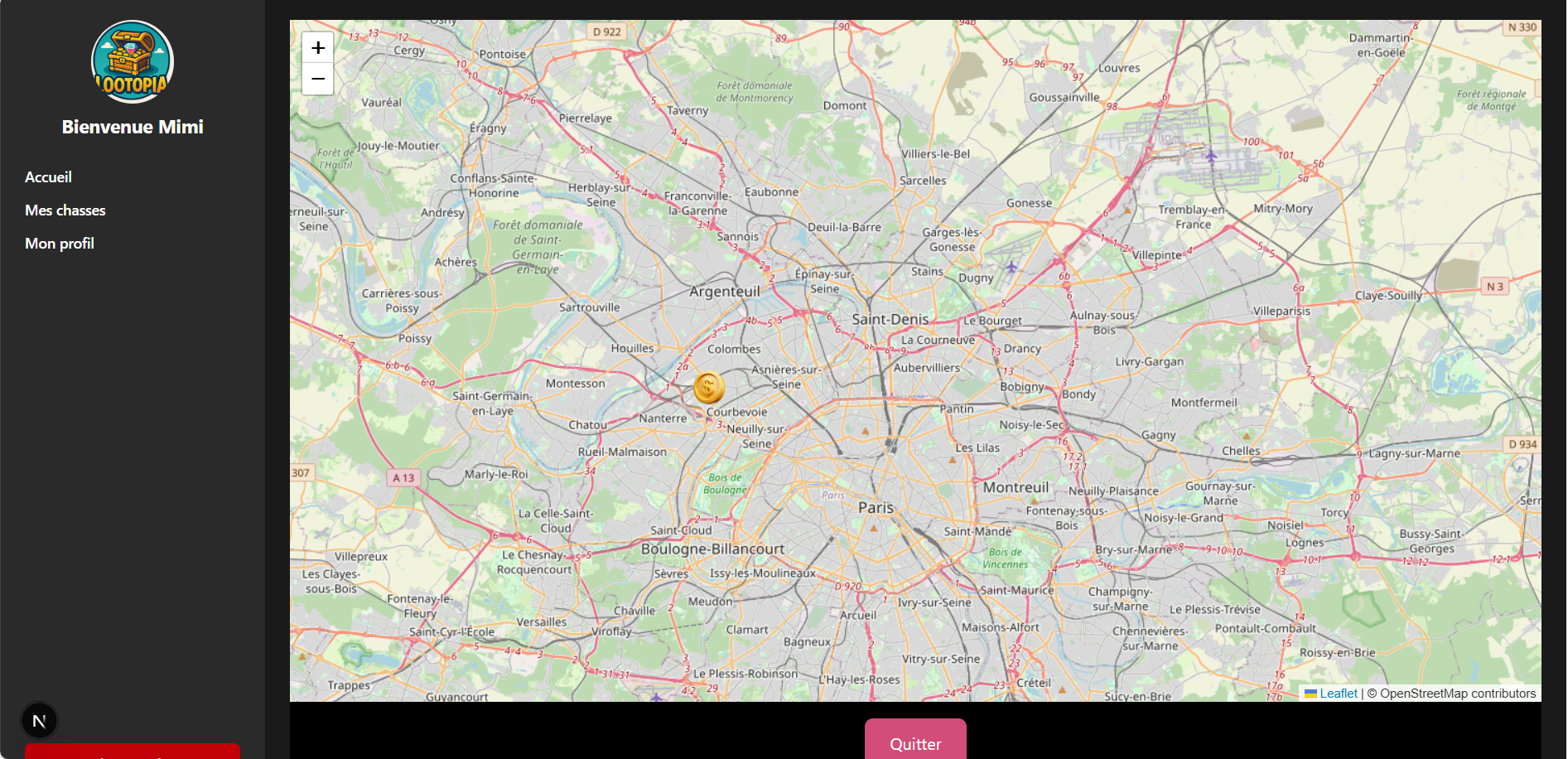
### 4.11 Rejoindre la chasse et carte interactive

Une fois sur la page de détails d’une chasse, l’utilisateur peut cliquer sur le bouton **« Rejoindre la chasse »**. Ce bouton déclenche sa participation et le redirige vers une **carte interactive** affichant la zone de recherche.

Sur cette carte, l'utilisateur peut :

* **Explorer librement** l’environnement à la recherche du trésor caché.
* Cliquer sur des zones pour recevoir un **indice de distance** s’il est loin du trésor.
* Recevoir un **message de succès** s’il trouve l’emplacement exact.

Ce mécanisme ludique offre une expérience immersive et gamifiée, rendant chaque chasse unique et engageante.



