Documentation Technique

1. Structure de l'Application

L'application est structurée autour de plusieurs écrans principaux, gérés par une navigation en pile (**Stack Navigator**) et une navigation par onglets (**Bottom Tab Navigator**). Les écrans sont organisés comme suit :

- **LoginScreen** : Écran de connexion.
- **HomeScreen** : Écran d'accueil avec des informations sur les objectifs, les progrès et les notifications.
- **GoalsScreen** : Écran de gestion des objectifs.
- **NotificationsScreen** : Écran de gestion des notifications.
- **ScanScreen** : Écran de scan de produits.
- **ProgressScreen** : Écran de suivi des progrès et des badges.
- **ArticlesScreen** : Écran d'articles et de tutoriels.

2. Navigation

La navigation est gérée par @react-navigation/native-stack et @react-navigation/bottom-tabs. La structure de navigation est définie dans le fichier _layout.tsx.

2.1 Stack Navigator

- **Login** : Écran de connexion.
- **HomeStack** : Contient les onglets de navigation.

2.2 Bottom Tab Navigator

- **HomeTab** : Écran d'accueil.
- **GoalsTab** : Écran des objectifs.
- **NotificationsTab** : Écran des notifications.
- **ScanTab** : Écran de scan.
- **ProgressTab** : Écran des progrès.
- **ArticlesTab** : Écran des articles.

3. Composants Principaux

3.1 LoginScreen

Fonctionnalités :

- Connexion de l'utilisateur via email et mot de passe.
- Redirection vers l'écran d'accueil après une connexion réussie.

Données :

- Utilise usersData pour valider les informations de connexion.

3.2 HomeScreen

Fonctionnalités :

- Affichage des informations de l'utilisateur (nom, avatar).
- Objectif du jour.
- Progression en pourcentage.
- Article suggéré.
- Dernière notification.

Données :

- Les données sont statiques pour le moment.

3.3 GoalsScreen

Fonctionnalités :

- Gestion des objectifs (ajout, suppression, suivi des mini-tâches).
- Filtrage des objectifs par catégorie.
- Calcul de la progression des objectifs.

Données :

- Utilise goalsData pour stocker les objectifs et les mini-tâches.

3.4 NotificationsScreen

Fonctionnalités :

- Gestion des paramètres de notification (rappels, messages motivants, notifications générales).
- Affichage des notifications récentes.

Données :

- Utilise notificationsData pour stocker les notifications.

3.5 ScanScreen

Fonctionnalités :

- Scan de codes-barres pour récupérer les informations sur les produits.
- Affichage des informations du produit (nom, marque, impact écologique).
- Conseils et alternatives écologiques.

Données :

- Utilise l'API Open Food Facts pour récupérer les informations des produits.

3.6 ProgressScreen

Fonctionnalités :

- Affichage des progrès de l'utilisateur.
- Classement des utilisateurs.
- Affichage des badges.

Données :

- Utilise usersData pour simuler les données des utilisateurs.

3.7 ArticlesScreen

Fonctionnalités :

- Affichage des articles suggérés en fonction des objectifs de l'utilisateur.
- Affichage de tous les articles disponibles.

Données :

- Les articles sont stockés dans articlesData.

4. Contexte d'Authentification

Le contexte d'authentification (AuthContext) est utilisé pour gérer l'état de connexion de l'utilisateur.

Fonctionnalités :

- Connexion et déconnexion de l'utilisateur.
- Persistance de l'état de connexion via AsyncStorage.

Composants :

- **AuthProvider** : Fournit le contexte d'authentification à l'application.
- **useAuth** : Hook pour accéder au contexte d'authentification.

5. Styles

Les styles sont définis dans chaque composant à l'aide de StyleSheet.create. Les styles sont réutilisables et modulaires.

6. Données

Les données sont actuellement simulées dans des fichiers JSON (usersData, goalsData, notificationsData, articlesData). Ces données peuvent être remplacées par des appels API dans une version future.

7. Dépendances

- **React Navigation** : Pour la gestion de la navigation.
- **Expo** : Pour l'accès à la caméra et aux icônes.
- **Axios** : Pour les requêtes HTTP.
- **AsyncStorage** : Pour la persistance des données locales.

8. Améliorations Possibles

- **Intégration d'une API** : Remplacer les données statiques par des appels API pour des données dynamiques.
- **Gestion des erreurs** : Améliorer la gestion des erreurs lors des appels API.
- **Internationalisation** : Ajouter la prise en charge de plusieurs langues.
- **Tests** : Ajouter des tests unitaires et d'intégration pour les composants.

Cette documentation technique fournit une vue d'ensemble des composants et des fonctionnalités de l'application. Elle peut être étendue pour inclure des détails supplémentaires sur les implémentations spécifiques ou les flux de données.