1. **🎯 Objectifs fonctionnels**

* Permettre à un utilisateur de :
  + Créer un compte, se connecter et modifier ses informations personnelles
  + Lancer une course avec géolocalisation en temps réel
  + Suivre sa distance, durée, vitesse, et le tracé GPS
  + Consulter son historique de courses
  + Refaire une course en mode “🎯 défi”
  + Commenter et aimer les courses de la communauté
  + Changer le thème clair/sombre
  + Donner un avis et supprimer son compte
* Offrir à un administrateur la possibilité de :
  + Visualiser tous les utilisateurs et courses
  + Supprimer un compte ou une course
  + Consulter les feedbacks reçus via l’app mobile

⚙️ Objectifs techniques

* Créer une application mobile performante avec React Native et Expo
* Gérer les données via une API Node.js sécurisée avec JWT et MySQL
* Optimiser l’expérience utilisateur grâce à un design responsive, des thèmes dynamiques et un système de partage
* Assurer la persistance des préférences utilisateur (thème, token, profil…)
* Permettre une administration complète via une interface web

🛠️ Technologies utilisées

🧩 Mobile (React Native)

* React Native + Expo
* expo-router (navigation)
* react-native-maps (tracé GPS)
* react-native-view-shot + expo-sharing (partage)
* @react-native-async-storage/async-storage (sauvegarde locale)
* Ionicons (icônes)
* Context API (authentification, thème)

🌐 Backend (Node.js + Express)

* Express
* MySQL2
* jsonwebtoken, bcrypt
* dotenv, cors

🖥️ Web Admin (React)

* React avec Vite
* React Router, Axios
* Tailwind CSS pour le style
* Chart.js (statistiques visuelles)

**2. 🧱 Architecture générale**

🗺️ Schéma de l’architecture

Le projet RunYnov repose sur une architecture modulaire et découpée en 3 grands blocs :

UTILISATEUR 👤

⬇️ ⬆️

📱 Application mobile (React Native)

- Authentification

- Géolocalisation en temps réel

- Suivi des performances

- Historique & commentaires

- Mode clair/sombre

- Partage et défi

⬇️ appels HTTP (fetch) via token JWT

🔗 API REST (Node.js + Express)

- Authentification & sécurité (JWT + bcrypt)

- Stockage des données (MySQL)

- Endpoints utilisateurs, courses, feedbacks, commentaires

- Middleware d’autorisation

⬆️ accès en lecture/écriture

🛢️ Base de données (MySQL)

- Tables : users, courses, goals, feedbacks, interactions…

⬇️ appels HTTP (Axios)

🖥️ Interface Web Admin (React + Vite)

- Tableau de bord admin

- Gestion des utilisateurs et feedbacks

- Suppression manuelle d’éléments

- Statistiques avec graphiques

📁 Arborescence des dossiers

📦 mobile/ (Application React Native avec Expo)

mobile/

├── app/ → Écrans (expo-router)

│ ├── (tabs)/ → Navigation principale (index, run, menu…)

│ ├── settings/ → Paramètres

│ ├── course/ → Détails de course

├── assets/ → Images et icônes

├── components/ → Composants réutilisables (ex: ShareCard, BackButton)

├── context/ → ThemeContext.tsx, AuthContext.tsx

├── utils/ → token.ts, helpers

├── theme.ts → Fichier de thème centralisé

├── app.config.js → Configuration Expo

📦 backend/ (API Node.js + Express)

backend/

├── routes/ → Routes API (auth, courses, feedback, interactions…)

├── controllers/ → Logique des endpoints

├── middlewares/ → authMiddleware, validations

├── config/ → db.js (connexion MySQL)

├── .env → Variables d’environnement

├── index.js → Point d’entrée de l’API

📦 admin/ (Interface Web Admin - React + Vite)

admin/

├── src/

│ ├── components/ → Table, Navbar, Cards, etc.

│ ├── pages/ → Utilisateurs, Feedbacks, Statistiques

│ ├── services/ → Requêtes API via Axios

│ ├── App.tsx

│ ├── main.tsx

├── public/

├── vite.config.ts

├── tailwind.config.js

💡 Choix techniques

| Élément | Choix | Justification |
| --- | --- | --- |
| Front Mobile | React Native + Expo | Cross-platform, rapide à mettre en place, écosystème riche |
| Navigation mobile | expo-router | Basé sur les routes fichiers, très intuitif |
| API Backend | Node.js + Express | Léger, rapide, facile à connecter avec React Native |
| Authentification | JWT + Bcrypt | Sécurisé, standard du web |
| Base de données | MySQL | Relationnelle, facile à structurer pour les courses / users |
| Web Admin | React + Vite + Tailwind | Rapide, moderne, Tailwind pour un design responsive |
| Cartes & GPS | react-native-maps | Intégration fluide avec le GPS du téléphone |
| Statistiques | Chart.js (admin), LineChart (mobile) | Visualisation simple et efficace |

