Documentation Technique – Projet de Recherche de Billets d'Avion

1. Objectif du projet

L'objectif est de développer un **agent intelligent** capable d'assister les employés de l'entreprise **SFM** (ou les utilisateurs finaux) dans la recherche et la sélection de billets d'avion selon des **critères personnalisés**.
L'agent combine plusieurs briques logicielles :

- **Scraping des données de vols** via Google Flights.
- **Analyse intelligente** et synthèse avec un **LLM** (LangChain + Ollama).
- **Génération d'un rapport Word** pour une utilisation facile.
- **Orchestration** des interactions grâce à **n8n** (formulaire + API HTTP).

2. Architecture générale

1. L'utilisateur remplit un **formulaire n8n** (DE = départ, AR = arrivée, date).
2. Le workflow **n8n** envoie une **requête HTTP POST** au backend Python (FastAPI).
3. Le backend exécute :
    - **Scraping Google Flights** avec Playwright.
    - Sauvegarde des résultats dans un **CSV local**.
    - Appel de l'**agent IA** (LangChain + LLM) pour résumer et recommander.

- Génération d'un **document Word** avec les résultats.

4. Le rapport final **LesVols.docx** est disponible pour l'utilisateur.

3. Description des fichiers

a) GoogleFlight_Scraping.py

- **Type** : Script principal (backend / API FastAPI).
- **Utilité** :
  - Reçoit la requête HTTP depuis n8n.
  - Construit l'URL Google Flights encodée.
  - Lance Playwright pour extraire les infos.
  - Sauvegarde les résultats en CSV.
  - Appelle l'agent IA et génère le rapport Word.

Fonctions principales :

- search_flights_iata(request: Request)
  - Entrée : JSON { "DE": "CDG", "AR": "FRA", "date": "2025-09-10" }.
  - Processus : construit l'URL, appelle le scraping, sauvegarde en CSV, appelle l'IA, génère LesVols.docx.
  - Sortie : chemin du fichier Word.
- scrape_flight_data(one_way_url: str)
  - Ouvre la page Google Flights.
  - Extrait vols : compagnie, horaires, durée, escales, prix.

    - Retour : list[dict].

- save_to_csv(data: list, filename: str)
    - Nettoie et enregistre les résultats en CSV.

b) agent.py

- **Type** : Module IA (LangChain).
- **Utilité** : Transformer les résultats bruts en **synthèse intelligente**.

Fonctions principales :

- model = OllamaLLM(model="llama3.2", temperature=0)
    - LLM local, déterministe.
- run_flight_agent(data: dict, flights: list) -> dict
    - Entrée : dictionnaire (vols + critères).
    - Transforme les vols en texte, appelle le LLM.
    - Sortie :
        - nombre total de vols,
        - aperçu des premiers résultats,
        - réponse textuelle (recommandation).

c) word.py

- **Type** : Génération de documents Word.

- **Utilité** : Fournir un rapport lisible par l'utilisateur.

Fonctions principales :

- clean_llm_text(text: str) : supprime caractères parasites.
- save_llm_to_word(text: str, path="LesVols.docx") : génère le fichier final.

d) Scraped_Data.csv

- Généré automatiquement.
- Contient les résultats du scraping :
    - heure départ, arrivée, compagnie, durée, escales, prix.

e) LesVols.docx

- Rapport final généré automatiquement.
- Contient la **synthèse IA** et un **tableau récapitulatif**.

f) requirements.txt

- Liste des dépendances :
- fastapi
- playwright
- pandas
- python-docx

- langchain
- ollama

g) env/

- Environnement virtuel Python isolé.

h) workflow.json (n8n)

- Décrit l'automatisation n8n.
- **Étapes principales** :
    1. **Formulaire n8n** : saisie utilisateur (champs DE, AR, date).
    2. **HTTP Request** : POST vers http://host.docker.internal:8000/search_flights_iata.

4. API – Endpoints

Endpoint principal

- URL : POST http://host.docker.internal:8005/recherche-billets

Paramètres d'entrée (JSON)

Le backend reçoit les données saisies par l'utilisateur dans le formulaire n8n :

```
{
  "DE": "CDG",
  "AR": "FRA",
```

"date": "2025-09-10",

"PrixMax": 350,

"Notes": "Préférence vol direct",

"Escale": "non"

}

- DE : code ou ville de départ (ex. CDG).
- AR : code ou ville d’arrivée (ex. FRA).
- date : date du vol (format AAAA-MM-JJ).
- PrixMax (optionnel) : budget maximum en euros.
- Notes (optionnel) : préférences de l’utilisateur (compagnie, horaires, etc.).
- Escale (optionnel) : "oui" / "non".

Réponse (JSON)

Exemple de retour du backend :

{

"status": "success",

"csv": "Scraped_Data.csv",

"word": "LesVols.docx",

"total_flights": 15,

"recommendation": "Le vol LH123 correspond le mieux à vos critères (direct, prix 320 €)."

}

- status : état de la requête (success ou error).
- csv : fichier CSV généré contenant les données brutes.
- word : chemin vers le rapport Word final.
- total_flights : nombre total de vols trouvés.
- recommendation : résumé IA avec recommandation personnalisée.

5. Solutions techniques retenues

- **Playwright** : fiabilité et gestion des pages dynamiques (Google Flights).
- **LangChain + Ollama** : IA locale sans dépendance cloud.
- **n8n** : automatisation low-code, intégration facile avec formulaire.
- **FastAPI** : API performante et simple à déployer.
- **Word (python-docx)** : sortie lisible pour les utilisateurs finaux.