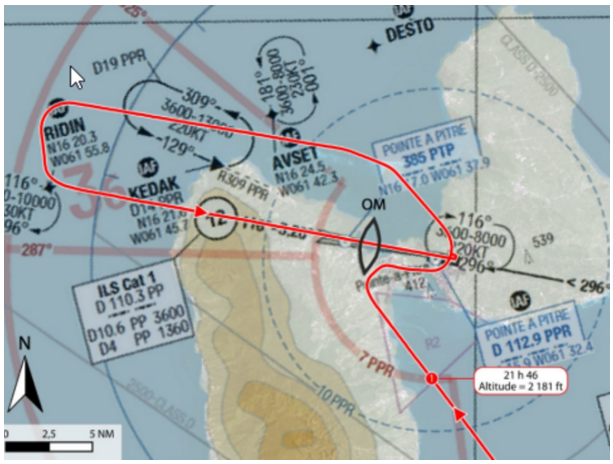
Ecole Nationale des
Sciences Géographiques

BEA

Rapport final du projet informatique

Cycle des Ingénieurs diplômés de l'ENSG 3ème année

Extraction d'informations géographiques fournies par le Service d'Information Aéronautique (SIA) pour la création d'une base de données géographique



Louis Steinmetz

Février 2023

☒ Non confidentiel ☐ Confidentiel IGN ☐ Confidentiel Industrie ☐ Jusqu'au ...

Ecole Nationale des Sciences Géographiques
6-8 Avenue Blaise Pascal - Cité Descartes - 77420 Champs-sur-Marne
Téléphone 01 64 15 31 00 Télécopie 01 64 15 31 07

Table des matières

Glossaire et sigles utiles **3**

Introduction **4**

1 Contexte et objectifs **5**
1.1 Contexte : l'apport des données du SIA pour les enquêtes du BEA 5
1.2 Objectifs de l'étude 5

2 Analyse fonctionnelle **7**
2.1 Présentation générale de notre outil 7
2.2 Format XML-SIA ou AIXM ? 8
2.3 Structure de la base de donnée géographique 10

3 Étude et Réalisation technique **15**
3.1 Choix des logiciels et bibliothèques utilisés 15
3.2 Présentation du résultat 16
3.3 Difficultés rencontrés 17
3.4 Etat d'avancement et amélioration possible 17

4 Gestion de projet **19**
4.1 GitHub 19
4.2 Planning 19

Glossaire et sigles utiles

ENSG *École Nationale des Sciences Géographiques*

BEA *Bureau d'Enquêtes et d'Analyses* - Le BEA est l'organisme officiel français qui réalise les enquêtes de sécurité sur les accidents et incidents grave de l'aviation civile survenus en France.

PPMD *Photogrammétrie, Positionnement et Mesure de Déformations* - Filière de 3ème année du cycle ingénieur de l'*ENSG*

SIA *Service d'Information Aéronautique* - Instance gouvernementale française chargées de rendre les services d'informations aéronautique en France.

AIXM *Aeronautical Information Exchange Model* - Format standard d'échange de données aéronautique nottament utilisé par l'Europe et les Etats-Unis.

VAC *Visual Approach Chart* - document relatif à un aérodrome et permettant l'approche et l'atterrissage en vol à vue.

SGBD *système de gestion de base de données* - logiciel système servant à stocker, à manipuler ou gérer, et à partager des données dans une base de données.

DGAC *Direction Générale de l'Aviation Civile* - institution qui régit l'aviation civile et le transport aérien en France.

BDDG *Base De Donnée Géographique*

Introduction

Dans le cadre de la troisième et dernière année du cursus ingénieur de l'ENSG en option *PPMD*, nous sommes amenés à réaliser un projet informatique de Novembre 2023 à Février 2024. Ce projet est découpé en deux grandes phases. Les deux premiers mois sont consacrés à l'analyse du projet, phase durant laquelle le but est d'analyser les besoins du commanditaire et réaliser l'état de l'art des outils disponibles pour mener à bien ce dernier. Enfin, cette partie a pour objectif de présenter les solutions qui seront implémentées lors de la seconde phase : le développement informatique.

J'ai choisi de travailler sur l'extraction d'informations géographiques fournies par le Service d'Information Aéronautique (SIA) pour la création d'une base de données géographique car étant un passionné d'aéronautique, ce sujet me permet de manipuler des données primordiales dans ce domaine là, tout en me faisant découvrir le BEA, entité clé dans la sécurité de l'aviation civile en France. Aussi, il me permet de voir un domaine d'application concret de la géomatique dans le milieu de l'aéronautique. Ce sujet est proposé par Jean-François VILLEFORCEIX, et a été encadré par ce dernier ainsi que par Darian MOTAMED.

Ce rapport présente le contexte dans lequel s'inscrit ce projet, ainsi que les objectifs et les solutions envisagées. Pour finir, il détaille les différentes méthodes mise en oeuvre pour mener à bien le développement informatique.

Chapitre 1 Contexte et objectifs

1.1 Contexte : l'apport des données du SIA pour les enquêtes du BEA

Dans le cadre des enquêtes de sécurité de l'aviation civile, le département technique du BEA s'intéresse notamment à l'environnement aéronautique lié à un évènement. Cependant, ces données évoluent régulièrement et par conséquent, nécessitent d'être mises à jour régulièrement. En France, c'est le SIA qui est en charge de publier ces données tous les mois. Plusieurs supports sont mis à disposition des utilisateurs, notamment grâce à un fichier XML contenant toutes les entités géographiques à jour, au format vectoriel.

