

Crayston Matt  
lafrate Thomas

## **Documentation Technique RunYnov**

# Sommaire :

<b>1. Introduction</b>	<b>3</b>
1.1 Présentation du projet RunYnov	3
1.2 Objectifs techniques détaillés	4
1.3 Technologies utilisées (récapitulatif)	6
<b>2. Architecture générale</b>	<b>8</b>
2.1. Schéma de l'architecture globale	8
2.2 API REST (Node.js + Express)	9
2.3. Base de données relationnelle (MySQL)	9
2.4 Interface Web Admin (React + Vite)	10
2.5 Arborescence des dossiers	11
2.6 Choix techniques	13
<b>3. Guide de Lancement du Projet</b>	<b>14</b>
3.1 Prérequis Généraux	Error! Bookmark not defined.
3.2 Installation Initiale	Error! Bookmark not defined.
3.3 Backend	Error! Bookmark not defined.
3.4 Application Mobile (App)	Error! Bookmark not defined.
3.5 Frontend (Interface Web)	Error! Bookmark not defined.
<b>4. Application Mobile</b>	<b>25</b>
4.1. Fonctionnalités principales	25
<b>5. API BACKEND</b>	<b>29</b>
5.1. Schéma général de l'API	30
5.2. Sécurité & Authentification	31
5.3 Principales routes REST	32
5.4. Structure de la base de données (MySQL)	34
<b>6. Interface Admin Web</b>	<b>36</b>
6.1. Objectifs de l'interface admin	36
6.2. Fonctionnalités développées	37
6.3. Technologies utilisées	39
<b>7. Déploiement</b>	<b>40</b>
7.1. Backend (API Node.js + MySQL)	41
7.2. Application Mobile (Expo / React Native)	42
7.3. Interface Admin Web (React + Vite)	43
<b>8. Bilan &amp; Perspectives</b>	<b>45</b>
a. Ce qui a bien fonctionné	45
b. Difficultés rencontrées	46
c. Améliorations possibles & pistes d'évolution	47
<b>Conclusions</b>	<b>49</b>

# 1. Introduction

## 1.1 Présentation du projet RunYnov

RunYnov est une application mobile complète et moderne dédiée à la pratique de la course à pied. Elle a été pensée comme un compagnon sportif personnel, permettant à chaque utilisateur de suivre ses performances, d'enregistrer ses courses, de défier ses anciens parcours, et de s'inscrire dans une dynamique communautaire (likes, commentaires, partage). Son objectif est de motiver les utilisateurs à courir régulièrement tout en fournissant une expérience fluide, personnalisée et engageante.

Le projet repose sur un triptyque technologique :

- Une application mobile développée avec React Native (Expo) pour Android et iOS
- Un backend Node.js/Express avec une base de données MySQL
- Une interface d'administration web (React + Vite) destinée à la supervision et à la modération

Ce projet pédagogique a été conçu pour couvrir toutes les facettes du développement fullstack : frontend mobile, API REST sécurisée, administration, base de données relationnelle, et aspects UX/UI.

### 1.1 Objectifs fonctionnels

Côté utilisateur mobile, l'application permet de :

- S'inscrire, se connecter, se déconnecter
- Modifier son profil, son mot de passe, supprimer son compte
- Lancer une course avec géolocalisation active
- Suivre en temps réel son tracé GPS, la distance parcourue, la durée, et la vitesse

- Refaire un parcours antérieur en mode "Défi" (guidage GPS vers le tracé précédent)
- Consulter l'historique de ses courses et leurs détails
- Aimer les courses de la communauté et laisser des commentaires
- Activer un mode clair/sombre (thème)
- Recevoir un objectif quotidien généré automatiquement (par ex. "Courir 2 km aujourd'hui")
- Partager une carte personnalisée de sa course (ShareCard) avec les données clés
- Envoyer un feedback écrit aux développeurs

Côté administrateur web, les fonctionnalités disponibles sont :

- Visualiser l'ensemble des utilisateurs inscrits
- Supprimer un compte utilisateur ou une course si nécessaire
- Consulter les feedbacks reçus depuis l'app
- Visualiser des statistiques (utilisateurs actifs, etc.)

## 1.2 Objectifs techniques détaillés

Le projet visait plusieurs objectifs techniques clairs :

Sécurité

- Authentification sécurisée par token JWT (stocké localement dans AsyncStorage)
- Hashage des mots de passe avec bcrypt côté serveur
- Middleware authMiddleware pour protéger toutes les routes sensibles de l'API

Structure & architecture

- Architecture modulaire et MVC sur le backend : routes, controllers, middlewares, config
- Organisation en contextes côté mobile (useAuth, useTheme)
- Réutilisation des composants avec props clairs (ShareCard, BackButton, etc.)

## Gestion des données

- Base MySQL relationnelle avec tables utilisateurs, courses, commentaires, feedbacks...
- Requêtes paramétrées et sécurisées avec mysql2
- Liaison correcte entre les entités (foreign keys : course.user\_id, feedback.user\_id)

## API RESTful

- Création d'une API complète : CRUD utilisateurs, CRUD courses, interactions (likes, commentaires)
- Respect des standards REST : GET, POST, PUT, DELETE, avec statuts HTTP cohérents
- Structure claire pour accéder aux courses publiques ou privées

## UX/UI & Design

- Intégration d'un thème clair/sombre dynamique (via useTheme + AsyncStorage)
- Interface moderne et responsive : flex layout, padding adaptés, couleurs contrastées
- Icônes expressifs avec react-native-vector-icons (Ionicons)

## Géolocalisation et cartographie

- Utilisation de react-native-maps avec suivi GPS en temps réel (background updates via expo-location)
- Tracés GPS enregistrés sous forme de coordonnées dans la base (format JSON)
- Mode défi qui trace un ancien parcours et guide l'utilisateur à le refaire

#### Stockage & persistance

- Stockage local via @react-native-async-storage : token, préférences, thème
- Gestion automatique du token et reconnection persistante
- Conservation des préférences même après redémarrage de l'application

#### Débogage & expérience dev

- console.log et logs côté backend pour suivi des erreurs
- Outils Postman pour tester les routes backend
- Espace sécurisé pour les feedbacks avec journalisation serveur

#### Déploiement et modularité

- Fichiers .env pour configuration : port, JWT\_SECRET, identifiants MySQL
- Architecture prête à être migrée en production avec Docker ou Railway
- Backend et frontend admin dissociés pour pouvoir scaler indépendamment

