CollègeAhuntsic

Département d'informatique

Applications web multiplateformes
420-713-AH

Projet Final:

Création d'une application React Native

Gestion des Trajets avec Firebase

Enseignant: KHALIL LOGHLAM

1. Contexte

Ce projet final est la continuité directe du Projet 1 « Création d'une application React Native, Gestion des Trajets ».

Dans le Projet 1, l'application permettait à l'utilisateur de : connexion, enregistrement, menu de navigation coulissant, liste des trajets, ajout de trajets par géolocalisation (positions GPS), affichage sur carte et paramètres, avec une base de données locale SQLite pour gérer les utilisateurs et les trajets.

Dans ce Projet final, l'application doit évoluer vers une vraie application multiplateforme connectée, en intégrant Firebase (Auth + Firestore) tout en gardant un support du mode hors ligne via SQLite.

2. Objectif général

Amener l'étudiant à transformer une application React Native locale en une application connectée, synchronisée et multiplateforme, incluant :

- Authentification sécurisée avec Firebase Auth
- Persistance des données dans une base de données infonuagique (Cloud Firestore)
- Gestion de session utilisateur
- Synchronisation entre stockage local (SQLite) et distant (Firestore)
- Optimisation de l'affichage des listes (FlatList + lazy loading)
- Gestion du mode hors ligne
- Thèmes (mode clair / mode sombre)
- Partage de trajets entre utilisateurs

3. Technologies

- React Native avec Expo (Expo Go / EAS)
- Firebase Auth (authentification par e-mail / mot de passe)
- Cloud Firestore (base de données infonuagique)
- SQLite (base locale utilisée comme cache / mode hors ligne – héritée du Projet 1)
- FlatList pour l'affichage des listes longues
- Expo Location / MapView (géolocalisation et carte, déjà mis en place au Projet 1)

4. Prérequis

- Le Projet 1 doit être fonctionnel :
  - Écrans : Connexion, S'enregistrer, Menu de navigation coulissant, Liste des trajets, Ajout d'un trajet (tracking GPS), Détails d'un trajet sur carte, Paramètres.
  - Base de données locale SQLite pour les utilisateurs et les trajets.
- Le projet final réutilise la même base de code et la même structure d'écrans, mais modifie / enrichit les comportements.

5. Nouvelles exigences fonctionnelles et techniques

5.1 Intégration de l'authentification Firebase (Sign In / Sign Up)

1. **Sign Up (S'enregistrer)**
   - Lors de l'inscription, l'application doit utiliser Firebase Auth pour créer un compte via e-mail + mot de passe.
   - Les informations de profil (prénom, nom, e-mail) sont enregistrées dans une collection `users` de Firestore, associée à l'`uid` retourné par Firebase Auth.
   - Validation : e-mail valide, mot de passe conforme (ex. longueur minimale), utilisateur non déjà existant.
2. **Sign In (Connexion)**
   - L'écran Connexion doit maintenant appeler Firebase Auth pour authentifier l'utilisateur.
   - En cas de succès :
     - L'utilisateur est redirigé vers le menu de navigation / liste des trajets.
     - La session est mémorisée (voir section 5.3).
   - En cas d'erreur : message d'erreur clair (mauvais e-mail, mauvais mot de passe, utilisateur inexistant).
3. **Mot de passe oublié**
   - Ajouter l'intégration de la fonction reset password via Firebase (`sendPasswordResetEmail`) ou, à défaut, laisser l'option pour une évolution future, bien commentée.

Le login ne doit plus reposer sur la seule base SQLite, mais sur Firebase Auth comme source d'authentification principale.

5.2 Intégration de Firestore (structure minimale)

L'application doit utiliser Cloud Firestore comme source principale de données. Toutes les données qui étaient auparavant sauvegardées uniquement en local doivent maintenant être persistées dans Firestore, tout en restant mises en cache dans SQLite pour le mode hors ligne.

5.2.1 Collections minimales à définir

1. **users**
   - `uid` (Firebase Auth)
   - `firstName`
   - `lastName`
   - `email`
   - éventuellement : `theme` (light/dark), autres préférences.

