Projet
Macro-tracker

Mathieu Noizet

Mars 2025

1 Introduction

Les sportifs, qu'ils soient amateurs ou professionnels, ont des objectifs spécifiques qui dépendent de leur activité : prise de masse, perte de poids, amélioration des performances, ou encore optimisation de la récupération. Pour atteindre ces objectifs, ils organisent leurs alimentations afin de répartir précisément les macronutriments (par exemple : 40% glucides, 30% protéines, 30% lipides) pour ajuster leur alimentation en fonction de leur dépense énergétique quotidienne et de leurs objectifs.

Un outil de suivi simple et efficace leur permet de rester autonomes, de mieux comprendre leur alimentation, et d'adapter leurs apports de manière réactive. Ce type d'application peut également servir de support au suivi personnalisé réalisé par un coach ou un diététicien.

Aujourd'hui, plusieurs applications commerciales répondent partiellement à ce besoin (MyFitnessPal, Yazio, Lifesum, etc.), mais elles sont souvent payantes, envahies par des options inutiles ou des interfaces complexes, et parfois verrouillées sur certaines fonctionnalités (export des données, ajout d'aliments personnalisés, etc.). Le besoin exprimé ici est celui d'un outil léger, gratuit et maîtrisable de bout en bout, qui se concentre uniquement sur le suivi des apports nutritionnels sans friction ni dépendance à des services tiers.

Cette solution s'adresse donc à un public soucieux de sa performance physique ou de son équilibre alimentaire, qui a besoin d'un outil fiable, rapide et sans fonctionnalités superflues. Il est envisageable que l'application, à terme, puisse ensuite être intégrée à une interface mobile ou web.

Dans ce cadre, vous allez développer une application dédiée au suivi nutritionnel. Elle devra permettre de consulter, manipuler et agréger des données alimentaires issues d'une base structurée et riche en informations nutritionnelles.

L'objectif est de fournir un outil simple, rapide et précis, permettant de *tracker* les apports nutritionnels quotidiens d'un utilisateur. L'application doit permettre :

- La recherche d'aliments dans une base de données (valeurs par 100g : protéines, glucides, lipides, fibres, vitamines, minéraux, etc.) ;
- L'ajout d'aliments consommés au fil de la journée, avec les quantités correspondantes ;
- La génération d'un bilan quotidien des macronutriments consommés, avec des totaux clairs et exploitables.

L'application sera découpée en plusieurs parties. Dans la suite du document, vous trouverez un *Overview* de ce qui sera demandé avec la structure des différents éléments, puis vous aurez des parties spécifiques pour chaque partie.

2 Coeur de l'application

Le principal objectif de l'application est d'aider les utilisateurs à suivre leurs apports nutritionnels de manière simple et efficace, tout en leur fournissant un outil personnalisé qui s'adapte à leurs besoins spécifiques. L'application ne doit pas seulement être un outil de suivi, mais aussi un assistant qui aide l'utilisateur à mieux comprendre ses choix alimentaires et à ajuster son alimentation pour atteindre ses objectifs physiques et de bien-être.

Ainsi, au cœur de l'application, nous retrouvons une interface qui permet de gérer les repas, d'agréger les données nutritionnelles, de suivre les progrès physiques et d'adapter les repas en fonction des besoins quotidiens, tout en offrant une solution souple et intuitive à l'utilisateur.

L'application que vous allez développer doit permettre aux utilisateurs de suivre facilement et efficacement leur consommation nutritionnelle tout au long de la journée, tout en leur fournissant un support personnalisé pour atteindre leurs objectifs de performance ou de bien-être physique. Le cœur de l'application repose sur plusieurs fonctionnalités essentielles, permettant une interaction fluide entre l'utilisateur et les données nutritionnelles, ainsi qu'une gestion complète de ses informations personnelles et de ses repas.

