**Projet Applicatif M2 : Gestion de vols et recommandations (MongoDB + Jupyter)**

# Membres du projet

Nous sommes un groupe de cinq étudiantes en M2 Management, Data & IA :  
  
- CASTELLI Ornella  
- KHALLOULI Nadia  
- CHOUMOUL Lamya  
- MOUNGAD Massilia  
- GUEYE Samira Bourgi  
  
Nous avons réalisé ce projet de manière collaborative dans un environnement Jupyter connecté à une base MongoDB Atlas, avec une attention portée sur la qualité des données, les opérations de gestion, et la personnalisation des services pour les utilisateurs.

# Objectif du projet

Nous avons réalisé un projet de système de gestion des vols pour l'aéroport Charles de Gaulle. Notre objectif était de concevoir une solution technique intelligente qui permet non seulement de gérer les vols, les passagers et les services associés, mais aussi de proposer des recommandations personnalisées aux voyageurs grâce à l’analyse des données.

Le projet a été conçu intégralement en Python via Jupyter Notebook, en interaction avec une base MongoDB hébergée sur MongoDB Atlas.

# Étape 1 : Création et modélisation de la base MongoDB

Dès le départ, nous avons choisi MongoDB Atlas pour sa flexibilité et sa capacité à gérer des données semi-structurées. Nous avons créé une base nommée airport contenant les collections suivantes :

* flights : chaque vol avec numéro, aéroports, horaires, capacité, statut
* passengers : informations des passagers et leurs profils
* services : services associés à un vol (contrôle, bagages...)
* bookings : réservation de vols avec place et classe
* baggage : gestion des bagages (poids, statut, position)

**✈️ Collection flights**

Cette collection contient tous les vols gérés par l'aéroport. Chaque document correspond à un vol avec :

* flight\_id : identifiant unique du vol (ex. "CDG001")
* flight\_number : numéro du vol (ex. "AF123")
* departure\_airport, arrival\_airport : aéroports d’origine et de destination
* departure\_time, arrival\_time : horaires programmés
* status : état actuel du vol ("On Time", "Delayed", "Cancelled")
* capacity : nombre total de sièges disponibles

**👤 Collection passengers**

Cette collection regroupe les données personnelles et contextuelles des passagers :

* passenger\_id : identifiant unique
* first\_name, last\_name : nom et prénom
* passport\_number : numéro de passeport
* flight\_id : identifiant du vol réservé
* booking\_date : date de réservation
* profile : dictionnaire contenant les préférences du passager :
  + preferred\_destinations : destinations favorites
  + flight\_type : type de vol préféré (ex : direct, escale)

**🧳 Collection baggage**

Cette collection permet de tracer les bagages enregistrés dans l’aéroport :

* baggage\_id : identifiant unique
* passenger\_id : à qui appartient le bagage
* flight\_id : vol associé
* weight : poids du bagage
* status : état du bagage ("Checked In", "Loaded", "Lost")
* position : dernière position connue ("Belt 4", "Cargo Zone")

**📅 Collection bookings**

Elle gère les réservations effectuées par les passagers :

* booking\_id : identifiant unique
* passenger\_id et flight\_id
* seat\_number : place attribuée (ex. "12A")
* class : classe de voyage ("Economy", "Business")
* booking\_date : date à laquelle la réservation a été effectuée

**🔐 Collection services**

Cette collection suit les services fournis autour d’un vol :

* service\_id : identifiant du service
* flight\_id : identifiant du vol concerné
* service\_type : nature du service ("Security Check", "Baggage Handling", "Boarding")
* service\_time : heure du service
* status : progression ("Planned", "In Progress", "Completed")

Ces collections sont reliées logiquement par les identifiants flight\_id et passenger\_id, permettant de croiser les informations dans les traitements analytiques ou fonctionnels (recommandation, suivi, reporting).

# Étape 2 : Opérations CRUD dans un Notebook Jupyter

Pour interagir avec notre base, nous avons utilisé la bibliothèque `pymongo` et développé des fonctions personnalisées dans un notebook. L'objectif était de pouvoir :  
- Créer (`insert\_one`) de nouveaux vols, passagers ou services  
- Lire (`find`, `find\_one`) les données par identifiant ou relation  
- Mettre à jour des champs précis comme le statut d’un vol ou le profil d’un passager (`update\_one` + `$set`)  
- Supprimer un document ciblé (`delete\_one`)

# Étape 3 : Algorithme de recommandation personnalisée

Le cœur du projet est un algorithme de recommandation basé sur les profils passagers. Il permet de suggérer des destinations à un passager donné en fonction :  
1. De ses destinations préférées et type de vol  
2. Du profil d’autres passagers similaires  
3. Des destinations qu’il n’a pas encore visitées



Une image contenant texte, capture d’écran, logiciel, Police

Le contenu généré par l’IA peut être incorrect.

Cette logique nous permet de générer dynamiquement des suggestions personnalisées à afficher ou envoyer par mail.

# Vers une API ou microservice

Notre code a été pensé de façon modulaire, afin de pouvoir le migrer vers :  
- une API REST (FastAPI ou Flask)  
- un module intégrable à une interface Django  
- un service automatisé d'email de suggestion

Nous avons tenté de creuser en utilisant Django mais à ce jour nous ne sommes pas arrivée à un résultat concluant.

# Architecture simplifiée

Utilisateur  
 └---> Notebook Jupyter (CRUD, Algo)  
 └---> MongoDB Atlas