

PROJET 9

RÉALISEZ UNE APPLICATION MOBILE DE RECOMMANDATION DE CONTENU

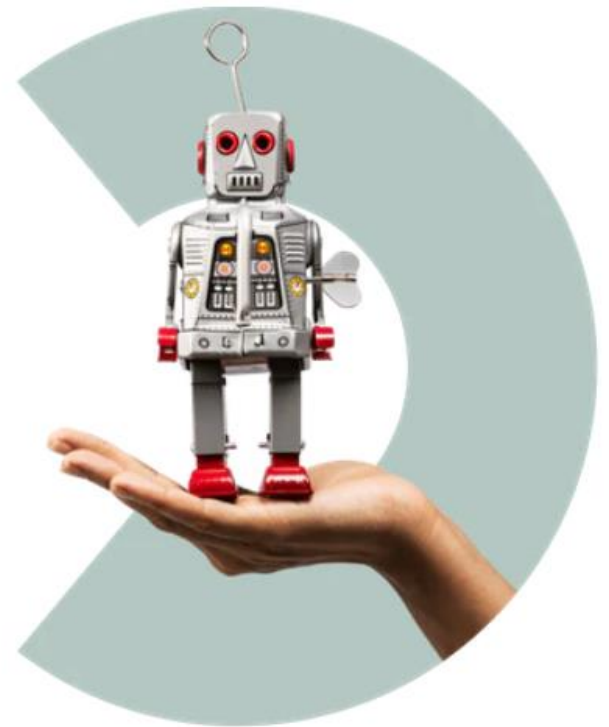
#CONTENT BASED #COLLABORATIVE FILTERING #HYBRID
#COSINE DISTANCE
#SERVERLESS #AZURE FUNCTIONS #MVP
#LIB SURPRISE

Ingénieur IA

Développez et intégrez des algorithmes de Deep Learning au sein d'un produit IA

OPENCLASSROOMS

OUDDANE NABIL



SOMMAIRE

Projet 9

Réalisez une application mobile de recommandation de contenu

A. INTRODUCTION

1. Contexte
2. Objectifs
3. Données

B. Préparation de l'environnement de développement

1. Installation
2. Test

C. Construction du modèle de Recommandation

1. Analyse rapide du jeu de données
2. Les types de système de recommandation
3. Différents systèmes de recommandation mis en œuvre
4. Collaborative filtering : test de la librairie Surprise
5. Comparaison de différents systèmes
6. Comparaison : des résultats à prendre avec des pincettes

D. Architecture cible: Bookshelf – Azure functions

1. 1ère possibilité: moteur de recommandation dans l'http trigger azure function
2. 2ème possibilité: l'http trigger azure function récupère le top5 dans un input binding sur cosmosDB
3. Test local/cloud des fonctions avec vscode
4. Push du dossier sur github

E. Annexes

A

INTRODUCTION

1. Contexte



- **ENJEU Compétences DU P9:**

- Concevoir des scripts permettant d'exécuter une chaîne de traitements IA bout-en-bout
- **Conception d'un MVP** sous forme d'une application mobile de recommandation de contenu
- **Architecture serverless** avec azure functions pour faire le lien entre le système de reco et l'appli

- **ENJEU global:**

- encourager la lecture en recommandant des contenus pertinents pour ses utilisateurs.
- MVP: réception de 5 articles recommandés



- Sélectionner l'architecture logicielle permettant de répondre au besoin métier

2. Objectifs

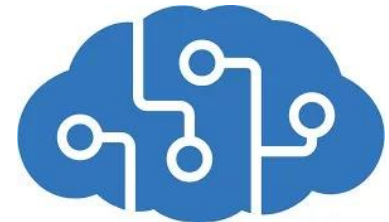
- **SCRIPT:**
 - **Développer une version de l'application mobile prenant en compte le système de recommandation en serverless:**
 - **Input: id utilisateur**
 - **Output : top 5 reco**
 - Utiliser GitHub
- **REFLEXIONS**
 - Architecture technique et description fonctionnelle de l'application
 - Architecture cible pour pouvoir prendre en compte l'ajout d'utilisateurs et de nouveaux articles

Délivrer

- Description fonctionnelle de l'application
- Schéma de l'architecture retenue
- Présentation du système e recommandation utilisé
- Schéma de l'architecture cible permettant de prendre ne compte les nouveaux utilisateurs ainsi que les nouveaux articles



Azure Machine Learning



3. données

- Données utilisateur :
 - interactions des utilisateurs avec les articles disponibles.
 - informations sur les articles (par exemple le nombre de mots dans l'article),
 - informations sur les sessions des utilisateurs (par exemple heures de début et de fin)
 - informations sur les interactions des utilisateurs avec les articles (Sur quel article l'utilisateur a-t-il cliqué lors de sa session ?).
 - <https://s3-eu-west-1.amazonaws.com/static.oc-static.com/prod/courses/files/Al+Engineer/Project+9+-+R%C3%A9alisez+une+application+mobile+de+recommandation+de+contenu/news-portal-user-interactions-by-globocom.zip>
- GitHub de l'application web de Julien :
 - <https://github.com/OpenClassrooms-Student-Center/bookshelf>
 - Mode opératoire: https://s3.eu-west-1.amazonaws.com/course.oc-static.com/projects/Ing%C3%A9nieur_IA_P9/Mode+ope%CC%81ratoire+test+Azure+function_V1.1.docx.pdf

B

PREPARATION DE L'ENVIRONNEMENT DE DEVELOPPEMENT

1- Installation de l'environnement

INSTALLATION ANDROID STUDIO:

Emulation d'un téléphone ANDROID 11
Avec Android Device Manager

INSTALLATION de Node.js et NPM

Node.js est un environnement d'exécution single thread pour des programmes écrits en javascript, coté serveur permettant de créer des applications en temps réel, rapides et évolutives.