- **Informations personnelles et suivi de la condition physique :** Chaque utilisateur pourra créer un profil personnel comprenant des informations telles que : nom, prénom, âge, poids, taille, et d'autres paramètres pertinents. Ces informations seront utilisées pour calculer son indice de masse corporelle (IMC) et, si nécessaire, son taux de masse grasse (body fat). Ces calculs permettront de personnaliser le suivi nutritionnel en fonction de son objectif de performance (perte de poids, prise de masse, maintien, etc.) et de son état physique.
- **Suivi de la composition corporelle :** Un aspect clé de l'application sera de permettre à l'utilisateur de suivre son évolution physique au fil du temps. En plus de l'IMC, le calcul du taux de graisse corporelle (body fat) sera réalisé à partir des informations personnelles (poids, taille, âge, etc.), avec une option pour saisir ces données de manière périodique (par exemple, chaque semaine ou chaque mois). Cela permettra au sportif d'adapter correctement sa nutritions et pour mieux atteindre ses objectifs physiques.
- **Gestion des repas et des journées types :** L'application permettra à l'utilisateur de créer et de sauvegarder une liste personnalisée de repas qu'il consomme régulièrement, comprenant des informations nutritionnelles pour chaque aliment ajouté. L'utilisateur pourra organiser ces repas dans des journées types (par exemple, "petit-déjeuner", "déjeuner", "dîner", "collation"). L'utilisateur pourra facilement ajouter des repas et modifier les repas existants, avec la possibilité de créer des combinaisons de repas pour mieux suivre ses habitudes alimentaires. Il est important que l'utilisateur puisse adapter ces journées types à son mode de vie, car le nombre de repas dans une journée peut varier d'un individu à l'autre, en fonction de ses besoins et de ses habitudes.
- **Ajout d'aliments et suivi des macronutriments :** Lorsqu'un utilisateur consomme un aliment, il pourra l'ajouter à l'un de ses repas. L'application récupérera les informations nutritionnelles de cet aliment (glucides, protéines, lipides, fibres, vitamines, minéraux, etc.) et l'ajoutera automatiquement au récapitulatif des macronutriments de la journée. Les macronutriments seront calculés en fonction des quantités indiquées par l'utilisateur pour chaque aliment, et un bilan quotidien sera généré pour afficher les totaux de chaque macronutriment, permettant à l'utilisateur de comparer ses apports avec ses besoins nutritionnels.
- **Suivi personnalisé des repas et de l'alimentation quotidienne :** L'application devra être capable de générer un récapitulatif des repas et des apports nutritionnels de chaque journée, en prenant en compte l'ensemble des aliments consommés. Ce suivi pourra être visualisé sous forme de graphique ou de tableau, montrant l'évolution des apports quotidiens (par exemple, glucides, protéines, lipides, etc.). Les repas pourront être modifiés ou supprimés si nécessaire, et de nouveaux aliments pourront être ajoutés selon les préférences de l'utilisateur.

- **Modification des informations personnelles :** L'utilisateur aura la possibilité de modifier ses informations personnelles à tout moment. Cela inclut des modifications sur son poids, sa taille, son âge, son objectif, etc. Toute modification affectera directement les calculs liés à la composition corporelle (IMC, body fat) et permettra à l'application de s'ajuster à ses nouveaux besoins nutritionnels et objectifs.
- **Affichage des objectifs et du suivi des progrès :** L'application permettra de définir des objectifs personnalisés en matière de nutrition, tels que des répartitions spécifiques de macronutriments (par exemple, 40% glucides, 30% protéines, 30% lipides). Ces objectifs seront comparés aux apports réels de l'utilisateur au cours de la journée. En outre, l'application pourra générer un historique des données pour suivre l'évolution de la condition physique et des habitudes alimentaires de l'utilisateur au fil du temps.

Pour avoir un aperçu visuel de l'utilisation d'une application similaire, vous pouvez consulter *MyFitnessPal* (application mobile), qui vous donnera une idée de la gestion des objectifs, des repas, de l'ajout d'aliments, etc.

3 Interaction utilisateur

Pour l'interaction avec l'utilisateur, et afin de valider le bon fonctionnement de votre application, vous développerez une interface en ligne de commande (CLI) simple et fonctionnelle. Cette interface permettra à l'utilisateur de rechercher un aliment, d'ajouter une consommation, ou de consulter son bilan nutritionnel journalier, directement depuis le terminal.

L'interaction devra être réactive, et permettre à l'utilisateur de saisir des commandes pendant que le programme est en cours d'exécution. C'est-à-dire que l'application ne devra être en aucun cas bloquée en attendant la saisie utilisateur. Pour cela, vous utiliserez une *go routine* dédiée à la gestion des entrées utilisateur. Cette routine devra :

- Lire en continu les commandes saisies depuis le terminal (par exemple via *bufio.NewReader(os.Stdin)*),
- Analyser les commandes (sous forme de mots-clés simples : *search, add, report, exit*, etc.),
- Gérer la fermeture propre de l'application (arrêt des autres go routines, sauvegarde éventuelle des données en mémoire, etc.).

