



Ecole Nationale des sciences géographiques

Rapport d'analyse final

Visualisation des résidus de traitements des mesures DORIS

par le centre d'analyse IGN-IPGP/JPL
de l'International DORIS Service

Projet informatique, filière PPMD, commandité par Arnaud Pollet, Samuel Nahmani et
Guillaume Lion, Ingénieurs - chercheurs de l'Institut de Physique du globe de Paris
(IPGP)

Réalisée par : Vanessa Monnier

Date : 27 janvier 2024

Table of Contents

1	Contexte du projet	2
1.1	Description du cadre du projet	2
1.2	Définitions et Notions clés	2
2	Expression du besoin	2
2.1	Contraintes	2
2.2	Les acteurs	2
2.3	Objectif de l'étude à terme	3
3	Analyse fonctionnelle - Solutions proposées	4
3.1	Détails des différentes fonctionnalités	4
3.2	Interface graphique	6
3.3	Différents types de visualisation	7
3.3.1	Visualisation actuelle	7
3.3.2	Visualisations futures	8
4	Étude technique : Choix des logiciels et langages - Architecture	10
4.1	Changement dans le diagramme de classe	10
4.2	Choix techniques	12
4.3	Librairies et modules utilisées	12
4.4	Données	12
4.5	Problèmes rencontrés	12
5	Réalisation et suivi de projet	13
5.1	Diagramme de Gantt	13
6	Évolutions et Perspectives	13
7	Annexe	14

1 Contexte du projet

1.1 Description du cadre du projet

Ce projet est commandité par des chercheurs du centre d'analyse IGN-IPGP-JPL de l'international DORIS service. Ce centre est chargé d'effectuer des recherches pour améliorer les traitements des mesures de la technique DORIS en analysant entre autre les résidus des traitements des mesures sur différents satellites et différentes stations au sol. Dans ce cadre, le besoin de disposer d'un outil de visualisation a émergé. Ce logiciel permettrait de faciliter l'étude et l'interprétation physique des différents produits du centre d'analyse et plus particulièrement les résidus des observations ainsi que certains paramètres orbitaux.

1.2 Définitions et Notions clés

Pour plus de précision, DORIS (Doppler Orbitography by Radiopositioning System) est une technique qui, à partir d'instruments embarqués sur satellite et de balises sur terre, permet de mesurer avec une grande précision à la fois la trajectoire du satellite et la localisation de la balise au sol. Il a été développé par le CNES. C'est un système ascendant. Les stations au sols émettent vers le satellite. Les données Doris permettent de déterminer les paramètres orbitaux des satellites et les coordonnées des stations au sol.

Les résidus associés à ces données correspondent aux différences entre les observations réelles (les données mesurées) et les valeurs calculées ou prédites par un modèle mathématique. L'étude de ces résidus est importante car elle permet la détection d'anomalies et une valeur résiduelle élevée indique que le modèle ne représente pas bien la réalité.

2 Expression du besoin

2.1 Contraintes

Les contraintes de ce projet sont définies ci dessous :

- Contrainte de temps qui limite le nombre de fonctionnalités qui seront employées.
- Il est préférable que le langage informatique utilisé soit Python, avec des librairies libres d'accès pour pouvoir être repris convenablement par le futur.
- Cette application peut s'organiser selon divers modules / packages qui doivent être modulables pour être repris facilement par la suite par d'autres développeurs et donc permettre l'ouverture vers de nouvelles fonctionnalités. Cela implique donc l'utilisation de plusieurs DesignPattern.
- Le logiciel doit être interopérable et fonctionner sur tous les systèmes (Windows, Linux et Mac).
- Le rendu final inclut le code du logiciel mais aussi un Jupyter Notebook contenant un guide de lancement du logiciel / guide d'utilisation, ainsi que des bouts de codes pertinents. Par exemple, le code ayant engendré certaines erreurs, ou bien pour pouvoir expliquer certains points importants, afin que le prochain développeur puisse reprendre en main le code facilement.

Une des contraintes majeurs, relève du temps imparti pour le projet. En effet, il n'est évidemment pas possible d'implémenter toutes les fonctionnalités décrites dans le diagramme de classe. L'objectif principal est d'avoir à la fin de ce projet, le squelette globale de l'architecture et si possible déjà quelques fonctionnalités. Nous n'allons pas ici rentrer dans les détails du diagramme 11, mais simplement évoquer les différences de couleurs qui marquent les différences de priorités. Ce diagramme de couleur avait été réalisé pendant la partie analyse. Maintenant que le projet est terminé, on se rend compte que la partie "*TraitementDonnee*" est moins importante que la réalisation dans un premier temps du corps de l'architecture du code.

2.2 Les acteurs

Les premiers acteurs du projet sont les commanditaires qui formulent leurs besoins. Ce sont les chercheurs du centre d'analyse IGN-IPGP-JPL de l'international DORIS Service. L'outil a pour vocation d'être utilisé par ces chercheurs en interne, et si le développement est assez avancé, par d'autres chercheurs, puis pourquoi pas pour un public averti. Cela constitue le deuxième acteur, c'est à dire les utilisateurs de l'outil. Le troisième acteur est le fournisseur de donnée. Ici, les fichiers d'entrées proviennent du logiciel GipsyX (développé par la NASA). Enfin, les différents développeurs qui interviendront sur ce projet constituent les acteurs finaux.

2.3 Objectif de l'étude à terme

L'objectif du projet à terme, c'est à dire une fois le projet aboutit dans quelques années, est de réaliser un logiciel de visualisation des résidus de traitements des mesures DORIS.

En effet, à l'issue de ce projet, une première version est à fournir au commanditaire, mais il n'est pas nécessaire qu'elle soit finit. Le développement de l'outil se poursuivra dans le cadre d'autres projets informatiques. Ainsi le diagramme de cas d'utilisation présenté ci-dessous montre la finalité voulue du projet à terme.

Il a donc pour but de faciliter les interprétations des comparaisons résiduelles de satellite à la surface du globe. Le logiciel doit permettre à l'utilisateur d'accéder à plusieurs types de représentations en fonction des fichiers pris en entrées. Dans un premier temps, une carte du monde s'affiche avec plusieurs informations et graphiques que l'utilisateur peut obtenir en cliquant sur les stations au sols présentes sur la carte. Ce projet a pour objectif pédagogique l'informatique, mais il n'exclut pas une petite partie recherche, entrant dans le cadre du mentorat recherche mis en place cette année par l'ENSG, pour implémenté par exemple le type de filtrage souhaité, ou bien les différents modèles et calculs statistiques.

Ensuite, un skyplot pourra être visualisé pour chaque station où l'on observe donc les valeurs des résidus en fonction de l'élévation et de l'azimut d'un ou plusieurs satellites. Il y aura donc possibilité d'observer un seul skyplot ou bien plusieurs skyplot à la fois sur des stations différentes pour permettre une comparaison. A cela s'ajoute la possibilité de faire apparaître différents types de statistiques ainsi que la possibilité de générer des modèles sur les données.

D'autres types de visualisation doivent être développées, comme un graphique d'évolution des résidus en fonction de l'élévation pour un satellite par exemple, ou bien la possibilité de visualiser les traces des satellites sur la carte.

L'utilisateur pourra donc se déplacer sur la carte, ce qui implique de créer une interaction dynamique pour lui (avec des zooms...). Les possibilités d'action que l'utilisateur peut réaliser sont décrites dans le diagramme de cas d'utilisation 1.