### 1.3 Technologies utilisées (récapitulatif)

#### Mobile (React Native + Expo) :

- React Native avec Expo CLI
- expo-router (navigation filesystem)

- react-native-maps
- react-native-view-shot + expo-sharing (capture de composant + partage natif)
- @react-native-async-storage/async-storage
- Ionicons
- Context API (useAuth / useTheme)

#### Backend (Node.js + Express) :

- Express.js
- mysql2
- bcrypt
- jsonwebtoken
- dotenv
- cors

#### Web Admin (React + Vite) :

- React + Vite
- Tailwind CSS
- React Router DOM
- Axios
- Chart.js

## 2. Architecture générale

### 2.1. Schéma de l'architecture globale

Le projet RunYnov repose sur une architecture modulaire composée de trois couches principales : l'application mobile utilisée par les coureurs, l'API backend assurant le traitement des données, et l'interface web dédiée à l'administration. Ces modules communiquent entre eux via des appels réseau sécurisés (HTTP + JWT), et leur séparation claire permet une évolution indépendante de chaque bloc.

Voici le découpage fonctionnel de chaque composant :

#### 1. Utilisateur (Application Mobile – React Native)

L'utilisateur interagit principalement via l'application mobile. Celle-ci constitue le cœur de l'expérience utilisateur. Elle est dotée de nombreuses fonctionnalités :

- Authentification (inscription, connexion, gestion du token JWT)
- Lancement de courses avec géolocalisation active
- Enregistrement automatique du tracé GPS (tableau de coordonnées)
- Affichage des statistiques de performance en temps réel (vitesse, distance, durée)
- Consultation de l'historique des courses et détail des anciennes sessions
- Ajout de commentaires, likes sur les courses des autres utilisateurs
- Refaire une course en mode défi (rejeu du tracé)
- Partage d'une carte personnalisée de la course (ShareCard)
- Thème clair/sombre dynamique
- Paramètres utilisateur : changement email, mot de passe, suppression compte



- Soumission d'un avis (feedback) aux développeurs

Les communications se font exclusivement via des requêtes HTTP, utilisant la méthode `fetch()` avec un token JWT pour l'authentification.

## 2.2 API REST (Node.js + Express)

L'API fait office de pont entre les données utilisateurs et l'application. Elle est conçue selon les bonnes pratiques RESTful, avec des routes claires et séparées par ressource (users, courses, feedbacks, goals...).

Elle comprend notamment :

- Authentification : vérification des credentials, génération du JWT, vérification middleware
- Sécurité : mot de passe haché avec `bcrypt`, vérification du token à chaque requête sensible
- Stockage et requêtes MySQL via `mysql2`
- Gestion des interactions (likes, commentaires)
- Génération automatique d'objectifs journaliers (daily goals)
- Traitement des feedbacks utilisateurs
- Accès public à certaines ressources (courses récentes)

Elle joue également un rôle central dans la validation, le contrôle des accès et la cohérence des données.

## 2.3. Base de données relationnelle (MySQL)

L'ensemble des données de l'application est structuré dans une base MySQL classique, dont la conception suit une logique relationnelle stricte avec des clés étrangères :

- Table users : id, email, username, mot de passe haché, photo de profil, rôle, etc.
- Table courses : distance, durée, tracé GPS, date, vitesse moyenne, FK vers user\_id
- Table goals : objectif quotidien par utilisateur et date
- Table feedbacks : contenu texte + FK user\_id
- Table interactions : type (like/comment), course\_id, user\_id, contenu, date
- Table sessions (admin) pour vérifier les tentatives de connexion éventuelles

Cette base offre une excellente scalabilité et permet une gestion fine des statistiques.

## 2.4 Interface Web Admin (React + Vite)

Cette interface est uniquement accessible aux administrateurs du système, après identification.

Elle permet notamment de :

- Visualiser l'ensemble des utilisateurs avec possibilité de suppression
- Accéder à toutes les courses enregistrées et les supprimer si nécessaire
- Lire les avis/feedbacks laissés par les utilisateurs depuis l'application
- Suivre les données statistiques : nombre d'utilisateurs, volume de courses, etc.
- Profiter d'un tableau de bord fluide, moderne et responsive

Les appels vers l'API se font avec Axios, et l'interface est construite avec des composants modulaires réutilisables, stylisés avec Tailwind CSS.

## 2.5 Arborescence des dossiers

Le projet est organisé en trois répertoires principaux : App/, Backend/, et Frontend/.

App/ (Application React Native avec Expo)

Ce dossier contient l'intégralité de l'application mobile, conçue avec expo-router :

App/

- | — app/    →    navigation par fichiers
- |   └ (tabs)/    →    navigation principale : index, run, profile, etc.
- |   └ course/    →    page detail d'une course (historyDetail.tsx)
- |   └ settings/    →    paramètres utilisateur
- |   └ feedback.tsx, about.tsx    →    écrans secondaires
- | — components/    →    composants réutilisables : ShareCard, BackButton...
- | — context/    →    AuthContext.tsx et ThemeContext.tsx
- | — utils/    →    fonctions utilitaires : token, formatDuration...
- | — theme.ts    →    couleurs, thèmes
- | — app.config.js    →    configuration Expo
- | — assets/    →    icônes, images, polices

Backend/ (API Node.js + Express)

Le backend est structuré de manière claire selon une architecture MVC :

Backend/

- | — routes/ → routes REST : auth, courses, feedbacks...
- | — controllers/ → logique des endpoints, traitement des requêtes
- | — middlewares/ → authMiddleware, validation, logs éventuels
- | — config/ → connexion MySQL
- | — .env → variables d'environnement (DB, JWT\_SECRET...)
- | — index.js → point d'entrée de l'API (serveur express)

Frontend/ (Interface Web Admin – React + Vite)

Interface web minimaliste et rapide avec React + Vite + Tailwind :

Frontend/

- | — src/
- | — components/ → UI : Table, Header, Navbar, StatCards...
- | — pages/ → pages dédiées : Utilisateurs, Feedbacks, Dashboard
- | — services/ → appels API via Axios
- | — public/ → favicons, logo, manifest
- | — App.tsx, main.tsx → point d'entrée de l'application React
- | — tailwind.config.js → configuration tailwind

| — vite.config.ts      →      configuration Vite (vite + React)