2. **trips** (trajets)
   - `id` (généré par Firestore)
   - `ownerId` (uid du propriétaire)
   - `name` (nom du trajet)
   - `description`
   - `type` (personnel / affaire)
   - `createdAt` (timestamp)
   - `updatedAt` (timestamp)
   - `positionsCount` (nombre de positions)
   - éventuellement : indicateur de synchronisation avec SQLite.
3. **positions** (selon votre design – au choix)
   - Option A : sous-collection `trips/{tripId}/positions`
   - Option B : collection globale `positions` avec champ `tripId`

   Champs:
   - `tripId`
   - `latitude`
   - `longitude`
   - `timestamp`
   - `order` ou `index` (optionnel, position dans le trajet).
4. **shared_trips** (partage des trajets)
   - `tripId`
   - `ownerId` (uid du propriétaire du trajet)
   - `targetUserId` (uid de l'utilisateur avec qui le trajet est partagé)
   - `targetEmail` (email du destinataire, pour affichage / recherche)
   - `sharedAt` (date du partage).

5.3 Gestion de la session

- À l'ouverture de l'application, vérifier si un utilisateur est déjà connecté via Firebase Auth :
  - Si oui : rediriger automatiquement vers l'écran principal (Trajets / Menu).
  - Si non : afficher l'écran Connexion.
- L'option Déconnecter (Logout) dans l'écran Paramètres doit :
  - Appeler `signOut` de Firebase Auth.
  - Réinitialiser les informations de session en mémoire.
  - Rediriger vers l'écran Connexion.

5.4 Remplacement de la persistance locale par Firestore (avec cache SQLite)

Toutes les données métier qui étaient enregistrées uniquement dans SQLite dans le Projet 1 doivent maintenant être persistées dans Firestore, tout en conservant SQLite comme cache local.

5.4.1 Données concernées

- Utilisateurs (profil)
- Trajets (nom, description, date de création, nombre de positions, type de trajet)
- Positions GPS d'un trajet
- Paramètres utilisateur (ex. thème clair/sombre)

5.4.2 Comportement attendu

- En mode en ligne :
  - Toute création / modification / suppression est d'abord envoyée à Firestore.
  - Les données sont ensuite mises à jour en local dans SQLite pour le cache.
- En mode hors ligne :
  - L'application lit les données à partir de SQLite.
  - Les nouveaux trajets ou modifications sont enregistrés dans SQLite avec un statut à synchroniser.
  - Lors du retour en ligne, l'application doit tenter de synchroniser les données en attente vers Firestore (au moins pour les trajets créés hors ligne et leurs positions).

Vous pouvez définir une stratégie simple de synchronisation (par exemple : au démarrage, si connexion disponible, envoyer tous les éléments marqués "non synchronisés").

5.5 Liste des trajets avec FlatList et lazy loading

L'écran Liste des trajets doit être revu pour utiliser :

- FlatList comme composant d'affichage principal.
- Un mécanisme de lazy loading / pagination :
  - Chargement des trajets par page (ex. 10 par 10) depuis Firestore.
  - Utilisation de `onEndReached` pour charger la suite de la liste.
  - Indicateur visuel simple (« Chargement… ») lorsque de nouveaux éléments sont récupérés.

La suppression d'un trajet doit :

- Supprimer le trajet de Firestore (et ses positions)
- Le retirer également de SQLite.

5.6 Mode hors ligne (support SQLite)

En mode hors ligne (absence de connexion) :

- L'écran Liste des trajets doit continuer à fonctionner en lisant les trajets stockés dans SQLite.
- L'utilisateur doit pouvoir :
  - Visualiser les trajets déjà synchronisés.
  - Créer un nouveau trajet (enregistrer des positions GPS) :
    - Les données sont enregistrées dans SQLite avec un marqueur de synchronisation.
- À la reconnexion, la synchronisation vers Firestore doit être tentée.

5.7 Types de trajets : personnel / affaire

L'application doit désormais supporter deux types de trajets :

- Trajet personnel
- Trajet affaire (business)

5.7.1 Création / modification de trajet

- Lors de la création d'un trajet, l'utilisateur doit pouvoir choisir le type de trajet (ex. via un bouton radio, un sélecteur ou un toggle).
- Le type doit être enregistré dans Firestore (champ `type`) et dans SQLite.

5.7.2 Écran Liste des trajets – deux onglets

L'écran Trajets doit proposer deux onglets / tabs :

- Onglet Personnels :
  - Affiche uniquement les trajets de type "personnel" appartenant à l'utilisateur ou partagés avec lui.
- Onglet Affaires :
  - Affiche uniquement les trajets de type "affaire" appartenant à l'utilisateur ou partagés avec lui.

Chaque onglet utilise une FlatList avec lazy loading.