**3. 📱 Application Mobile**

a. ✅ Fonctionnalités principales

L'application mobile RunYnov offre un ensemble riche de fonctionnalités destinées à améliorer l’expérience de running, l’engagement communautaire et la personnalisation :

| Fonctionnalité | Description |
| --- | --- |
| 🔐 Authentification | Inscription / Connexion sécurisée avec JWT et vérification par email |
| 📍 Suivi GPS en temps réel | Utilisation de expo-location pour enregistrer la position toutes les 3s |
| 🏁 Enregistrement de courses | Distance, durée, vitesse moyenne, tracé GPS, sauvegarde en base de données |
| 🎯 Mode défi | Rejouer une course déjà effectuée avec un objectif de distance |
| 📜 Historique & détails | Vue liste de toutes les courses + détail complet avec carte et stats |
| ❤️ Likes et 💬 commentaires | Interaction entre utilisateurs : aimer et commenter une course |
| 🎯 Objectif journalier | Un challenge automatique chaque jour, validé dès 2 km effectués |
| 🎨 Thèmes clair / sombre | Gestion via ThemeContext, avec persistance du thème |
| 🌐 Langue (prévu) | Prévu via i18n mais non encore intégré |
| ⚙️ Paramètres utilisateur | Changer mot de passe, adresse email, supprimer son compte |
| 📤 Partage d’une course | Génération d’un visuel via react-native-view-shot + partage natif |
| 🗣️ Feedback utilisateur | Formulaire d’avis enregistré en base pour usage admin |

**4. 🛠️ API Backend**

a. 🔌 Schéma général de l’API

L’API backend centralise toutes les interactions entre l’application mobile, l’interface admin et la base de données MySQL. Elle suit une architecture RESTful et est développée en Node.js avec Express.

* Les routes sont organisées par modules fonctionnels (auth, user, run, challenge, feedback…)
* Utilisation de JWT pour l’authentification
* Middleware pour la vérification des rôles (admin / utilisateur)
* Toutes les données sont persistées dans une base de données MySQL (via Sequelize ORM)

—

b. 🔐 Sécurité & Authentification

L’authentification utilise un token JWT, stocké côté mobile en local via AsyncStorage.

* Hachage des mots de passe avec bcrypt
* JWT généré à la connexion, vérifié sur chaque requête protégée
* Système de validation d’email à l’inscription (optionnel)
* Middleware auth utilisé pour sécuriser les routes

—

c. 🌐 Principales routes REST

Voici les routes principales exposées par l’API (toutes en /api) :

| Méthode | Route | Description |
| --- | --- | --- |
| POST | /auth/register | Inscription d’un nouvel utilisateur |
| POST | /auth/login | Connexion, retour du JWT |
| GET | /users/me | Récupère les infos du user connecté |
| PUT | /users/me | Mise à jour email, mot de passe, etc. |
| DELETE | /users/me | Supprime le compte utilisateur |
| POST | /runs | Enregistre une nouvelle course |
| GET | /runs | Liste toutes les courses de l’utilisateur |
| GET | /runs/:id | Détail d’une course |
| GET | /runs/:id/comments | Récupère les commentaires d’une course |
| POST | /runs/:id/comments | Ajoute un commentaire |
| POST | /runs/:id/like | Aime une course |
| GET | /challenges/today | Challenge journalier de l’utilisateur |
| POST | /feedback | Soumission de feedback utilisateur |
| GET | /admin/users | Liste des utilisateurs (admin uniquement) |
| GET | /admin/feedbacks | Liste des feedbacks (admin uniquement) |

—

d. 🗃️ Base de données (MySQL via Sequelize)

Les principales tables :

* users : id, email, password, role, createdAt, updatedAt
* runs : id, userId, distance, duration, speed, coords[], date
* likes : id, userId, runId
* comments : id, userId, runId, content, createdAt
* challenges : id, userId, date, completed
* feedbacks : id, userId, message, rating

1. **🖥️ Interface Admin Web**

a. Objectifs de l’interface admin

L’interface admin a été pensée pour permettre à l’équipe de gestion de la plateforme RunYnov d’avoir un aperçu rapide et maîtrisé des éléments clés du service :

* Gestion et visualisation des utilisateurs
* Visualisation des courses publiques
* Modération des feedbacks utilisateurs
* Statistiques générales d’utilisation

Elle est indépendante de l’app mobile mais utilise la même API sécurisée pour récupérer les données.

b. Fonctionnalités développées

Voici les principales pages/fonctionnalités de l’interface admin :

* 🔐 Connexion administrateur : authentification via JWT
* 👥 Liste des utilisateurs : nom, email, date d’inscription, nb de courses
* 📋 Liste des feedbacks : contenu, note associée, utilisateur, date
* 🏃‍♂️ Vue des courses : liste des dernières courses publiques avec infos clés (distance, durée, vitesse, likes)
* 📈 Statistiques : total users, total courses, feedback moyen, etc.

