Thi Quynh-Nhu PHAM
Florian NOR

React Native course

Document d'avancée du projet :

NutriChef App

Université Gustave Eiffel
ESIEE PARIS

Table des matières

Le lien du repository du git : https://github.com/FlrianNK/health_care_app

Objectifs

L'objectif de notre projet est de développer une application en React Native. Cette dernière aura pour but d'aider les utilisateurs à mieux gérer leur alimentation et ainsi leur santé en définissant des objectifs de santé personnels, en suivant leur apport calorique quotidien et en planifiant leurs repas en fonction de leurs objectifs de santé personnels.

Dans le projet et lors des commits, Laetitia PHAM et TQNLP sont la même personne. Il s'agit de Thi Quynh-Nhu PHAM.

Architecture de l'application

Notre application se divise en trois parties qui sont représentées par 3 onglets différents :

- Le premier onglet : **Health goals :** Les utilisateurs peuvent saisir leurs informations personnelles et leurs objectifs de santé. Ils peuvent aussi y renseigner leur âge, leur sexe, leur taille, leur poids, leur niveau d'activité et leur objectif global de santé afin d'obtenir un résultat qui représente la quantité de calories journalière quotidienne nécessaire pour atteindre leurs objectifs.
- Le second onglet : **Food Database :** Les utilisateurs peuvent effectuer des recherches sur des plats ou des aliments. Ils obtiennent alors plusieurs résultats correspondant à ce qu'ils ont recherché. Par la suite, ils ont la possibilité d'ajouter les plats qu'ils souhaitent à leur planning repas qu'ils pourront visualiser dans l'onglet suivant.
- Le troisième onglet : **Meal Planning :** Les utilisateurs peuvent planifier et visualiser les repas qu'ils ont planifié. Ils peuvent supprimer des plats et les modifier et ainsi avoir une vision globale du nombre de calories total pour chacun des jours qu'ils ont planifié.

Dans le fichier App.js cela s'organise de la façon suivante :

- les imports
- la fonction du premier onglet : HealthGoalsScreen
- la fonction du second onglet : FoodDatabaseScreen
- la fonction du troisième onglet : MealPlanningScreen
- les styles des onglets triés
- le composant principal du projet

Fonctionnalités développées fonctionnelles

Dans ce projet, presque la totalité des fonctionnalités ont été développées et sont fonctionnelles. Nous les avons testés pour simuler une utilisation classique de l'application. Cependant, sur le plan des performances globales, aucun test n'a été réalisé. De même, nous n'avons pas testé le cas où l'utilisateur entrerait une grande quantité de données dans le tableau des repas. Sur la durée, il serait plus judicieux de procéder autrement.

Pour citer quelques exemples de fonctionnalités fonctionnelles :

- Le formulaire donne les bonnes informations et inclut des vérifications en cas de soumission sans avoir complété correctement le formulaire. Un pavé numérique avec filtrage est utilisé pour éviter que l'utilisateur ne renseigne des informations inadéquates pour le calcul.
- Une bonne gestion des modals, datepicker, des hooks et du context avec la page FoodDatabase et l'ajout d'aliments dans le plan des repas .
- La possibilité de supprimer un plat depuis le plan des repas
- La possibilité de modifier la quantité d'un plat depuis le plan des repas
- La possibilité d'ajouter un aliment existant au même repas, à la même date ne rajoute pas de ligne supplémentaire. La quantité est seulement ajoutée à la quantité de l'aliment déjà existant dans le tableau.
- le asyncStorage pour sauvegarder le plan des repas même si on ferme entièrement l'application
- La possibilité de renvoyer l'utilisateur vers la page foodDatabase
- etc

Fonctionnalités développées non fonctionnelles

Lorsque l'on effectue une requête API, on a remarqué que certains aliments retournés sont doublés voire plus. On a essayé de filtrer l'affichage en créant une map dans laquelle on va venir ajouter un élément si et seulement si ce dernier n'est pas encore dans la liste. On crée pour cela une clé unique qui est la concaténation de toutes les informations sur l'aliment qui va être ajouté et si cette clé n'existe pas dans la map alors on ajoute l'aliment sinon on ne le met pas et il ne s'affiche pas. Cette méthode

a enlevé plusieurs répétitions. Cependant, il reste des doublons qu'on arrive pas à enlever et on ne comprend pas la cause de cet effet.

