École Nationale des Sciences Géographiques

ENSG

Master spécialisé PPMD

Rapport de programmation du projet informatique

Création d'un plugin Qgis de visualisation de données aéronautiques

Élève :
Louise Redlinger

Encadrants :
Jean-François Villeforceix
Darian Motamed

29 janvier 2024

Table des matières

Introduction

Le projet informatique s'inscrit dans le cadre de ma troisième année dans le cursus des ingénieurs diplomés de l'ENSG, École Nationale des Sciences Géographiques, en master spécialisé PPMD, Photogrammétrie, Positionnement et Mesure de Déformation. Ce projet se scinde en deux phases disctintes dans le temps, une phase d'analyse informatique et une phase de développement.

La phase de développement a pour but de concrétiser les attentes du sujet et trouver des solutions techniques aux fonctionnalités requises par le commanditaire tant en assurant une gestion pérenne du projet par l'écriture d'une documentation à l'usage des futurs développeurs et utilisateurs.

Ainsi, le présent rapport de programmation fait suite au précédent rapport d'analyse du sujet qui m'a été proposé par Jean-François Villeforceix et Darian Motamed, deux enquêteurs du BEA, Bureau d'Étude et d'Analyse pour la sécurité de l'aviation civile : soit la création d'un plugin Qgis de visualisation de données aéronautiques. Je vous y exposerai donc comment s'est déroulé l'implémentation du plugin mais aussi en quoi il a changé l'analyse informatique intialement faite.

1 Contexte et Objectifs

1.1 Le BEA et le SIA

Le Bureau d'enquêtes et d'analyses pour la sécurité de l'aviation civile (BEA) réalise des enquêtes de sécurité sur des accidents survenus au sein de l'aviation civile en France.

Lors de ces investigations, le BEA est ammené à visualiser les données publiées par le SIA, Service de l'Information Aéronautique qui comprennent toutes les réglementations de l'aviation civile en vigueur le jour de l'incident. Pour le moment, l'accès et la visualisation de ces données se fait manuellement, ce qui rend les procédures d'enquête assez longues.

Pour répondre à cette problématique, le BEA a proposé deux sujets complémentaires : d'une part la création automatisée d'une base de données géographiques à partir des données du SIA, et d'autre part, la conception d'un plugin QGIS chargé de récupérer les objets de cette base de données et des les afficher dans l'environnement QGIS. Mon projet s'articule donc plutôt autour du développement du plugin Qgis de visualisation des informations extraites de cette nouvelle base de données qui elle, est le coeur du projet de Louis Steinmetz, un de mes camarades.

L'aboutissement de ces deux projets améliorera significativement le bon déroulé des enquêtes faites au sein du BEA.

1.2 Objectifs de l'étude

Les données qui servent aux investigations du BEA sont fournies tous les mois par le SIA. Ces données du SIA ont été extraites, triées et organisées dans le cadre du projet réalisé en parallèles par Louis Steinmetz, "Extraction d'informations géographiques fournies par le Service d'Information Aéronautique (SIA) pour la création d'une base de données géographique". L'objectif principal de ce projet était donc de maintenir à jour mensuellement une nouvelle base de données avec les nouvelles réglementations.
Quant au présent projet, il consistait à implémenter un plugin Qgis doté d'une interface graphique connectée à cette nouvelle base de données géographiques. Cette interface graphique permet à l'utilisateur de réaliser des requêtes spatiales et temporelles dans la base de données, tout en permettant de sélectionner parmis les données récupérées celles que l'on souhaite visualiser dans l'environnement QGIS. En résumé, ce projet a permis l'implémentation d'un plugin QGIS qui permettra à terme, par le biais d'une interface graphique, de visualiser dans l'environnement QGIS une représentation des espaces aériens et autres éléments importants dans le déroulé d'une enquête au sein du BEA.

2 Étude technique

L'étude technique de ce projet réside essentiellement dans la compréhension de la structure de la base de données associée au plugin Qgis. Le rapport de Louis Steinmetz fournie une explication détaillée à cet égard. Cependant, voici ci-dessous, à titre d'illustration la structure de la base de données géographiques en figure 1.

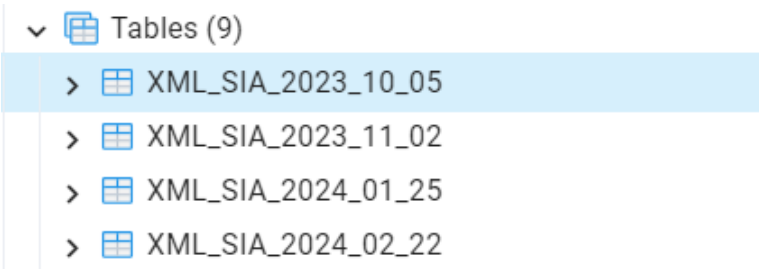

FIGURE 1 – Noms des tables de la BDDG au format XML_SIA_aaaa_mm_jj

Cette base de données géographiques consiste en un ensemble de tableaux dont la nomenclature est illustrée sur la figure 1. Un nouveau tableau est intégré à la base de données chaque mois. Ainsi chaque nom de table commence par "XML_SIA_" suivi de sa date de publication au format "aaaa_mm_jj". Cette information prend tout son sens car, depuis l'interface graphique, l'utilisateur peut rentrer la date de l'accident de l'enquête en cours pour effectuer une requête temporelle dans la base de données. Ainsi, en fournissant la date de l'investigation en cours, la table avec la date antérieure la plus proche lui sera fournie car il est capital de se référer aux réglementations aéronautiques qui étaient en vigueur le jour de l'accident pour la bonne compréhension des événements.