Des actions d’administration supplémentaires sont prévues (désactivation d’un compte, suppression de feedback, etc.), mais non implémentées pour cette première version.

c. Technologies utilisées

L’interface admin est une web app développée avec :

* React + TypeScript
* React Router DOM pour la navigation
* Axios pour les appels API
* Chart.js pour les graphiques de statistiques
* Tailwind CSS pour le design
* JWT stocké en localStorage
* Accès restreint aux administrateurs uniquement (vérification du token + rôle admin)

1. **📦 Déploiement**

a. Backend (API Node.js + MySQL)

Le backend de l’application RunYnov est un serveur Node.js avec Express, connecté à une base de données MySQL. Il gère l’authentification, la logique métier (courses, objectifs, feedbacks, interactions…), et expose une API REST sécurisée par JWT.

Déploiement local :

* Lancement avec nodemon ou npm start
* Variables d’environnement via .env (DB\_HOST, DB\_USER, JWT\_SECRET…)
* Connexion à MySQL locale (XAMPP, MAMP ou Docker)

Déploiement distant (prévu) :

* Hébergement prévu sur un VPS (type Render, Railway, ou serveur école)
* API exposée sur un sous-domaine type api.runynov.fr
* Sécurisation avec HTTPS + logs (morgan / winston)
* Base MySQL distante sur MariaDB ou PlanetScale

b. Application Mobile (Expo / React Native)

L’app mobile RunYnov est construite avec Expo, ce qui facilite les tests, le build et le déploiement sur les stores.

Déploiement et distribution :

* Développement avec Expo Go (scan QR code)
* Build via EAS Build (expo build:android / ios)
* Publication sur Play Store (Android) prévue
* Possibilité d’exporter un .apk pour installation manuelle

Configuration :

* app.json pour définir l’icône, splash screen, nom, version
* Permissions : localisation, partage, média, notification
* Connexion à l’API en HTTP (ou HTTPS en prod)

c. Interface Admin Web (React Web)

L’interface admin web est conçue comme une Single Page Application (SPA) indépendante. Elle peut être déployée sur n’importe quel hébergement statique (Netlify, Vercel, GitHub Pages, etc.).

Déploiement local :

* npm run dev (Vite ou React Scripts)
* Accès à l’API backend via fetch/Axios
* Variables d’environnement (.env)

Déploiement distant :

* Build : npm run build
* Upload du dossier dist/ ou build/ sur Netlify / Vercel
* Custom domain admin.runynov.fr possible

1. **📊 Bilan & perspectives**

a. Ce qui a bien fonctionné

* Le suivi GPS en temps réel a été implémenté avec précision grâce à react-native-maps et une gestion efficace du state.
* L'enregistrement de course fonctionne parfaitement avec l'API Express, la base de données MySQL et les fonctionnalités de stockage.
* Le mode défi est une vraie plus-value, offrant un challenge basé sur des parcours précédemment réalisés.
* L’authentification avec token JWT et la gestion des utilisateurs sont stables et sécurisées.
* Le système de feedback et d’interactions (likes, commentaires) est simple mais efficace.
* L’interface mobile est intuitive, responsive, et agréable à utiliser en thème clair comme sombre.
* La modularité du code avec les contextes (useAuth, useTheme), composants réutilisables (BackButton, ShareCard…) permet une bonne maintenabilité.
* L’application reste fonctionnelle hors-ligne grâce à un manifest bien configuré et l’utilisation du cache local.

b. Difficultés rencontrées

* L'intégration des permissions natales (localisation, galerie, stockage) a nécessité plusieurs tests selon les plateformes.
* Le passage de mysql2 en mode Promise a généré des erreurs lorsqu’on n’appelait pas correctement con.promise().query().
* Certaines erreurs d’affichage sont survenues quand l’utilisateur n’a pas encore de données (ex. division par null, toFixed sur undefined).
* Le maintien du thème entre les sessions a nécessité une implémentation via AsyncStorage.
* Les performances sur cartes complexes (tracés longs) peuvent parfois souffrir sur des appareils plus anciens.
* Le changement de langue a été prévu mais non implémenté par manque de temps.

c. Améliorations possibles

* Implémenter la sélection de langue (fr/en) avec un fichier de traductions et AsyncStorage.
* Ajouter une page de statistiques avancées (graphique, moyenne par semaine, progression...).
* Déployer l’application sur le Play Store / App Store avec une configuration EAS plus poussée.
* Renforcer l’aspect social (partage externe, ajout d’amis, défis entre utilisateurs).
* Ajouter des notifications push pour rappeler les objectifs quotidiens ou féliciter une performance.
* Mettre en place un back-office complet pour l’admin avec suppression / ban / statistiques globales.
* Optimiser les performances sur les longues sessions de tracking GPS.
* Ajouter des récompenses (badges, XP) pour améliorer la fidélisation et la motivation.