meal-planner-esiee

Objectif de l'application

L'objectif de notre application mobile est de fournir aux utilisateurs un moyen pratique et personnalisé de planifier leurs repas afin d'atteindre leurs objectifs de santé. En calculant leur taux métabolique de base (BMR) et à l'aide d'une base de données de recettes alimentaires, nous aidons les utilisateurs à organiser leur alimentation en fonction de leurs besoins et objectifs.

Architecutre

Notre application a été développée en utilisant TypeScript et React Native. Nous avons organisé notre code en plusieurs dossiers pour maintenir une structure claire et modulaire :

Screens

Les écrans de l'application sont tous situés dans le dossier **screens** et chacun a un rôle spécifique. Nous avons les écrans suivants :

- HealthGoalScreen.tsx : Permet aux utilisateurs de renseigner leurs données personnelles pour calculer leur BMR et définir leurs objectifs de santé.
- FoodDatabaseScreen.tsx : Affiche la base de données d'aliments et permet aux utilisateurs de rechercher et de sélectionner des plats.
- MealPlanningScreen.tsx : Permet aux utilisateurs de planifier leurs repas en choisissant des plats et en les ajoutant à leur calendrier de repas.
- RecipeScreen.tsx : Affiche les détails d'une recette sélectionnée, tels que les ingrédients et les instructions de préparation.

Routes

Le dossier **routes** contient le fichier **index.ts** où nous définissons les routes de navigation avec React Navigation. Nous utilisons un **RootStack** qui comprend un onglet inférieur (**BottomTab**) contenant les écrans **Health Goals**, **Food** et **Meal Planner**. L'autre écran dans le **RootStack** est **Food Detail**, qui permet d'afficher les détails d'une recette à partir de n'importe quel endroit de l'application.

- RootStack
  - BottomTab
    - **Health Goals**: `HealthGoalScreen.tsx`
    - **Food** : `FoodDatabaseScreen.tsx`
    - **Meal Planner** : `MealPlanningScreen.tsx`
  - **Recipe** : `RecipeScreen.tsx`

Components

Le dossier **components** contient des composants réutilisables utilisés dans plusieurs écrans de l'application ou dans le but de maintenir une structure claire et modulaire en respectant le **single responsibility principle**. Nous avons les composants suivants :

- AddMealPlanDialog.tsx : Affiche une boîte de dialogue pour ajouter un plat à la planification des repas.
  - Props
    - **visible** : booléen pour afficher ou masquer la boîte de dialogue.
    - **onDismiss** : fonction à appeler lorsque la boîte de dialogue est fermée.
    - **onAddMeal(date: Date, mealCategory: MealCategory)** : fonction à appeler lorsque l'utilisateur clique sur le bouton Ajouter pour ajouter le plat à la date et au repas sélectionnés.
- DateWheel.tsx : Affiche une roue de sélection de la date.
  - Props
    - **selectedDate** : date sélectionnée.
    - **onDateChange(date: Date)** : fonction à appeler lorsque la date est modifiée.
- EnumButtonGroup.tsx : Affiche un groupe de boutons pour sélectionner une valeur d'énumération.
  - Props
    - **values** : tableau des valeurs d'énumération.
    - **selected** : valeur d'énumération sélectionnée.
    - **setSelected(value: T)** : fonction à appeler lorsque la valeur d'énumération est modifiée.
    - **icons** : dictionnaire des icônes à afficher pour chaque valeur d'énumération.
    - **title** : titre du groupe de boutons.
    - **style** : style du groupe de boutons.
- FoodCard.tsx : Affiche une carte de plat avec une image, un titre et un bouton pour ajouter le plat à la planification des repas.
  - Props
    - **food** : plat à afficher.
    - **addAction(food: Food)** : fonction à appeler lorsque l'utilisateur clique sur le bouton Ajouter pour ajouter le plat à la planification des repas.
    - **deleteAction(food: Food)** : fonction à appeler lorsque l'utilisateur clique sur le bouton Supprimer pour supprimer le plat de la planification des repas.
    - **openAction(food: Food)** : fonction à appeler lorsque l'utilisateur clique sur la carte pour afficher les détails du plat.
- StackNavigatorHeader.tsx : Affiche un en-tête de navigation avec un bouton de retour et un titre en utilisant React Navigation et React Native Paper.

Context

Le dossier **context** contient le fichier **MealPlannerContext.tsx**, où nous définissons le contexte de l'application. Il contient l'objectif journalier de calories, la planification des repas et les fonctions pour les modifier.

- MealPlannerContext.tsx
  - **dailyCaloriesGoal** : objectif journalier de calories.
  - **mealPlan** : planification des repas.
  - **setDailyCaloriesGoal(goal: number)** : fonction pour définir l'objectif journalier de calories.
  - **addMealPlan(date: Date, mealCategory: MealCategory, food: Food)** : fonction pour ajouter un plat à la planification des repas.
  - **removeMealPlan(date: Date, mealCategory: MealCategory, index: int)** : fonction pour supprimer un plat de la planification des repas.