	lk text
1	[LF][Aer 9657-2][.][10]
2	[LF][Aer 9657-1][.][10]
3	[LF][Aer 9636][.][10]
4	[LF][Aer 9627][.][10]
5	[NT][SUR 050][.][10]
6	[NT][SUR 040][.][10]
7	[NT][SUR 030][.][10]
8	[LF][Aer 9657-3][.][10]
9	[LF][Aer 9561][.][10]

FIGURE 2 – Identifiant "lk" unique à chaque objet de la table

En tant que développeur du plugin QGIS, ce qui nous intéresse dans cette base de données sont les colonnes nommées par Louis "lk" et "geometry" dont vous avez un aperçu en figure 2 et 3. L'attribut "lk" est un identifiant unique à chaque objet qui compose les tables de la BDD et sa nomenclature est parlante aux enquêteurs du BEA. Quant à l'attribut "geometry" il fournit au plugin les informations 2D des géométries selon si elles sont des points, des lignes ou des polygones.

	geometry geometry
1	0101000020E6100000295C8FC2F528064024F25D4A5DCE4640
2	0101000020E6100000D2A8C0C9365006405000C5C892CF4640
3	0101000020E6100000780DFAD2DB1F0A406E88F19A57B34640
4	0101000020E610000059880E8123010F4073F22213F0ED4540
5	0103000020E61000000100000028000000027840EBA048361C0DCA0F65B3BD121C0F488D1730B8361C0FB3E1C2444D121C0BB438A01128361C0D63A71395ED121C0828B153518836
6	0103000020E61000000100000028000000 8E20956247EE62C0AFB14B546FB930C019A9F7544EEE62C01E8997A773B930C0C34A051555EE62C00C0742B280B930C06C79E57A5BEE62
7	0103000020E61000000100000028000000B0E2546BE1B262C014200A664C8D31C018080264E8B262C083F755B9508D31C09225732CEFB262C0717500C45D8D31C017F19D98F5B262
8	0101000020E61000006D3656629E150640B6662B2FF9CD4640

FIGURE 3 – Colonne "geometry", informations 2D des objets de la table

En effet dans un premier temps, l'utilisateur effectue depuis l'interface, une requête spatiale en définissant manuellement une enveloppe spatiale dans QGIS et une requête temporelle en spécifiant la date de l'évènement sur lequel il travaille.

À l'aide de ces informations, l'interface va pouvoir afficher à l'investigateur l'ensemble des identifiants "lk" des objets de la table à la date spécifiée, dont la géométrie "geometry" est comprise dans l'emprise spatiale définie.

Et enfin, l'utilisateur peut cocher les objets qui l'intéressent, établissant ainsi une liste d'attributs "lk" dont le plugin va se servir pour afficher les différentes géométries correspondantes dans l'environnement QGIS.

3 Analyse fonctionnelle : mise à jour

Dans cette partie, vous trouverez la mise à jour des outils d'analyse fonctionnelle qui décrivent les caractéristiques du plugin Qgis développé. Depuis la fin de la phase d'analyse, le plugin QGIS nommé "Aeronautical Data Visualizer" a été développé et ses fonctionnalités ont évoluées. Cette section contient aussi un visuel de l'interface graphique liée au plugin QGIS pour faciliter la compréhension de son fonctionnement.

3.1 Cas d'utilisation

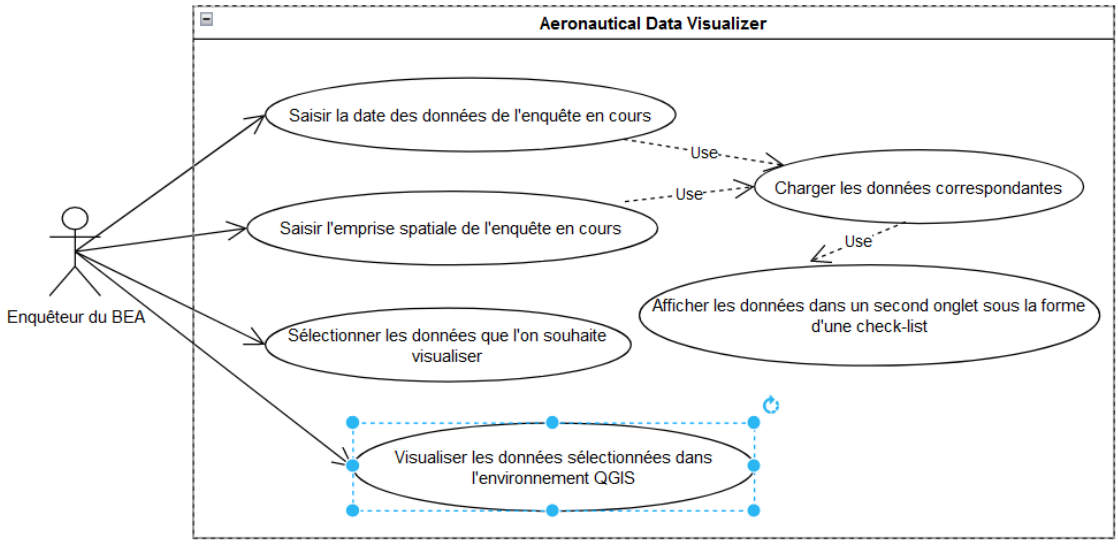

FIGURE 4 – Diagramme des cas d'utilisation

Le plugin Aeronautical Data Visualizer a été développé à des fins d'utilisation par des enquêteurs du BEA. L'utilisateur, en tant qu'investigateur du BEA peut depuis l'interface graphique saisir la date de l'incident sur lequel il travaille mais il peut aussi saisir l'emprise spatiale de l'enquête en cours. La requête spatiale est une des nouvelles fonctionnalités que nous avons décidé de mettre en place afin d'optimiser la taille des requêtes effectuées dans la base de données. Cette action permettra au plugin de charger la base de données correspondante en interne. L'utilisateur pourra ensuite sélectionner parmis les éléments récupérés, ceux qu'il souhaite afficher, puis les visualiser dans l'environnement QGIS.

