

MSPR EPSI - HealthAI

Rapport technique et guide de déploiement

Abdellah Ait Abdelkarim
Awenn lelu
Noah Lejolivet
Thibau Lasblei

<https://gitlab.com/la-team-du-sang-de-la-veine/healthai-coach>

Table de matières

■	Mise en contexte
■	Choix technologiques
■	Backend
■	Framework
■	Bases de donnée
■	Visualisation
■	Environnement de développement
■	Environnement de déploiement
■	résultat obtenus
■	difficultés
■	évolutions

Mise en contexte

L'entreprise healthAI coach cherche à mettre en place une application de conseil d'hygiène de vie orienté autour d'une approche holistique (considérer la santé comme un ensemble de système interagissant entre eux plutôt qu'un seul point précis) de la santé.

Dans ce cadre, le but de ce projet est de concevoir une solution logicielle satisfaisant les points suivants:

■ Business model

La monétisation de l'application doit être assurée par 2 paliers d'abonnements.

Les utilisateurs non abonnés ont accès aux fonctionnalités suivantes:

- journal alimentaire, suivis d'activité, IMC, tableaux de progressions simples Les 2 paliers suivants donnent accès aux fonctions suivantes:
- Premium (**9.99€ / mois**)
 - accès à des recommandations spécialisés fournis par IA, des plans nutritionnels et sportifs plus détaillés, et un suivi plus fin des objectifs.
- Premium +
 - Gestion des données liée au sommeil / poids / fréquence cardiaque, consultation en ligne avec nutritionnistes partenaires.

En plus de ces paliers d'abonnements, l'application doit pouvoir être vendue en **B2B** à des entreprises partenaires avec la possibilité pour ces dernières d'apposer leur branding.

Mise en contexte

■ Cibles

Les cibles de l'application sont:

- Millenials / GenZ
- urbains actifs ne disposant que d'un temps limité pour leur activité sportive
- débutant en nutrition / sport
- personne avec des objectifs spécifiques
 - perte de poids
 - renforcement musculaire
 - amélioration de sommeil
 - ...

■ Différentiation concurrentielle

Le marché étant fortement concurrentiel, l'application se différencie par les points suivants:

- approche holistique de la santé
- intégration de l'IA générative
- stratégie B2B permettant une source de revenu / visibilité supplémentaire

■ Scalabilité technologique

Afin de soutenir sa croissance future et son modèle B2B, l'application doit pouvoir être scalable au niveau de ses différents composants.

Ainsi, chaque module de l'application doit être construit sur de technologies modernes, documentées et permettant d'assurer la stabilité et la scalabilité de chaque module séparément, ainsi que leur déploiement et maintenance pour des clients B2B.

Mise en contexte

■ Objectif

Dans le cadre de ce bloc de développement, l'objectif est de fournir un prototype de la partie **Ingestion de données** de l'application finale.

Il s'agit de définir le socle technique et structurel sur lequel les développements ultérieurs vont se baser.

Le rendu final doit être capable d'ingérer des sets de données provenant de multiples sources de données et de les convertir dans un format standardisé afin d'être exploité par les parties ultérieures de l'application.

Les objectifs concrets sont les suivants:

- Mettre en place un système de collecte et de traitement automatisé des données de manière fiable et sécuriser.
- Développer un processus de transformation de ces données afin de garantir l'exploitation de ces dernières dans des outils ultérieurs
- Proposer une **API REST** permettant aux clients mobiles et web d'accéder à ses données.
- Livrer une interface de visualisation permettant de suivre les indicateurs clés de l'application (revenu, nb d'utilisateurs, abonnements, ect...)

Choix technologique

■ Backend

▒ Framework

Le choix à été fait d'utiliser **Django** (<https://www.djangoproject.com/>) et **Django Rest Framework** (<https://www.django-rest-framework.org/>) afin de gérer la partie **API** de l'application.

Ce choix est motivé par le fait que *Django* est un framework reconnu comme étant particulièrement maintenable sur le long terme, il est conçu avec le langage *Python*, qui, bien que simple à apprendre, ce qui en fait un plus pour l'intégration de nouveaux arrivant, est un langage extrêmement facile à déployé sur de multiples plateformes, et disposant d'un écosystème riche autant pour le web que pour le traitement de données et l'IA.