Pour des applications à forte intensité de données, on préférera du multithread comme java.

- **Facile et incontournable** pour les débutants en dev web
- **Evolutif**: étant single thread, il peut traiter un grand nombre de connexions simultanées avec un débit élevé
- **Rapide**
- **Beaucoup de paquets node.js dans l'écosystème NPM** permettant de faciliter le travail
- **Solide** en c/c++
- **Multiplateforme**
- **Maintenable** car js pour gérer frontend et backend

CLONAGE DE L'APP Bookshelf de Julien faite en React Native:

React Native est framework d'applications mobiles open source

INSTALLATION de VSCODE:

Visual Studio Code est un éditeur de code source qui peut être utilisé avec une variété de langages de programmation, Il intègre les commandes Git

Creation de

- P9/bookshelf
- P9/functionOC avec un environnement virtuel

Connexion azure et creation d'une azure function via vscode

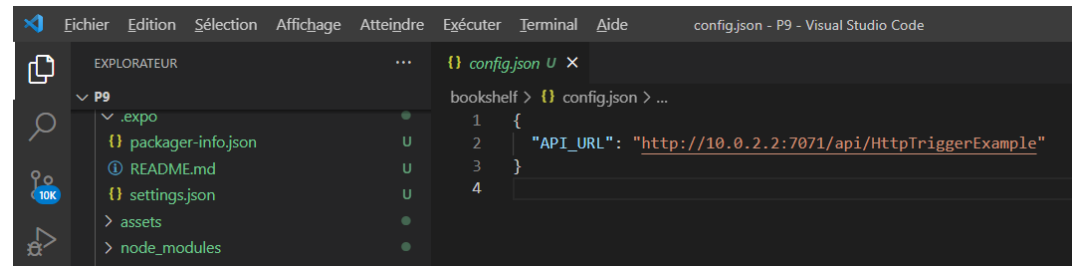
Exemple sur http trigger

<https://docs.microsoft.com/fr-fr/azure/azure-functions/create-first-function-vs-code-python>

2- Test de l'environnement

MODIFICATION DE L'URL dans config.json de bookshelf URL local / URL Azure

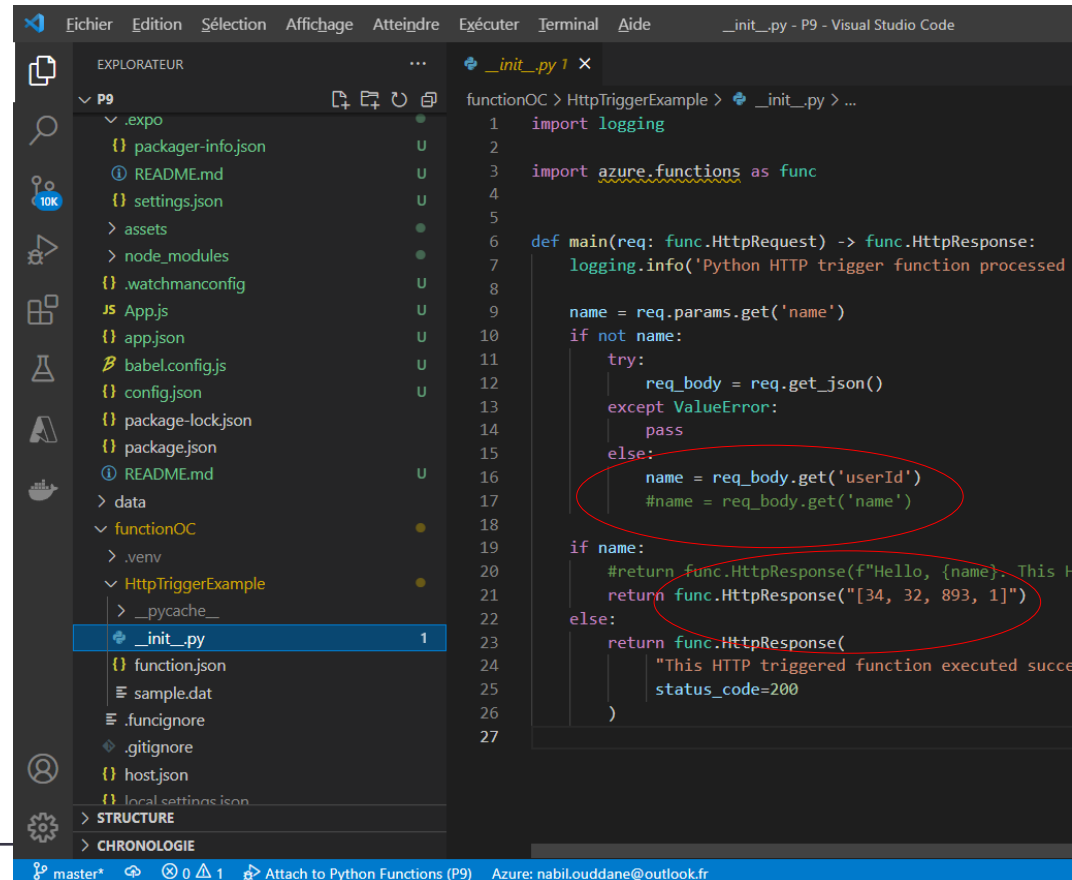
Pour le local il faut mettre 10.0.2.2 au lieu de localhost



```
config.json U X
bookshelf > {} config.json > ...
1 {
2   "API_URL": "http://10.0.2.2:7071/api/HttpTriggerExample"
3 }
4
```

