Projet Innovation
Etat de l'art
Version 2

Equipe-Q : Eatrack
Soufiane Aourinmouche, Hugo Croenne, Alix Humbert, Mohamed Younes

Critères de comparaison et résultat de recherches :

Un tableau comparatif pour les fonctionnalités :

Le tableau a été mis à jour durant la semaine, en découvrant d'autres applications essentiellement sur App Store comme il a été mentionné dans le feedback.

https://docs.google.com/document/d/1ijGnPt6zbqTBbeiQHS0N5cXkeE8zZ4MleYjidfr9QZ8/edit?usp=sharing

Comme mentionné dans la première version, Eatrack s'appuie sur un vrai système de recommandation utilisant des méthodes scientifiques pour l'analyse des profils des historiques.
Il était mentionné que le traitement de plusieurs contexte possibles (Amis, Familles …) nous démarquent des autres concurrents. Ceci est toujours le cas après une semaine de développement et de recherche.

L'application ***BigOven*** que nous avons découverte récemment traite le cas d'un ensemble de personnes invités, mais pas de la même façon que ***Eatrack.*** En effet, ***BigOven*** est basée sur un principe de réseau social où l'utilisateur peut suivre d'autres personnes et vérifier les recettes qu'ils publient, il pourra alors se baser sur cela pour décider une recette adaptée à eux s'il les invite. Ce choix dépend alors de l'utilisateur et n'est pas forcément pertinent à l'ensemble du groupe. ***Eatrack*** fait dans ce cas une recommandation plus pertinente en traitant tout l'historique des utilisateurs concernés, sans avoir besoin à publier leur historique, ensuite l'utilisateur principal a le choix entre plusieurs recettes recommandées.

Pour la partie monétisation, il s'est avéré que les partenariats avec des marques (comme Charal, Panzani …) peut être difficile pour nouvelle application dans le marché. Notre alternative est alors de monétiser l'application grâce à des partenariats avec des super-marchés de proximités. Ceci est faisable puisqu'il permet aux super-marchés de liquider leur marchandise proche de péremption. Notre application peut aussi être un moyen de mettre en avant certaines marques ou produits que les super-marchés (ou la marque elle même) veulent afficher plus aux clients. Toutefois, nous restons sur notre avis de ne pas utiliser la publicité envahissante, ce qui pourra nous démarquer également. D'autres recherches sur la monétisation alternative est en cours.

Choix et réorientation

Architecture logicielle :
Le besoin d'une architecture basée sur des micro-services a été confirmé pendant notre phase de développement. En ajoutant la fonctionnalité de la gestion des groupes, nous avons eu besoin d'un service qui s'occupe des profils afin de les fusionner ou les traiter individuellement. Nous avons alors ajouté un micro-service à notre architecture, cela a été facile et bénéfique puisqu'il n'a pas affecté le reste.

Système de recommandation basé sur la librairie Surprise :
Surprise propose plusieurs algorithmes pour configurer notre système de recommandation. En plus de ça, il permet de les comparer avec un Benchmark. Pour avoir un système de recommandation pertinents, on devait décider quel algorithme est le plus adapté à notre besoin, et le plus performant.
La judaïcité de ce choix a été confirmé en faisant un test de qualité où nous avons dû essayé plusieurs algorithmes différents afin de comparer leurs performances. L'algorithme que nous utilisions à la base était "SVD", nous avons décidé finalement d'utiliser "BaseLine" qui est plus performant.

Reconnaissance des images :
Concernant cette partie, elle n'est pas encore traité dans notre application. Nous ne pouvons pas alors nous décider. Cependant, il était mentionné dans le feedback que 2 technologies n'étaient pas suffisantes. Pytorch nous semble un bon choix éventuelle puisqu'il permet d'implémenter facilement des algorithmes complexes puisqu'il est basé sur le flot de contrôle standard de Python, et il a une syntaxe assez simple. Il propose en plus des API pour faciliter la tâche.

Critères de comparaison de ces technologies

Architecture modulaire : Nous utilisons une architecture micro-services, qui nous permet d'intégrer/modifier des services sans impacter le reste. Elle nous permet également d'utiliser des services implémentés avec de différents langages (service en Python pour le système de recommandation, .Net pour les web-services ainsi que les clients et possibilité d'ajouter un client android/IOs).

Système de recommandation et Valorisation de données : Il s'agit du point le plus important dans notre projet. En effet, les concurrents utilisent un filtrage des recettes en fonctions des ingrédients du frigo, les préférences et les exigences. ***Eatrack*** s'appuie sur un vrai système de recommandation basé sur des modèles mathématiques implémentés en Python dans un service externe. Nous avons utilisé une librairie **"Surprise"** parce qu'elle propose toutes les fonctionnalités nécessaires au système de recommandation dont notre projet a besoin. Cette librairie est facilement configurable, ce qui permet de différencier les différents contextes d'utilisations de l'application. De plus celle-ci est implémentée en Python, langage étudié dans notre parcours scolaire, open-source et propose des tutoriels pour la prendre en main.
Il y a d'autres implémentations pour les systèmes de recommandation, comme la library fast.ai, Crab et recsys qu'on n'a pas utilisé parce que Surprise est plus populaire, meilleur en terme de qualité de code et optimisé pour l'analyse des données. Le choix de cette librairie nous paraît encore aujourd'hui comme un choix judicieux.

Reconnaissance des aliments : Nous comptons utiliser Keras plutôt que **Tensorflow**. En effet, pour construire un projet "from-scratch", **Keras** est recommandé puisqu'il est facile à prendre en main (*pour un projet de 3 semaines)*. Cependant **Tensorflow** étant plutôt orienté Deep Learning, il ne répond pas précisément à notre besoin.
Il y a d'autres implémentations pour machine learning comme: scikit-learn et scikit-learn.

Résultat de l’évaluation selon critères mentionnés (Positifs)

Il était mentionné en V1 que nous comptons évaluer notre système de recommandation avec la satisfaction de l’utilisateur. L’application n’étant pas terminée, nous avons implémenté des tests unitaires, fonctionnels et de qualité.

Ces tests simulent la satisfaction de l’utilisateur, en comparant ses préférences avec les recettes qui lui sont recommandées.

Un test de qualité compare la performance de notre système en utilisant différents algorithmes.

Il était mentionné également que nous comptons évaluer notre architecture en ajoutant d’autres fonctionnalités dans autres services. Notre avancement nous a poussé à ajouter un service responsable de la gestion profils utilisateur “User_profiles”. Cet ajout n’a pas impacté le reste, et notre application est restée stable en modifiant des bouts de l’application.

Résultat de l’évaluation selon critères mentionnés (Négatifs)

Notre architecture en micro-services est basée sur des web-services pour la communication entre des services implémentés dans des langages différents.

Nous nous sommes rendus compte finalement que nous aurions pu nous en passer des web-services qui sont alors une étapes de plus pour traduire la communication entre deux services.

Cette architecture nous a forcé également à dupliquer une partie du code, mais vu les bénéfices, nous restons satisfaits de ce choix modulaire.