De plus, Django propose nativement une interface d'administration extensible permettant d'interagir avec les données stockée dans la base de données, ainsi qu'une gestion des rôles et des utilisateurs, permettant de répondre à ces besoins pour un coût de développement minime.

Django propose de plus un ORM pouvant s'intégrer dans des scripts externes permettant de standardiser la manière dont la base de données est accéder, fiabilisant les opérations de lectures et d'écritures.

▒ Bases de donnée

▒ Postgres (<https://www.postgresql.org/>)

Base de donnée principale servant au stockage "*froid*" des données, postgres dispose d'un large écosystème d'utilisateur et de librairies associés et est reconnu comme une solution performante et open source.

▒ Valkey (<https://valkey.io/>)

Base de données de cache pour les requêtes api et les données d'entrainement du futur modèle d'IA. **Valkey** est une base clé/valeur stockée en RAM qui implémente la spécification de *Redis*, ce qui la rend compatible avec le parc de librairie déjà en place pour cette dernière.

Choix technologique

■ Visualisation

■ Grafana (<https://grafana.com/oss/grafana/>)

Grafana est une solution open source simple d'utilisation et extrêmement versatile concernant les types et sources de données qu'elle peut agréer.

Elle permet de concevoir des dashboard contenant une variété non négligeable de graphiques et, de son côté open source, peut être étendue pour satisfaire les besoins spécifiques du projet.

■ Environnement de développement

■ nix (<https://nixos.org/>)

Nix est un gestionnaire de paquets permettant de créer des environnements de développements et des paquets de manière déclarative et reproductible.

Ce mécanisme permet d'assurer la reproductibilité de l'environnement de développement d'un pc à un autre, et de manière isolée, permettant de réduire les erreurs liées à l'environnement du développeur.

De plus, cela peut être utilisé comme outil de build avec les mêmes avantages, permettant de réduire les outils de builds différents et de simplifier drastiquement la maintenance de scripts de builds et de CI/CD.

■ git (<https://git-scm.com/>)

Le projet est versionné avec git afin d'assurer un historique des modifications du code et que chaque développeur puisse intervenir sur des branches spécifiques du code et les fusionner avec celles des autres. Git est la référence moderne de ce type d'outil et est un choix par défaut dans la plupart des projets.

■ gitlab (gitlab.com)

Gitlab est la forge logicielle concurrente à github, contrairement à ce dernier, il n'est pas détenu par Microsoft et est self hostable, ce qui fait que le projet pourra être facilement porté si l'entreprise décide de self hoster son infrastructure dans le futur. Gitlab intègre de même des outils de gestion de projet via les issues boards, ainsi qu'une CI/CD solide.

Choix Technologique

■ Environnement de déploiement

■ Virtualisation

■ Podman

Dans un soucis d'isolation des différents modules du projet, chaque composant sera fournis sous la forme d'une image **podman** qui est une implémentation open source du standard de **Docker**, ce dernier permet de plus de fonctionner *rootless* par défaut, ce qui permet d'éviter les escalations de privilèges si la daemon est compromis. De plus, les images seront buildées par **nix** afin d'assurer la reproductibilité des builds.

Le choix de conteneurs comme environnement de déploiement permet de pouvoir isoler les dépendances de chaque module, ainsi que de pouvoir les déployer, stopper et redémarrer séparément. De plus, dans une optique de scalabilité, il est possible de lancer plusieurs instances d'un module afin de distribuer la charge sur toutes les instances disponibles.

Difficultés

La plus grosse difficulté du projet fut de déterminer l'approche que l'on allait avoir concernant le traitement des données par l'IA, quel données était pertinentes, comment utilisé l'IA dans des fonctions où elle aurais plus de valeur ajouté qu'une approche déterministe.

Au final, l'approche retenue est d'utilisé l'IA comme un moteur de suggestions et de concevoir l'interface de l'application de manière "AI-first", c'est à dire de présenter l'application comme une interface de conversation avec l'agent, ce dernier pouvant répondre aux questions de l'utilisateur basée sur ses données biométriques et le dataset auquel il a accès. L'utilisateur converse ainsi avec l'agent afin de lui énoncé ses objectifs, lui demandé des exemples de repas et des exercices sur le moment de son activité, conseils qu'il peut retrouver dans son historique.