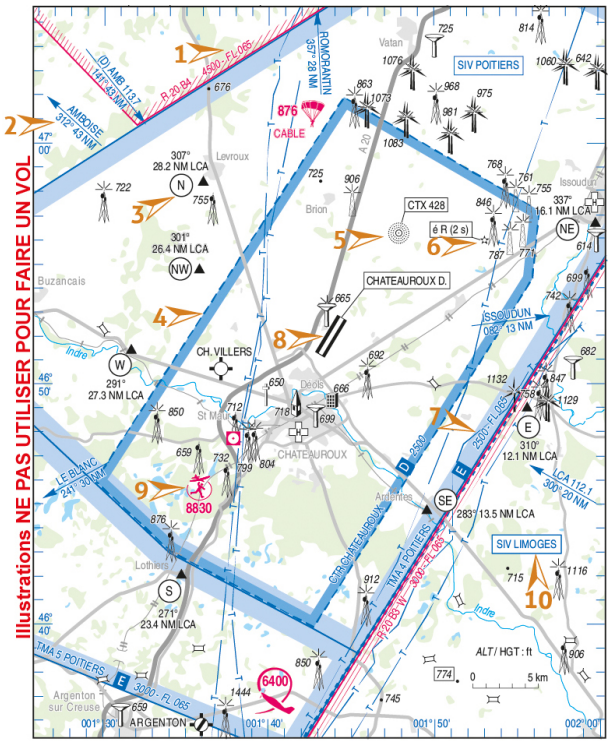


FIGURE 1.1 – Exemple de carte VAC, présentant des données de réglementations de l'aviation civile

1.2 Objectifs de l'étude

À l'heure actuelle, le BEA ne possède pas d'outils permettant de récupérer de manière automatique les données publiées par le SIA, ce qui pose problème, car le processus peut s'avérer long et fastidieux. Par exemple, le fichier XML est publié sous deux formats différents : un dans le format *AIXM* (voir glossaire), et dans un format propre au SIA, que l'on appellera dans la suite XML-SIA. Il faut donc savoir lequel choisir en fonction de son besoin. De plus, ces bases de données sont très

complètes et toutes les informations contenues dans ces dernières ne sont pas forcément utiles dans le cadre des enquêtes du BEA. Il faut donc trier cette base de donnée, ce qui implique de comprendre comment elle est structurée, ce qui n'est pas toujours intuitif. C'est pourquoi l'objectif de ce projet est de créer un outil permettant de récupérer tous les mois le nouvel export du SIA afin de créer une nouvelle base de données géographique contenant uniquement les données utiles au BEA. Cette base de données pourra ensuite être réutilisée par tous les autres outils du bureau d'enquête, y compris le plugin QGIS développé dans le projet : "Création d'un plugin QGIS de visualisation de données aéronautique" proposé par le même commanditaire.

Nous allons donc décomposer cet objectif en deux sous-objectifs :

— comprendre la nature et l'organisation des informations géographiques contenues dans les bases de données XML du SIA et d'y identifier les informations d'intérêt pour le BEA.

— à partir de cette base de données XML, de constituer un SGBD spatial interrogeable par les différents logiciels du BEA. Ce SGBD devra gérer les mises à jour des XML fournis par le SIA, leur archivage et permettre la consultation d'informations géographiques antérieures, contemporaines de la période à laquelle s'est produit l'accident ou l'incident aérien.

Chapitre 2
Analyse fonctionnelle

Cette partie présente les différents solution à disposition afin de répondre à notre besoin, et doit permettre de conclure sur quelle solution est la plus adaptée.

2.1 Présentation générale de notre outil

Les diagrammes suivants décrivent l'outil qui sera dévellopé dans le cadre de ce projet.

2.1.1 Diagramme de cas d'utilisation

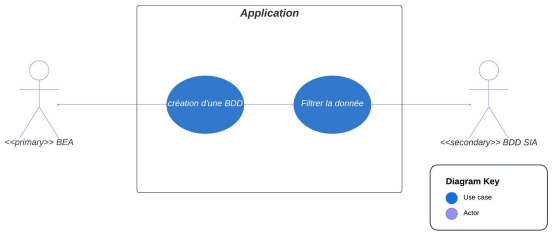


Figure 2.1 – Diagramme de cas d'utilisation de notre application

Notre application aura deux fonctionnalités principales, à savoir trier une donnée en entrée pour ne garder uniquement les informations qui intéressent le BEA. La deuxième fonctionnalité consiste en la création d'une base de donnée géographique dont la structure est détaillée dans la section "Structure de la base de donnée géographique". L'utilisateur principal est donc le BEA en général, c'est-à-dire que cette base de données va être utilisable par de nombreux outils déjà développés par le bureau d'enquête. Pour finir, notre application utilise comme donnée en entrée un fichier XML extérieur (fourni par le SIA), c'est donc l'utilisateur secondaire.

2.1.2 Diagramme de classe

Pour comprendre le diagramme de classe il faut d'abord s'interesser aux données d'intérêt pour le BEA dont voici la liste :

— Les espaces (avec les parties et volumes qui les composent)

— Les waypoints (contenus dans la classe *NavFix*, avec notament leurs positions en plus de leurs noms)

— {Facultatif} Les obstacles (notament leurs noms, leurs positions et leurs hauteurs)

— {Facultatif} Hélistations (notament leurs noms, leurs positions)

— {Facultatif} Les aérodromes (notament leurs noms, leurs positions ainsi que les caractéristiques des pistes)

Notre application doit donc implémenter une méthode de récupération par type de donnée recherchée, que l'on définira dans ce que l'on appelle une toolbox. Ensuite, nous ferons appel à ces dernières dans un script plus général qui enchaînera les méthodes pour construire notre base de données. A l'issue de ce projet, le script Python comprend un algorithme qui permet d'assembler