## 2.6 Choix techniques

Chaque technologie adoptée pour RunYnov a été soigneusement sélectionnée selon des critères de performance, de simplicité de mise en œuvre, et d'adéquation avec un projet mobile/temps réel :

Élément	Choix	Justification technique
Frontend mobile	React Native + Expo	Cross-platform, rapide, fort écosystème
Navigation mobile	expo-router	Navigation déclarative intuitive, dossier = route
Backend API	Node.js + Express	Léger, performant, adapté aux APIs REST
Base de données	MySQL	Structure relationnelle, bien adaptée à notre modèle
Authentification	JWT + bcrypt	Sécurisé, standard dans le web
Gestion préférences	AsyncStorage + Context	Persistant même après redémarrage
Géolocalisation / Cartes	react-native-maps	Support natif GPS + tracé Polyline
Statistiques visuelles	Chart.js (web) + LineChart (mobile)	Données lisibles, interaction rapide
Admin Web	React + Vite + Tailwind CSS	Développement rapide, interface moderne, responsive

Élément	Choix	Justification technique
Partage	react-native-view-shot + expo-sharing	Capture de composant + partage natif
Icônes	Ionicons	Cohérence graphique dans tout le projet

Cette base technologique permet à RunYnov d'être :

- Léger et performant
- Sécurisé sur les échanges (JWT)
- Responsive et agréable à utiliser (dark/light)
- Facile à maintenir (code modulaire)
- Extensible à terme (API publique, nouveaux écrans...)

### 3. Guide de Lancement du Projet

Avant Cette section fournit **l'ensemble des étapes nécessaires pour configurer, installer et lancer** correctement l'ensemble du projet, qui comprend **trois modules** principaux :

- un **backend** Node.js + MySQL (API REST),
- une **application mobile** React Native via Expo,
- un **frontend web** React.

Elle est conçue pour un développeur ou un contributeur technique reprenant le projet.

---

### 3.1 Prérequis Généraux

Avant toute chose, assurez-vous que votre environnement de développement est correctement configuré.

#### Outils et dépendances requises

Outil	Version Recommandée	Utilité
Node.js	18.x.x	Pour exécuter les serveurs JS (backend + frontend)
npm ou Yarn	npm 9+ ou Yarn 1.22+	Gestionnaire de paquets JavaScript
Expo CLI	Dernière version	Démarrage de l'application mobile avec React Native
MySQL	8.x	Base de données relationnelle
phpMyAdmin	Web UI	Interface graphique pour gérer MySQL

#### Installation rapide

Voici comment installer les éléments si ce n'est pas encore fait :

```
# Node.js (installe aussi npm)
```

```
https://nodejs.org/
```

```
# Yarn (facultatif)
```

```
npm install -g yarn
```

```
# Expo CLI
```

```
npm install -g expo-cli
```

Pour **MySQL + phpMyAdmin**, vous pouvez :

- Télécharger [XAMPP](#) (inclut Apache, MySQL, phpMyAdmin).
- Ou [MAMP](#) si vous êtes sur Mac.

---

## 3.2 Installation Initiale du Projet

### 1. Récupérer le code source

Clonez le dépôt Git du projet :

```
git clone https://github.com/nom-utilisateur/nom-du-projet.git  
cd nom-du-projet
```

Si le projet est distribué sous forme d'archive .zip, décompressez-le et accédez au dossier.

---

### 2. Scripts de configuration automatique

Deux scripts bash sont fournis pour faciliter la configuration initiale :



```
./start.sh  
./replace-ip.sh
```

➤ **start.sh**

- Installe les dépendances dans les sous-dossiers (Backend/, App/, Frontend/)
- Prépare les fichiers .env si nécessaire
- Vérifie les versions de Node

➤ **replace-ip.sh**

- Remplace l'adresse IP locale dans les fichiers de configuration des clients
- Nécessaire pour que l'application mobile accède à l'API en réseau local

Si l'exécution échoue, appliquez :

```
chmod +x start.sh replace-ip.sh  
Puis relancez les scripts.
```

---

### 3.3 Backend — API Node.js & Base de Données

#### Structure du dossier

```
/Backend
```

```
|— database/  
|   └─ Setup.sql  
|— controllers/  
|— routes/  
|— models/  
|— .env  
└─ index.js
```

## 1. Préparation de la base de données

Ouvrez **phpMyAdmin** et effectuez les opérations suivantes :

### a) Créez une base de données (si elle n'existe pas déjà)

```
CREATE DATABASE runynov_db DEFAULT CHARACTER SET utf8mb4  
COLLATE utf8mb4_general_ci;
```

### b) Importez le script Setup.sql

Accédez à votre base, puis :

- Cliquez sur "**Importer**",
- Sélectionnez le fichier Setup.sql,
- Cliquez sur **Exécuter**.

Le script va créer les tables nécessaires (users, sessions, etc.)

---

## 2. Ajouter un compte administrateur

Dans l'onglet SQL de phpMyAdmin, exécutez :

```
INSERT INTO users (email, password, username, role, verified)
VALUES (
  'admin@runynov.com',
  '$2a$10$KIX/8k8uUvHvfQXFdTq9MeWZb9Jc5mK1H2gqx4XrX4aVqWTfKh
  Uoy',
  'admin',
  'admin',
  1
);
```

Le mot de passe est déjà chiffré avec Bcrypt. Il correspond à un mot de passe fort. Aucun besoin de le modifier.

---

## 3. Configuration du backend

Créez un fichier .env dans /Backend :

```
PORT=3000
DB_HOST=localhost
DB_USER=root
```

```
DB_PASSWORD=your_password
```

```
DB_NAME=runynov_db
```

```
JWT_SECRET=your_jwt_secret
```

Remplacez your\_password par votre mot de passe MySQL.

Le JWT\_SECRET est utilisé pour sécuriser les jetons d'authentification.

---

#### 4. Installation et démarrage du backend

```
cd ./Backend
```

```
npm install
```

```
npm run dev
```

- Le backend démarre sur `http://localhost:3000`
  - Un message de confirmation s'affiche dans le terminal si la connexion DB est réussie.
- 

### 3.4 Application Mobile (React Native)

#### Aperçu

L'application mobile permet aux utilisateurs de se connecter, consulter leurs données, et interagir avec l'API REST.