Afin d'assurer une communication fluide et sécurisée entre la *go routine* dédiée à l'interface utilisateur et le reste de l'application, vous devrez mettre en place un *channel*. Celui-ci servira à transmettre les informations saisies par l'utilisateur (sous forme de messages), comme la commande avec ses arguments, au cœur de votre application où elles seront traitées (recherche, ajout d'aliment, génération de bilan, etc.).

L'utilisation d'un *channel* permet de bénéficier de la concurrence sécurisée de Go, en évitant les conflits d'accès ou les partages de mémoire explicites.

Pour faciliter et homogénéiser l'ensemble des échanges de votre application, il est fortement conseillé de commencer à mettre en place des structures pour la gestion de vos messages et des erreurs éventuelles.

4 Base de données pour les aliments

Pour récupérer le détail des aliments (macronutriments et micronutriments), nous utiliserons la base de données *FoodData Central* (FDC : `https://fdc.nal.usda.gov/`), mise à disposition par le *National Agricultural Library* (NAL : `https://www.nal.usda.gov/`) dépendant du *United States Department of Agriculture* (USDA : `https://www.usda.gov/`). Un guide complet d'utilisation de l'API est également

accessible à l'adresse suivante : `https://fdc.nal.usda.gov/api-guide`.

La connexion avec la FDC permet de récupérer l'ensemble des données nutritionnelles disponibles : description des aliments, valeurs par portion, listes des macro- et micro-nutriments, etc. Afin d'intégrer cette interaction dans votre application, il vous est demandé de gérer la communication avec l'API au sein d'une *go routine*.

Pour cela, vous utiliserez le package standard `net/http`. L'API de la FDC est accessible par des requêtes HTTP, en `GET` ou `POST`, selon les besoins de votre implémentation.

Étant donné que ce type de requêtes n'a pas été abordé dans les TD précédents, un exemple de code, intitulé `exemple_fdc.go`, vous est fourni pour illustrer comment effectuer une première requête simple.

Pour pouvoir interagir avec l'API de la FDC, vous devrez obtenir une clé d'API personnelle. Celle-ci peut être générée gratuitement en créant un compte sur le site officiel via le lien suivant : `https://fdc.nal.usda.gov/api-key-signup`.

Attention : l'utilisation de cette API est soumise à une limitation de quota, fixée à 1000 requêtes par jour pour chaque clé.

Afin de vous guider dans les différentes requêtes à réaliser, voici une liste non exhaustive des requêtes possibles à implémenter :

- **Recherche d'un aliment par nom** :
  - Endpoint : `/fdc/v1/foods/search` (méthode POST)
  - Permet d'obtenir une liste d'aliments correspondant à un mot-clé donné (ex. "banana").
  - La réponse doit inclure les noms d'aliments, leur `fdcId`, et éventuellement un résumé des valeurs nutritionnelles.
- **Récupération des détails nutritionnels d'un aliment :**
  - Endpoint : `/fdc/v1/food/{fdcId}` (méthode GET)
  - Permet de récupérer tous les nutriments disponibles (macro et micro), pour un aliment donné à partir de son `fdcId`.
  - Vous devrez extraire au minimum : protéines, glucides, lipides, calories, fibres.
- **(Optionnel) Filtrage par type de base (Foundation, Branded, etc.) :**
  - Le champ `dataType` dans la recherche peut être utilisé pour restreindre les résultats (ex. éviter les produits industriels).
- **(Optionnel) Recherche par marque ou catégorie :**
  - La FDC permet aussi de filtrer les aliments par marque (pour les produits transformés) ou par groupe (ex. "Vegetables and Vegetable Products").
- **(Bonus) Gestion de l'unité de mesure :**
  - Par défaut, les valeurs nutritionnelles sont données pour 100g. Vous pouvez enrichir votre traitement pour adapter les valeurs aux quantités saisies par l'utilisateur (ex. : 150g de riz).
  - *Cette option peut facilement être géré dans votre code ou directement par la requêtes. Vous avez le choix.*

5 Gestion des données utilisateur et stockage local

Vous avez désormais mis en place les deux piliers fonctionnels de votre application : l'interaction avec l'utilisateur via le terminal, et la récupération des données nutritionnelles via la base FoodData Central (FDC). Toutefois, cette base ne permet pas de stocker les informations propres à vos utilisateurs. Chaque requête doit être rejouée à chaque utilisation, ce qui nuit à l'expérience utilisateur et à l'efficacité globale de l'application.