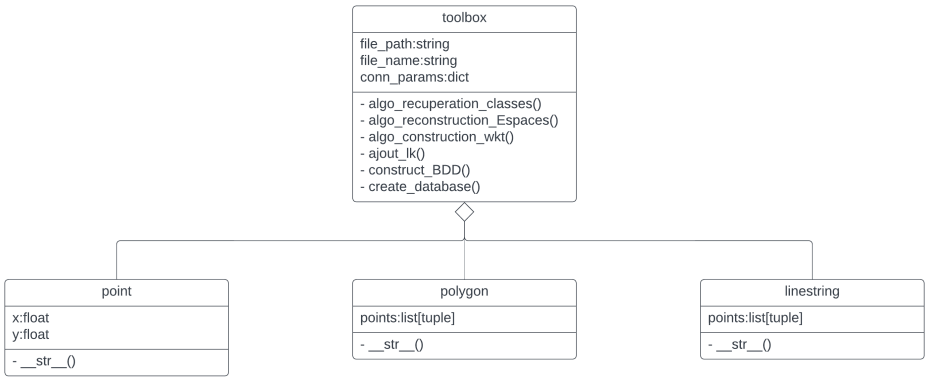


FIGURE 2.2 – Diagramme de classe de notre application

les espaces, comme décrit par la suite. Une fonction est également implémentée pour lire n'importe quelle classe du fichier XML de départ et le convertir en DataFrame. De plus, une fonction permet d'empiler les tableaux de différents types entre eux, comme les espaces et les NavFixs, par exemple. Pour gérer la partie géographique de notre base de données, une fonction construit les requêtes au format WKT, un format compatible avec GeoPandas, afin de les envoyer dans une colonne *geometry* dans PostgreSQL. Cette dernière nécessite la mise en place de classes : point, polyligne et polygone. Selon la classe de l'objet étudié, la requête WKT n'est pas la même. Pour implémenter la classe des polygones dans le script python, nous utilisons la propriété qu'une telle géométrie est considérée comme une polyligne dont le dernier point coïncide avec le premier.

2.1.3 Diagramme d'activité global

Les principales étapes de notre script sont décrites dans ce diagramme d'activité ci-dessous. Tout d'abord, nous commençons par télécharger la donnée du SIA, puis on filtre cette dernière pour ne garder que ce qui interesse le BEA, ce qui revient à enchaîner les méthodes pour récupérer uniquement certaines informations. Ensuite, nous créons une nouvelle base de donnée géographique dont le nom contient la date de publication du rapport du SIA (date de publication de la donnée). Puis nous remplissons cette base de données avec les données qui ont été filtrées dans les étapes précédentes.

2.2 Format XML-SIA ou AIXM ?

Comme mentionné ci-dessus, le SIA livre la donnée dans deux formats différents. Le format propre au service français (XML-SIA), et le format standard d'échange d'information aéronautique (*AIXM*). L'objectif est donc de comprendre et comparer les deux formats pour en déduire le plus adapté à notre besoin.

Pour commencer, il est important de mentionner que le format n'a aucun impact sur les informations contenues par ces fichiers : les deux fichiers contiennent la même donnée. Nous allons donc baser notre choix sur la facilité d'utilisation, tout en tenant compte de la durabilité dans le temps de cet outil. En effet, si les formats d'échange évoluent, cet outil ne fonctionnera plus et devra lui aussi être mis à jour. Il est donc important de tenir compte de l'aspect temporel de ces formats.

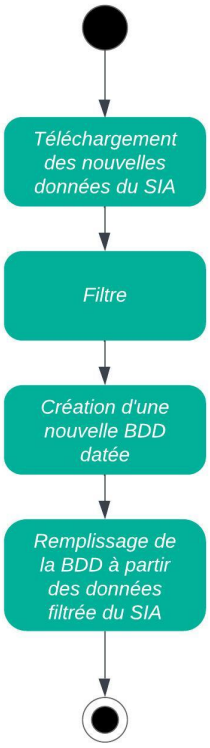


FIGURE 2.3 – Diagramme d'activité simplifié de notre application

Ceci dit, le format AIXM est le format standard Européen, donc il est forcément plus pratique pour échanger les informations à l'international. De plus, cela signifie que notre outil serait aussi compatible avec les autres données européennes. Cependant, le BEA s'intéresse principalement aux accidents et incidents sur le territoire français. Autrement dit, avoir des données sur les autres pays n'est pas une nécessité, bien qu'intéressante.

De plus, le SIA étant une instance gouvernementale française, le format XML-SIA est livré avec une documentation, ainsi qu'avec des guides d'utilisation. Ce qui signifie qu'en téléchargeant le rapport, tout est fourni pour l'inteprétation de ce dernier. Tout ceci rend ce format très pratique à utiliser, et plus facile à comprendre.

Par ailleurs, le SIA dit explicitement que le format XML-SIA n'évolura que très peu dans le temps car c'est un format "historique" et par conséquent, un format utilisé dans un grand nombre d'outils, dont ceux de la DGAC, alors que le format AIXM peut être amené à évoluer selon la version du standard, rendant ainsi notre outil obsolète.

Pour finir, Il est facile de rentrer en contact avec le SIA, ce qui permet un échange privilégié avec ces derniers. Cela permet au BEA d'être plus au courant de la direction que le SIA suit, ainsi qu'obtenir de l'aide en cas de besoin.

Toutes ces raisons nous orientent donc vers l'utilisation du format XML-SIA au détriment du format AIXM.

2.3 Structure de la base de donnée géographique

Maintenant que nous avons choisi le format à utiliser de la donnée livrée par le SIA, il faut s'intéresser à la structure de ce dernier afin de déterminer comment organiser notre base de données géographiques.