MODIFICATION du _init_.py de l'azure function:

- Lecture du paramètre d'entrée
- Variable de réponse pour un test de l'app

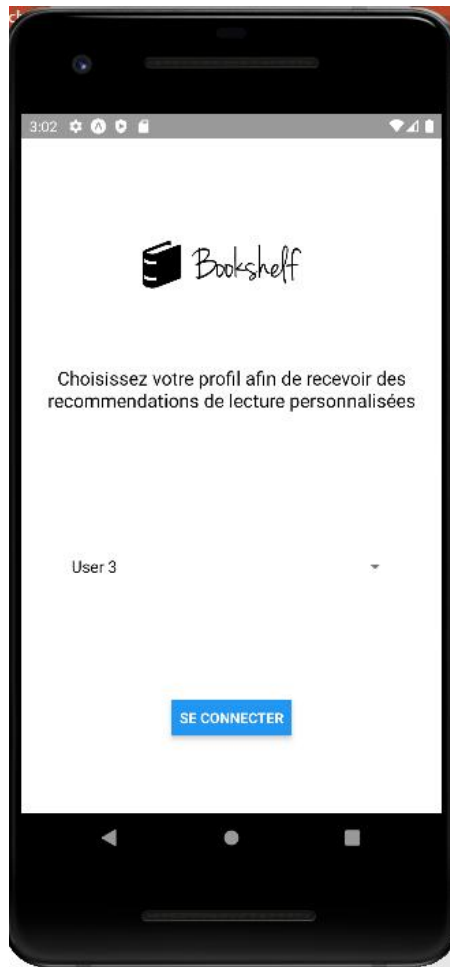


```
_init_.py P9 - Visual Studio Code
functionOC > HttpTriggerExample > _init_.py > ...
1 import logging
2
3 import azure.functions as func
4
5
6 def main(req: func.HttpRequest) -> func.HttpResponse:
7     logging.info('Python HTTP trigger function processed a request')
8
9     name = req.params.get('name')
10    if not name:
11        try:
12            req_body = req.get_json()
13        except ValueError:
14            pass
15        else:
16            name = req_body.get('userId')
17            #name = req_body.get('name')
18
19    if name:
20        #return func.HttpResponse(f'Hello, {name}. This HTTP triggered function executed successfully')
21        return func.HttpResponse("[34, 32, 893, 1]")
22    else:
23        return func.HttpResponse(
24            "This HTTP triggered function executed successfully",
25            status_code=200
26        )
27
```

- **Lancement de l'émulateur ANDROID**
Local/réseau
Sans oublier d'autoriser EXPO
(l'application client react native)
- **Lancement de NPM dans P9/bookshelf**
avec la commande NPM start
- **F5** pour lancer le mode debug

On doit faire avec quelques petites
instabilités nécessitant de nettoyer parfois
l'émulateur et de le relancer. Ca finit par
fonctionner