3.2 L'interface graphique

Pour améliorer notre compréhension du fonctionnement du plugin Qgis, voici à quoi ressemble l'interface graphique du plugin Aeronautical Data Visualizer. Initialement il était prévu que l'interface graphique se compose en deux fenêtres distinctes. Toutefois, il a été jugé plus convivial en terme de programmation et d'usage de penser l'interface graphique sous la forme d'une unique fenêtre avec deux onglets : l'un pour le filtre spatial et temporel appliqué aux requêtes dans la base de données et l'autre pour la sélection des données correspondantes que l'on souhaite visualiser.

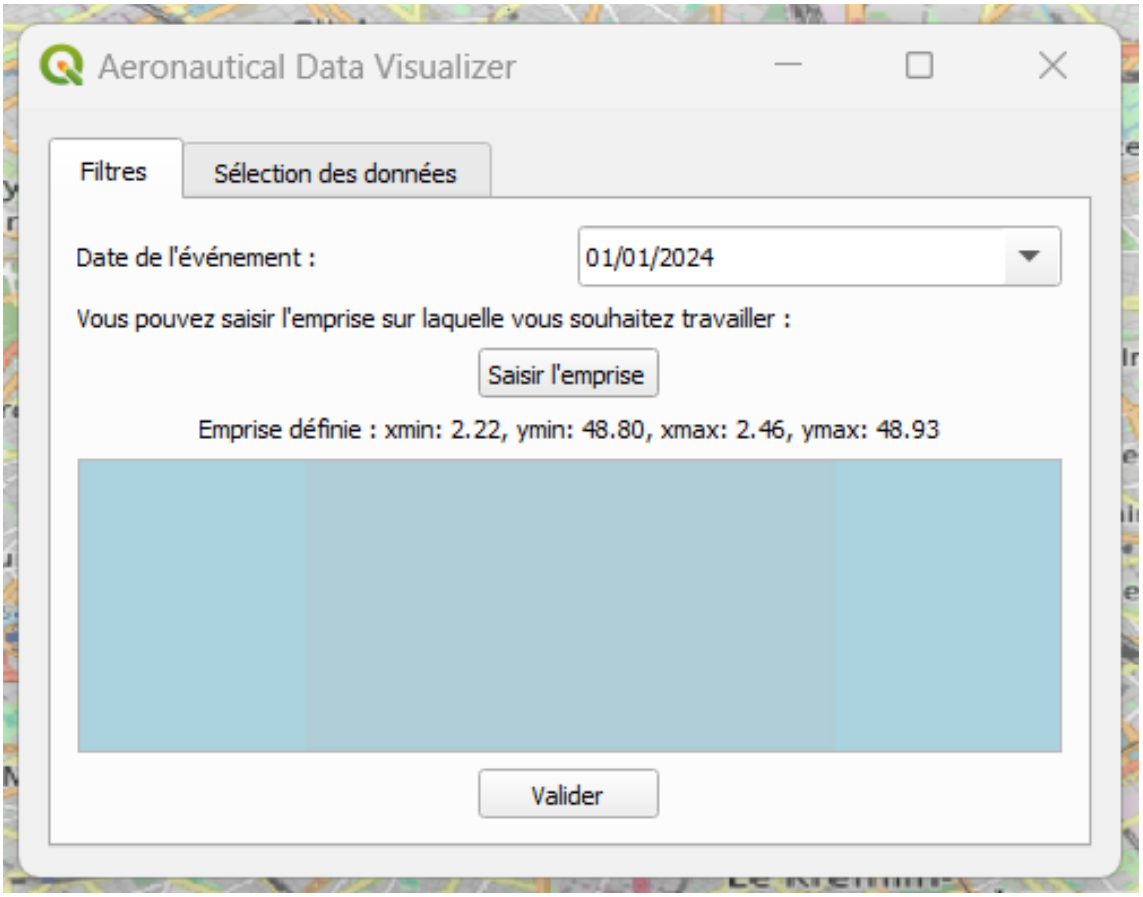

FIGURE 5 – Onglet de filtrage spatial et temporel

L'onglet que vous pouvez observer en figure 5 est la première interaction de l'utilisateur avec l'interface du plugin. Il peut y rentrer la date de l'incident sur lequel il travaille au format "jj/mm/aaaa" mais il peut aussi saisir l'emprise spatiale sur laquelle il souhaite travailler en cliquant sur le bouton "Saisir l'emprise". Il confirme ensuite sa saisie en appuyant sur le bouton "Valider". Cette action permettra en interne de charger l'ensemble des publications aéronautiques en vigueur à cette date et dans la zone géographique définie depuis la base de données et d'en afficher les identifiants des éléments dans un second onglet qui est illustré en figure 6.