5.8 Mode clair / mode sombre (Dark / Light mode)

L'écran Paramètres doit permettre à l'utilisateur de choisir :

- Mode clair
- Mode sombre

Comportement attendu :

- Le thème sélectionné doit être appliqué à l'ensemble de l'application (couleurs de fond, textes, boutons).
- Le choix de thème doit être persisté :
  - Localement (SQLite ou AsyncStorage) pour application rapide au démarrage.
  - Optionnellement dans Firestore (champ `theme` dans `users`) pour synchronisation entre appareils.

L'interface doit rester cohérente et lisible dans les deux modes.

5.9 Partage des trajets entre utilisateurs

Le système de **partage** permet à un propriétaire de trajet de :

- Partager un trajet avec un autre utilisateur, identifié par son adresse e-mail.
- Retirer le partage pour un utilisateur.

5.9.1 Règles de partage

- Seul le propriétaire du trajet peut :
  - Ajouter un partage
  - Supprimer un partage
- Un utilisateur qui reçoit un trajet partagé :
  - Peut consulter le trajet dans sa liste (marqué comme "partagé").
  - Ne peut pas partager, modifier ou supprimer le trajet d'origine (sauf spécification contraire).

5.9.2 Implémentation Firestore

Collection `shared_trips` :

- `tripId`
- `ownerId`
- `targetUserId` (uid du destinataire)
- `targetEmail`
- `sharedAt` (date de partage).

La liste des trajets affichés pour un utilisateur doit donc combiner :

- Ses propres trajets (`trips` où `ownerId == currentUser.uid`)
- Les trajets qui lui sont partagés (`shared_trips` où `targetUserId == currentUser.uid`).

5.9.3 Écran Détails du trajet – gestion des partages

Dans l'écran Détails du trajet, en plus de la carte et des informations déjà présentes :

- Afficher la liste des utilisateurs autorisés (partages actifs) :
  - Affichage du courriel et, si possible, du prénom / nom.
- Permettre au propriétaire :
  - D'ajouter un partage en saisissant l'adresse e-mail de l'utilisateur.
  - De retirer un partage (ex. bouton "Retirer" à côté de chaque utilisateur partagé).

6. Style et langue

- Tous les textes affichés dans l'interface doivent être en français correct.
- Les libellés, messages d'erreur, confirmations et titres d'écrans doivent être cohérents et compréhensibles.
- Les noms d'écrans, boutons et messages doivent refléter clairement leur rôle (ex. « Se connecter », « S'enregistrer », « Trajets personnels », « Partager », « Retirer le partage », « Mode sombre », etc.).

Remise:

- Date de remise : Jeudi le 4 décembre 2025 avant 8:00
- Remettre via LÉA (Omnivox) un fichier ZIP contenant vos fichiers du travail.
- Le 4 décembre 2025 vous devez présenter votre réalisation pratique à votre enseignant.
- Pendant la présentation l'enseignant vous posera des questions qui seront notées.
- Un travail qui a été remis sur Omnivox mais qui n'a pas été présenté est considéré comme n'ayant pas été remis.

Barème

Section	Critère	Points	Total
1. Authentification Firebase	Sign Up fonctionnel (Firebase Auth) Enregistrement du profil dans Firestore (users) Sign In fonctionnel Gestion des erreurs (email invalide, mdp, utilisateur inexistant) Gestion de la session (auto-login / logout)	5 3 5 3 4	20
2. Firestore – Base de données infonuagique	Sauvegarde des trajets dans Firestore Sauvegarde des positions GPS Structure Firestore conforme (trips, positions, users) Mise à jour des données (update) Suppression Firestore fonctionnelle	5 4 5 3 3	20
3. Mode hors ligne + SQLite	Conservation du SQLite (cache local) Lecture en mode hors ligne Synchronisation locale → Firestore au retour en ligne Gestion du statut "synced" / file d'attente	5 5 5 5	20
4. Interface Trajets (FlatList + types)	Utilisation correcte de FlatList Lazy loading / pagination Firestore Onglets "Personnels" et "Affaires" Sélecteur du type de trajet à la création	5 4 3 3	15
5. Partage des trajets	Ajout d'un partage par e-mail Enregistrement dans shared_trips Affichage des utilisateurs ayant accès Retrait du partage Application correcte du thème	4 3 4 4	15
6. Paramètres – Mode clair / sombre	Persistance du thème (local + option Firestore) Bouton / commutateur fonctionnel dans Paramètres	4 3 3	10
Total		**100**	**100**