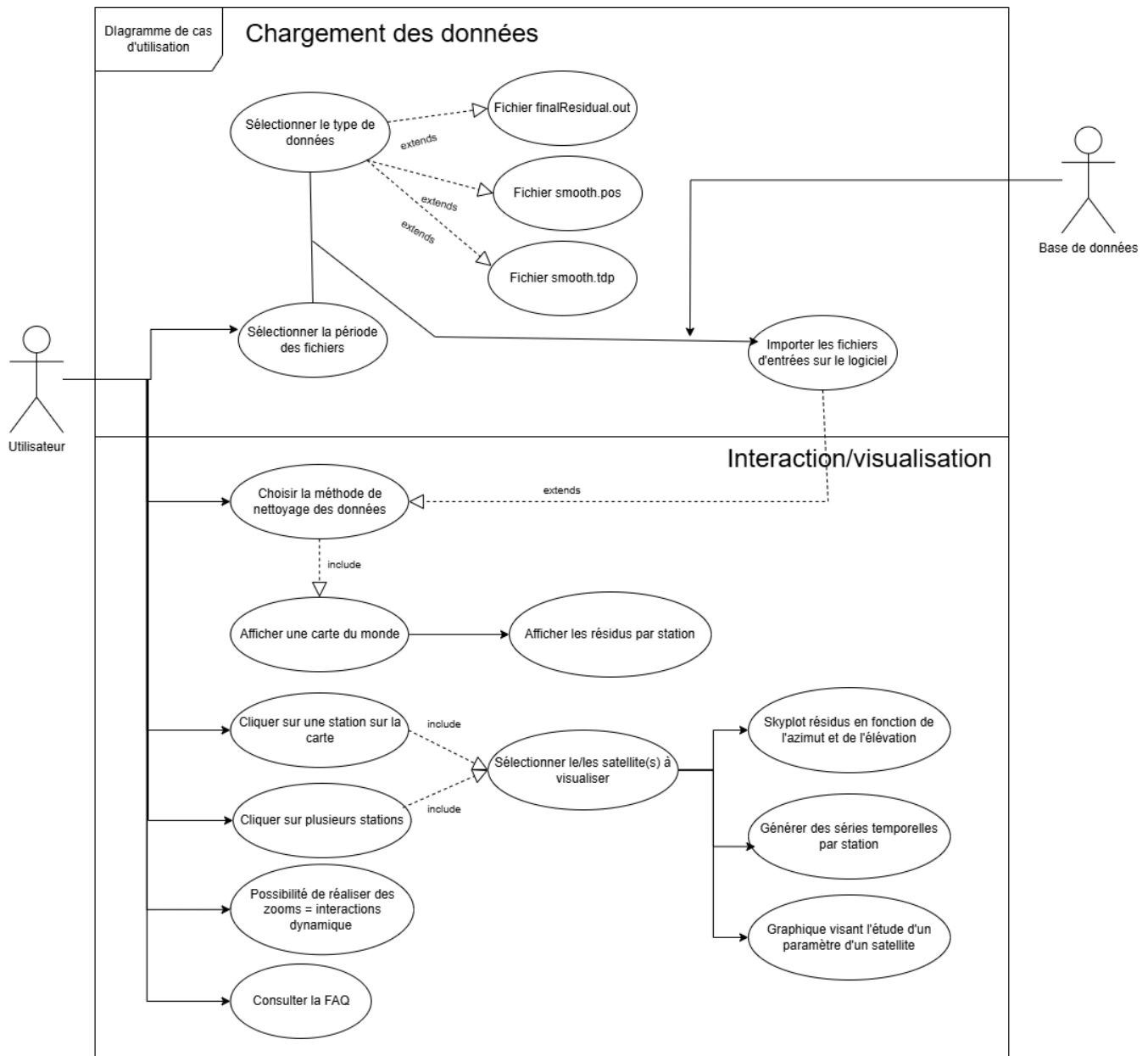


FIGURE 1 – Diagramme de Cas d'utilisation

3 Analyse fonctionnelle - Solutions proposées

3.1 Détails des différentes fonctionnalités

Comme déjà évoqué dans les objectifs du projet, l'utilisateur aura la possibilité de naviguer sur l'interface graphique pour obtenir plusieurs visualisations différentes.

Voici un détail des fonctionnalités présentes actuellement sur le logiciel :

- **Position des stations** : Affichage des positions des stations DORIS sur une carte mondiale avec survol interactif des coordonnées. Il est donc possible de voir pour chaque station le nom de la station, son abréviation associé et le détail de ses coordonnées géographiques (longitude, latitude).
- **Possibilité de zoom et de clique sur chaque station** : Permet de voir le descriptif détaillé.
- **Génération de Skyplots** : Cette génération de skyplot est possible lorsque l'utilisateur renseigne une période de temps, un satellite qu'il veut observer, et une ou plusieurs stations associées. Dans notre projet actuel, deux types de graphique skyplot sont possibles.
 - *Skyplot-station* : Correspond aux traces de la station en se plaçant dans le référentiel du satellite fixe.

– *Skyplot-satellite* : Correspond à la trace du satellite dans le ciel vue depuis la station au sol.

Le Skyplot représente l'évolution des résidus en fonction de l'azimut et de l'élévation (de la station ou du satellite).

- **Nettoyage et analyse statistique des données** : Un nettoyage a été effectué dans les fichiers de données brutes pour supprimer les lignes contenant des "DELETED". Cependant, l'implémentation d'un algorithme de nettoyage pour exclure certaines données n'a pas été faite par manque de temps.