Pour remédier à cela, vous devrez mettre en place une base de données locale, qui permettra de persister les informations importantes entre chaque exécution. Cette base servira notamment à :

- Stocker les informations personnelles des utilisateurs : nom, prénom, âge, poids, taille, etc.
- Sauvegarder la composition de repas personnalisés : cela évitera à l'utilisateur de reconstituer manuellement chaque repas à chaque fois.
- Organiser les repas selon des journées types : petit-déjeuner, collation, déjeuner, dîner, etc. Attention : la structure des repas peut varier d'un utilisateur à l'autre (certains peuvent avoir plusieurs collations, d'autres non).

Ce stockage local vous permettra également de combiner plus facilement les repas d'un utilisateur pour générer un bilan nutritionnel quotidien ou hebdomadaire.

Afin de simplifier la mise en place de cette base, il est fortement conseillé d'utiliser **Docker** pour exécuter un conteneur de base de données relationnelle (comme `PostgreSQL`, `MySQL` ou `SQLite` selon vos préférences). Cela vous permettra d'interagir facilement avec la base depuis votre application Go tout en conservant un environnement maîtrisé et reproductible.

(Optionnel) Vous pouvez envisager de passer le déploiement de votre application intégralement sous ***Docker****.*

Bien entendu, il est à noter que la sauvegarde locale des informations ne doit pas bloqué le reste de l'application. Vous utiliserez donc également de *go routine* et des *channels*.

6 Pour aller plus loin (facultatif)

Cette section s'adresse aux groupes désireux de pousser leur projet plus loin en ajoutant une interface utilisateur graphique sous forme d'application web. Il s'agit d'une fonctionnalité complémentaire, optionnelle mais très intéressante qui ne doit en aucun cas perturber ou remplacer le cœur de l'application, basé sur l'interaction via le terminal. L'interface web viendra donc s'ajouter en surcouche d'affichage, en s'appuyant uniquement sur les fonctionnalités exposées par votre *backend*.

L'objectif est de proposer une expérience utilisateur plus agréable et intuitive : l'utilisateur pourra consulter ses repas journaliers, rechercher des aliments, suivre sa progression ou encore visualiser l'évolution de ses apports nutritionnels dans le temps. Il sera aussi possible de modifier ses informations personnelles (taille, poids, âge, etc.) ou encore de visualiser des statistiques comme la répartition des macronutriments ou la courbe de poids. Cette partie est donc une opportunité d'intégrer des notions de visualisation de données et d'interaction web moderne.

Pour le développement de cette interface, vous pouvez partir sur une approche simple en `HTML,CSS, JavaScript` pur, ou utiliser des frameworks modernes comme `React, Vue.js`, etc. Ces frameworks permettent une gestion efficace de l'état et une mise à jour dynamique des composants en fonction des données reçues depuis le *backend*. Pour les échanges entre le *frontend* et le serveur Go, le format *JSON* peut être utilisé, tout comme l'*XML*, ou autres formats de votre choix.

Côté Go, vous pourrez exposer vos endpoints REST en utilisant le package standard net/http, ou un routeur plus ergonomique comme *gorilla/mux*. Ces outils vous aideront à structurer proprement

vos routes (`GET, POST, PUT, DELETE`), à gérer les paramètres et les données JSON reçues ou renvoyées. Si vous le souhaitez, vous pouvez également utiliser des *middlewares* pour gérer les erreurs, l'authentification ou le *logging* des requêtes.

Pour afficher graphiquement les données, comme la répartition des macronutriments ou l'évolution du poids, vous pouvez intégrer des bibliothèques JavaScript comme Chart.js, Recharts, ou Plotly.js, qui facilitent la création de graphiques dynamiques (camemberts, barres, lignes, etc.).

Enfin, vous êtes libres de l'architecture *frontend* que vous souhaitez mettre en place, tant qu'elle respecte l'autonomie du *backend*. Aucun traitement métier ne doit être réalisé côté client : l'interface se contente d'afficher, de récupérer ou de transmettre des données à l'application Go, qui en reste le cœur logique.

Bon courage !