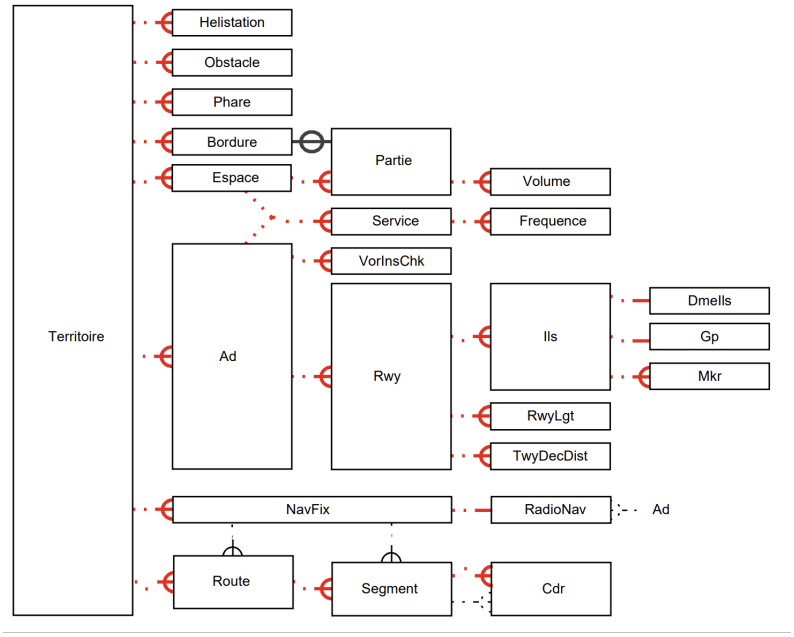


FIGURE 2.4 – Diagramme représentant la structure du format XML-SIA - extrait de la doc "SiaExport V6.0 1/20"

Le modèle est décrit sous la forme entités/associations, une entité représentant une collection d'éléments dotés de caractéristiques communes (attributs), une association représentant les liens ou relations des entités entre elles. Symboles utilisés dans le diagramme :

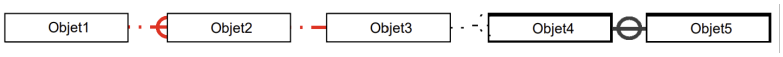


FIGURE 2.5 – Description des associations entre les différentes entités - extrait de la doc "SiaExport V6.0 1/20"

- Un Objet1 est associé à un nombre quelconque d'Objet2, un Objet2 est associé à un Objet1 et un seul.
- Un Objet2 est associé un Objet3 au maximum, un Objet3 est associé à un Objet2 et un seul.
- Un Objet3 est associé à un nombre quelconque d'Objet4, un Objet4 est associé à un Objet3 au maximum
- Lorsque le symbole de la relation est en gras, la relation est identifiante pour l'objet situé à sa droite.
- Cas particulier : un Objet4 est associé à un nombre quelconque d'Objet5, un Objet5 est associé à un nombre quelconque d'Objet4. Dans ce cas, l'un des deux objets contient un attribut complexe (cas de l'attribut Contour dans l'objet Partie)

Maintenant que nous avons une vision globale de la structure du XML-SIA, intéressons nous aux données d'intérêt pour le BEA. Voici un rappel de la liste de ces dernières :

— Les espaces (avec les paties et volumes qui les composent)

— Les waypoints (contenus dans la classe *NavFix*, avec notament leurs positions en plus de leurs noms)

— {Facultatif} Les obstacles (notament leurs noms, leurs positions et leurs hauteurs)

— {Facultatif} Hélistations (notament leurs noms, leurs positions)

— {Facultatif} Les aérodromes (notament leurs noms, leurs positions ainsi que les caractéristiques des pistes)

Commençons par étudier le cas particulier des espaces aériens. Tous sont séparés en plusieurs entités comme décrit ci-dessous :

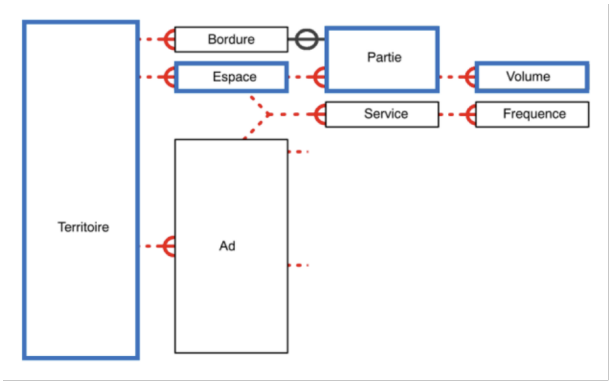


FIGURE 2.6 – Comment retrouver les informations sur les Espaces Aériens en XML-SIA ? - extrait de la doc FAQ

avec :

o <Espace> : informations générales (nom et type)
o <Partie> : parties d'espaces aériens (géométrie et informations)
o <Volume> : caractéristiques associées au découpage vertical des espaces aériens (planchers et plafonds).

Les signes d'association indiquent que chaque espace est décomposé en plusieurs parties et chaque partie en plusieurs volumes. Notre objectif est de récupérer toutes les informations liées aux espaces aériens. Donc pour tous les espaces du XML, on veut récupérer toutes les parties de ces derniers, et pour toutes les parties, on veut récupérer tous les volumes associés. Or dans notre base de donnée géographique (BDDG), il faut que chaque "ligne" soit une entité unique. Donc, premier constat, nous ne pouvons pas organiser notre BDDG avec une ligne par espace aérien. En effet, en faisant cela, étant donné que tous les espaces n'ont pas le même nombre de parties, et donc de volumes, nous ne pourrions pas organiser notre tableau avec une colonne par partie et une colonne par volume.
Pour réaliser cet objectif, il faut parcourir le XML d'une autre manière : on parcourt tous les espaces du XML. Pour chaque entité, on récupère son identifiant, puis on parcourt toutes les entités "parties". Pour chaque entité, on regarde si l'espace auquel la partie appartient est celui que l'on recherche. Si c'est effectivement le cas, alors on récupère l'identifiant de la partie et on parcourt les entités "volume", en regardant la partie auquel le volume appartient. Si le volume coïncide avec celui étudié, alors on crée une nouvelle ligne dans notre BDDG avec toutes les informations du volume, ainsi que toutes les informations de la partie à laquelle il appartient ainsi que toutes les informations de l'espace auquel la partie appartient. Il faut donc une colonne par données contenues dans chaque entité étudiée (décrit par la suite, ici une colonne par données dans la classe espaces, par données dans la classe partie et par données dans la classe volume.). En procédant ainsi, chaque ligne est

unique et on retrouve par simple requête SQL, tous les vecteurs qui composent un même espace, sans perte d'information.
Voici le diagramme d'activité associé à cette fonctionnalité :