2- Test de l'environnement



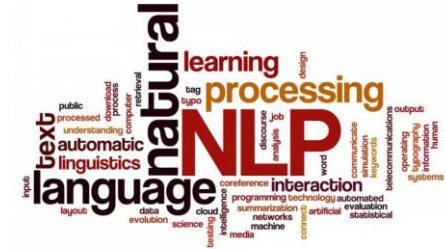
























CONSTRUCTION DU MODÈLE DE RECOMMANDATION



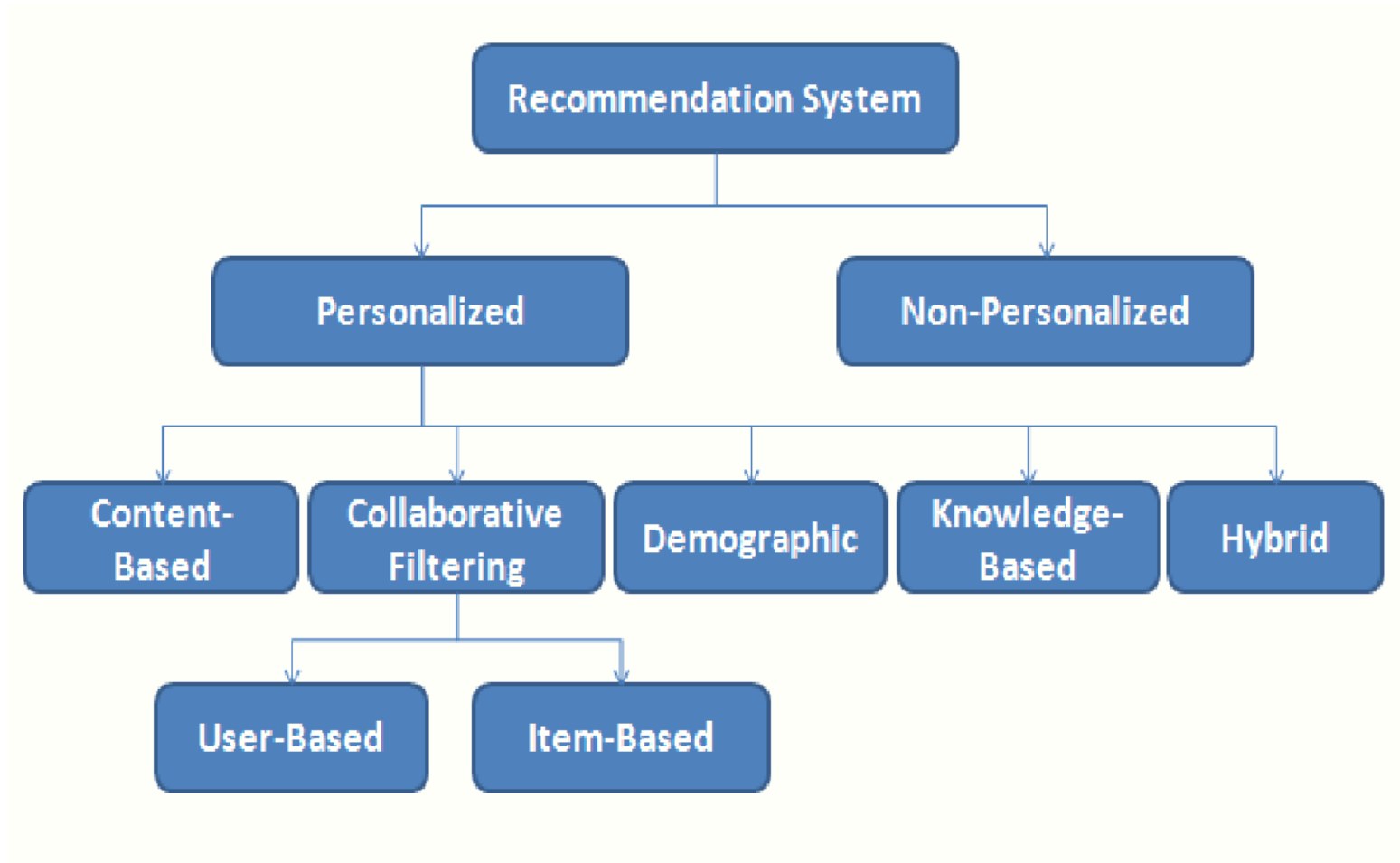
1. Analyse rapide du jeu de données

- Des interactions
 - User_id/article_id/click
- Des articles avec leur représentation vectorielle:
 - basée sur leur contenu
 - 250 dimensions
- Des informations temporelles de click et de session
 - pas assez précises pour être exploitables
- D'autres caractéristiques peu exploitables
- Pas de note explicite user_id/article_id



	 Book 1	 Book 2	 Book 3	 Book 4	 Book 5
 User A					
 User B					
 User C					
 User D					

2. Les types de système de recommandation

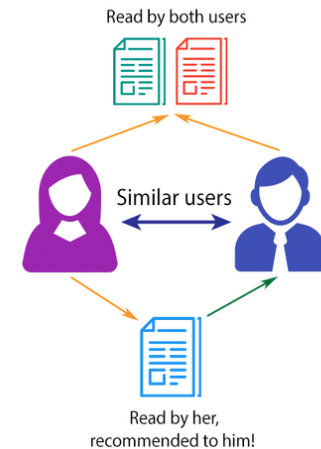


3. Différents systèmes de recommandation mis en œuvre

- **Collaborative Filtering:**

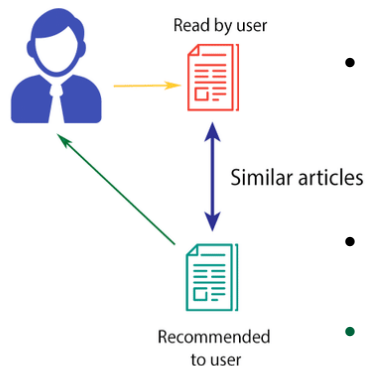
- **Similitude entre les utilisateurs**
- factorisation de matrice de rating uid/iid en général pour l'approche model based
- **Ne peut recommander que**
 - **des articles présents dans le jeu TRAIN pour des utilisateurs également présents dans le jeu TRAIN**
- **Incapable de recommander de nouveaux articles**

COLLABORATIVE FILTERING



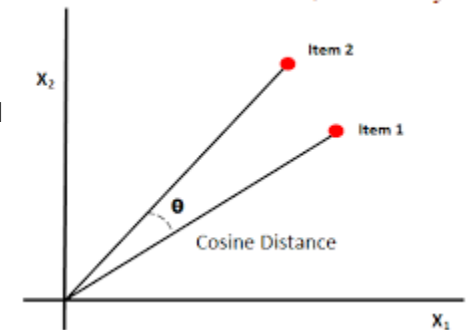
CONTENT-BASED FILTERING

- **Content based :**



- **similitudes entre les profils uid et les articles** dans un espace à 250 dimensions
 - word embeddings basé sur le contenu de la description des articles
- Seuls les uid dont on peut calculer un profil peuvent recevoir une recommandation
- **Permet de recommander de nouveaux articles des qu'on connaît son vecteur word embeddings**