Ce second onglet présente sous forme de menu déroulant, la check-list des données récupérées. Les données disponibles qui devaient initialement être triées par type (par exemple un espace ou un waypoint) sont en fait reconnaissables à l'aide d'une nomenclature connue des investigateurs et donc non triées. L'utilisateur peut donc cocher les

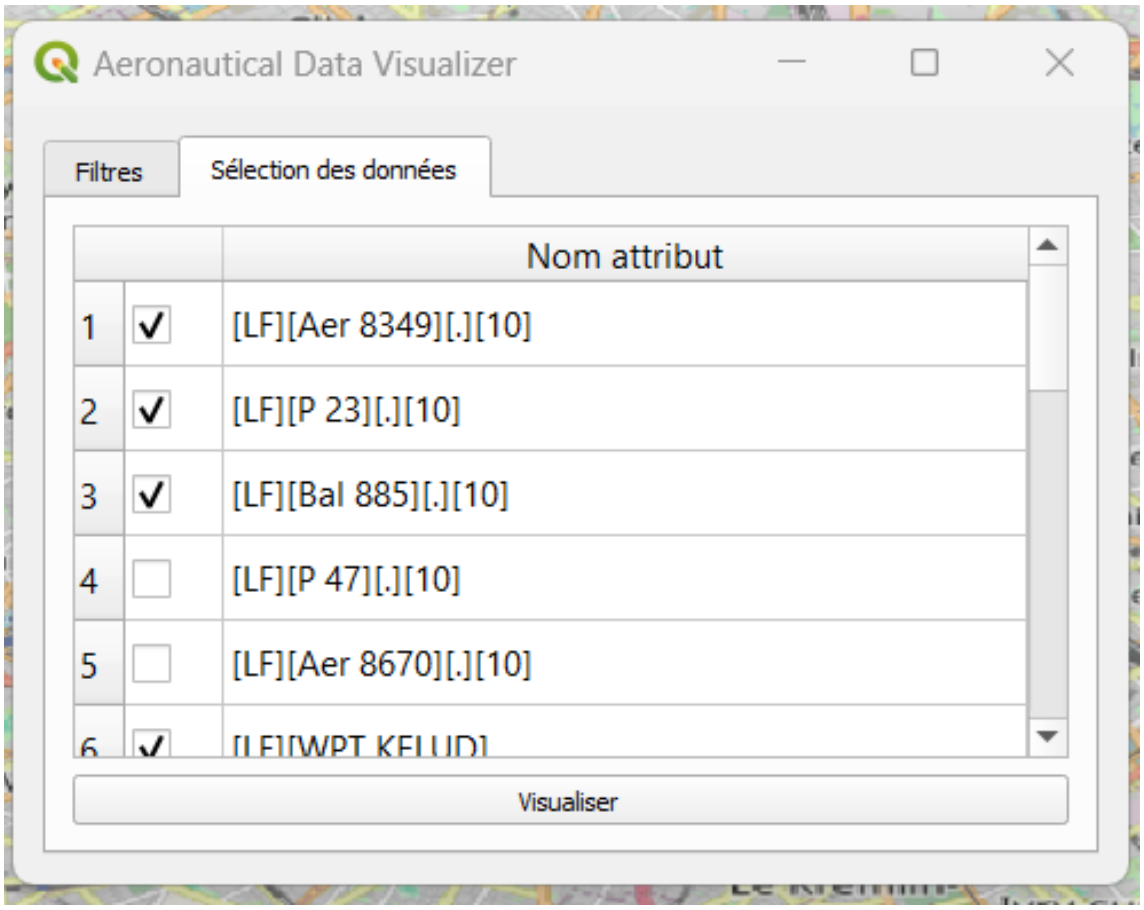

FIGURE 6 – Onglet de sélection des données à visualiser

éléments qui l'intéressent et les visualiser dans l'environnement QGIS en cliquant sur le bouton "Visualiser".

Comme perspectives d'amélioration, il est envisageable que tous les attributs de l'élément sélectionné apparaissent dans un espace dédié de la fenêtre afin de s'assurer des caractéristiques de l'élément que l'on souhaite afficher. Il avait été aussi pensé de mettre une barre de recherche en tête du menu déroulant pour faciliter la sélection des données. Cepednant, le développement de ces otpions facultatives aurait nécessité un temps supplémentaire, ce qui n'était pas envisageable dans le cadre de la durée de ce projet.

3.3 Diagramme de classes mis à jour

Le diagramme de classes en figure 7 synthétise les classes implémentées lors du développement du plugin QGIS. Le plugin QGIS constitue donc la classe AeroDataVisualizer et il est doté d'une interface graphique iface de type QgsInterface ainsi que d'une fonction run(self) qui s'occupe de lancer le plugin au sein de QGIS, cette classe est essentielle mais n'a pas été au coeur de la programmation puisque l'outil Plugin Builder fournit une version déjà oppérationnelle. En revanche la phase de programmation s'est articulée autour de la classe AeroDataVisualizerDialog qui hérite en quelques sortes de la classe précédente puisqu'elle est aussi dotée d'une interface graphique. Cette interface a deux onglets, aussi dits TabWidget, de type QWidget.