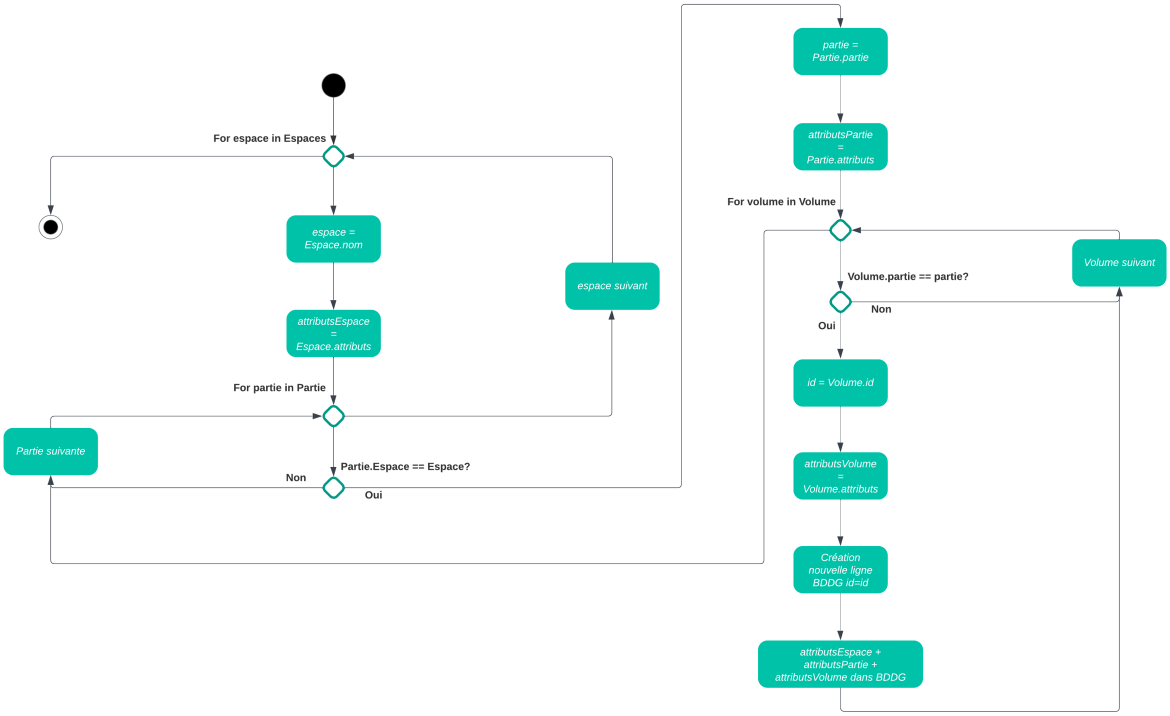


FIGURE 2.7 – diagramme d'activité décrivant les étapes pour la récupération des espaces aériens recherchés

Ceci dit, comme montré ci-dessus, les espaces aériens ne sont pas les seules données qui intéressent le BEA. Prennons l'exemple des *waypoints* (points de passage). Ces derniers sont stockés dans la classe *Navfix*. Or la décomposition n'est pas du tout la même que pour les espaces aériens. Nous allons donc parcourir le XML de manière différente. D'ailleurs, seules les données contenues dans *Navfix* nous intéresse, donc il n'y a même pas de décomposition. Ce qui signifie que nous allons juste parcourir les entités de cette classe et créer une nouvelle ligne par élément. Or, les attributs ne sont plus du tout les mêmes que pour les espaces aériens. Pourtant, nous voulons que tous les éléments soient placés dans la même BDDG. Pour gérer ce problème, nous rajoutons une colonne *type* qui renseigne sur le type d'élément : est ce que c'est un volume, qui décrit un espace aérien ? Ou est-ce que c'est un *Waypoints* ? Ensuite, on rajoute une colonne par attribut contenu dans la classe *Navfix*. Puis on affecte des valeurs *Nul* ou *Nan* pour toutes les colonnes qui concernent les attributs des espaces aériens. Ainsi, peu importe le type de la donnée qui nous intéresse, tous sont dans la même BDDG.

Par ailleurs, pour ajouter une nouvelle classe contenue dans le XML du SIA, il suffit de rajouter une colonne par attribut de cette dernière, attribuer des valeurs nulles pour toutes les lignes qui ne sont pas de ce nouveau type, et enfin implémenter dans le script la méthode de parcours de cette nouvelle classe.

Faisons un exemple simple : imaginons que seuls les *espaces aériens* et les *Waypoints* nous intéressent. Voici les attributs contenus dans chacune de ces classes :

Espace			Représente les espaces aériens de toutes natures.
nom de l'attribut		domaine	définition
cle	Territoire	relation(Territoire)	Désigne le territoire ou se situe l'espace
cle	TypeEspace	enum(TypeEspace)	Type de l'espace
cle	Nom	texte(40)	Nom de l'espace
?	AltrFt	entier(-1000,30000)	Altitude de transition de l'espace (ft AMSL)
?	AdAssocie	relation(Ad)	désigne l'aérodrome principal associé

Partie			Représente les parties d'espaces aériens
nom de l'attribut		domaine	définition
cle	Espace	relation(Espace)	Désigne l'espace auquel appartient la partie
cle	NomPartie	texte(20)	Nom de la partie
!	NumeroPartie	entier(0,32767)	détermine l'ordre de tri par défaut des parties d'un espace
?	NomUsuel	texte(50)	Nom usuel de l'espace
!	Contour	contour	Représente la spécification des limites latérales des espaces
?	Geometrie	geometrie	Représente la géométrie résolue des limites sous forme de polyline (pour SIG, redondant avec Contour)