La roadMap

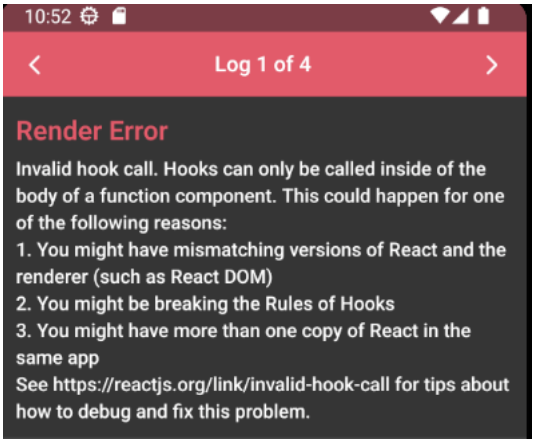
Dans cette roadMap, nous définissons les deadlines maximales à ne pas dépasser pour chacune des fonctionnalités de l'application. Cela nous permet d'éviter tout retard de développement et ainsi de rendu.

	Avant : 06/06	**Avant : 15/06**	**Avant : 19/06**	**Avant : 21/06**
Bottom navbar	**X**			
Formulaire	**X**			
Food Database		**X**		
Meal Planning			**X**	
CSS Final				**X**
Vidéo Démo				**X**

Retours sur le planning à la fin du développement :
Dans l'ensemble, nous avons respecté notre planning global même si bien sûr, le développement n'est pas linéaire. Nous avons dû faire des aller retour dans le développement des onglets pour corriger des bugs ou ajouter de petites fonctionnalités supplémentaires. La communication entre deux onglets rend le respect des deadlines plus délicat car un problème sur l'onglet suivant peut venir de l'onglet précédent.

Quelques problèmes rencontrés

1. Dès le développement du second module, nous avons rencontré un problème de hook call :



Ce problème venait du fait qu'on utilisait une librairie qui utilisait React 17.0.2 tandis que le reste de notre application utilise React 18.2.0. Le problème est que la librairie utilisée n'avait pas de version plus récente et ne fonctionnait donc pas avec notre version. Nous avons décidé d'utiliser la librairie `react-native-picker/picker` qui est compatible avec React 18.2.0.

2. Un problème lors de l'implémentation de la vérification des champs venait du fait que l'affichage du résultat était toujours présent avant même que l'utilisateur ne valide son formulaire. Il fallait ainsi trouver un moyen de faire en sorte que ce texte n'apparaisse que lorsque l'utilisateur valide son formulaire et s'il le modifie le texte disparaît jusqu'au moment où le formulaire est de nouveau soumit.

3. Nous avons eu un problème avec la seconde API lorsque nous étions en train de développer le second onglet. La deuxième API proposée dans le sujet du projet nous renvoyait une erreur du serveur : erreur 503. Nous avons donc basculé sur l'API 1 Edamam.

4. Comme évoqué dans la partie : *fonctionnalités développées non fonctionnelles*, la requête nous renvoyait des doublons et même plus. Nous avons essayé de les filtrer pour n'afficher qu'une seule instance de chaque résultat mais rien n'y fait, il y a toujours des doublons pour certains aliments.

5. Dans les inputText du formulaire de renseignement, l'utilisateur avait la possibilité d'entrer des valeurs à virgule pour l'âge et aussi pour la taille en cm. Ce comportement n'est pas attendu c'est pour cette raison que nous avons ajouté des regex pour filtrer l'input.

6. Nous avions essayé de personnaliser nos boutons pour changer leur forme mais aussi leur couleurs. Cependant, nous ne comprenions pas pourquoi le style appliqué ne fonctionnait pas. Finalement, il s'avère que le composant bouton ne peut pas contenir de style. Nous sommes passés par des composants "TouchableOpacity" qui offrent la possibilité d'appliquer un style sur le texte du bouton et le bouton lui-même.