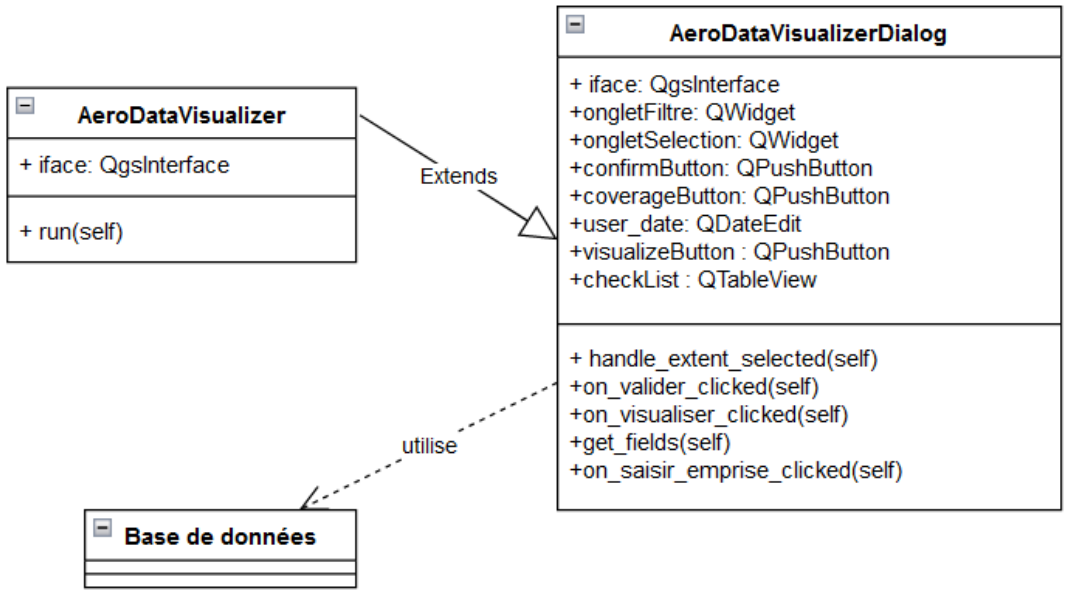

FIGURE 7 – Diagramme de classes mis à jour

Dans le premier onglet que j'ai nommé "ongletFiltre" dans la figure 7 on retrouve essentiellement les éléments suivants : le bouton confirmButton et le bouton coverageButton de types QPushButton qui correspondent respectiverment aux boutons "Valider" et "Saisir l'emprise" de l'interface et aussi l'élément "user_date" de type QDateEdit qui correspond au champs de date que l'utilisateur peut saisir depuis l'interface. Ces éléments intéragissent avec le plugin, QGIS et la base de données par le biais de fonctions suivantes donc nous allons résumer le rôle : on note les méthodes handle_extent_selected(), on_valider_clicked() et on_saisir_emprise_clicked().

La méthode handle_extent_selected() gère la sélection de l'étendue sur la carte QGIS par l'utilisateur. Elle récupère l'étendue sélectionnée par l'utilisateur, s'assure que ces informations soient dans le système de coordonnées WGS84, et affiche les coordonnées du coin inférieur gauche et supérieur droit de l'emprise correspondante dans une étiquette visible depuis l'interface. De plus, elle met à jour une prévisualisation de la carte en affichant l'étendue sélectionnée dans l'interface. Quant à on_saisir_emprise_clicked(), elle gère l'événement de "clic" sur le bouton "Saisir l'emprise". Cette méthode active l'outil de sélection d'emprise sur la carte puis fait appel à la méthode précédente handle_extent_selected().

Enfin on a la méthode on_valider_clicked() s'occupe de l'événement de clic sur le bouton "Valider". Cette méthode récupère la date saisie par l'utilisateur et effectue une première requête à la base de données pour obtenir le nom de la table correspondante avec la date antérieure la plus proche. En fonction de l'étendue spatiale sélectionnée, une requête SQL est construite pour extraire les identifiants 'lk' abordés précédemment des objets dans la base de données. Les identifiants 'lk' sont ensuite affichés dans l'objet checkList qui se trouve dans le second onglet nommé ici "ongletSelection".

Une fois ce second onglet ouvert par la méthode que nous venons de détailler. On peut y retrouver les éléments suivants : la liste avec des éléments qui se cochent ainsi que le bouton "Visualiser" qui correspond au visualizeButton de type QPushButton.

La méthode get_fields() s'occupe de récupérer les champs sélectionnés par l'utilisateur dans la liste. Elle retourne une liste des noms des champs qui ont été cochés. Et finalement, la gestion de l'événement de clic sur le bouton "Visualiser" est régie par la méthode on_visualiser_clicked(). Elle récupère les champs sélectionnés par l'utilisateur grâce à la fonction get_fields(), puis crée des couches vectorielles dans QGIS à partir des données correspondantes dans la base de données. Les couches sont organisées dans un groupe de couches portant le même nom de la table de données dont elles sont issues.

3.4 Diagramme d'activités

Ici, en figure 8 on retrouve le diagramme d'activités qui décrit le fonctionnement global du plugin Qgis. On distingue également quelles actions se font d'un onglet à l'autre.