Volume			Représente les caractéristiques associées au découpage vertical des espaces aériens.
nom de l'attribut		domaine	définition
cle	Partie	relation(Partie)	Désigne la partie à laquelle appartient le volume
cle	Sequence	entier(0,32767)	Complète l'identification du volume par un numéro
!	PlafondRefUnite	enum(AltiCode)	Code indiquant la référence et l'unité du plafond
!	Plafond	entier(-1000,100000)	Valeur du plafond
?	Plafond2	entier(0,100000)	Valeur du plafond en ft ASFC, pouvant surcharger le plafond (la valeur effective étant la plus élevée des deux)
!	PlancherRefUnite	enum(AltiCode)	Code indiquant la référence et l'unité du plancher
!	Plancher	entier(-1000,100000)	Valeur du plancher
?	Plancher2	entier(0,100000)	Valeur du plancher en ft ASFC, pouvant surcharger le plancher (la valeur effective étant la plus élevée des deux)
?	Classe	enum(Classe)	Code indiquant la classe du volume
!	HorCode	enum(Hor)	Code indiquant l'horaire d'activité
?	HorTxt	texte(240)	horaire d'activité (si non codable)
?	Rtba	enum(Rtba)	Indique l'appartenance au réseau basse altitude (zones R uniquement)
?	Activite	texte(240)	Activite
?	Remarque	texte(1000)	Remarque

FIGURE 2.8 – Attributs des classes espaces, partie et volume - extrait de la doc SiaExport V6.0 1/20

NavFix			Représente tous les points d'appui du réseau de routes (aides radio comprises)
nom de l'attribut		domaine	définition
cle	Territoire	relation(Territoire)	Désigne le territoire où se situe le nav_fix
cle	NavType	enum(NavType)	Désigne le type de nav_fix
cle	Ident	texte(10)	Indicatif du nav_fix
!	Wgs84	enum(Wgs84)	précision des coordonnées wgs84
!	Latitude	Latitude	latitude du nav_fix
!	Longitude	Longitude	longitude du nav_fix
!	Geometrie	geometrie	fournit la position sous la forme Latitude,Longitude (redondant)

FIGURE 2.9 – Attributs de la classe Navfix - extrait de la doc SiaExport V6.0 1/20

Voici la forme que prendra la BDDG résultante :

Id de l'élément		Attributs de la classe espace			Attributs de la classe partie			Attributs de la classe Volume			Attributs de la classe NavFix		
id	type	TypeEspace	Nom	[...]	Espace	Partie	[...]	Partie	Plafond	[...]	Long	Lat	[...]
309660	Volume	CTL	303931		303931	301627		301627	7700		Nan	Nan	
309708	Volume	CTL	303931		303931	301342		301342	5000		Nan	Nan	
309489	Volume	CTL	303641		303641	301302		301302	12000		Nan	Nan	
[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]
18	NavFix	Nan	Nan		Nan	Nan		Nan	Nan		-3.602481	48.33265	
110	NavFix	Nan	Nan		Nan	Nan		Nan	Nan		4.1423	44.066431	
262	NavFix	Nan	Nan		Nan	Nan		Nan	Nan		0.406961	46.703939	
[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]

FIGURE 2.10 – Exemple simple d'une BDDG issue du XML-SIA

Pour finir, l'objectif est de lancer ce script pour chaque nouveau jeu de données du SIA, tout en gardant en archive les BDDG précédentes. Étant donné que les rapports sont publiés tous les mois, le plus simple est donc de lancer le script tous les mois, et de créer une nouvelle table avec comme nom la date de publication du rapport. Une autre solution était de concaténer tous les rapports dans une même BDDG, or cette dernière deviendrait trop grosse donc on préférera le choix d'une BDDG par rapport.

Étude et Réalisation technique

Chapitre 3

3.1 Choix des logiciels et bibliothèques utilisés

3.1.1 Logiciel pour BDD

Figure 3.1 –

Pour créer une base de données géographique, plusieurs logiciels sont disponibles. Tout d'abord, nous voulons pouvoir utiliser cette dernière en faisant des requêtes spatiales, donc on élimine les bases de données n'utilisant pas de gestionnaires dédiés à une telle utilisation. Cependant, plusieurs choix s'offrent encore à nous tels que *PostgreSQL*, *MySQL*, ou encore *SQLite*. Cependant, après réflexion avec mon commanditaire Jean-François, nous avons choisi *PostgreSQL* car ce dernier est parfaitement compatible avec l'environnement informatique du BEA, et intègre un module spatial très complet : *PostGIS*. En outre, l'interface utilisateur peut être gérée par *pgAdmin* sur une page Web, ne nécessitant ainsi aucune installation particulière sur la machine qui stockera nos bases de données. Pour finir, ce dernier est aussi supporté par QGIS en natif, chose très pratique pour le projet du plugin QGIS dévellopé en parallèle.

3.1.2 Logiciel pour le script

Pour implémenter les différentes méthodes de récupération de données, aucun logiciel n'est imposé et étant un nouveau projet, je n'ai pas de structure de code à reprendre. Je suis donc libre de choisir le langage utilisé dans le cadre de ce projet du moment qu'il répond aux attentes. J'ai donc choisi de coder sur python.

3.1.3 Bibliothèques utilisées

Lors de ce projet, les différentes étapes réalisées ont requis l'utilisation d'un certain nombre de bibliothèques :

- numpy : permet l'intégration de `nan` dans mon tableau tout en étant compatible avec pandas.

- pandas : permet de gérer des bases de données et tableaux.
- geopandas : permet de gérer l'aspect géographique de notre base de données.
- ast : permet d'interpréter un texte en tant que dictionnaire.
- psycopg2 : permet de connecter mon script Python à PostgreSQL.
- SQLAlchemy : permet de faire des requêtes SQL dans Python.
- geoalchemy2 : permet de faire des requêtes SQL géographiques dans Python.