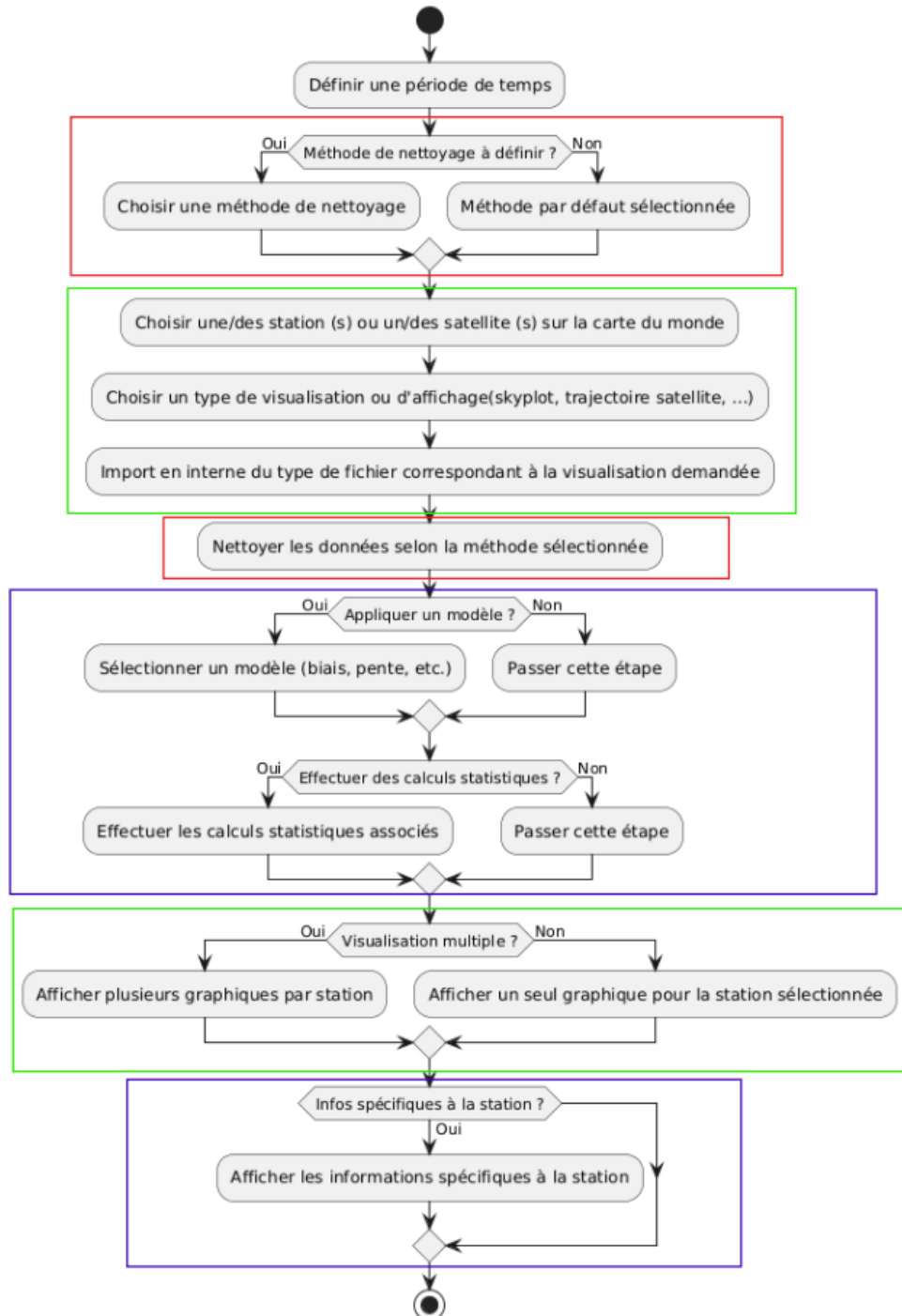


FIGURE 2 – Diagramme d'activité proposé pour la solution

Le diagramme d'activité 2 représente les étapes principales et les options offertes à l'utilisateur lors de l'utilisation du logiciel de visualisation. Ce diagramme représente les fonctionnalités finales qui devraient être implémentées. Actuellement, pour la partie nettoyage des données, l'utilisateur n'a pas la possibilité de choisir le nettoyage qu'il veut donc l'encadré en rouge n'est pas encore implémenté. Le seul nettoyage présent est un nettoyage de base sur les fichiers brutes qui est obligatoirement à faire donc l'utilisateur n'a pas à choisir ou non s'il le veut. Ce nettoyage élimine des valeurs déjà supprimés dans la génération de ces fichiers mais qui subsistent encore. Les nettoyages mentionnés dans les encadrés rouges permettraient

à l'utilisateur d'avoir un jeu de donnée plus fin et donc d'extraire certaines valeurs un peu grandes, même si ces valeurs ne sont pas considérées comme aberrantes.

De même le premier encadré en violet sur les calculs statistiques ou modèles pouvant être générés sur les données n'a pas encore été implémenté. Ce n'était pas la priorité du projet.

Enfin, les encadrés verts et le deuxième encadré violet reflètent de ce qui a été implémenté. L'utilisateur peut en effet choisir une station, un satellite, une période de temps pour générer des skyplots et obtenir des informations sur une station en cliquant dessus. La suite du projet est de vouloir d'autres types de visualisation comme évoquées plus haut. L'export de ces graphiques de skyplot est actuellement possible.

3.2 Interface graphique

Nous allons maintenant développer en particulier la forme qu'a le logiciel actuellement, en évoquant donc l'interface graphique, et ce que voit l'utilisateur en lançant le logiciel. Cette interface est codée en Python, suivant les volontés des commanditaires.

Lorsque le logiciel se lance, on arrive automatiquement sur la carte du monde en fond avec une barre latérale en haut contenant plusieurs onglets. Voici ci dessous l'interface actuelle :

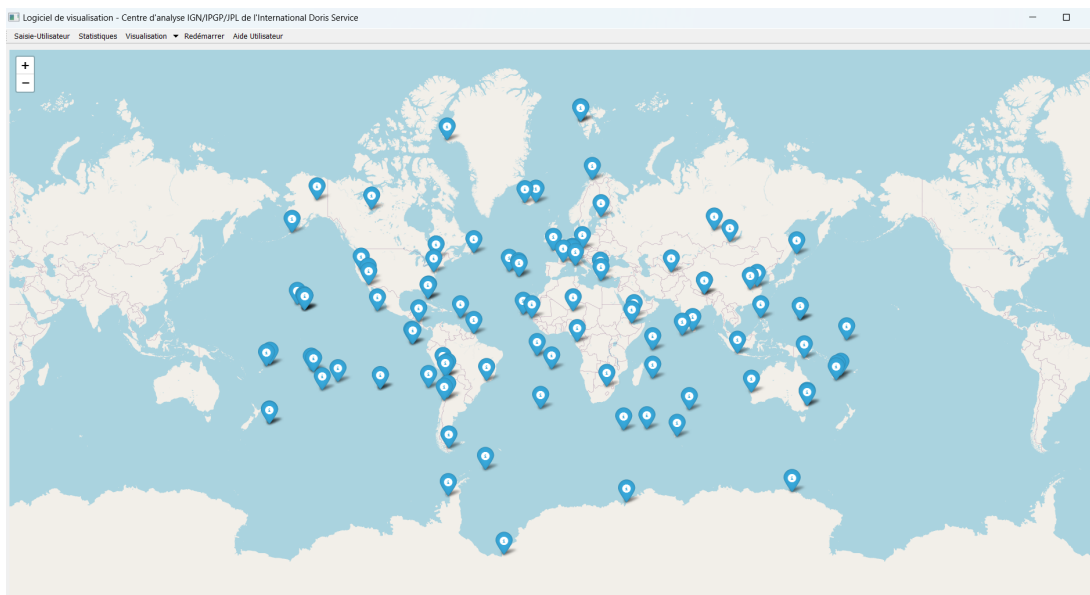


FIGURE 3 – Capture d'écran de la page principale du logiciel

Voici la liste des onglets que contient cette barre latéral :

- Onglet "Aide Utilisateur" : Cet onglet mène à un descriptif des différentes fonctionnalités du logiciel et au lien du dépôt git du projet. Cela permet à l'utilisateur de trouver sur le dépôt git une documentation utilisateur sous forme de jupyter notebook, ce qui lui permet de visualiser toutes les divers fonctionnalités qu'il va pouvoir effectuer.
- Onglet "Statistique" : Doit contenir toutes les méthodes statistiques que l'utilisateur va pouvoir effectuer sur ses données, sous forme de menu déroulant. Actuellement, ces méthodes n'ont pas encore été implémentées donc cet onglet renvoie le message "Statistique en cours de développement" dans la console python.
- Onglet "Visualisation" : Permet d'afficher les différents types de visualisation possible. L'objectif à terme serait de rajouter des onglets dans ce menu déroulant pour avoir plusieurs type de visualisation. Actuellement, il est possible d'afficher le "skyplot-station" et "skyplot-satellite" qui seront décrit dans la partie suivante.
- Onglet "Redémarrer" : Il est possible pour l'utilisateur de cliquer sur le bouton 'Redémarrer' afin de ré-actualiser toutes les variables qui étaient enregistrés. Cela ferme les fenêtres Skyplot ouvertes, réinitialise les données dans le modèle et recharge les données pour réafficher la vue principale
- Onglet "Saisie Utilisateur" : Ce bouton permet à l'utilisateur de rentrer la période de temps des données qu'il veut étudier, le nom du satellite sur lequel il souhaite se focaliser, et une liste de station associée au satellite où il doit en choisir au moins 1, sachant qu'un maximum de 4 est préférable pour garder une bonne visibilité et gestion des graphiques.

Concernant la carte du monde, les stations y sont représentées par des points. L'utilisateur peut effectuer des zooms et se déplacer sur la carte.