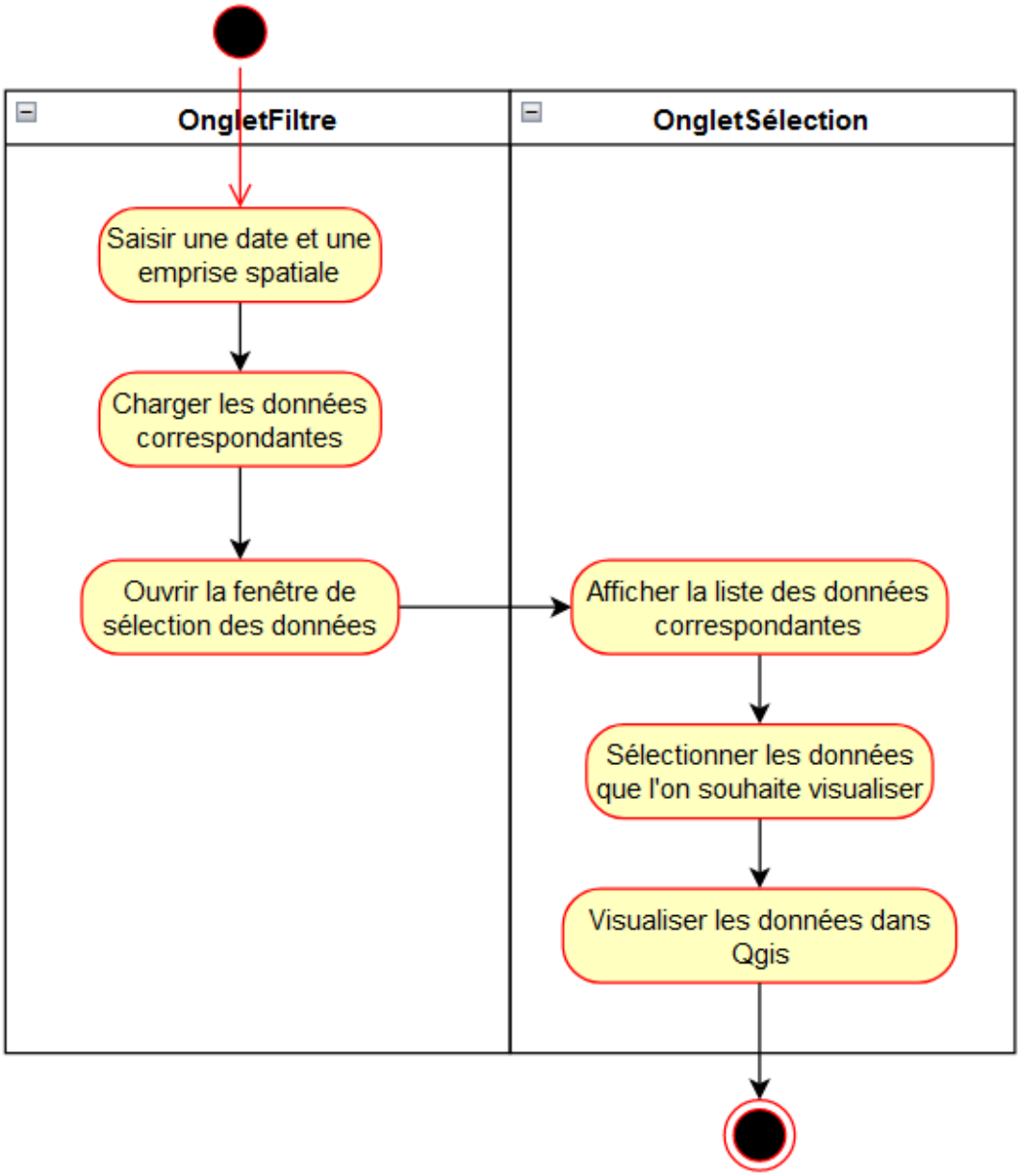

FIGURE 8 – Diagramme d'activités mis à jour

Par rapport à la fin de la phase d'analyse le diagramme d'activités est globalement resté le même dans l'enchaînement des activités du plugin si ce n'est que l'on a rajouté la saisie de l'emprise spatiale et que les intéractions se font d'un onglet à l'autre plutôt qu'entre deux fenêtres.

4 Développement

Les fonctionnalités qui sont au coeur du plugin viennent d'être amplement détaillées par une analyse fonctionnelle plus fournie qu'en début de projet. Je propose donc que l'on aborde dans cette partie la structure du code qui a été livré au BEA.

4.1 Architecture

Comme nous l'avons vu au début de ce présent rapport, ce projet est complémentaire d'un second projet qui s'articulait autour de la mise en place d'une base de données géographiques. Un dépôt git à donc était ouvert afin d'assurer la pérennité et le bon déroulement simultané des deux projets.

Les deux projets étant corrélés, afin de pouvoir utiliser le plugin développé il faut en amont pouvoir exécuter le scrpit de Louis qui intéragit avec l'environnement du gestionnaire de données PostgreSQL. Toutes les informations à cet effet sont fournies dans son rapport. Quant au plugin réalisé dans le cadre de ce présent projet, il s'organise autour d'un certains nombre de fichiers que dont vous pouvez avoir un aperçu en figure 9.