Cosine Distance/Similarity



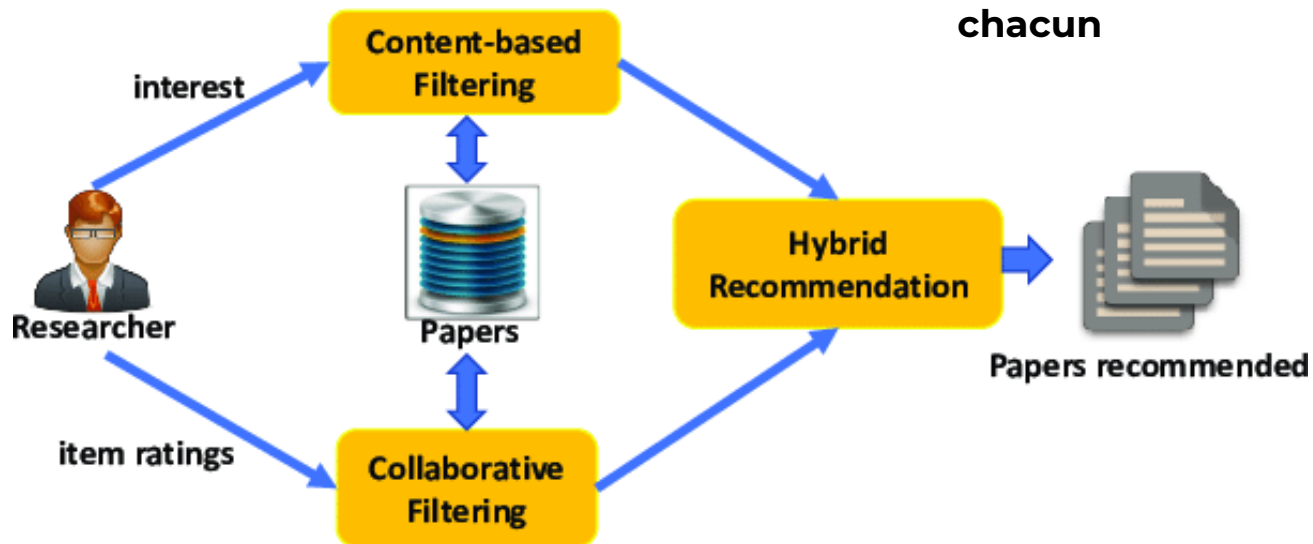
3. Différents systèmes de recommandation mis en œuvre

- **Popularity based:**

- Basé sur la popularité des articles
- **Assez statique** si construit de façon basique
- Permet de donner des recommandations **aux nouveaux utilisateurs**
- La popularité n'est jamais un mauvais choix

- **Hybrid:**

- Mix des systèmes précédents
- Prend les bons aspects de chacun

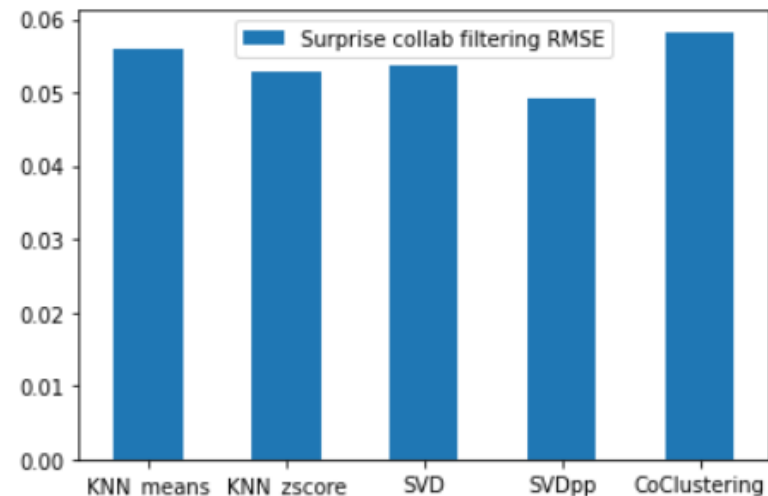


4. Collaborative filtering : test de la librairie Surprise

- Nos données représentent des **interactions** sous forme de click `user_id/articles`
 - Nous n'avons **pas de ratings explicites**
 - Dans ce cas la **librairie Implicit** semble plus adaptée
 - Nous pouvons tout de même utiliser le nombre de clicks comme un rating afin d'utiliser la librairie Surprise
- Différentes approches testées par **GRIDSEARCHCV**:
 - Memory based - KNN**
 - User_based
 - Item_based
 - Les matrices de similarité uid/uid peuvent rapidement exploser les capacités mémoires en fonction du nombre d'utilisateur
 - Model based – factorisation**
 - SVD
 - SVDpp

Le meilleur modèle SVDpp semble être la meilleure approche en terme de RMSE

Test su 5% des données:
27924 couples uid/iid interagis
244854 articles



5. Comparaison de différents systèmes

- Principe de l'évaluation:
 - Séparation du jeu de données en **TRAIN/TEST**
 - **Fit** des modèles sur le TRAIN
 - **Pour chaque uid du train** ou pour un nombre fini d'uid aléatoire (gain de temps de calcul),
 - **on prédit le vecteur iid complet de recommandation avec son vecteur note correspondant**
 - on calcul le rang de chaque iid interagi par l'uid issu de l'échantillon dans un vecteur aléatoire de 100 iid recommandés issus du vecteur complet
 - On calcul pour tout l'échantillon test le **pourcentage d'iid interagis dans le top5/top10**

```
{'modelName': 'Popularity-Filtering',  
  'recall@5': 0.9874357509994289,  
  'recall@10': 0.989149057681325}
```

```
{'modelName': 'Content-Based',  
  'recall@5': 0.29878048780487804,  
  'recall@10': 0.40853658536585363}
```

```
{'modelName': 'Collaborative-Filtering-SVDpp',  
  'recall@5': 0.7944031981724728,  
  'recall@10': 0.9868646487721302}
```

```
{'modelName': 'Hybrid-Filtering',  
  'recall@5': 0.2804878048780488,  
  'recall@10': 0.8902439024390244}
```