Évolutions

■ CI/CD

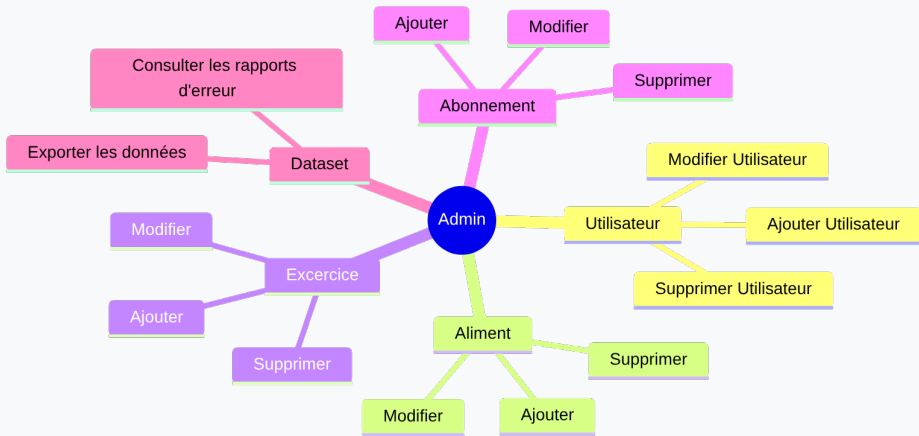
La mise en place d'un processus de développement continue permettant d'analyser de façon automatique la qualité du code serait un plus non négligeable afin d'assurer la stabilité long terme du produit.

De plus, un système de déploiement continu permettrait de réduire les marges d'erreurs et les coûts liés à des déploiements manuels, ainsi que de mettre en place différents environnements (production / développement) de manière automatisée.

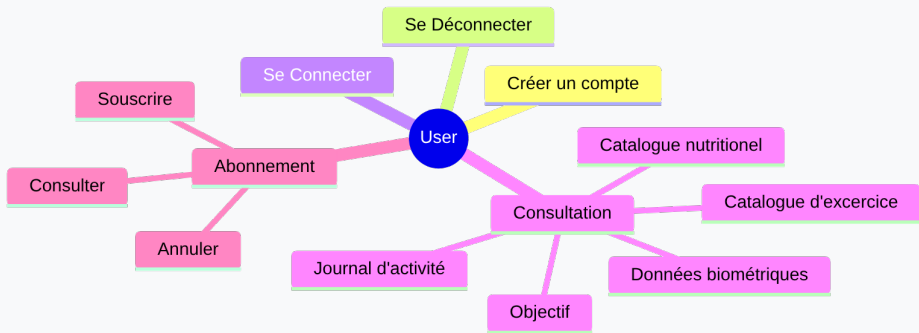
■ Tests

De la même façon que la mise en place d'une CI, la mise en place de tests unitaires pour les pipelines d'ingestion et l'API permettrait de s'assurer de l'intégrité du code au cours des développements successifs. Ces points pourraient être directement intégrés au sein du processus de développement via des méthodologies de *TDD*

