Guide d'Introduction

Réalisez une application de recommandation de contenu (Projet OC IA P10)

Auteur : **Patrice Preudhomme**
Date : **02/2025**

Table des matières

1. Contexte et objectifs
2. Structure des livrables
3. Trois dépôts GitHub
4. Comment s'articule l'architecture serverless ?
5. Procédure de navigation
6. Conclusion

1.Contexte et objectifs

La start-up **My Content** souhaite offrir à ses utilisateurs un **service de recommandation d'articles**. Mon projet (P10) vise à :

- **Analyser** un large volume de données d'interaction (clics, vues) sur de nombreux articles,
- Mettre en place un **moteur de recommandation** (le filtrage collaboratif, avec un modèle ALS),
- **Industrialiser** ce moteur via **Azure Functions** (pour bénéficier d'une architecture *serverless*),
- Proposer une **interface** (Streamlit) permettant de tester la recommandation à partir d'un `user_id`.

L'objectif global est de **démontrer** la **validité** et la **scalabilité** de cette solution :

- Du Notebook d'entraînement,
- Jusqu'au déploiement sur le cloud (Azure),
- En passant par une application web (Streamlit) accessible au public.

2.Structure des livrables

Le dossier transmis,

`OC_IA_P10_Realisez_une_application_de_recommandation_de_contenu_preudhomme_patrice`

contient **trois** sous-dossiers :

1. **preudhomme_patrice_1_application_022025**
    - Contient l'**application** (`app.py` pour Streamlit).
    - Le lien déployé est : https://p10-streamlit-app-2025.azurewebsites.net/
2. **preudhomme_patrice_2_scripts_022025**

- Rassemble les **scripts** pour l'industrialisation :
  - Le code de la **fonction Azure** (par ex. `recommend_articles/__init__.py`, qui contient la fonction `main` et le chargement du modèle, ainsi que `shared/azure_blob.py` pour la lecture du CSV sur Blob Storage),
  - Le **Notebook** (entraînement du modèle ALS, évaluations).

3. **preudhomme_patrice_3_presentation_022025**
   - Contient un **PDF** présentant :
     - Le besoin fonctionnel,
     - Les modèles testés (filtrage collaboratif, ALS, etc.),
     - Le schéma d'architecture,
     - Les performances du modèle,
     - Les pistes d'évolution (nouveaux utilisateurs/articles, mise à jour du modèle, etc.).

3.Trois dépôts GitHub

En complément, le code complet est versionné dans **trois dépôts** distincts sur GitHub :

1. **OC_IA_P10_Recommandation_contenu**
   - **Lien GitHub** : https://github.com/preudh/OC_IA_P10_Recommandation_contenu
   - **Rôle** : c'est ici que je développe et teste mon Notebook (Jupyter). J'y explore différentes méthodes de recommandation, j'y entraîne mes modèles (notamment ALS), et je documente mes résultats. Il fait office de "laboratoire" de R&D.
2. **OC_IA_P10_RecoFunction**
   - **Lien GitHub** : https://github.com/preudh/OC_IA_P10_RecoFunction
   - **Rôle** : ce dépôt contient mon projet **Azure Functions**, qui constitue le "back-end serverless". On y trouve notamment le fichier `recommend_articles/__init__.py` qui :
     - **Charge** le modèle ALS (fichier `.npz`),
     - **Lit** le CSV des clics via `shared/azure_blob.py` pour récupérer les données sur le Blob Storage,
     - **Retourne** la liste d'articles recommandés lorsqu'une requête HTTP arrive.
3. **OC_IA_P10_STREAMLIT_APP**
   - **Lien GitHub** : https://github.com/preudh/OC_IA_P10_streamlit_app
   - **Lien de l'app déployée** : https://p10-streamlit-app-2025.azurewebsites.net/
   - **Rôle** : c'est l'interface *Streamlit*, qui sert de "front-end" à mon système. L'utilisateur sélectionne un `user_id`, et l'appli appelle l'API (hébergée dans **OC_IA_P10_RecoFunction**) pour récupérer les articles recommandés.

4.Comment s'articule l'architecture serverless ?

- **Azure Functions** : j'y déploie une API sans gérer de serveur complet. Les fonctions principales (dans `recommend_articles/__init__.py`) :
  - Charge le modèle ALS via un fichier `.npz` local,

    - Appelle `load_clicks_csv()` depuis `shared/azure_blob.py` pour récupérer les clics depuis Blob Storage,
    - Retourne la liste d'articles recommandés au format JSON.
- **Azure Blob Storage** : y sont stockés le CSV (`clicks_sample.csv`) et d'autres fichiers importants (matrices de popularité, etc.). Cette méthode permet à la fonction Azure et à l'application Streamlit de partager les mêmes données.
- **Application Streamlit** :
    - L'utilisateur final se connecte à l'interface (sur Azure Web App).
    - Lorsqu'il clique sur "Obtenir des recommandations", l'appli envoie une requête à l'API (Azure Functions).
    - L'API exécute la recommandation (ALS) et renvoie la liste d'articles, qui s'affiche aussitôt.

En résumé :

- **OC_IA_P10_Recommandation_contenu** : Notebook de R&D (exploration, entraînement du modèle).
- **OC_IA_P10_RecoFunction** : Fonction Azure (API serverless) gérant la logique de recommandation et l'accès à Blob Storage.
- **OC_IA_P10_STREAMLIT_APP** : Interface utilisateur (Streamlit).

Cette séparation clarifie la logique :

- un dépôt pour la **recherche** (Notebook),
- un pour la **logique d'API** (Azure Functions),
- un pour la **partie front-end** (Streamlit).

5.Procédure de navigation

1. **Application (preudhomme_patrice_1_application_022025)**
    - Explorez le fichier `app.py` (ou similaire).
    - Testez éventuellement l'application via le lien déployé : https://p10-streamlit-app-2025.azurewebsites.net/.
2. **Scripts (preudhomme_patrice_2_scripts_022025)**
    - Consultez le Notebook d'entraînement (pour voir le filtrage collaboratif, ALS, etc.).
    - Découvrez la fonction Azure (`recommend_articles/__init__.py`) et `shared/azure_blob.py` pour comprendre comment l'API est exposée et où sont stockées les données.
3. **Présentation (preudhomme_patrice_3_presentation_022025)**
    - Parcourez le PDF pour une vue **synthétique** : architecture, choix du modèle, performances, perspectives (gestion de nouveaux utilisateurs, nouveaux articles, etc.).

6.Conclusion

En combinant ces trois livrables :

- **Une application** (Streamlit) qui démontre l'usage concret pour l'utilisateur final,

- **Des scripts** (Notebook, code Azure Functions) pour l'industrialisation et la maintenance,
- **Un support de présentation** expliquant l'architecture, les modèles et les résultats,

je montre la **faisabilité** et la **pertinence** d'une **architecture serverless** pour recommander des articles à grande échelle.

Pour plus de détails, vous pouvez consulter les **dépôts GitHub** listés ci-dessus.