Voici à quoi ressemble l'interface pour saisir les différentes variables d'entrées afin d'afficher des skyplots. Pour le moment, cette interface est relatif à des skyplots, mais il est facilement envisageable de la relier à d'autres types de graphiques, si des nouvelles visualisations sont implémentées.

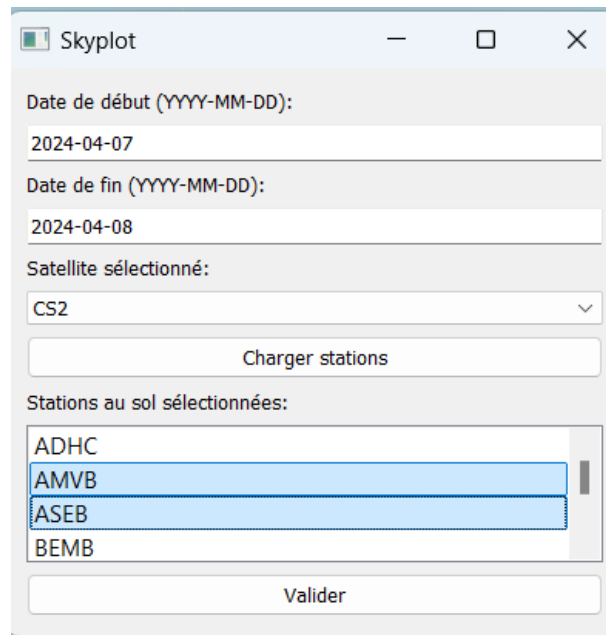


FIGURE 4 – Interface Saisie Utilisateur

3.3 Différents types de visualisation

3.3.1 Visualisation actuelle

Une des visualisations principales est le skyplot. Il représente l'évolution résiduelles des satellites en fonction de l'azimut et de l'élévation c'est à dire la variation des résidus du satellite par rapport à sa position en orbite autour de la terre. Comme évoquer précédemment, il y a deux possibilités de tracer le skyplot. Cela se fait grâce aux deux sous-boutons "Skyplot-Station" et "Skyplot-satellite" permettent d'afficher une fenêtre skyplot. Le premier considère le satellite immobile, et permet donc d'afficher les traces de la station au sol, et le deuxième considère le cas classique d'une station au sol fixe et permet donc de visualiser la trace du satellite dans le ciel. Ainsi pour chaque satellite, l'utilisateur peut charger les stations qui lui sont associés (les stations qui peuvent voir le satellite à son passage) et afficher plusieurs skyplots en même temps afin d'effectuer des comparaisons.

Voici un exemple de Skyplot pour la station de Toulouse après le lancement du logiciel. 5.

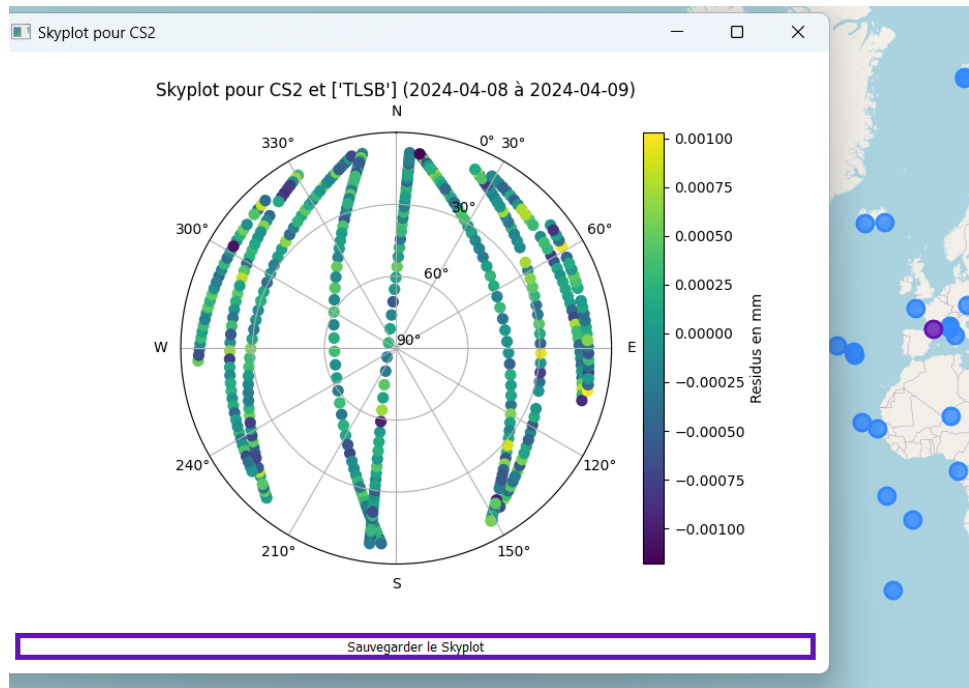


FIGURE 5 – Visualisation spatiale

Dans la figure 5, on peut voir la correspondance de couleur entre la fenêtre en violet et le point au sol violet. L'intérêt de cela intervient lors de la trace de plusieurs skyplot pour comparer spatialement les variations résiduelles. Cette figure correspond au cas "Skyplot - station".

De plus, voici ci-dessous le graphique de la figure 6, lorsqu'elle est sauvegardée, et cette fois-ci dans le cas "Skyplot - satellite".

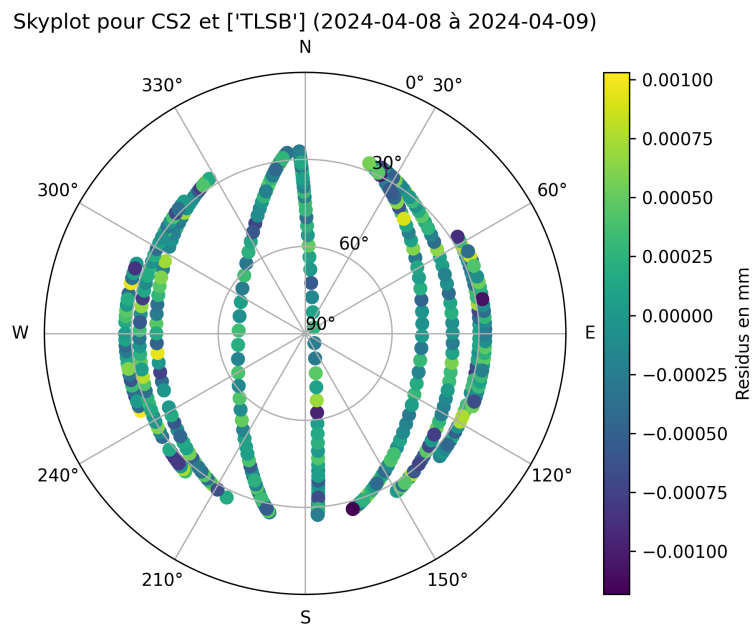


FIGURE 6 – Visualisation spatiale

3.3.2 Visualisations futures

D'autres visualisations seront intéressantes comme la possibilité pour l'utilisateur de réaliser un stacking des résidus (empilement des résidus) pour observer les effets du multi-trajets. Le stacking ou empilement des résidus consiste à superposer plusieurs séries de données (ici, des résidus) pour les analyser collectivement. Par exemple, si l'utilisateur a plusieurs mesures de résidus à différents moments et à différentes élévations (en fonction de la position du satellite ou de la station), le stacking permet de visualiser

l'effet de ces résidus de manière combinée, ce qui facilite la détection de motifs ou d'irrégularités. Le multi-trajet fait référence à un phénomène où un signal de satellite ne suit pas une trajectoire directe entre le satellite et le récepteur, mais est réfléchi ou réfracté par des objets comme des bâtiments, des arbres, ou le sol avant d'atteindre le récepteur. Ce phénomène peut entraîner des erreurs dans les mesures de position, car le signal réfléchi parcourt un trajet plus long que celui du signal direct, provoquant un décalage dans la mesure de la distance.