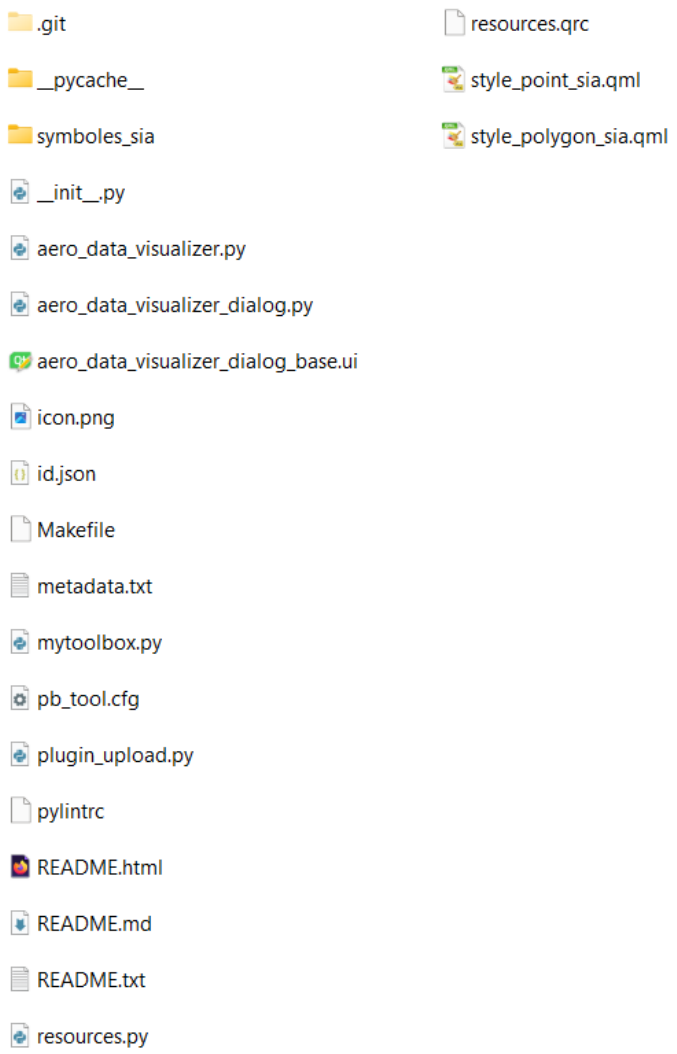

FIGURE 9 – Fichiers qui constituent le plugin AeroDataVisualizer

La pluspart des éléments sont déjà inclus lors de l'éxécution du plugin PluginBuilder qui fournit une base solide pour le développement d'un plugin QGIS. Les modifications ont essentiellement été faites dans les fichiers aero_data_visualizer.py et aero_data_visualizer_dialog.py qui ont déjà été détaillées plus haut et ainsi que dans la documentation développeur du code. En revanche je vous propose de détailler le éléments propres à ce plugin qui ont été ajoutés.

4.2 Boîte à outils

Le fichier mytoolbox.py contient une méthode externe à la classe AeroDataVisualizerDialog. Il s'agit de la méthode que l'on a nommé nearest_table_date(user_date , db_dates_list) qui prend en argument la date rentrée par l'utilisateurn et la liste de toutes les dates de mise à jour de la BDD et renvoie la date de mise à jour antiérieure la plus proche de la date saisie par l'enquêteur.

4.3 Connexion à la base de données

La connexion à la base de données se fait grâce à des paramètres qui sont propres à l'utilisateur. C'est pourquoi nous avons un fichier .json sur le même format qu'en figure 10, où il est plus simple pour l'utilisateur d'aller modifier ce fichier sans connaissance du code plutôt qu'aller modifier des variables internes au code du plugin.

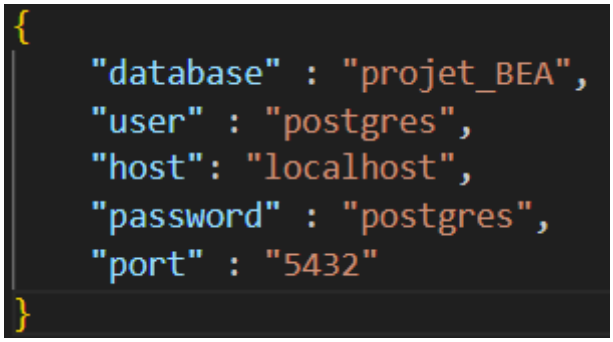

FIGURE 10 – Paramètres de connexion à la base de données

4.4 Respect de la charte graphique

Ce projet proposé par le BEA invitait aussi à répondre à la question du respect de la charte graphique des cartes aéronautiques de l'OACI, l'Organisation de l'Aviation Civile Internationale. Cette problématique a été traitée par Darian Motamed et Jean-François Villeforceix et intégrée au plugin par le biais des fichiers ".qml". QML signifie Qgis Style Markup ce qui veut dire que les fichiers .qml contiennent l'ensemble des informations sur la manière dont les données doivent figurer sur la carte. Cette solution a été apportée en fin de projet puisqu'elle ne faisait pas partie du cadre du sujet même. Vous avez à titre d'illustration ce que cette solution apporte à l'exemple de la figure 11

