# Rapport — Portfolio interactif (Timéo Tessier)

## Introduction

Dans le cadre de ce projet, une application web interactive a été développée à l'aide du framework **Streamlit**. Cette application a pour objectif de présenter mon portfolio, en regroupant différents éléments tels que le curriculum vitae, les projets réalisés, les compétences acquises et les expériences professionnelles.

L'enjeu principal du projet est de proposer une interface simple et intuitive permettant à un utilisateur de poser des questions en langage naturel et d'obtenir des réponses pertinentes et contextualisées. Ces réponses sont générées à partir de documents locaux, préalablement indexés, et exploitées par un agent conversationnel (« chat bot »). Le projet vise ainsi à illustrer la mise en œuvre d'un pipeline de type **Retrieval-Augmented Generation (RAG)**, combinant recherche sémantique et génération de texte, ainsi que son déploiement sur la plateforme **Streamlit**.

## Mise en place et organisation du projet

Le projet repose sur une architecture modulaire qui permet de séparer clairement les différentes responsabilités de l'application. L'interface utilisateur est entièrement gérée par le fichier app.py, qui constitue le point d'entrée de l'application Streamlit. Ce fichier prend en charge l'affichage de l'interface de discussion, de la barre latérale ainsi que des sources associées aux réponses générées.

La logique métier est isolée dans des scripts dédiés. Le fichier agent.py est responsable de la création et de la configuration de l'agent conversationnel, ainsi que de la définition des outils nécessaires à la recherche d'informations au sein du portfolio. Le découpage des documents textuels est quant à lui assuré par le script chunking.py, qui transforme les fichiers « *Markdown »* en segments plus courts, appelés *chunks*, afin de faciliter leur indexation et leur exploitation ultérieure.

Les données utilisées par l'application sont regroupées dans un dossier spécifique contenant les fichiers « *Markdown »* décrivant le profil, les projets et les compétences, ainsi qu'un fichier *JSON* résultant du processus de découpage. La configuration de l'environnement repose sur des variables d'environnement documentées dans un fichier « *env.example »*, ce qui permet de décrire les paramètres requis sans exposer d'informations sensibles dans le dépôt.

## Gestion du code

Le code source du projet est versionné à l'aide de **GitHub**, ce qui facilite le suivi des modifications et le travail collaboratif. Un workflow basé sur l'utilisation de branches et de *Pull Requests* est privilégié afin de garantir la stabilité du projet. Les fichiers contenant des clés ou des secrets ne sont jamais intégrés au dépôt et sont exclus via le fichier « gitignore ».

La recherche sémantique repose sur l'utilisation d'un index vectoriel distant fourni par **Upstash Vector**. Après la création d'un index sur la plateforme Upstash, les informations nécessaires à son accès sont renseignées sous forme de variables d'environnement. Un script d'indexation dédié permet alors d'envoyer les vecteurs correspondant aux chunks de texte vers l'index, rendant possible la recherche de documents pertinents lors des requêtes utilisateur.

## Environnement de développement

Le développement de l'application a été réalisé à l'aide de **Visual Studio Code**, en s'appuyant sur un environnement virtuel Python afin d'isoler les dépendances du projet. Celles-ci sont listées dans un fichier requirements.txt, ce qui permet de reproduire facilement l'environnement de travail sur une autre machine.

Chaque fichier du projet joue un rôle précis : le point d'entrée Streamlit gère l'interface, le script de l'agent centralise la logique de recherche et de génération, tandis que les fichiers « Markdown » constituent la base de connaissances consultée par l'agent conversationnel.

## Fonctionnement de l'application Streamlit

L'application se présente sous la forme d'une interface de discussion dans laquelle l'utilisateur peut formuler librement ses questions (comme sur chat gpt ou un autre site permettant la discussion avec une IA générative). L'agent conversationnel analyse ces requêtes, recherche les informations pertinentes dans l'index vectoriel, puis génère une réponse en s'appuyant sur les documents correspondants.

Une barre latérale complète l'interface principale en fournissant des informations sur l'auteur ainsi que des liens permettant de télécharger certains documents, comme le CV ou des bilans personnels. Afin d'améliorer la transparence du système, les sources utilisées pour chaque réponse sont affichées sous forme de badges, permettant à l'utilisateur d'identifier les documents mobilisés. L'historique de la conversation est conservé durant toute la session grâce à un mécanisme de stockage local propre à Streamlit.

## Exécution locale et déploiement

L'application peut être exécutée localement après le clonage du dépôt, l'installation des dépendances et la configuration des variables d'environnement. Si nécessaire, les documents peuvent être découpés et indexés à nouveau avant le lancement de l'application.

Le déploiement sur **Streamlit Community Cloud** s'effectue directement à partir du dépôt GitHub. Les paramètres sensibles sont renseignés via le système de secrets de la plateforme, garantissant la sécurité des clés d'accès. Les journaux d'exécution disponibles sur l'interface de Streamlit permettent de diagnostiquer rapidement d'éventuels problèmes après la mise en ligne.

## Conclusion

Ce projet a permis de concevoir une application web fonctionnelle offrant un accès interactif à un portfolio professionnel. L'architecture adoptée assure une séparation claire entre l'interface utilisateur, la logique de l'agent conversationnel et le système d'indexation vectorielle. La structure du projet et la documentation associée facilitent à la fois son utilisation en local et son déploiement en ligne.

À terme, plusieurs pistes d'amélioration peuvent être envisagées, notamment l'ajout d'une gestion des erreurs plus avancée, la mise en place d'un mode hors ligne reposant sur une recherche locale, ainsi qu'une amélioration de l'ergonomie et de la présentation des sources.