|  |
| --- |
| **BOOTCAMP « Data Engineer »**  **(Session de février 2023)**  **PROJET DST AIRLINES**  **Rapport sur l’étape n°1** |

|  |  |
| --- | --- |
| **Participants** | **Rôle** |
| **Bruno Bevert** | **Réalisateur** |
| **Philippe Rérole** | **Réalisateur** |
| **Thimothée Baures** | **Réalisateur** |
| **Darshi Khatri** | **Réalisatrice** |
| **Dan (@Datascientest)** | **Mentor** |
| **Dimitri (@Datascientest)** | **Backup Mentor** |
| **Yaniv (@Datascientest)** | **Chef de cohorte** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Historique du document** | **Auteur** | **Version** | **Date** |
| **Initialisation** | **Bruno Bevert** | **0.1** | **02/03/2023** |
| **Relecture** | **Toute l’ équipe** | **0.2** | **10/03/2023** |
| **Version finale pour relecture Dan** | **Bruno Bevert** | **1.0** | **14/03/2023** |
| **Mise à jour (nouvelles données et services utilisés)** | **Bruno Bevert** | **1.1** | **16/03/2023** |
| **Ajout des exemples Données dynamiques** | **Bruno Bevert** | **1.2** | **24/03/2023** |

**Remarque :** les documents du projet sont stockés sous [GitHub](https://github.com/DataScientest-Studio/FEV23_Airlines).

Table des matières

[Préambule 3](#_Toc130511893)

[Introduction 3](#_Toc130511894)

[Collecte et exemples de données 3](#_Toc130511895)

[Les données accessibles 4](#_Toc130511896)

[Traitement des données 5](#_Toc130511897)

[Aperçu de la solution « API » 5](#_Toc130511898)

[Champs d’applications 5](#_Toc130511899)

[Annexe : 6](#_Toc130511900)

[Exemples de données collectées 6](#_Toc130511901)

[Les données statiques 6](#_Toc130511902)

[Les données dynamiques 11](#_Toc130511903)

[Les services non retenus 13](#_Toc130511904)

# **Préambule**

Nous devions avant tout prendre en compte certaines contraintes inhérentes à tout projet : les délais à respecter, la complexité du projet, les ressources disponibles pour le réaliser, etc. De ce fait, nous avons adopté une stratégie « Agile » consistant à mettre en œuvre le plus rapidement possible une fonctionnalité significative qu’un utilisateur lambda pourrait utiliser depuis une application que nous déploierons en fin de parcours. Selon note capacité à faire et le temps qui nous restera, nous pourrions alors ajouter, par itération et d’une manière incrémentale, d’autres fonctionnalités (prévision de retards avec des paramètres issus de sites «  météo », analyse de l’empreinte carbone d’un avion).

# **Introduction**

Dans cet esprit, l’étude de l’API du [site Lufthansa](https://developer.lufthansa.com/docs) nous a conduit à définir un premier périmètre du projet. Après l’analyse des données retournées par chacun des services, un élément s’est vite détaché des autres : la différence entre les heures de départs et d’arrivées des différents avions planifiés et ceux réalisés. Nous avons donc décidé de fonder la première version de notre application sur l’exploitation de statistiques s’appuyant sur différents types de retards sous la forme d’une API. A noter que seules les lignes commerciales avec des passagers sur l’année 2023 ont été retenues du fait des contraintes du site de l’API de la Lufthansa.

# **Collecte et exemples de données**

Nous avons à faire à 2 grandes familles : les données dynamiques, issues des services de l’API, et celles que l’on peut considérer comme « statiques », ou « quasi-statiques » (des aéroports ne sont pas construits tous les jours dans le monde…), en fait celles qui vont surtout servir à renseigner notre référentiel. La récupération de ces dernières se feront également avec des techniques de « webscraping » bien que nous avons constaté que les services de l’API permettent d’en avoir une bonne partie.

Pour chaque service de l’API, l’implémentation se fondera dur la librairie « request » de Python. Un « POC » a démontré que nous pourrons alimenter une base de données relationnelle qui sera hébergée sur une VM. Dans ce cadre, des services de tests ont permis de valider la faisabilité technique de la collecte des données depuis les API Lufthansa. On pourra trouver des exemples dans [l’annexe](#_Annex_:_exemple).

Les diagrammes ci-dessous présentent les principales données qui seront collectées et stockées en base de données :

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

# **Les données accessibles**

L’application proposera des services rassemblés dans une API. En tenant compte des restrictions de l’API de Lufthansa, les paramètres qui seront à disposition seront les aéroports, les lignes des compagnies aériennes ainsi que les différentes routes aériennes associées.

# **Traitement des données**

Toutes les données fournies par les services de l’API ne sont pas forcément utiles pour notre besoin. De ce fait, un filtre sera appliqué afin de ne considérer que les données que nous considérons comme essentielles.

Notre objectif est double :

1. Stocker les données de l’API « Luthansa », et autres éventuelles données complémentaires, dans une base de données (en suivant un schéma défini auparavant).
2. Réaliser notre API de services. Ces derniers répondraient à un certain nombre de cas d’utilisations tournant, pour commencer, autour des retards (lister les lignes cumulant le plus de retards, les retards des avions, les lignes qui présentent le plus de retards, les aéroports les plus concernés par les retards, etc.).