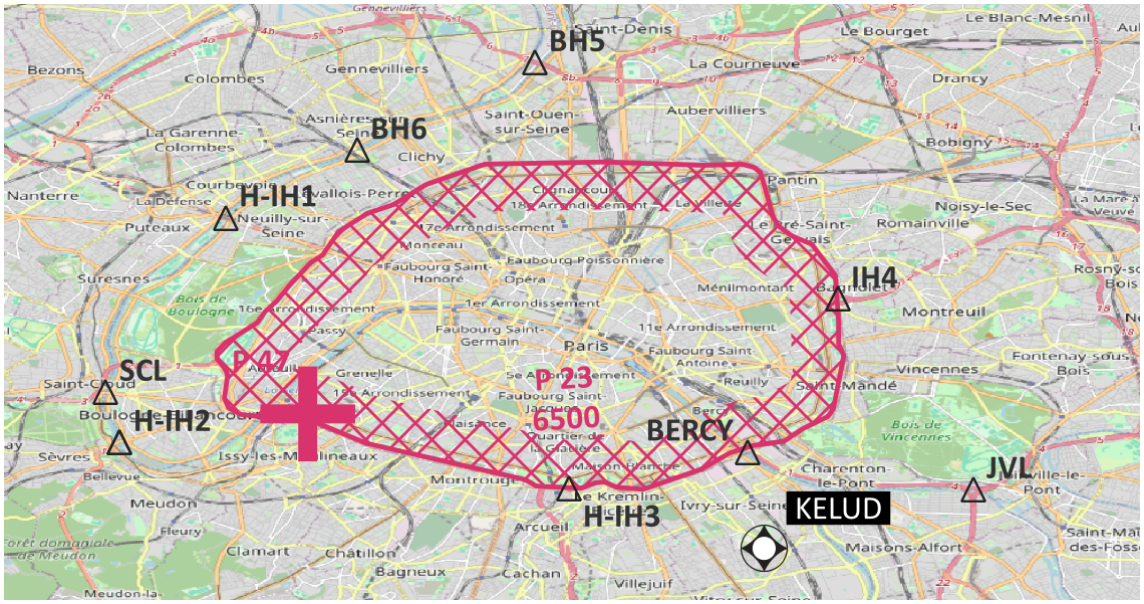

FIGURE 11 – Exemple d'application des feuilles de style dans QGIS

5 Gestion de projet

5.1 Outils et méthode de travail

La réalisation du plugin Qgis impose l'usage du langage Python, de l'API pyQgis et en particulier, l'interface graphique se réalise avec la librairie graphique Qt. De plus, il s'est avéré utile pour la réalisation de cette dernière d'utiliser l'outil Qt Designer directement pour ce qui est de la construction des éléments de base de l'interface graphique. Pour ce qui est de la création de la base de données, quelques installations préalables ont été faites comme l'installation de PostgreSQL comprenant pgAdmin et l'extension postgis ainsi que l'installation des librairies utilisées dans le scrypt de Louis. Enfin, un dépôt git a été créé en collaboration avec Louis Steinmetz pour le partage et la sauvegarde de nos travaux respectifs.

5.2 Planning prévisionnel

La figure 12 décrit le planning adopté lors de cette précédente phase développement.

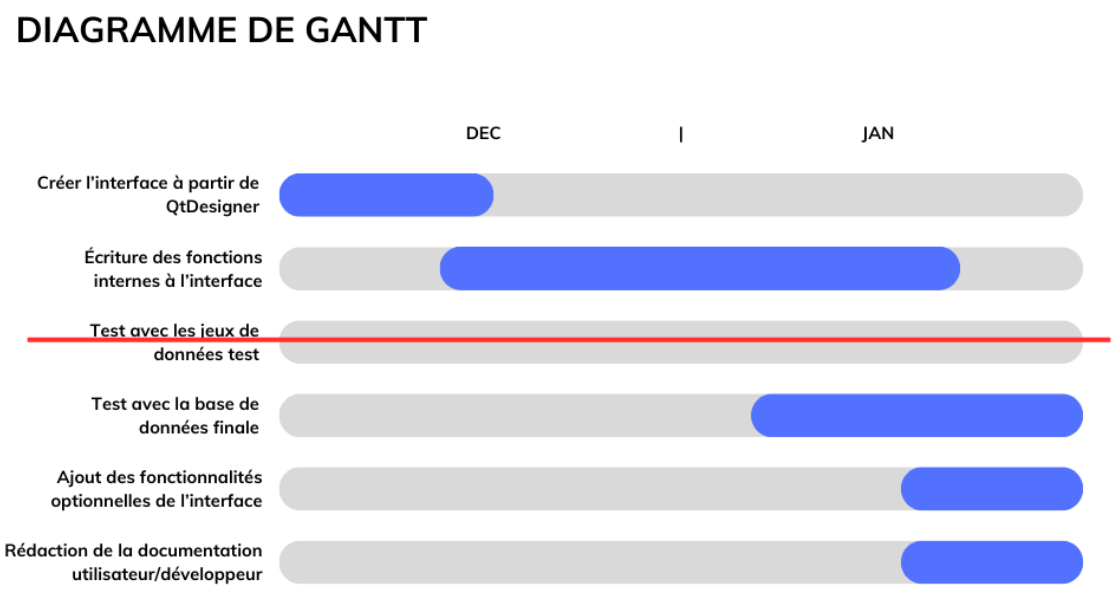

FIGURE 12 – Diagramme de Gantt mis à jour

5.3 Contraintes

Les principaux défis de ce travail se situaient à l'intersection des deux projets proposés par le BEA cette année. En effet, l'interface graphique développée à l'issue de ce projet est destinée à être connectée à la base de données géographiques développée à la fin du second projet, réalisé en parallèle par Louis Steinmetz. La difficulté résidait donc dans la connexion de cette future interface avec la nouvelle base de données. C'est d'ailleurs la raison pour laquelle nous n'avous pas fait d'essais avec des jeux de données test, par peur d'implémenter des méthodes qui ne concordent pas avec le format de la base de données finale puisque elle a été terminée assez tard dans la phase de développement et son format a auss beaucoup évolué.

Conclusion

Tous les points abordés durant la phase d'analyse ont été pris en compte. En effet le plugin développé présente toutes les fonctionnalités qui avaient été initialement demandées. De plus, des fonctionnalitées additionnelles ont été ajoutées afin d'améliorer le fonctionnement global du plugin et l'interaction de l'utilisateur avec l'interface graphique. Toutefois, bien que terminé, on voit quelques pistes d'améliorations à intégrer au plugin. Dans les améliorations envisageables il y a déjà celles que l'ont a citées dans ce rapport : l'intégration d'une barre de recherhce pour faciliter la sélection des données et l'affichage des attributs des données sélectionnées.

Table des figures