## Structure du dossier

```
/App
|— assets/
|— components/
|— screens/
|— navigation/
|— services/
└─ App.js
```

---

### 1. Lancer l'application mobile

```
cd ./App
npm install
npx expo start
```

Une interface web Expo s'ouvre automatiquement.

---

### 2. Accès depuis un smartphone

#### a) Installez Expo Go :

- Android : Play Store
- iOS : [App Store](#)

## b) Scannez le QR code généré

Votre application mobile se lance sur l'appareil.

**Important** : Le téléphone **et l'ordinateur doivent être connectés au même réseau Wi-Fi** pour que cela fonctionne.

---

## Débogage mobile

- **L'application ne charge pas ?**
  - Assurez-vous que `replace-ip.sh` a bien modifié l'adresse de l'API dans les fichiers `.env` ou `constants.js`.
- **Erreur de dépendance ?**
  - Faites un `npm install` dans `/App`.

---

## 3.5 Interface Web — Frontend React

### Objectif

Permet d'afficher une interface responsive de gestion côté utilisateur ou administrateur. Connexion, affichage de données, graphiques, etc.

### Structure du dossier

```
/Frontend
|— src/
|— |— components/
|— |— pages/
|— |— services/
```

```
| — └─ App.jsx  
| — .env  
└─ vite.config.js
```

---

## 1. Lancer l'application web

```
cd ./Frontend  
npm install  
npm run dev
```

Par défaut, Vite servira l'application sur <http://localhost:5173>.

Ouvrez cette URL dans votre navigateur.

---

## 2. Configuration API

Vérifiez que votre fichier `.env` contient bien l'URL de l'API backend :

```
VITE_API_URL=http://localhost:3000
```

---

## Vérifications

Composant	Fonctionne si...
API Backend	curl http://localhost:3000/health renvoie "ok"
Frontend Web	http://localhost:5173 affiche l'interface React
Mobile avec Expo	L'appli charge les données depuis l'API

---

## 3.6 Sécurité et Bonnes Pratiques

### Sécurisation

- Ne jamais versionner vos fichiers .env
- Ajouter .env au fichier .gitignore
- Générer un JWT\_SECRET aléatoire (via openssl rand -hex 32)

### Commandes utiles

```
# Nettoyage des modules
rm -rf node_modules
npm cache clean --force
# Réinstallation
npm install
```

---

## 3.7 Prochaine Étape : Déploiement

Une fois l'environnement local validé, vous pouvez envisager un déploiement :



<b>Module</b>	<b>Méthode recommandée</b>
Backend	VPS (ex : Render, Railway, Vercel backend)
Frontend	Vercel, Netlify, ou Github Pages
Base MySQL Plan Cloud	(PlanetScale, AWS RDS, etc.)

## 4. Application Mobile

L'application mobile représente le cœur de l'expérience utilisateur du projet RunYnov. Développée avec React Native et Expo, elle est pensée pour offrir une expérience fluide, immersive et complète aux coureurs, tout en conservant une interface simple et intuitive.

Elle intègre des fonctionnalités avancées comme le suivi GPS en temps réel, les objectifs personnalisés, le mode défi ou encore l'interaction communautaire avec likes et commentaires.

### 4.1. Fonctionnalités principales

Ci-dessous une revue détaillée de chaque fonctionnalité clé intégrée dans l'application :

Authentification sécurisée (JWT + email vérifié)

- Le système d'inscription permet à chaque utilisateur de créer un compte avec une adresse email valide, un mot de passe et un nom d'utilisateur.
- La connexion vérifie les identifiants et retourne un JWT (JSON Web Token) signé côté serveur.
- Le token est stocké localement (AsyncStorage) pour maintenir la session entre les relances de l'application.
- Une vérification par email est exigée à la première connexion (champ verified dans la base), renforçant la sécurité.

## Suivi GPS en temps réel

- L'application utilise expo-location pour accéder à la position de l'utilisateur via le GPS du téléphone.
- Un système de géolocalisation continue est mis en place : une position est enregistrée toutes les 3 secondes pendant la course.
- Chaque point GPS est stocké sous forme de tableau d'objets { latitude, longitude }, qui est ensuite converti en JSON pour stockage en base.

## Enregistrement des courses

- Une fois une course lancée, l'application enregistre :
  - La distance totale parcourue (somme entre chaque paire de points GPS)
  - La durée écoulée (timer en secondes)
  - La vitesse moyenne (distance / temps)
  - Le tracé GPS (path)
- À la fin de la course, ces données sont transmises au backend via un POST vers /api/courses.
- Une course peut être publique (visible dans l'historique global) ou privée (utilisateur uniquement).

## Mode défi

- Une course déjà enregistrée peut être rejouée via un mode "défi".
- Ce mode charge le tracé GPS d'une course passée et le rejoue en arrière-plan.
- Le but est de parcourir une distance équivalente ou supérieure pour battre son ancien score.
- Cela favorise la motivation et la progression de l'utilisateur.

## Historique & détails des courses



- Une liste complète des anciennes courses est affichée dans l'onglet Historique.
- Pour chaque course : date, distance, durée, carte (MapView), vitesse moyenne.
- Il est également possible d'accéder à une course d'un autre utilisateur si elle est publique.
- Le détail comprend aussi les likes reçus, les commentaires et les actions possibles (partage, mode défi, etc.).

## Likes et commentaires (communauté)

- Chaque utilisateur peut :
  - Aimer une course d'un autre utilisateur (like avec cœur jaune / rouge selon l'état)
  - Ajouter un commentaire visible par tous sur une course publique
- Les commentaires sont affichés sous forme de bulles :
  - À droite et en gris foncé s'il s'agit de ses propres commentaires
  - À gauche et en gris clair pour les autres utilisateurs
- Cela apporte une dimension communautaire et sociale à l'application.

## Objectif journalier automatique

- Chaque jour, un objectif est généré automatiquement (par exemple : "Faire 2 km aujourd'hui").
- L'objectif est considéré comme atteint si l'utilisateur a une course  $\geq 2$  km dans la journée.
- L'état de validation est enregistré dans la table goals.

- L'objectif est affiché sur la page d'accueil avec une icône  ou  selon l'état.

### Thème clair / sombre

- L'utilisateur peut basculer entre un thème clair et un thème sombre dans les paramètres.
- Le thème affecte :
  - Le fond d'écran
  - Les textes, icônes et composants
- Le choix du thème est stocké en local (AsyncStorage) pour persister entre les relances de l'application.
- Tout le projet est basé sur un ThemeContext permettant de centraliser la logique et d'adapter dynamiquement les composants.