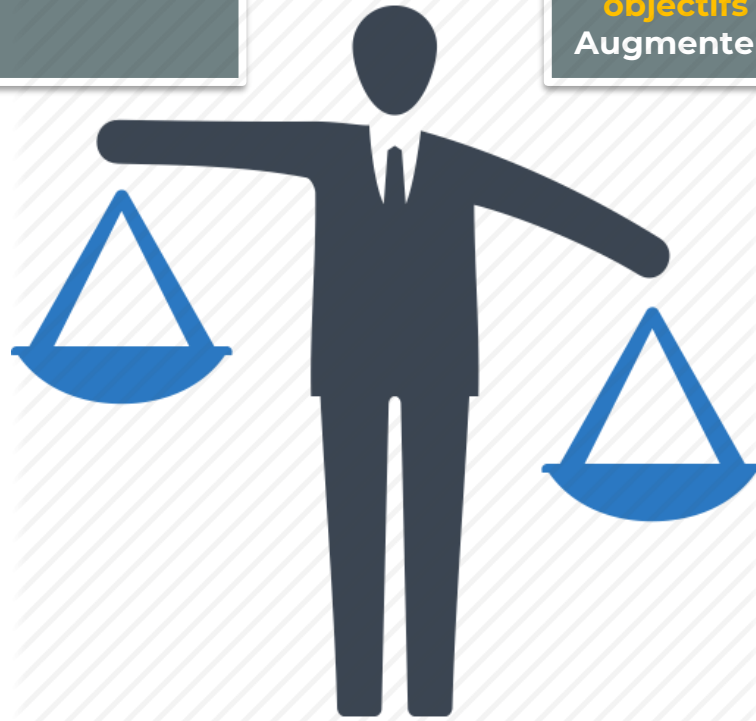
6. Comparaison : des résultats à prendre avec des pincettes

Dépendance des jeux de données aux systèmes de recommandation présents lors de leur création

Quel est le meilleur modèle?

- Celui qui donne les meilleurs résultats de test
- celui qui correspond le mieux aux objectifs ciblés:

Augmenter le CA ou favoriser la diversité



D

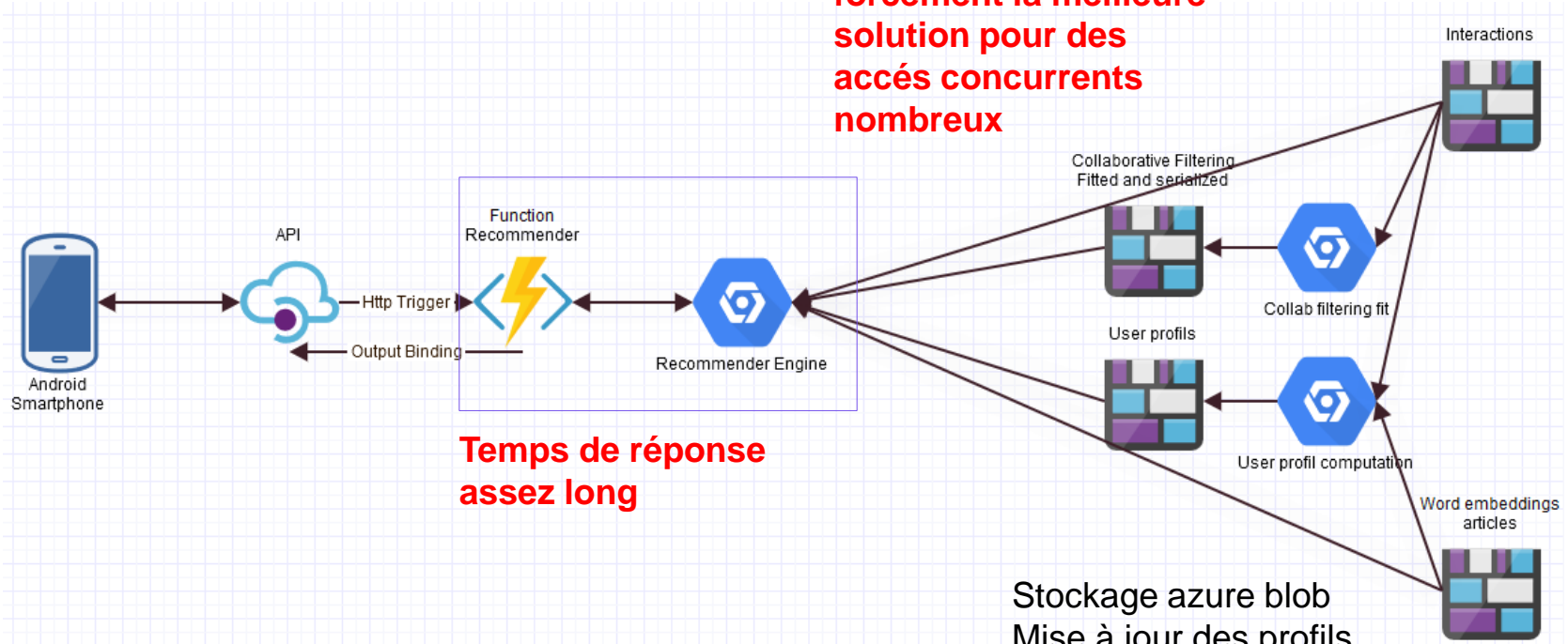
ARCHITECTURE CIBLE BOOKSHELF – AZURE FUNCTIONS



1. Architecture cible

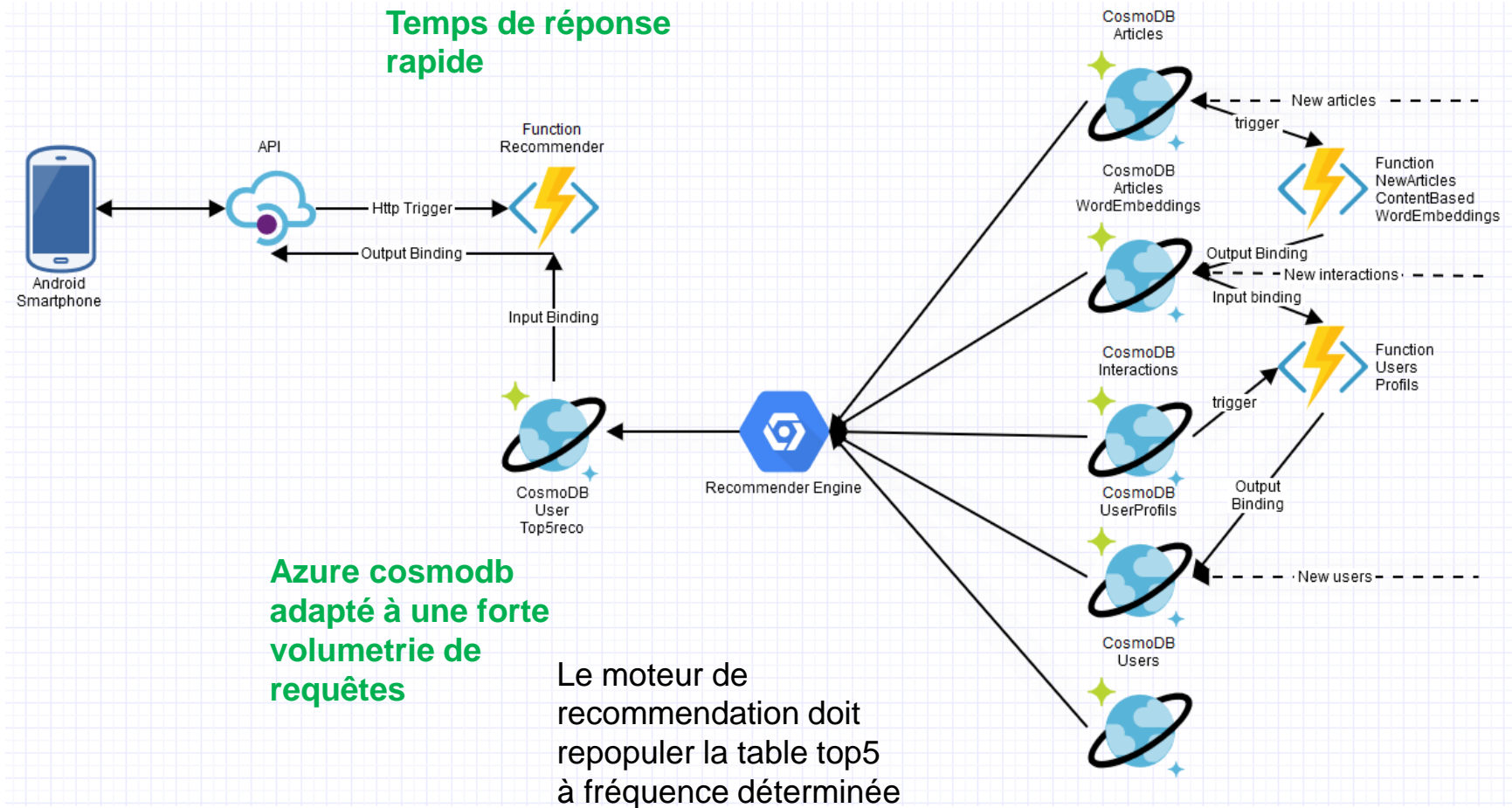
1^{ère} possibilité: moteur de recommandation dans l'http trigger azure function

Azure blob pas
forcement la meilleure
solution pour des
accès concurrents
nombreux