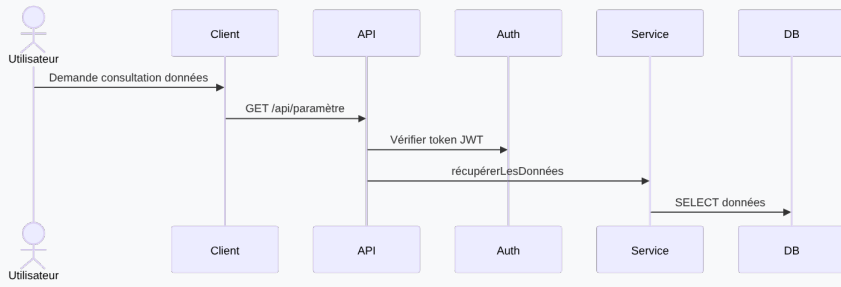
cas d'utilisations Admin



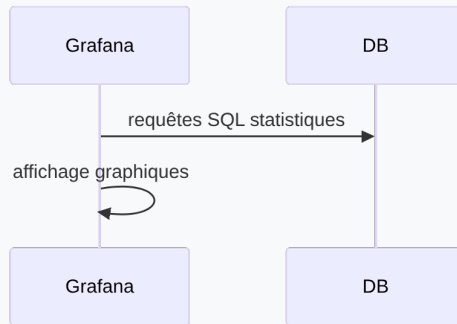
User



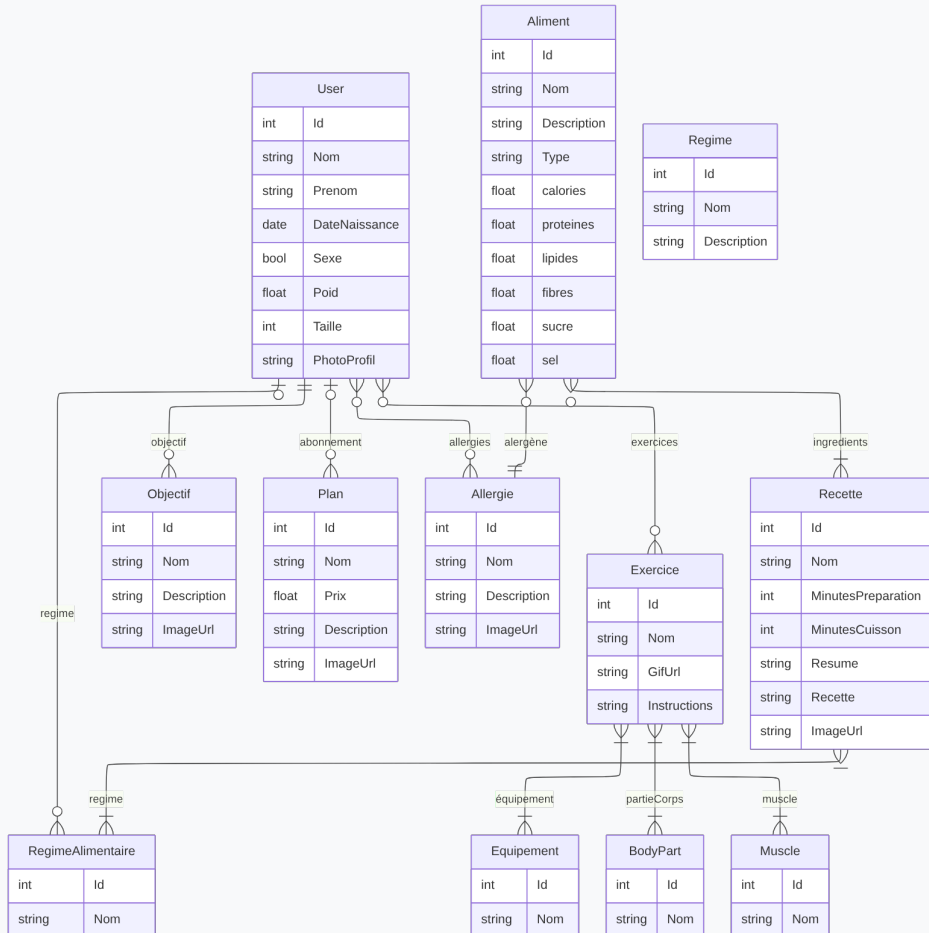
Séquence Consultation des données via API



Grafana



Entity Relations



Dossier de déploiement

■ HealthAI Coach

An ai coach that helps you for a better shape.

■ Développement

■ Backend

① Pré requis

Vous devez avoir nix et docker/podman d'installer
Téléchargement des dépendences et lancement d'un environnement de développement.

```
nix develop # télécharge les dépendences du projet
```

Lancement de la base de donnée et de grafana

```
docker compose up -d
```

Lancement du backend

```
cd backend
# créer les tables nécessaires dans la base de donnée
python manage.py migrate
python manage.py runserver # lance le serveur de développement
```

l'api sera disponible sur l'url <http://localhost:8080/api> l'interface d'administration sur l'url <http://localhost:8080/admin> la documentation swagger sur l'url <http://localhost:8080/api/schema/swagger-ui/>

■ Déploiement

```
# clone le repository
git clone
https://gitlab.com/la-team-du-sang-de-la-veine/healthai-coach.git
cd healthai-coach
# copiez la configuration d'exemple et remplissez les champs de mot
# de passe pour la base de donnée.
cp docker-compose.production.example.yml
docker-compose.production.yml
# la commande de deploy vas builder les composants nécessaires et
lancer les
# services via docker
docker compose up -d -f docker-compose.production.yml
```