L'objectif ici est ainsi de permettre à l'utilisateur d'observer comment les erreurs causées par le multi-trajet varient en fonction de l'élévation des satellites, en combinant plusieurs mesures pour une meilleure analyse.

De plus, il pourrait être judicieux de tracer des graphiques d'évolutions des résidus en fonction de l'élévation seulement, pour obtenir des séries temporelles en indiquant les valeurs des paramètres sur la station. Il serait aussi intéressant d'observer les traces des satellites sur la carte mondiale avec un jeu sur les couleurs pour observer les différences résiduelles en fonction de l'endroit où l'on se trouve et de la trace de la station.

4 Étude technique : Choix des logiciels et langages - Architecture

4.1 Changement dans le diagramme de classe

Les changements dans le diagramme de classe entre le diagramme du rapport d'analyse et celui du rapport final résident plutôt dans les méthodes au sein des classes. Le corps du diagramme n'a pas changé en lui-même.

Voici le diagramme actuel avec toujours la même architecture MVC (figure 7).

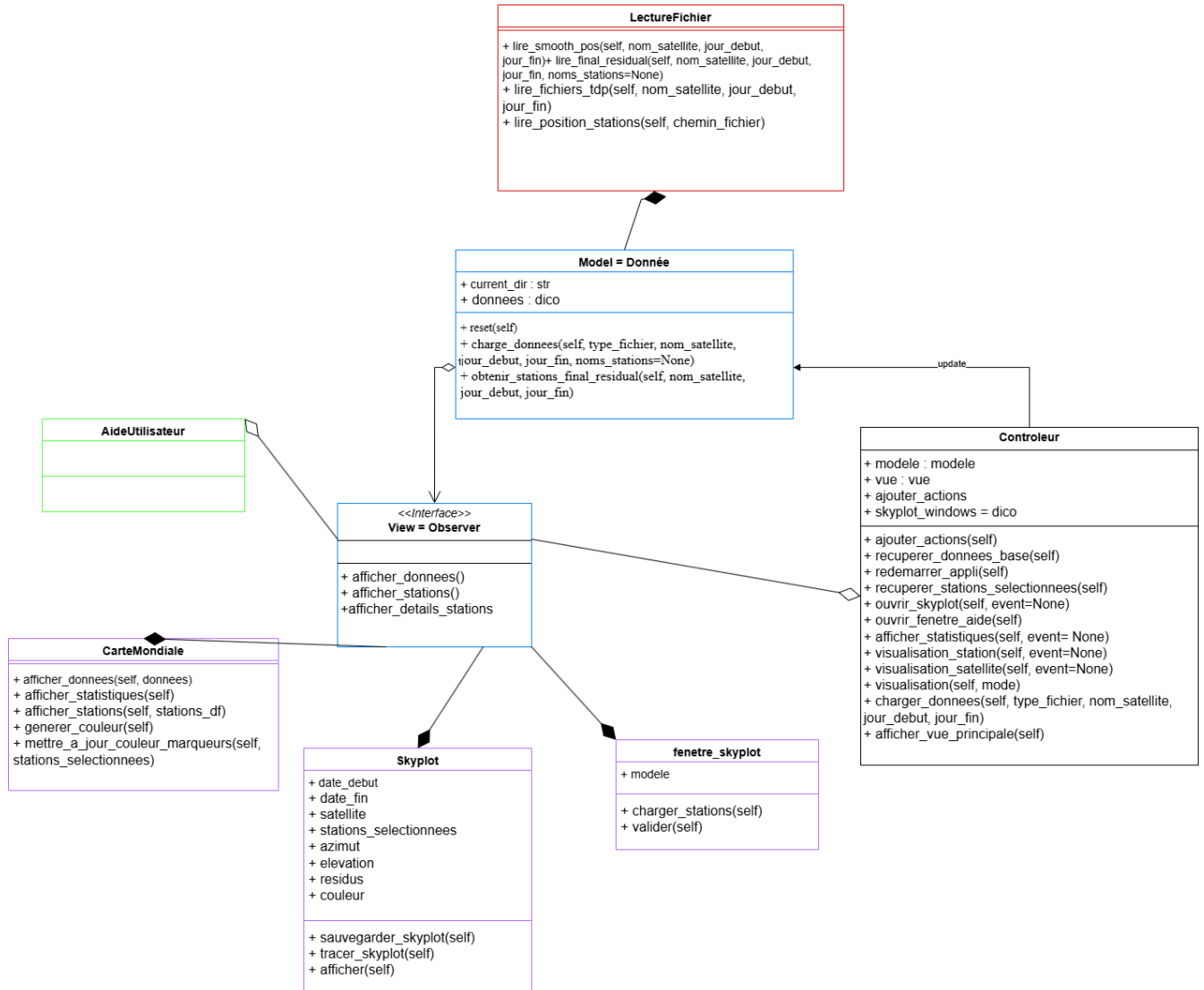


FIGURE 7 – Diagramme de classe final

Ce diagramme suit les processus de divers Design Pattern, nécessaires pour répondre aux contraintes et besoins des commanditaires. L'architecture globale du code (présentée sur la figure 11) suit le design Pattern MVC : Model-View-Controller. Ce pattern permet de séparer la structure d'une application en trois entités distinctes : le modèle (qui gère la logique, les données et leur traitement), la vue (qui est responsable de la présentation et de l'affichage des données aux utilisateurs) et le contrôleur qui agit comme un intermédiaire entre le modèle et la vue. Il reçoit les entrées utilisateurs (par exemple, des clics ou sélections), les interprète, et déclenche des actions sur le modèle ou met à jour la vue.

Tout d'abord, concernant la partie *Modele*, elle a peu changé entre le diagramme de classe ci-dessus, et le diagramme de classe fait pour la partie analyse figure 13. Cette classe permet de centraliser les données et les opérations nécessaires pour leur traitement, leur nettoyage, et leur analyse statistique. Elle récupère le type de données souhaité à l'aide de la classe *LectureFichier*, et permet ainsi de charger les données souhaitées avec sa méthode *Chargerdonnees*. La différence réside dans l'absence dans le diagramme final de la classe *TraitementDonnee* qui n'a pas été implémenté par manque de temps.

Pour le traitement des données, il faudrait d'ailleurs utiliser un StrategyPattern. Cela permettrait

de choisir entre différentes méthodes de nettoyage et de laisser l'ouverture à de futures implémentations dans le cas où ce projet informatique sera repris par d'autres personnes.

Un nettoyage interne a été effectué sur le jeu de donnée, notamment concernant le fichier "final-Residual.out". Ce nettoyage s'observe bien sur les graphiques de skyplot, tels qu'on le voit dans ces graphiques :

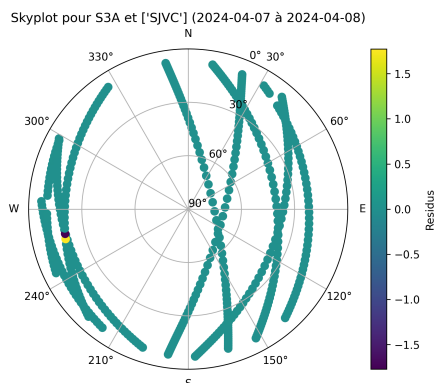


FIGURE 8 – Graphique sans nettoyage

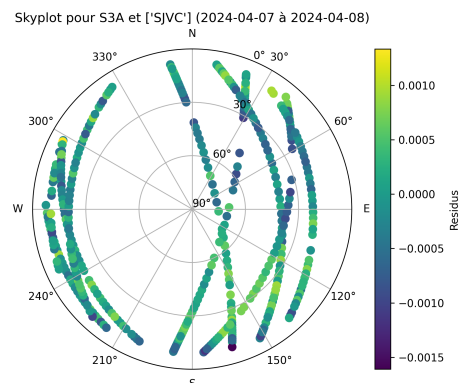


FIGURE 9 – Graphique avec nettoyage

On constate bien la différence dans l'échelle des résidus où sans nettoyage, les valeurs aberrantes sont encore présentes, et empêche la visualisation correct et non biaisée de l'ensemble du jeu de donnée.