3.2 Présentation du résultat

3.2.1 Explication de la structure du code

Dans un premier temps, pour que le script fonctionne, il faut que l'utilisateur renseigne ses paramètres de connexion à pgAdmin ainsi que le fichier XML au format du SIA, avec son chemin. Ensuite, nous exécutons (dans le main) la fonction *create_database()*. Cette dernière commence par parser le fichier, puis construit un DataFrame avec la fonction *construct_BDD()*. À l'heure actuelle, cette dernière crée un tableau unique composé de tous les NavFixs du fichier, ainsi que de tous les espaces reconstruits avec leurs parties et leurs volumes, comme détaillé précédemment. À ce tableau, nous ajoutons une colonne *'lk'*, qui permet d'identifier un objet de manière unique et parlante. Cette dernière est particulièrement utile pour le plugin QGIS développé par ma camarade. Ensuite, nous ajoutons une colonne *'wkt'* servant de clé dans notre BDDG. Enfin, la fonction *create_database()* établit la connexion avec PostgreSQL et crée une nouvelle table construite avec la date de publication du rapport.

3.2.2 Présentation du résultat

Avant le développement de ce projet, le BEA utilisait une autre alternative proposée par le SIA. En effet, ces derniers publient aussi des cartes de navigation aérienne contenant une grande partie des informations contenues dans les fichiers XML. Le tout était de récupérer ces cartes (en réalité des images) qu'on venait référencer pour ensuite pouvoir superposer avec des données extérieures. Cependant, la quantité d'informations étant très grande, le résultat n'était pas très lisible et simple à comprendre.

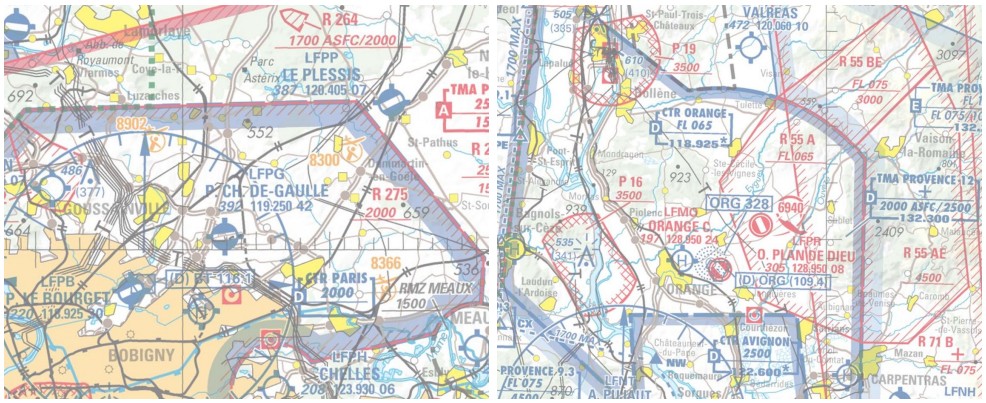


FIGURE 3.2 – Exemple de cartes proposé par le SIA, alternative aux fichiers xml.

Et donc, avec ce nouvel outil, il est désormais possible de filtrer la donnée et n'afficher que ce qui nous intéresse. On gagne alors grandement en clarté et en lisibilité.

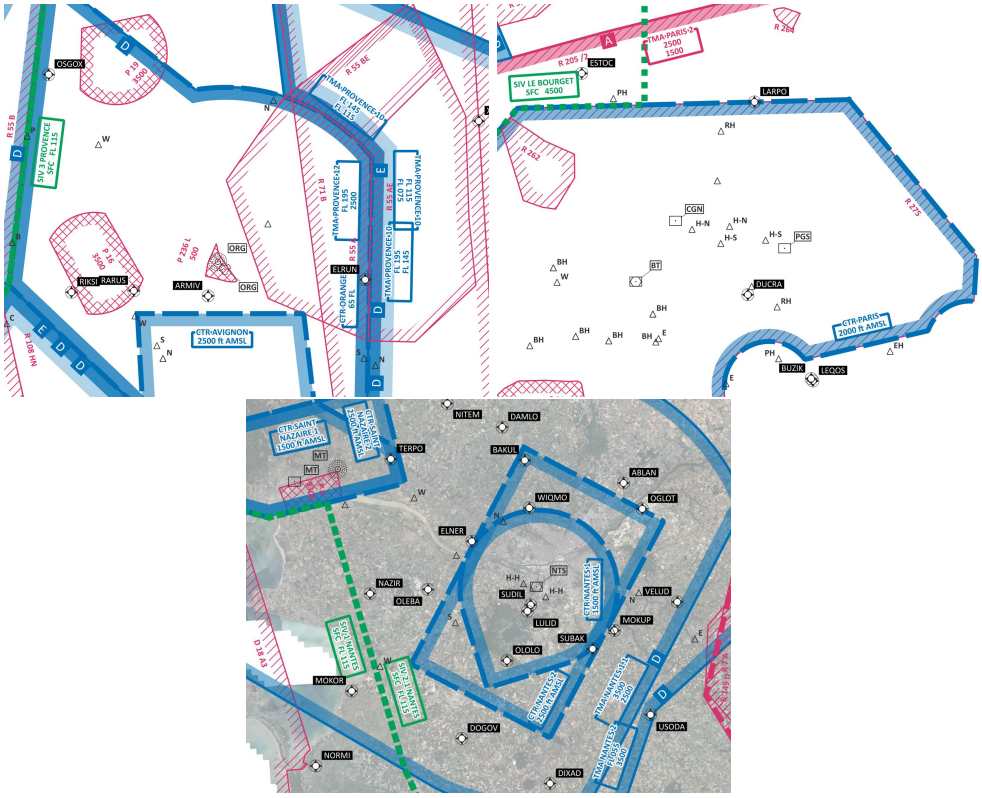


FIGURE 3.3 – Exemple de cas d'utilisation de la BDDG construite par l'outil développé dans ce projet.