## **5. Documentation API :**

**1. POST /api/users/hunts/[id]/participate**

* **Description** : Permet à un utilisateur de participer à une chasse au trésor.
* **Authentification** : Requise (utilisateur connecté via JWT).
* **Méthode** : POST
* **Entrée** : ID de la chasse (dans l’URL)
* **Sortie** : Participation créée ou message si déjà inscrit

**2. GET | PUT | DELETE /api/admin/hunts/[id]**

* **Description :** Récupérer, modifier ou supprimer une chasse (réservé à l’admin).
* **Authentification :** Requise (ADMIN)
* **Méthodes :**
  + **GET :** détails d’une chasse
  + **PUT :** modification d’une chasse
  + **DELETE :** suppression d’une chasse et de ses éléments associés

**3. GET | POST /api/admin/hunts/hunts**

* **Description :**
  + **GET :** Lister toutes les chasses existantes
  + **POST :** Créer une nouvelle chasse
* **Authentification :** Requise (ADMIN)
* **Entrée (POST) :** titre, description, lieu, dates, etc.
* **Sortie :** Liste des chasses ou nouvelle chasse créée

**4. POST /api/users/participation/status**

* **Description** : Mettre à jour le statut (PENDING, IN\_PROGRESS, COMPLETED) d'une participation.
* **Authentification** : Requise (utilisateur connecté)
* **Entrée** : huntId et status
* **Sortie** : Objet participation mis à jour

**5. GET | POST | PUT | DELETE /api/hunts/[id]/review**

* **Description** : Gérer les avis d’une chasse.
* **Authentification :** Requise (utilisateur connecté)
* **Méthodes :** 
  + **GET :** liste des avis
  + **POST :** création d’un avis
  + **PUT :** modification d’un avis (si auteur)
  + **DELETE :** suppression d’un avis (si auteur)

**6. POST /api/admin/users/promote**

* **Description :** Promouvoir un utilisateur au rôle d’administrateur.
* **Authentification :** Requise (ADMIN)
* **Entrée :** id de l’utilisateur
* **Sortie** : Utilisateur mis à jour avec rôle ADMIN

**7. DELETE /api/admin/users/[id]**

* **Description** : Supprimer un utilisateur via son ID.
* **Authentification** : Requise (ADMIN)
* **Sortie :** Message de confirmation de suppression

**8. POST /api/admin/users/create**

* **Description :** Créer un utilisateur depuis l’espace admin.
* **Authentification :** Requise (ADMIN)
* **Entrée :** prénom, nom, email, mot de passe, rôle
* **Sortie :** Nouvel utilisateur créé

**9. GET /api/admin/users**

* **Description :** Lister tous les utilisateurs enregistrés.
* **Authentification :** Requise (ADMIN)
* **Sortie :** Liste des utilisateurs avec leurs rôles

## **6. Annexe :**

|  |  |  |
| --- | --- | --- |
| **Package** | **Version** | **Usage** |
| **next** | **15.3.1** | **Framework principal de l'application (Next.js)** |
| **react** | **19.0.0** | **Bibliothèque UI principale (React)** |
| **react-dom** | **19.0.0** | **Rendu DOM côté client (ReactDOM)** |
| **axios** | **1.9.0** | **Requêtes HTTP vers l’API backend** |
| **jsonwebtoken** | **9.0.2** | **Génération et vérification des JWT** |
| **bcryptjs** | **3.0.2** | **Hashing des mots de passe** |
| **prisma** | **6.11.1** | **ORM pour la base de données PostgreSQL** |
| **nodemailer** | **7.0.3** | **Envoi d'e-mails (mot de passe oublié, etc.)** |
| **pg** | **8.15.5** | **Driver PostgreSQL pour Node.js** |

**7. Conclusion :**

Le projet **Lootopia** a permis de concevoir une application web complète et innovante autour du concept de chasse au trésor interactive. En combinant des technologies modernes comme **Next.js, React, Prisma et PostgreSQL**, nous avons développé une plateforme à la fois performante, sécurisée et facile à utiliser. Les utilisateurs peuvent vivre une expérience ludique immersive, tandis que les administrateurs disposent d’outils efficaces pour gérer les chasses, les comptes et les avis. L’intégration d’une carte interactive renforce l’aspect jeu et rend l’exploration plus engageante. L’authentification sécurisée via **JWT/JWK** assure la protection des données et le contrôle des accès. Lootopia répond ainsi à un besoin croissant d’activités numériques originales et personnalisables.

Ce projet a également été une occasion d’appliquer nos compétences en développement full-stack dans un cadre concret et stimulant. Nous en tirons une expérience enrichissante, tant sur le plan technique que collaboratif.