En ce qui concerne la classe "*LectureFichier*", elle contient les données fournis en entrées, qui seront développées dans la sous-partie Données.

Une des différences majeurs est l'absence de la classe "*CalculStatistiques*" dans le diagramme actuel car celle-ci n'a pas pu être implémenté. Ce n'était pas la priorité du projet, et c'était prévu avec les commanditaires que cette classe ne soit pas implémenté lors de ce projet.

Nous allons maintenant nous intéresser à la partie concernant les classes Vue et Contrôleur.

Concernant le contrôleur, il gère les interactions entre le modèle et la vue. Il effectue le chargement des données selon les sélections de l'utilisateur c'est à dire qu'il déclenche le chargement et traitement des fichiers en fonction de la période ou des stations choisies. Il gère les événements de visualisation en répondant aux actions de l'utilisateur pour mettre à jour les vues (ex., choisir une station, modifier la période). Pour finir, il permet de gérer les configuration dynamique en intégrant la modularité pour permettre de nouvelles fonctionnalités et nouveaux affichages sans changements majeurs dans le code.

La classe Vue est responsable de la représentation graphique et de l'interaction avec l'utilisateur. Elle agit comme une interface entre : le modèle (les données et leur logique métier) et l'utilisateur (qui interagit avec le système). Elle affiche les données issues du Modèle sous une forme compréhensible pour l'utilisateur (cartes, graphiques, etc.). Elle met à jour l'affichage en fonction des changements dans les données.

4.2 Choix techniques

Ce projet est particulièrement adapté à de l'orienté objet. En effet, la POO permet de structurer chaque composant comme un objet autonome, ce qui facilite la compréhension et la maintenance du système. Dans notre cas, de nombreuses méthodes sont indépendantes, c'est pourquoi ces principes s'appliquent bien à notre code.

4.3 Librairies et modules utilisées

Pour le fonctionnement du code, quelques modules vont devoir être importés. Il y a les modules de base comme **numpy** et **matplotlib**. Concernant la lecture des fichiers, les données seront représentées sous forme de `dataFrame` par l'intermédiaire de la librairie **Pandas**. La carte du monde sera implémentée en utilisant la librairie **cartopy**. Concernant le graphique `skyplot`, il utilisera la librairie **gnns_lib_py**.

Pour l'interface graphique, **PyQt** est la librairie qui sera utilisée. Elle dispose de plusieurs avantages la rendant idéal pour un projet de ce type. En effet, cette librairie est très puissante pour des interfaces complexes avec plusieurs onglets, widgets et graphiques. Elle est aussi très adaptée pour des cartes interactives et des visualisations avancées comme le `skyplot`. Elle supporte des fichiers multi-onglets, les barres de menu et les systèmes de gestion de logs.

Pour la gestion de la carte, la bibliothèque **folium** sera utilisée. La librairie **Folium** est un outil Python permettant de créer des cartes interactives en utilisant `Leaflet.js`, une bibliothèque JavaScript pour la cartographie. Elle permet de visualiser des données géospatiales de manière dynamique et interactive.

4.4 Données

Les données constituent des fichiers fournis par le logiciel `GipsyX` (logiciel développé par la NASA). 3 types de fichiers nous sont fournis.

Tout d'abord un fichier "***smooth.pos***" qui contient les paramètres orbitaux d'un satellite (position et vitesse), ce qui donne par colonnes les variables : nom du repère (ici repère terrestre fixe), le nom du satellite, la date de la position du satellite en seconde, et ensuite les positions `x`, `y` et `z` du satellite en coordonnées cartésiennes suivit du vecteur vitesse selon ses trois composantes `vx`, `vy` et `vz`. L'origine de ce repère est le centre des masses de la terre. `x` pointe vers l'intersection de l'équateur terrestre et du méridien de Greenwich, `y` est orthogonal à `x` dans le plan équatorial, dans la direction est et `z` est aligné avec l'axe de rotation de la Terre.

Ensuite, pour extraire les résidus utile à notre visualisation, on utilise le fichier "***finalresiduals.out***" qui correspond par colonne aux variables : la date d'acquisition des résidus en seconde depuis le 1er janvier 2000, le nom du satellite associé au nom de la station qui le mesure, le type de modèle, les valeurs résiduelles associées au satellite en m/s, l'élévation du satellite, l'azimut du satellite, l'élévation de la station et pour finir l'azimut de la station.

Le troisième fichier fournit en entrée se nomme "***smooth.tdp***". C'est un fichier de paramètre supplémentaire comme les paramètres de tropo ou d'accélération empirique (en sinus et cosinus) contenant : date mesure, apriori (ce qu'on estime), valeur du paramètre et écart-type associé.

Enfin, le quatrième fichier en entrée est "***stationdata.txt***" qui a été créé manuellement et qui contient les noms de station, leur abréviation et les positions en longitude et latitude.

Pour lancer le logiciel, la base de donnée se fait pour le moment avec un dossier en local contenant les fichiers voulus (généralement correspondant à la période de temps souhaité). Il n'existe pas de base de donnée non locale.

4.5 Problèmes rencontrés

Tout d'abord concernant l'interface graphique, des problèmes sont survenus rapidement sur la gestions des vues. En effet, le fait d'avoir plusieurs sous-vues à longterm posé problème pour le contrôleur pour relier ses méthodes aux différentes vues.

Ensuite, concernant les marqueurs. On peut remarquer qu'au lancement du projet, les points représentant les stations au sol sont des marqueurs. Lorsque l'on souhaite tracer le `skyplot`, les marqueurs deviennent alors des cercles. Cela est du à un problème de gestion des marqueurs sur les valeurs hexadécimal. En effet, on utilise une fonction "`genererCouleur`" qui permet pour chaque station sélectionnée de générer une couleur aléatoirement en utilisant des couleurs hexadécimal. L'utilisation de ces cercles n'est pas problématique pour la compréhension de la logique du code.

Pour finir, la prise en main des différentes librairies a été une difficulté, pour gérer les liens d'appartenance entre chaque méthode. La création du `skyplot` a aussi été une longue étape, car le code a subit

de nombreux ajustement pour finir par bien fonctionner et afficher les bons tracés.

5 Réalisation et suivi de projet

5.1 Diagramme de Gantt

Ce diagramme de Gantt 10 permet de connaître le détail de la planification des différents modules qui ont été implémentés durant le projet. Il se base sur les couleurs de priorité présentes dans le diagramme de classe.