### Langue (i18n – prévu mais non encore implémenté)

- La structure est prête pour l'internationalisation via un système de dictionnaires et une librairie comme react-i18next.
- L'idée est de proposer au moins deux langues : Français et Anglais.
- L'utilisateur pourra changer la langue depuis les paramètres.
- À ce jour, cette fonctionnalité est prévue mais n'a pas été priorisée dans cette version de l'application.

### Paramètres utilisateur (compte)

- L'utilisateur a accès à une page "Paramètres" lui permettant :
  - De changer son adresse email (requête PUT /api/auth)
  - De modifier son mot de passe (avec vérification de l'ancien)
  - De supprimer son compte (après vérification par mot de passe + alerte modale)

- Les paramètres sont sécurisés avec vérification via token JWT.

#### Partage d'une course

- Pour chaque course, un bouton de partage est proposé.
- Une carte visuelle est générée dynamiquement via un composant ShareCard.
- Elle affiche :
  - Le nom de l'utilisateur
  - La distance, la durée, la vitesse
  - Le tracé GPS stylisé
- Le composant est capturé avec react-native-view-shot et partagé avec expo-sharing.

#### Feedback utilisateur (avis)

- Dans l'onglet Menu, l'utilisateur peut écrire un avis via un champ texte + note sur 5 étoiles.
- L'avis est ensuite envoyé à l'API backend (table feedbacks) et consultable par les administrateurs dans l'interface web.
- Cela permet de centraliser les retours utilisateurs sans passer par un store (Google Play ou App Store).

## 5. API BACKEND

L'API REST du projet RunYnov constitue le socle de communication entre l'application mobile, l'interface web administrateur et la base de données relationnelle MySQL. Développée avec Node.js et Express.js, cette API a été pensée de manière modulaire, sécurisée et évolutive. Elle couvre tous les aspects fonctionnels de l'application : authentification, gestion des utilisateurs,

des courses, des commentaires, des likes, des feedbacks, ainsi que des objectifs journaliers.

## 5.1. Schéma général de l'API

L'API suit les conventions REST : chaque entité (utilisateur, course, commentaire...) dispose de routes spécifiques organisées par ressource, en respectant les bonnes pratiques HTTP (GET, POST, PUT, DELETE). Elle est hébergée localement (ou sur un serveur Node) et communique avec la base de données MySQL via le module mysql2.

Caractéristiques générales :

- Express.js utilisé comme serveur HTTP minimaliste et performant.
- Architecture MVC : séparation claire des routes, contrôleurs, middlewares et configuration.
- Découpage modulaire des routes : chaque domaine fonctionnel (auth, user, courses, feedbacks, interactions...) possède son propre fichier de routes.
- Les réponses sont normalisées : codes HTTP explicites, messages clairs, structure homogène.
- Utilisation intensive de middlewares pour la vérification des tokens, la gestion des rôles, le contrôle des erreurs, etc.

Flux général :

- Mobile envoie des requêtes HTTP à l'API (ex: /api/courses).
- API vérifie l'authenticité du JWT et traite la requête si l'utilisateur est authentifié.

- MySQL est interrogé (lecture / écriture / modification) via mysql2 ou Sequelize.
- Réponse retournée à l'application avec données JSON ou message d'état.

Modules principaux :

- auth : inscription, connexion, suppression de compte
- users : récupération et mise à jour des données personnelles
- courses : enregistrement, récupération, modification de courses
- interactions : likes & commentaires
- feedbacks : système de retour utilisateur
- goals : objectifs journaliers
- admin : gestion admin des utilisateurs et feedbacks

## 5.2. Sécurité & Authentification

La sécurité des échanges est un pilier central du backend.

L'authentification est basée sur un système de token JWT (JSON Web Token), et toutes les données sensibles (mots de passe) sont hachées avant stockage.

Éléments de sécurité mis en place :

- Hachage de mots de passe : bcrypt est utilisé avec un salt pour crypter les mots de passe. Les données en base sont non lisibles.
- JWT :
  - Généré à la connexion et signé avec un secret côté serveur.
  - Expire automatiquement au bout de 24 heures.

- Transmis dans le header Authorization de chaque requête protégée.
- Middleware d'authentification :
  - Chaque route sensible est protégée par un middleware authMiddleware.
  - Ce middleware vérifie le JWT, décode le token et injecte les informations du user (id, rôle) dans la requête.
- Vérification d'email :
  - Lors de l'inscription, un champ verified est enregistré.
  - L'utilisateur ne peut pas se connecter tant que son email n'a pas été vérifié (dans un système complet, cela passerait par un lien de validation).
- Séparation des privilèges :
  - Des middlewares spécifiques vérifient si le rôle du user est "admin" avant d'autoriser certaines routes.
  - Cela permet de cloisonner proprement les fonctionnalités utilisateur / administrateur.

### 5.3 Principales routes REST

Le backend expose un ensemble de routes organisées par fonctionnalité. Voici une table résumant les endpoints majeurs, tous préfixés par /api :

Méthode	Route	Description
POST	/auth/register	Crée un nouvel utilisateur, hachage



Méthode	Route	Description
		du mot de passe, retour message
POST	/auth/login	Vérifie les identifiants, retourne un token JWT si valide
GET	/users/me	Retourne les infos du compte connecté (id, email, username, rôle...)
PUT	/users/me	Permet de modifier son mot de passe, email ou image de profil
DELETE	/users/me	Supprime le compte (avec vérification du mot de passe)
POST	/courses	Enregistre une nouvelle course (distance, durée, path GPS...)
GET	/courses	Liste toutes les courses de l'utilisateur connecté
GET	/courses/:id	Détail complet d'une course par son id
GET	/courses/public/:id	Détail d'une course d'un autre utilisateur (si publique)

Méthode	Route	Description
GET	/interactions/:id/comments	Liste tous les commentaires d'une course
POST	/interactions/:id/comments	Ajoute un commentaire à une course
POST	/interactions/:id/like	Like / Unlike une course (toggle)
GET	/interactions/:id/likes	Retourne le nombre de likes + état actuel
GET	/goals/daily	Renvoie l'objectif journalier généré ou en cours
GET	/goals/history	Historique des objectifs atteints
POST	/feedback	Envoie un avis (content + rating) à la base
GET	/admin/users	Liste tous les utilisateurs (admin uniquement)
GET	/admin/feedbacks	Liste tous les feedbacks utilisateurs (admin uniquement)