# **Aperçu de la solution « API »**

Une image contenant diagramme

Description générée automatiquement

# **Champs d’applications**

En prenant l’exemple d’un avion, un utilisateur de l’API pourrait lancer une action auprès des acteurs concernés pour identifier les raisons de retards et, le cas échéant procéder à des inspections de sécurité, des maintenances ou même interdire de vol ledit avion : On voit là un intérêt majeur car la sécurité des passagers peut être affectée si rien n’est fait. On peut aussi y voir un intérêt économique : même si l’avion s’avère sûr, il est possible qu’il soit devenu inefficient, avec une consommation trop importante de kérosène, doublé d’un temps allongé de vol.

Annexe :

## Exemples de données collectées

### Les données statiques

#### Le référentiel des pays (issu du service mds-references/countries/)



#### Le référentiel des villes (issu du service mds-references/cities/)



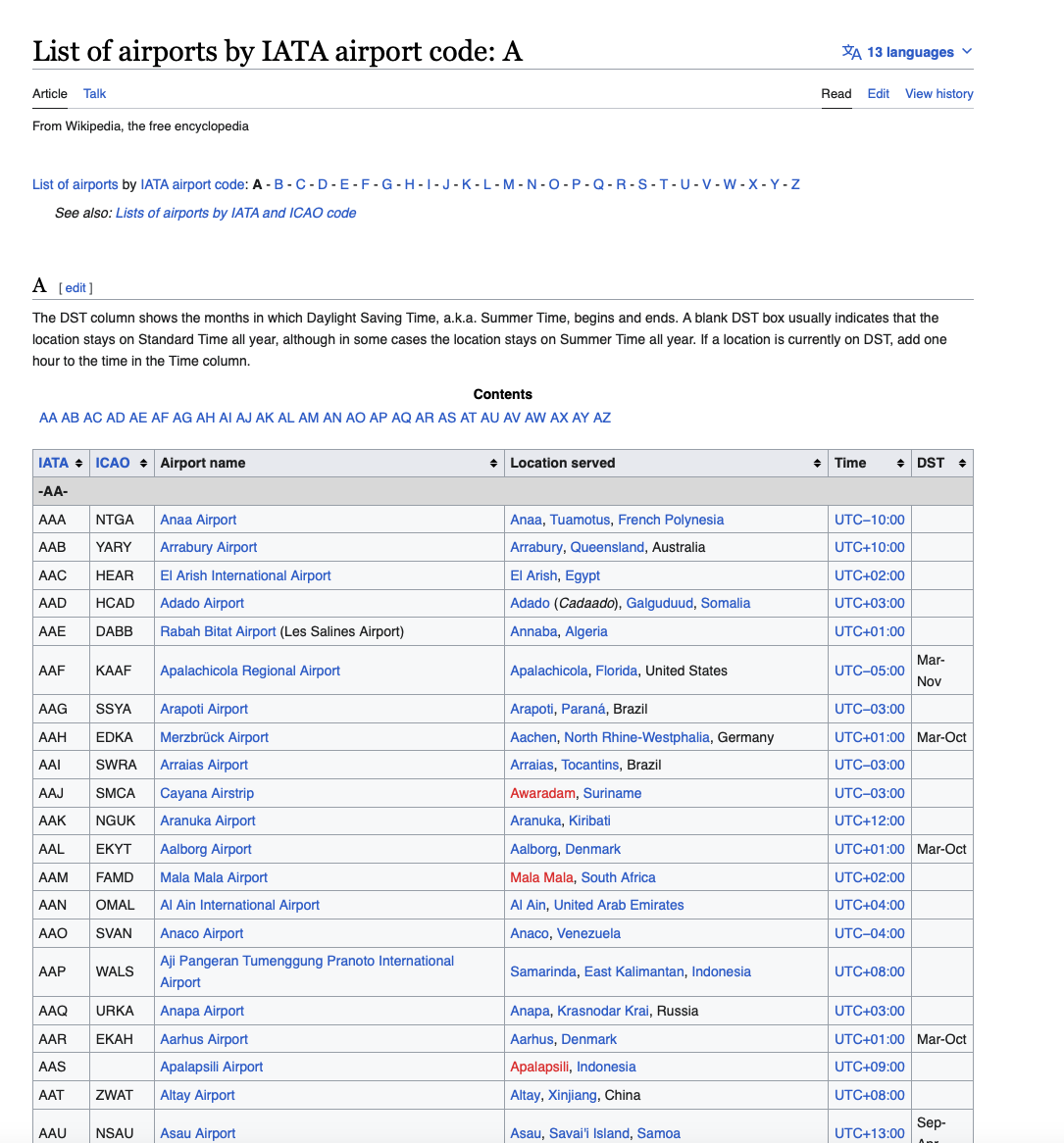
#### Le référentiel des avions (issu du service / mds-references/aircraft/)

#### Le référentiel des lignes aériennes (issu du service /mds-references/airlines)



#### Le référentiel des aéroports

La technique du webscraping est utilisé. Elle se fonde sur des pages Wikipédia (<https://en.wikipedia.org/wiki/List_of_airports_by_IATA_airport_code:_A>, etc.)



### Les données dynamiques

#### Les vols planifiés (issus du service flight-schedules/flightschedules/passenger)

Une image contenant texte

Description générée automatiquement

#### Le statut des vols (Issus du service operations/flightstatus)

Une image contenant table

Description générée automatiquement

### Les services non retenus

* operations/schedules/
* operations/customerflightinformation/
* operations/customerflightinformation/arrivals/
* operations/customerflightinformation/route/
* operations/customerflightinformation/departures/
* operations/flightstatus/route/
* operations/flightstatus/arrivals/
* operations/flightstatus/departures/
* mds-references/airports/nearest/ (Webscraping à la place)
* offers/seatmaps/
* offers/lounges/