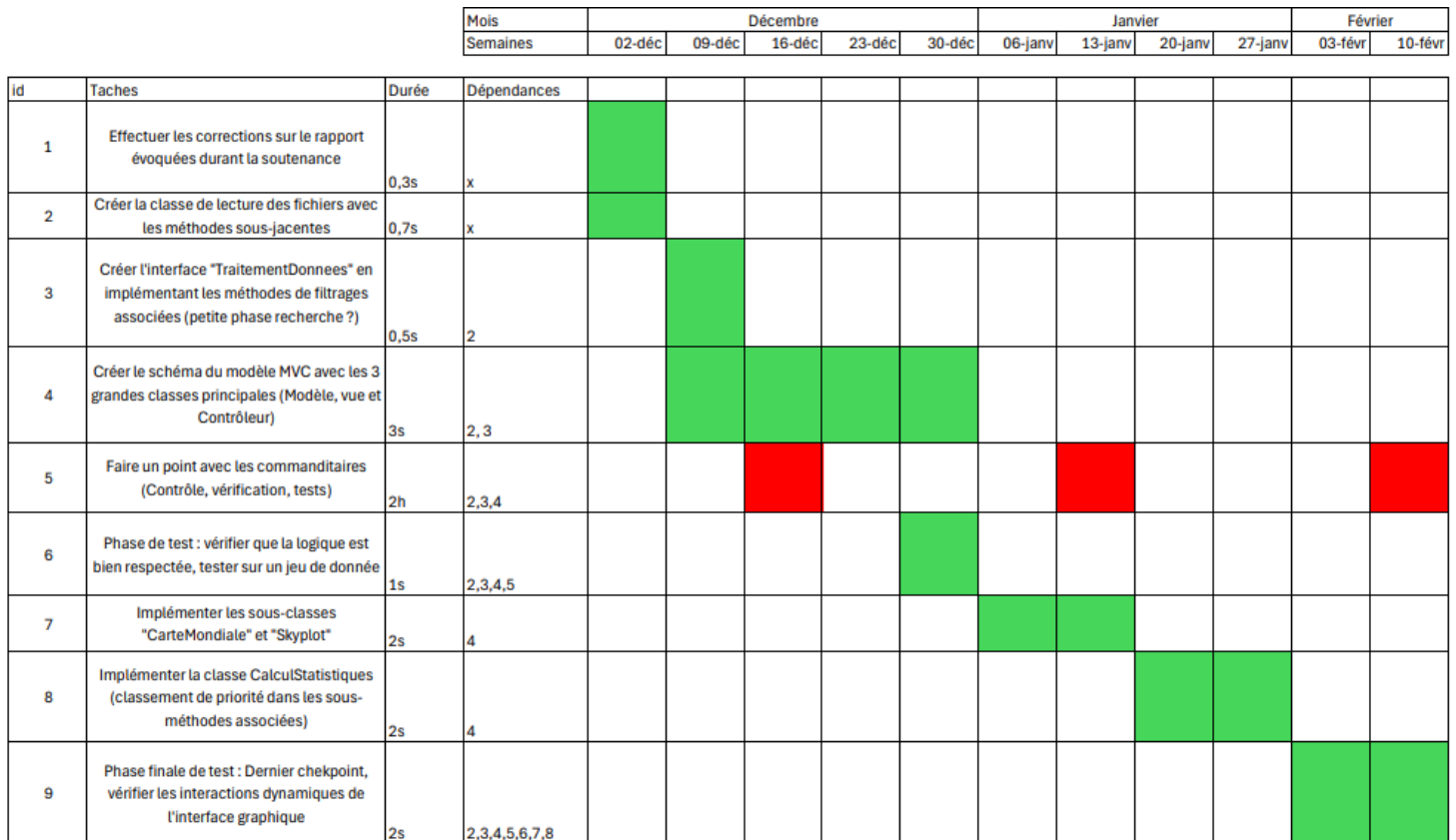


FIGURE 10 – Diagramme de Gantt

6 Évolutions et Perspectives

Le projet actuel a été conçu de manière modulaire, ce qui facilite sa reprise et son évolution par d'autres étudiants dans les années à venir. Grâce à une documentation claire et un code structuré, les futurs utilisateurs pourront facilement comprendre le travail effectué et l'étendre selon les besoins. Plusieurs axes d'amélioration sont envisageables, notamment l'optimisation des performances, l'ajout de nouvelles fonctionnalités (comme les différents types de visualisation et la partie traitement des données avec le nettoyage, et les calculs statistiques), ainsi que l'intégration de tests unitaires et d'une vraie base de donnée.

Le logiciel de visualisation développé présente un grand intérêt pour l'IPGP, tant dans le cadre de la recherche que de l'enseignement. En permettant une visualisation claire et interactive des données géospatiales, il offre aux chercheurs un outil puissant pour analyser et interpréter les phénomènes naturels, tels que les mouvements tectoniques, les risques sismiques et volcaniques. Cette interface interactive facilite l'exploration des données complexes et améliore la prise de décision dans le cadre des recherches.

7 Annexe

Pattern MVC - architecture globales

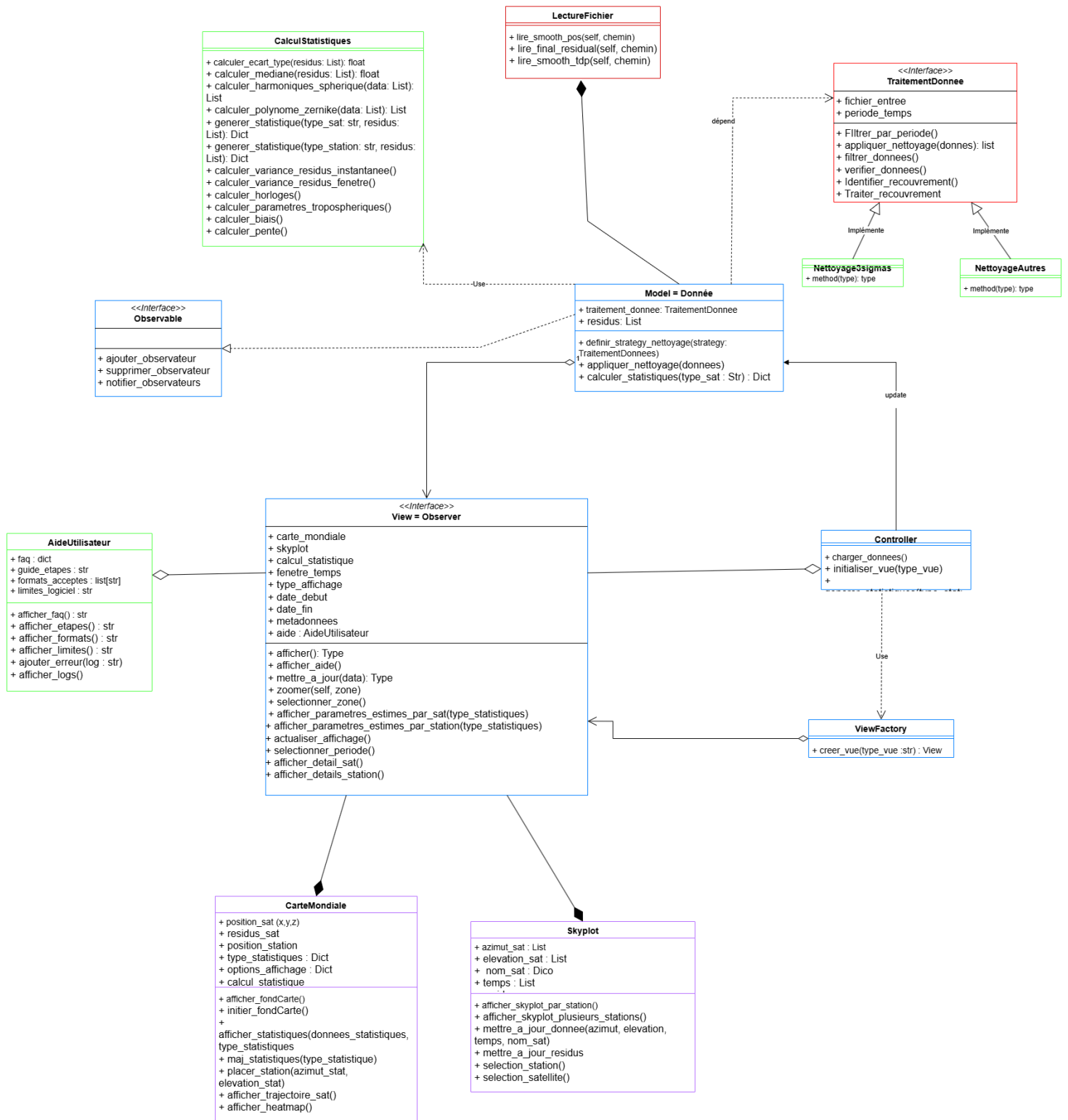


FIGURE 11 – Diagramme de classe complet

Partie modèle du diagramme de classe initial : Partie vue-contrôleur du diagramme de classe initial :