## 5.4. Structure de la base de données (MySQL)

Le backend repose sur une base MySQL normalisée. Chaque entité est représentée par une table avec ses relations. Voici les principales :

Table	Champs	Description
users	id, email, password, username, profile_picture, verified, role, created_at	Stocke les utilisateurs
courses	id, user_id, distance, duration, path, avg_speed, start_time	Stocke les données de chaque course avec son tracé GPS
interactions_likes	id, user_id, course_id	Relation de like entre user et course
interactions_comments	id, user_id, course_id, content, created_at	Commentaires d'une course
goals	id, user_id, date, label, completed	Objectifs journaliers associés à un utilisateur
feedbacks	id, user_id, content, rating, created_at	Feedback utilisateur avec note
admin_logs	id, action, target, admin_id, date	(optionnel) Historique des actions admin

Structure relationnelle :

- Un utilisateur possède plusieurs courses (1-N)
- Une course peut être liée par plusieurs utilisateurs (N-N)
- Une course peut recevoir plusieurs commentaires
- Chaque utilisateur peut avoir un objectif journalier généré automatiquement
- Un feedback est associé à un utilisateur unique

Optimisations techniques :

- Index sur user\_id, course\_id pour accélérer les requêtes
- Utilisation de JSON pour le champ path (table courses) contenant le tracé GPS
- Transactions SQL pour les opérations critiques (ex: suppression utilisateur + feedbacks liés)

## 6. Interface Admin Web

L'interface administrateur de RunYnov constitue un outil stratégique destiné à superviser, modérer et analyser l'activité des utilisateurs sur la plateforme. Elle permet de centraliser toutes les données issues de l'application mobile dans un tableau de bord ergonomique et intuitif, conçu pour offrir un maximum de contrôle aux administrateurs tout en garantissant la sécurité et la clarté de l'information.

### 6.1. Objectifs de l'interface admin

L'interface web admin a été conçue pour répondre à plusieurs besoins fonctionnels du point de vue de l'équipe de gestion :

- Visualisation en temps réel de l'activité des utilisateurs (inscriptions, feedbacks, nouvelles courses).
- Gestion des comptes utilisateurs : consultation des profils, statistiques personnelles, nombre de courses.
- Supervision de la qualité de service via les retours d'expérience des utilisateurs (feedbacks).
- Accès à des indicateurs clés de performance (KPI) via des tableaux de bord et graphiques.
- Sécurité d'accès renforcée : seules les personnes avec le rôle administrateur peuvent accéder à cette interface.
- Indépendance technique : l'interface fonctionne comme une Single Page Application (SPA) autonome, distincte de l'app mobile, mais reposant sur la même API sécurisée (authentification JWT, middlewares admin).

En résumé, l'objectif est de doter les administrateurs d'une interface fiable, rapide, intuitive et extensible.

## 6.2. Fonctionnalités développées

L'interface admin RunYnov propose différentes vues accessibles une fois l'administrateur connecté avec son compte personnel. Voici un aperçu des fonctionnalités principales développées à ce jour :

### Connexion administrateur

- Formulaire de connexion sécurisé
- Authentification via appel à `/auth/login`
- Vérification du rôle "admin" dans le token JWT
- Stockage du token dans le `localStorage` du navigateur

- Redirection automatique vers le dashboard après connexion
- Protection des routes non autorisées (guard sur les routes privées)

### Gestion des utilisateurs

- Liste complète de tous les utilisateurs enregistrés dans la base
- Données affichées : id, nom, email, date de création, rôle, statut vérifié, nombre de courses
- Tri et pagination sur la liste (amélioration future possible)
- Affichage conditionnel du bouton “Supprimer” ou “Désactiver” (non encore implémenté)

### Feedbacks utilisateurs

- Vue synthétique de tous les avis envoyés via l’application mobile
- Affichage du contenu du feedback, note (de 1 à 5 étoiles), utilisateur concerné, date
- Fonctionnalité de suppression ou de modération prévue
- Vue utile pour détecter les dysfonctionnements ou points de friction dans l’app mobile

### Courses publiques

- Affichage des dernières courses réalisées par la communauté (filtrées via l’API)
- Affichage des données clés : distance, durée, vitesse moyenne, nombre de likes
- Possibilité d’exporter ou de trier les données (fonctionnalité envisagée)
- Intérêt : observer les tendances, identifier les utilisateurs actifs

### Statistiques globales (Tableau de bord)

- Vue “Dashboard” regroupant les chiffres clés de l’application :
  - Nombre total d’utilisateurs
  - Nombre total de courses enregistrées
  - Nombre de feedbacks reçus
  - Note moyenne des feedbacks (ex: 4.3 / 5)
  - Courbe d’évolution des utilisateurs ou des courses (via Chart.js)
- Objectif : offrir une vue macro de l’évolution de la plateforme

### 6.3. Technologies utilisées

L’interface admin a été développée avec un stack moderne, basé sur React.js et Tailwind CSS, afin d’assurer un développement rapide, une performance optimale et un design cohérent avec l’identité de l’application.

Technologie	Rôle / Usage
React + Vite	Framework principal pour construire la SPA avec performances optimisées
TypeScript	Typage statique pour sécuriser et structurer le code
React Router DOM	Gestion de la navigation entre les pages (routes sécurisées, redirections...)
Axios	Requêtes HTTP vers l’API sécurisée (/api), avec en-tête Authorization
Tailwind CSS	Framework CSS utilitaire pour un design moderne, responsive et personnalisable rapidement
Chart.js	Librairie de graphiques intégrée pour les statistiques (barres, lignes, camemberts...)
localStorage	Stockage persistant du token JWT après login, pour session utilisateur

Technologie	Rôle / Usage
JWT (JSON Web Token)	Authentification sécurisée avec vérification de rôle (admin) sur chaque route protégée

Architecture technique simplifiée :

- Le frontend admin est entièrement découplé de l'app mobile.
- L'ensemble des appels API REST se fait sur les mêmes endpoints que l'app mobile.
- Seules les routes protégées (admin uniquement) sont exposées (ex : /api/admin/users).
- Le token JWT est validé par l'API et vérifie le rôle de l'utilisateur.
- Les erreurs sont interceptées et affichées dans l'interface (gestion des erreurs Axios + redirections si non autorisé).