7. Un problème majeur s'est posé lorsque l'on souhaitait afficher les données depuis le FoodDatabase vers le MealPlanning. La structure de données proposée dans l'énoncé du projet ne permettait pas de transférer l'ensemble des données voulues. Sachant que nous utilisions un datepicker comme évoqué en TD, nous devions pouvoir gérer non seulement la semaine mais aussi les semaines suivantes. Nous avons modifié l'entièreté de la structure de données et par conséquent, nous avons modifié les codes des deux onglets concernés. La structure adopté est la suivante :

   '3023-06-XX' : {
   'Breakfast' :[foodItem and information],
   'Lunch' :[foodItem and information],
   'Snack' :[foodItem and information],
   'Dinner' :[foodItem and information],
   }

   Cette structure permet de passer les informations nécessaires peu importe le jour en prenant la date du jour en tant que structure principale et non le nom d'un jour.

8. Une fonctionnalité stylistique supplémentaire que nous voulions implémenter est le changement de couleur de l'icone de la page active de la tabBar en fonction de l'onglet actif. Pour cela nous sommes passés dans un premier temps par un tabBarOptions mais cette fonction est dépréciée. Nous sommes donc passés par ScreenOptions pour appliquer ce comportement.

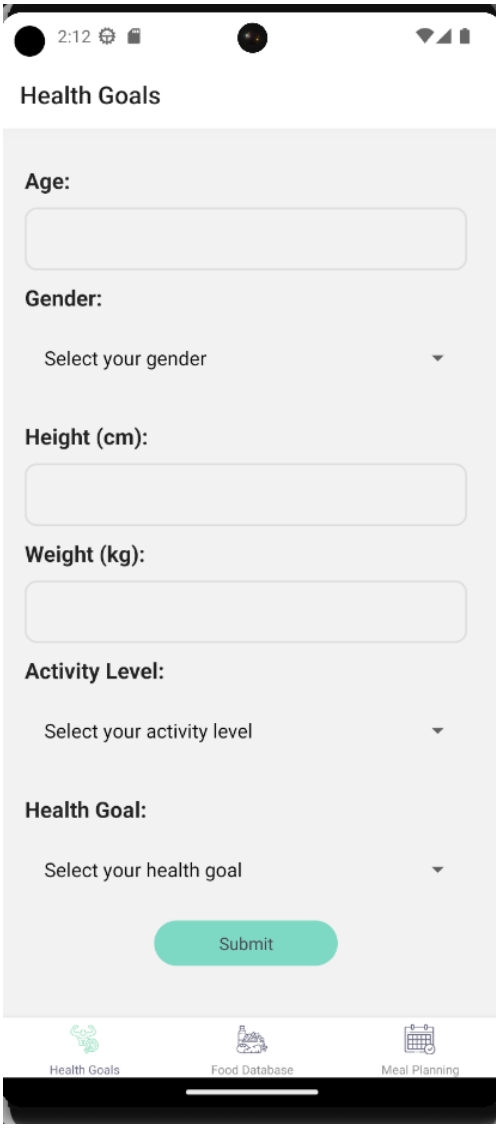
Pistes d’améliorations

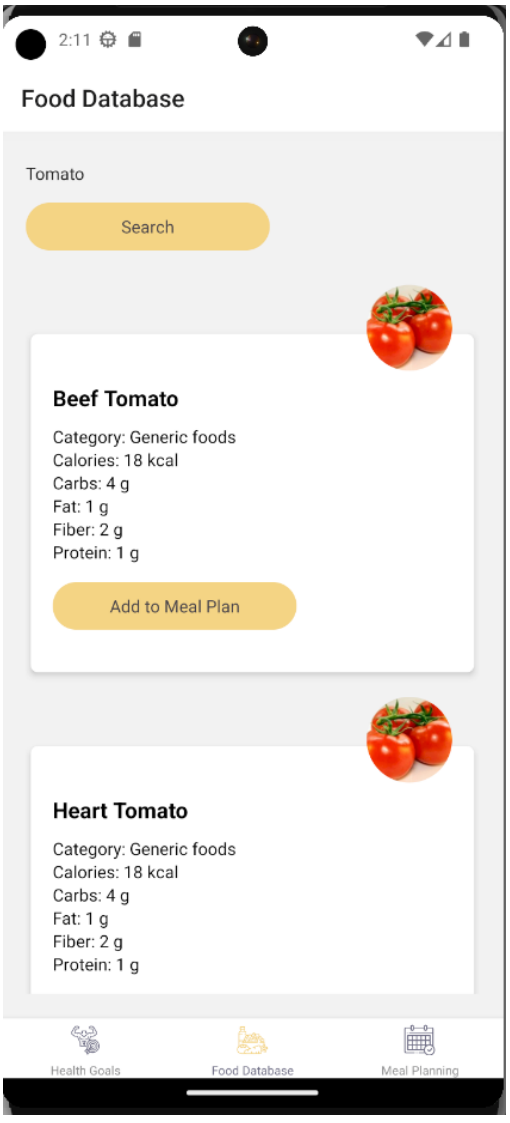
Cette application n'implémente que peu de fonctionnalités pour satisfaire pleinement une application de planification de repas. Par exemple, on pourrait par la suite développer les fonctionnalités suivante :

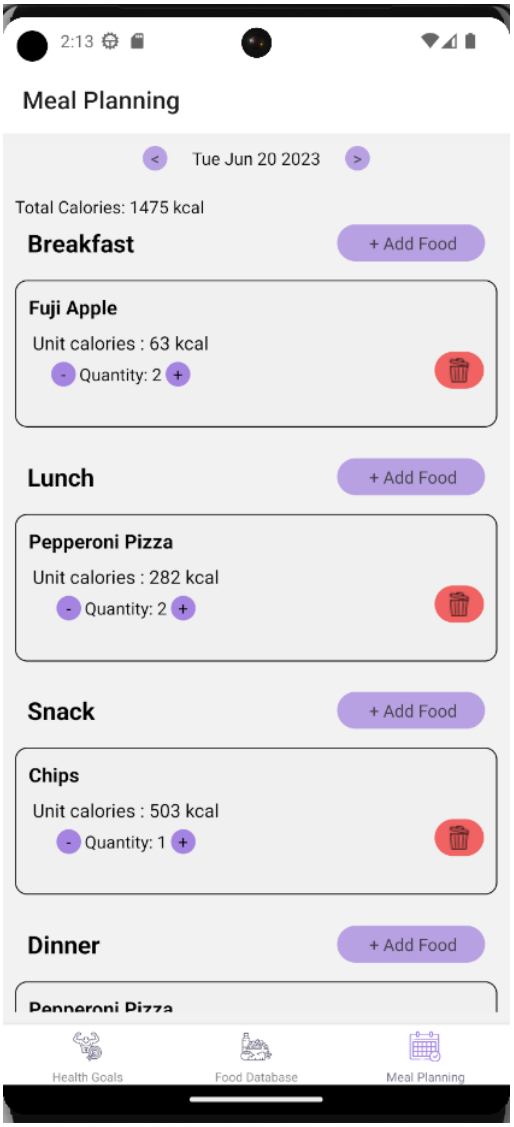
- la possibilité de déplacer un aliment vers un autre repas de la journée en drag and drop
- l’ajout d’une page de statistique pour voir les calories, les glucides, les matières grasses consommées et toutes les informations intéressantes
- la possibilité d’implémenter un IA capable de proposer des aliments en fonction des allergies et des habitudes alimentaires de l’utilisateur.

Quelques Screenshots de l’application

Ci-dessus les 3 onglets de l’application :






2:13
Meal Planning
< Tue Jun 20 2023 >
Total Calories: 1475 kcal
Breakfast
+ Add Food
Fuji Apple
Unit calories : 63 kcal
- Quantity: 2 +
Lunch
+ Add Food
Pepperoni Pizza
Unit calories : 282 kcal
- Quantity: 2 +
Snack
+ Add Food
Chips
Unit calories : 503 kcal
- Quantity: 1 +
Dinner
+ Add Food
Health Goals
Food Database
Meal Planning