Services

Le dossier **service** contient le fichier FoodApiService.ts, où nous définissons l'instance d'Axios et les appels d'API vers Spoonacular, notre source de données pour les recettes alimentaires.

Types

Le dossier **type** contient plusieurs fichiers où nous définissons les types utilisés dans l'application. Les types liés à l'API Spoonacular sont regroupés dans FoodApi.ts.

Utils

Le dossier **utils** contient des fonctions utilitaires réutilisables. Elles sont regroupées dans des fichiers spécifiques à leur utilisation, par exemple healthGoal.ts et recipe.ts.

Fonctionnalités

- Calcul du BMR en temps réel à mesure que les utilisateurs modifient leurs données.
- Recherche de plats préférés dans la base de données alimentaire.
- Ajout de plats sélectionnés à la planification des repas en choisissant la date et le repas appropriés.
- Affichage d'un aperçu des calories totales pour la journée sélectionnée et comparaison avec l'objectif calorique quotidien.
- Navigation fluide entre les écrans et affichage des détails d'une recette à partir de n'importe où dans l'application.

Roadmap

Module 1 - Création du projet

- ☒Créer un nouveau projet React Native avec Expo.
- ☒Configurer eslint et prettier.
- ☒Installer les dépendances React Navigation et React Native Paper.
- ☒Créer des placeholders pour les écrans de l'application.
- ☒Configurer la navigation entre les écrans.

Module 2 - Calcul du BMR et définition des objectifs de santé

- ☒Afficher un formulaire pour que les utilisateurs saisissent leurs données personnelles.
- ☒Lire les données saisies par les utilisateurs et calculer leur BMR.
- ☒Afficher le BMR calculé
- ☒Sauvegarder les données personnelles et le BMR dans le stockage local.
- ☒(Bonus) Améliorer l'interface utilisateur du formulaire.

Module 3 - Base de données alimentaire

- ☒Créer un compte sur l'API Spoonacular.
- ☒Créer un service pour effectuer des appels d'API vers Spoonacular.
- ☒Afficher un champ de recherche pour que les utilisateurs saisissent le nom d'un plat.
- ☒Lire le nom du plat saisi par les utilisateurs et effectuer une recherche dans la base de données Spoonacular.
- ☒Afficher les résultats de la recherche.
- ☒Afficher le nom, l'image, les calories et le temps de préparation de chaque plat.

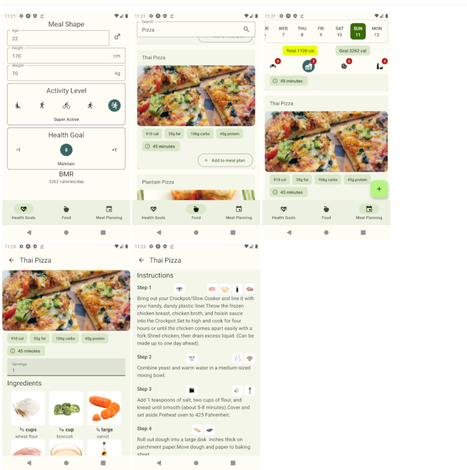
Module 4 - Planification des repas

- ☒Créer un contexte pour stocker la planification des repas.
- ☒Afficher un calendrier pour que les utilisateurs sélectionnent une date.
- ☒Afficher un groupe de boutons pour que les utilisateurs sélectionnent un repas.
- ☒Afficher les résultats de la recherche de plats.
- ☒Afficher un bouton pour ajouter un plat à la planification des repas.
- ☒Lire la date, le repas et le plat sélectionnés par les utilisateurs et ajouter le plat à la planification des repas.
- ☒Afficher un aperçu des calories totales pour la journée sélectionnée.
- ☒Lire la date sélectionnée par les utilisateurs et afficher l'aperçu des calories totales pour cette journée.
- ☒Afficher un bouton pour supprimer un plat de la planification des repas.
- ☒Lire la date, le repas et l'index du plat sélectionnés par les utilisateurs et supprimer le plat de la planification des repas.

Bonus - Affichage des détails d'une recette

- ☒Afficher un bouton pour afficher les détails d'une recette.
- ☒Lire le plat sélectionné par les utilisateurs et afficher les détails de la recette.
- ☒Afficher les ingrédients de la recette.
- ☒Afficher les ustensiles de cuisine nécessaires pour préparer la recette.
- ☒Afficher les instructions de la recette.
- ☒Choisir le nombre de portions et afficher les informations nutritionnelles de la recette en fonction du nombre de portions.

Screenshots



Difficultés rencontrées

Pendant le développement de l'application, nous avons rencontré quelques difficultés, notamment :

- L'intégration avec l'API Spoonacular et la gestion des requêtes asynchrones.
- La gestion des états et des mises à jour en temps réel des données lors de la planification des repas.
- L'organisation efficace du code pour maintenir la lisibilité et la maintenabilité à mesure que l'application se développait.