Enjeux futurs :

- Ajout d'un système de logs admin (qui a supprimé quoi, quand ?)
- Possibilité de filtrer les utilisateurs inactifs ou les feedbacks malveillants
- Export CSV des utilisateurs / courses
- Gestion plus fine des droits (modérateur vs. super admin)

## 7. Déploiement

Le déploiement du projet RunYnov s'articule autour de trois briques indépendantes mais interconnectées : l'API backend (Node.js + MySQL), l'application mobile (Expo / React Native) et l'interface d'administration web (React + Vite). Chacune de ces parties a ses propres contraintes et modalités de



déploiement, que ce soit en environnement local (développement) ou distant (production).

## 7.1. Backend (API Node.js + MySQL)

L'API est le cœur de l'infrastructure RunYnov. Elle est responsable de la logique métier, de la gestion des utilisateurs, des courses, des objectifs, des interactions (likes/commentaires), du feedback utilisateur, de l'authentification, et plus encore. Elle expose une interface RESTful sécurisée par JWT, consommée aussi bien par l'application mobile que par l'interface admin web.

### Déploiement en local (développement)

- Lancement du serveur :
  - nodemon index.js pour le développement à chaud
  - npm start pour un lancement standard
- Utilisation d'un fichier .env contenant les variables sensibles :
  - DB\_HOST, DB\_USER, DB\_PASSWORD, DB\_NAME, JWT\_SECRET, PORT, etc.
- Connexion à MySQL local :
  - Via XAMPP ou MAMP (phpMyAdmin local)
  - Ou via une image Docker (mysql:5.7)
- Architecture REST :
  - Toutes les routes sont préfixées par /api
  - Accès protégé aux routes sensibles via authMiddleware
- Logs serveur :
  - En développement : console.log
  - En production : prévu avec morgan ou winston

### Déploiement distant (production – prévu)

- Hébergement sur un serveur distant (VPS, Render, Railway, ou machine d'école)

- Exposition de l'API sur un sous-domaine dédié :
  - Exemple : <https://api.runynov.fr>
- Configuration HTTPS avec certificat SSL (Let's Encrypt ou proxy via Cloudflare)
- Sécurisation accrue :
  - Headers HTTP via helmet
  - Logger centralisé
  - Rate limiting / anti-spam (prévu)
- Base de données distante :
  - Serveur MySQL externe (MariaDB sur Render ou PlanetScale)
  - Authentification forte et backup régulier

## 6.2. Application Mobile (Expo / React Native)

L'app mobile est développée avec Expo, ce qui simplifie les phases de build, test, debug et publication sur les stores. Le développement est centré sur l'itération rapide, avec un workflow fluide entre tests locaux et builds en cloud.

Développement et test :

- Utilisation d'Expo Go :
  - Scan QR code depuis un terminal ou navigateur
  - Test instantané sur appareil physique (iOS/Android)
- Fonctionnement local avec connexion à l'API en HTTP (IP locale, ex : 192.168.1.42)
- Simulation d'authentification avec token stocké en AsyncStorage
- Utilisation de expo-dev-client pour debug avancé
- Déploiement (production)
- Build via EAS Build (Expo Application Services) :

- expo build:android
    - expo build:ios (nécessite un compte Apple)
  - Génération d'un fichier .apk ou .aab (Android App Bundle)
  - Publication sur le Google Play Store :
    - Création d'un compte développeur
    - Génération de la clé de signature
    - Ajout de l'icône, screenshots, catégorie, etc.
  - Publication sur l'App Store (prévue) :
    - Via Transporter / Xcode
- Configuration avancée
- app.json ou app.config.js pour :
    - Définir l'icône, splash screen, nom de l'application, version
    - Permissions requises : localisation, accès galerie, notifications, etc.
  - Gestion des thèmes via Context
  - Reconnexion automatique à l'API via AsyncStorage

### 6.3. Interface Admin Web (React + Vite)

L'interface d'administration est une Single Page Application (SPA) autonome développée avec React + TypeScript + Vite. Elle est conçue pour être légère, performante et facilement déployable sur un hébergement statique.

#### Déploiement local (développement)

- Lancement de l'environnement dev :
  - npm run dev
  - Serveur local accessible sur <http://localhost:5173> (ou autre port)
- Appels à l'API via Axios :
  - Gestion du token JWT dans les headers

- Vérification du rôle “admin” sur les routes protégées
- Variables d’environnement :
  - VITE\_API\_URL pour pointer vers l’API distante ou locale
  - .env à la racine du projet

#### Déploiement distant (production)

- Build de production :
  - npm run build (génère un dossier dist/)
  - Minification, transpilation, purge CSS automatique avec Tailwind
- Hébergement sur :
  - Netlify (drag and drop du dossier dist)
  - Vercel (build auto via GitHub)
  - GitHub Pages (avec ajustement de base URL)
- Configuration DNS :
  - Sous-domaine : admin.runynov.fr (via CNAME)
  - Sécurisation HTTPS automatique
- Redirection automatique en cas d’erreur 404 vers index.html

#### Résumé global du déploiement :

Composant	Local	Production
API Backend	nodemon + MySQL local (XAMPP)	VPS avec Node.js + MySQL (PlanetScale / MariaDB)
App Mobile	Expo Go + IP locale	EAS Build > Play Store (.apk ou .aab)
Admin Web	Vite + React local dev	Netlify / Vercel (dist/ statique)

## 8. Bilan & Perspectives

Après plusieurs semaines de développement, de tests, de validation et d'itérations fonctionnelles, l'équipe est en mesure de tirer un bilan constructif du projet RunYnov. Ce projet a permis de mettre en œuvre une application mobile robuste, une API backend performante et une interface d'administration claire, le tout avec une interconnexion maîtrisée.