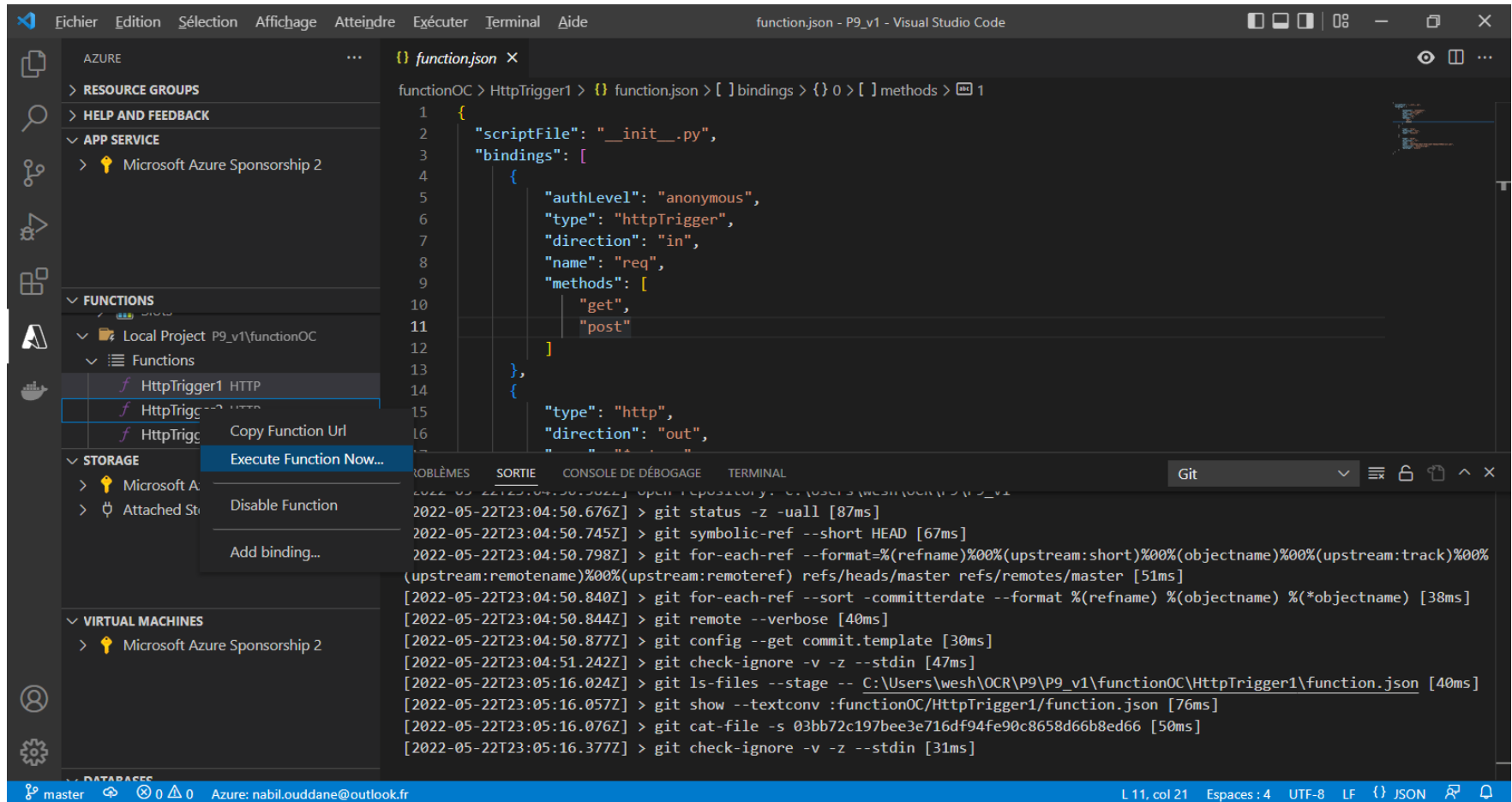
2. Architecture cible:

2ème possibilité: l'http trigger azure function recupere le top5 dans un input binding sur cosmosDB





3. Test local/cloud des fonctions avec vscode

```
{ "userId": "7" }
```



4. Push du dossier sur github

 Search or jump to... Pull requests Issues Marketplace Explore


 Private

Unwatch 1 Fork 0 Star 0

<> Code Issues Pull requests Actions Projects Security Insights Settings

master 1 branch 0 tags

Go to file Add file Code

 bilnab c'est fait 2996bba 2 hours ago 1 commit

.vscode	c'est fait	2 hours ago
bookshelf	c'est fait	2 hours ago
functionOC	c'est fait	2 hours ago

Add a README with an overview of your project. Add a README

About

No description, website, or topics provided.

0 stars
1 watching
0 forks

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Languages

Python 81.7% JavaScript 18.3%



ANNEXES

Ressources complémentaires

- Principes d'un système de recommandation base sur le contenu:
 - ❖ <https://www.youtube.com/watch?v=YMZmLx-AUvY>
 - ❖ <https://heartbeat.fritz.ai/recommender-systems-with-python-part-i-content-based-filtering-5df4940bd831>
 - <https://realpython.com/build-recommendation-engine-collaborative-filtering/>
- Azure functions:
 - ❖ https://www.youtube.com/watch?v=coT4lIGQLCw&list=PLbl2SbVli-Wo2W81Jyqlv5B375W_EcUsj
 - https://www.youtube.com/watch?v=9RLbuEnW-6g&list=PLbl2SbVli-Wo2W81Jyqlv5B375W_EcUsj&index=13
- Web-reco:
 - Librairie surprise: <https://surprise.readthedocs.io/>
 - Exemple de systeme de reco avec surprise: <https://medium.com/hacktive-devs/recommender-system-made-easy-with-scikit-surprise-569cbb689824>
 - Exemple de reco sur Kaggle: <https://www.kaggle.com/gspmoreira/recommender-systems-in-python-101>
 - Librairie implicit : <https://github.com/benfred/implicit>
 - Liste de modèle de collaborative filtering: <https://github.com/microsoft/recommenders>

Ressources complémentaires

- **Azure:**
 - ❖ Bonnes pratiques: https://s3.eu-west-1.amazonaws.com/course.oc-static.com/projects/Ing%C3%A9nieur_IA_P1/Bonnes_pratiques_consmmation_Azure.pdf
 - Azure functions et vscode: <https://docs.microsoft.com/fr-fr/azure/azure-functions/create-first-function-vs-code-python>
 - ❖ Creation azure functions sur azure: <https://docs.microsoft.com/fr-fr/learn/modules/create-serverless-logic-with-azure-functions/>
 - ❖ Liaison d'entrée de blobs dans azure function: <https://docs.microsoft.com/fr-fr/azure/azure-functions/functions-bindings-storage-blob-input?tabs=python>
 - Liaison d'entrée de cosmo db dans azure function: <https://docs.microsoft.com/fr-fr/azure/azure-functions/functions-bindings-cosmosdb-v2-input?tabs=python>