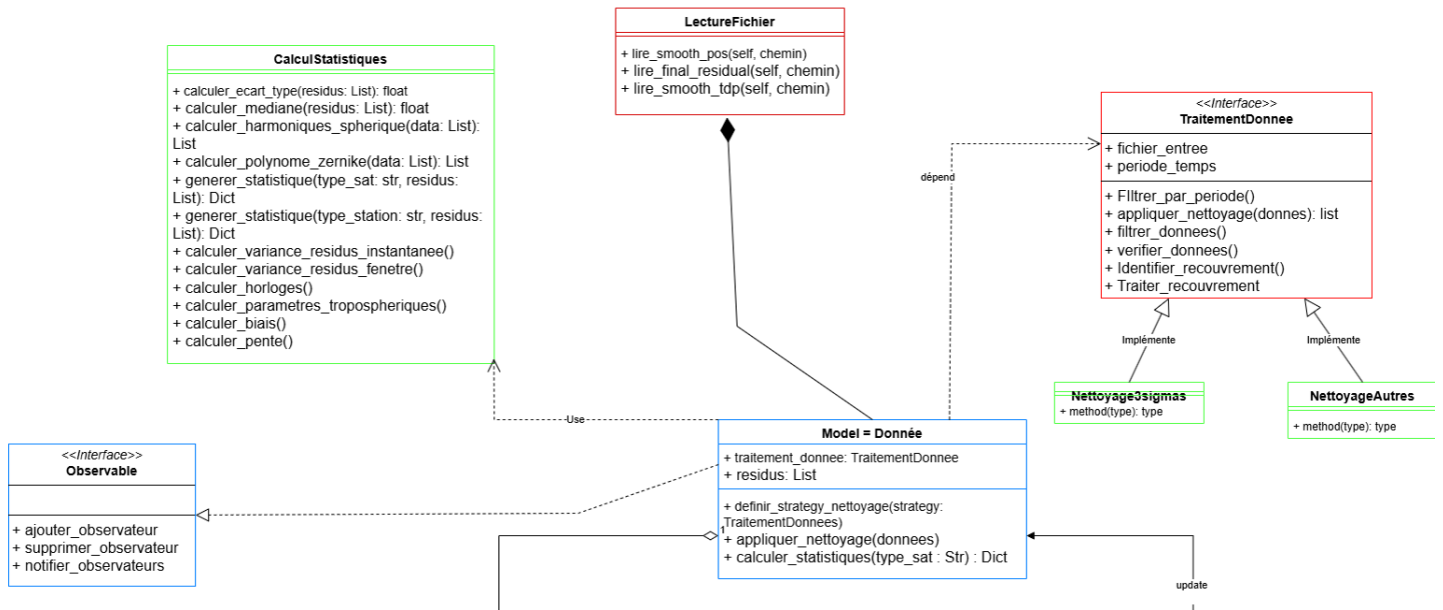


FIGURE 12 – Entité modèle du diagramme de Classe

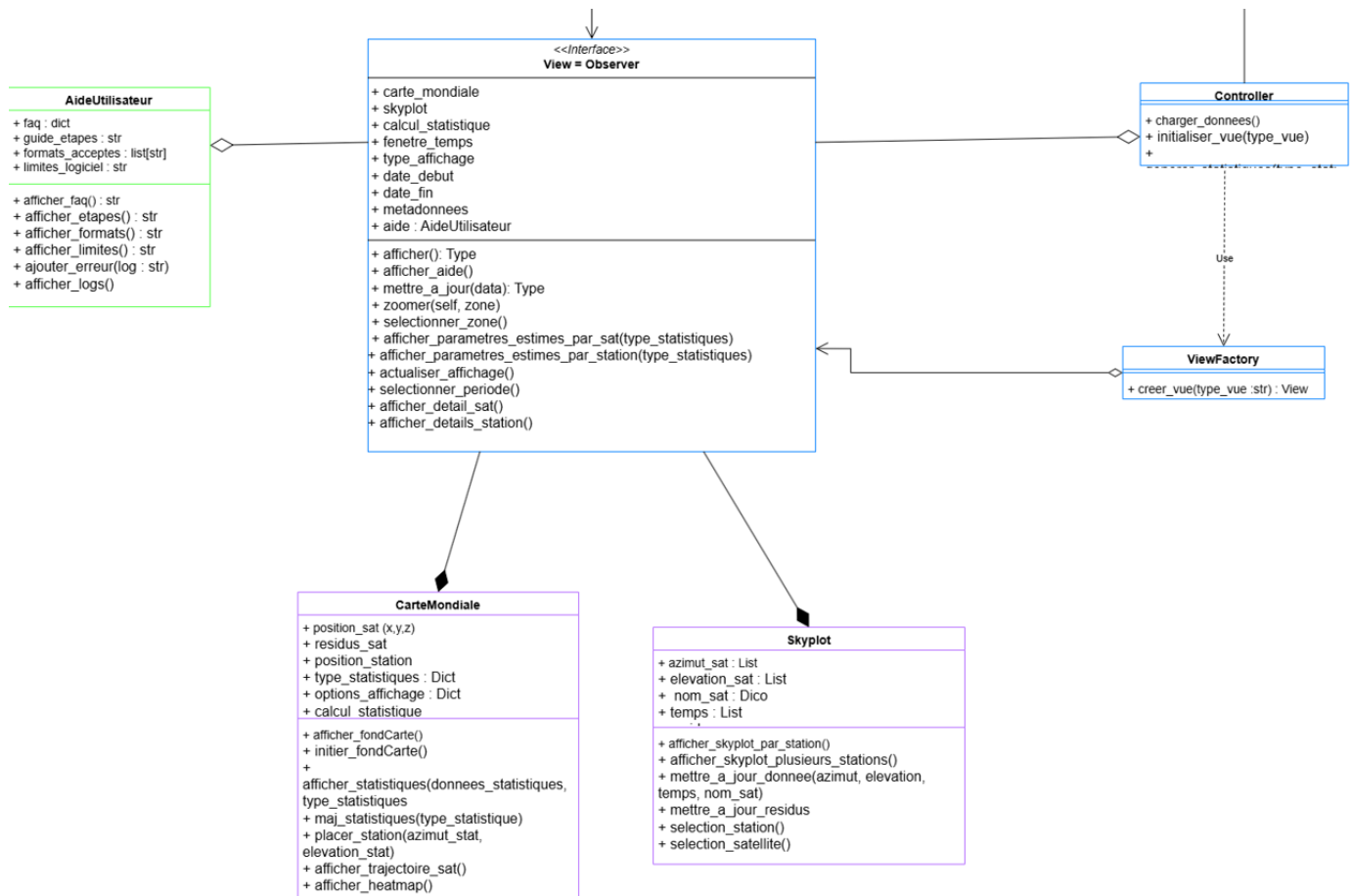


FIGURE 13 – Entité vue-Contrôleur du diagramme de Classe