### a. Ce qui a bien fonctionné

Le développement de RunYnov a été ponctué de nombreuses réussites techniques et fonctionnelles :

- **Suivi GPS en temps réel** : La gestion de la géolocalisation, grâce à expo-location combiné à react-native-maps, a été un vrai point fort. Les coordonnées sont enregistrées toutes les 3 secondes et stockées sous forme de tracé JSON, permettant une restitution fluide de la course sur une carte. La précision est au rendez-vous, et le rendu visuel est agréable, même pour les longues sessions.
- **Enregistrement des courses** : Grâce à une bonne séparation entre les composants front et la logique backend, l'enregistrement de chaque course (distance, durée, tracé, vitesse) s'effectue de manière fluide. Le lien avec MySQL est stable et fiable, permettant une persistance immédiate après la fin de course.
- **Mode défi** : Cette fonctionnalité apporte une réelle dimension gamifiée à l'app. Elle permet à un utilisateur de relancer une course qu'il a déjà effectuée, en se fixant pour objectif de battre son ancien score ou simplement de répéter le tracé. Cela ajoute une couche de motivation et de rejouabilité intéressante.
- **Authentification sécurisée** : L'intégration de JWT (JSON Web Token) permet une gestion sécurisée des sessions utilisateurs. Le token est généré à la connexion, stocké localement (via AsyncStorage) et utilisé

pour chaque appel protégé. Le middleware de vérification rend l'accès aux ressources conforme au rôle (utilisateur/admin).

- **Système d'interactions** : Le fait de pouvoir "liker" et commenter les courses renforce l'aspect communautaire de l'app. La simplicité du système (un like = un appel POST) permet une bonne performance, même sur connexions mobiles.
- **Interface utilisateur** : L'interface mobile a été pensée en termes d'ergonomie et d'accessibilité. Le design adaptatif clair/sombre s'adapte bien à tous types d'écrans, et le contraste des composants permet une lecture facile.
- **Modulation du code** : Le recours à des hooks personnalisés (useAuth, useTheme), des contextes partagés et des composants réutilisables (comme BackButton, ShareCard, Avatar, etc.) améliore nettement la lisibilité, la maintenance et la scalabilité de l'application.
- **Fonctionnalités hors-ligne** : Grâce au manifest.json bien configuré et au cache local, l'application peut fonctionner même en l'absence de connexion internet, au moins pour l'accès aux données locales et à certaines pages.

## **b. Difficultés rencontrées**

Comme dans tout projet de cette ampleur, plusieurs défis techniques et blocages ont dû être surmontés :

- **Gestion des permissions natives** : L'utilisation de fonctionnalités sensibles (géolocalisation, accès à la galerie, capture d'écran, stockage) nécessite la gestion des permissions différemment selon les OS (Android vs iOS). Des comportements inattendus sont apparus en cas de refus ou de permissions mal configurées.
- **mysql2 & Promesses** : Lors du passage de mysql2 en mode asynchrone (via .promise()), des erreurs classiques sont survenues, notamment l'appel direct à .then() sur une méthode non promise. La correction a

nécessité un refactoring des appels de requêtes pour les rendre compatibles.

- Erreurs null/undefined : Certaines erreurs d’affichage se sont produites dans des cas limites où l’utilisateur n’avait pas encore enregistré de course (ex. `.toFixed()` sur `undefined`, division par zéro pour la vitesse moyenne...). Des conditions de garde ont été ajoutées pour y remédier.
- Persistance du thème : La sauvegarde du choix de thème entre les sessions n’était pas automatique. Il a fallu utiliser `AsyncStorage` pour le conserver localement, puis le restaurer au lancement de l’app.
- Performances des longues courses : Lors de la visualisation de tracés GPS très longs (plusieurs centaines de points), les performances chutent légèrement sur des téléphones moins récents. L’optimisation de ces tracés via simplification ou clustering pourrait être envisagée.
- Langue non implémentée : Bien que l’internationalisation ait été prévue (via `i18next`), le manque de temps n’a pas permis son intégration finale. L’app reste pour l’instant francophone.

### **c. Améliorations possibles & pistes d’évolution**

Plusieurs fonctionnalités supplémentaires ou refontes techniques sont envisageables pour faire évoluer RunYnov vers une version plus complète et professionnelle :

#### **1. Gestion multilingue :**

- Intégration d’un système de traduction (ex. `react-i18next`)
- Choix de la langue dans les paramètres
- Persistance via `AsyncStorage`

#### **2. Statistiques avancées :**

- Ajout d’un écran dédié avec graphiques : progression hebdomadaire, temps moyen, performance par distance

- Intégration de recharts ou victory-native pour une visualisation élégante

### 3. Publication sur les stores :

- Configuration complète du pipeline EAS Build (certificats, icône, splash, etc.)
- Publication sur Google Play Store avec fiche descriptive, screenshots
- Version iOS via Xcode ou Transporter

### 4. Réseau social minimal :

- Système de “suivi” entre utilisateurs (amis)
- Défis communautaires (ex : “Parcours de la semaine”)
- Fil d’actualité des dernières courses

### 5. Notifications push :

- Rappel quotidien si objectif non atteint
- Félicitations après un record
- Utilisation de expo-notifications et d’un cron backend

### 6. Panel admin amélioré :

- Ajout de fonctions de suppression de comptes, modération des commentaires, désactivation
- Statistiques dynamiques sur les utilisateurs (activité, feedbacks...)

### 7. Optimisation GPS :

- Compression des tracés via algorithme de Douglas-Peucker
- Mode économie de batterie (relevé toutes les 10s au lieu de 3s)

### 8. Système de récompense :

- Ajout de badges, niveaux, XP
- Gamification de l’expérience (progression + motivation)



## Conclusions

RunYnov représente une solution mobile moderne, complète et robuste pour l'accompagnement des coureurs, que ce soit dans leur suivi d'activité physique, leur motivation quotidienne ou leur engagement communautaire. Grâce à une architecture modulaire et bien structurée (React Native côté mobile, Node.js côté serveur, MySQL pour la persistance des données et une interface web dédiée aux administrateurs), le projet a su allier performances techniques et expérience utilisateur.

Toutes les fonctionnalités implémentées — du suivi GPS en temps réel au mode défi, en passant par la gestion des objectifs quotidiens et les interactions sociales (likes, commentaires, partage) — témoignent d'un souci du détail et d'une volonté de créer un écosystème sportif complet, agréable et motivant.

L'interface administrateur permet de superviser l'ensemble de la plateforme avec clarté et simplicité, assurant un bon équilibre entre liberté utilisateur et modération.

Les fondations posées sont solides, et l'ensemble du projet est prêt à évoluer vers une V2 plus ambitieuse : support multilingue, notifications push, gamification, ouverture vers d'autres types d'activités, ou encore publication sur les stores mobiles.

RunYnov n'est pas simplement une application de suivi de course : c'est un outil intelligent et communautaire, pensé pour accompagner durablement les utilisateurs dans leur pratique du running.