3.3 Difficultés rencontrés

Durant cette phase de développement, j'ai rencontré trois problèmes majeurs : en premier lieu, la concaténation des tableaux ne fonctionnait pas de la manière attendue. En effet, les index étaient étranges, ce qui entraînait des problèmes lors de la reconstruction. La solution a été de réinitialiser les index à chaque étape. Ce problème a occasionné une perte de temps significative et a retardé la création de la base de données de test pour le plugin QGIS. Après concertation avec ma camarade, elle s'est concentré sur le développement de l'IHM pour continuer à avancer en parallèle.

Ensuite, j'ai envisagé de créer la base de données "test" manuellement, mais cela s'est révélé être trop complexe et hasardeux en raison de la gestion imprévisible des types de colonnes tant que je n'avais pas implémenté la fonction permettant d'extraire une base de données dans PostgreSQL.

Enfin, lors du développement de l'algorithme permettant de reconstruire les aéroports, un autre problème s'est présenté. En concaténant les différentes classes, plusieurs colonnes portaient le même nom (geometrie), ce qui empêchait pandas d'exécuter correctement la concaténation des différents tableaux. Par manque de temps, ce problème n'a pas été résolu, et c'est pourquoi l'algorithme en question n'a pas été implémenté dans la version finale du script (bien qu'il soit présent dans le fichier *script dev.py*).

3.4 Etat d'avancement et amélioration possible

Dans le script final seuls les espaces, les NavFixs et les obstacles sont construits et ajoutés à la BDDG PostgreSQL. L'algorithme permettant d'ajouter les aérodromes est bien avancé mais n'est pas encore fonctionnel. Ainsi, la première piste d'amélioration consiste à résoudre le problème décrit dans la partie précédente afin d'ajouter les aérodromes. Les hélistations sont également à ajouter. De plus, une fonctionnalité automatisant le téléchargement du rapport du SIA serait un ajout intéressant.

Chapitre 4
Gestion de projet

4.1 GitHub

Tous les documents et les scripts produits durant ce projet ont été mis sur un *repository* GitHub créé à l'occasion, qui sera commun avec le projet de ma camarade intitulé "Création d'un plugin QGIS de visualisation de données aéronautiques" qui utilise directement la base de donnée créée dans mon projet. Pour cela, nous avons travaillé chacun sur une branche dédiée. Cette organisation nous permet d'avoir accès au travail de l'autre et facilite les échange et le partages de doccuments. De plus nos commanditaires ont aussi accès à ce GitHub leur permettant de suivre l'évolution de nos recherche et ainsi nous guider si nous nous éloignons de leurs attentes.

4.2 Planning

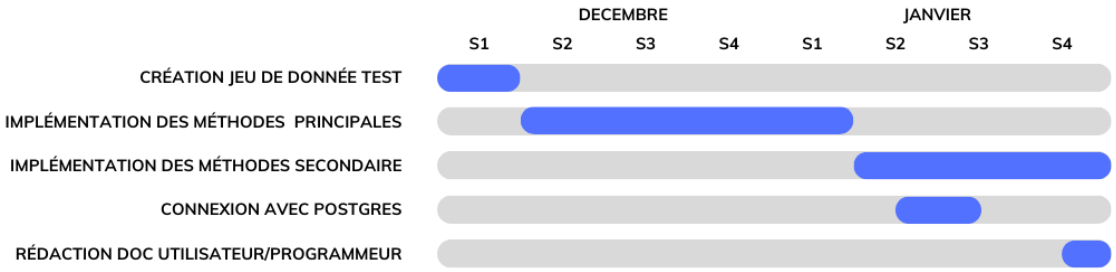


Figure 4.1 – Planning suivi durant la phase de développement

Nous pouvons décomposer ce projet en plusieurs objectifs à réaliser. Le premier est la construction d'un jeu de données test permettant à ma camarade de tester son code sur une base de données "type". Cela a demandé de commencer par développer les méthodes principales de récupération de la donnée pour construire ce jeu de données. Cette étape a été plus longue que prévu, d'autant que la connexion à PostgreSQL m'a également demandé du temps. En parallèle, j'ai commencé à développer les méthodes secondaires qui n'ont pas été finies. Cette partie ne devait pas prendre autant de temps car à l'issue de la fonction principale, les méthodes de parcours et de reconstruction sont maîtrisées. Cependant, de nouveaux problèmes ont ralenti leur implémentation. Pour finir, la rédaction de toutes les documentations et du rapport a clôturé ce projet.

Table des figures

1.1 Exemple de carte VAC, présentant des données de réglementations de l'aviation civile 5

2.1 Diagramme de cas d'utilisation de notre application 7
2.2 Diagramme de classe de notre application 8
2.3 Diagramme d'activité simplifié de notre application 9
2.4 Diagramme représentant la structure du format XML-SIA - extrait de la doc "SiaExport V6.0 1/20" 10
2.5 Description des associations entre les différentes entités - extrait de la doc "SiaExport V6.0 1/20" 10
2.6 Comment retrouver les informations sur les Espaces Aériens en XML-SIA ? - extrait de la doc FAQ 11
2.7 diagramme d'activité décrivant les étapes pour la récupération des espaces aériens recherchés 12
2.8 Attributs des classes espaces, partie et volume - extrait de la doc SiaExport V6.0 1/20 13
2.9 Attributs de la classe Navfix - extrait de la doc SiaExport V6.0 1/20 13
2.10 Exemple simple d'une BDDG issue du XML-SIA 13

3.1 15
3.2 Exemple de cartes proposé par le SIA, alternative aux fichiers xml. 16
3.3 Exemple de cas d'utilisation de la BDDG construite par l'outil développé dans ce projet. 17

4.1 Planning suivi durant la phase de développement 19