RAPPORT DE TRAVAIL DE FIN D'ÉTUDES

Développement d'un plugin QGIS pour l'accès à des mesures *in situ* de rayonnement solaire à l'échelle locale respectant le standard Sensor Observation Service (SOS)

Stagiaire :

Alexandre BARBUSSE

Promotion 56

EIVP

Maître de stage :

M. LIONEL MENARD

Ingénieur de recherche

MINES ParisTech

Tutrice :

Mme Cécile HAECK

Professeur de SIG à l'EIVP

IGN

Centre Observation Impacts Énergie (O.I.E.), MINES ParisTech
Sophia Antipolis, 1er Février - 31 Juillet 2017

Résumé

Des séries temporelles de mesures *in situ* de grandeurs physiques sont utilisées pour acquérir une connaissance fine - dans l'espace et dans le temps - des grandeurs physiques caractéristiques de notre environnement, comme par exemple le rayonnement solaire surfacique. Ces mesures sont enregistrées sous la forme de données numériques d'observation de la Terre. Les Technologies de l'Information et de la Communication (TIC) permettent de stocker, puis de partager ces données numériques. En particulier, l'Open Geospatial Consortium (OGC) propose le service *Sensor Observation Service (SOS)* – qui repose sur des standards ouverts, ce qui lui confère une grande interopérabilité - pour l'encodage de ces données. Elles peuvent alors être stockées au sein de serveurs d'implémentation du service SOS. Ce dernier service permet également de mettre à disposition les données par l'intermédiaire de clients, c'est-à-dire des interfaces numériques qui rendent possible la communication entre des utilisateurs et un serveur SOS. Les données de mesures *in situ* incluent la localisation des stations où elles ont été effectuées. Les clients développés utilisent cette information géospatiale pour générer la couche cartographique numérique des stations de mesures associées à un serveur SOS. Si la plupart des clients pour le service SOS actuellement disponibles sont des clients Web, une réelle demande a été exprimée en faveur de la création d'un client permettant à des utilisateurs de récupérer, de visualiser, et de télécharger des séries temporelles de données de mesures *in situ* stockées sur des serveurs SOS depuis un Système d'Information Géographique (SIG). Ce stage a permis le développement et la publication d'une extension Python du SIG Open Source QuantumGIS (QGIS) qui contribue à répondre à cette demande. A l'issu des tests effectués, le bon fonctionnement de l'ensemble des fonctionnalités attendues de l'extension a été observé pour 4 serveurs parmi les 10 serveurs SOS testés. Si son développement n'est pas achevé, et qu'elle se heurte à de nombreuses limitations, le plus souvent générées en amont de cet outil, cette extension démontre la faisabilité technique d'une réponse à la demande sous la forme d'une extension QGIS. Un certain nombre de problèmes techniques à résoudre ont été précisément identifiés. Les développeurs souhaitant fournir une réponse robuste et complète à la demande initiale pourront chercher à résoudre ces problèmes, et réutiliser le code source de l'extension développée au cours de ce stage, qui est en libre distribution sur la plateforme de développement et de partage de code GitHub.

Thésaurus – **QGIS, SIG, Open Source, OGC, Sensor Observation Service, séries temporelles, mesures *in situ*, énergies renouvelables**

Abstract

Time-series of in-situ measurements are used to obtain precise knowledge – both in time and space – of the environment's physical values such as surface solar irradiance. These measurements are recorded in the form of digital data, which are stored and shared using Information and Communication Technologies (ICT). In particular, the web service Sensor Observation Service (SOS) – built with open standards and therefore highly interoperable - has been proposed by the Open Geospatial Consortium (OGC) to encode these data, which can be stored in servers according to the SOS specifications. Clients for the SOS interface allow users to communicate with SOS servers. As in-situ measurements data include information on the measurement location, those clients can generate vector layers to display all measurement stations relative to an SOS server. Although most of those clients are web clients, there has been a clear demand for a client which can be used in a Geographic Information System (GIS) in order to retrieve, visualize, and then download time-series of in-situ measurements data stored in SOS servers. This internship led to the development and the release of a Python plugin for the Open Source GIS QuantumGIS (QGIS) which aims at contributing to respond to such demand. This plugin was tested on a set of 10 SOS servers, 4 of which showed a perfect compatibility with all its different features, with a high level of compatibility as to the remaining 6 servers. This tool, whose development is still in progress and which faces a few limitations mostly due to those of the dependencies, shows that a response to the demand is technically possible using QGIS. Several of remaining technical issues have been precisely identified: developers who wish to bring a robust and complete response to the initial demand can try to find solutions and re-use the source code which was developed during this internship, as it is available freely on the open-source code sharing platform GitHub.

Keywords - **QGIS, SIG, Open Source, OGC, Sensor Observation Service, time series, *in situ* measurements, renewable energy**

Remerciements

Je souhaiterais en tout premier lieu remercier M. Lionel Ménard, qui m'a fait l'honneur d'être mon maître de stage pour ce travail de fin d'études. Il m'a accueilli avec beaucoup d'enthousiasme et de gentillesse, et a su me guider pendant ces 6 mois grâce sa clairvoyance et à une rare hauteur de vue. L'attention qu'il m'a portée et son sens de l'humour m'ont mis à l'aise dès le début de ce stage. J'exprime aussi toute ma gratitude à M. Thierry Ranchin, directeur du Centre Observation, Impacts, Energie (O.I.E.) de MINES ParisTech où j'ai effectué ce stage, pour avoir accepté de m'accueillir dans l'équipe du centre. Il m'a immédiatement placé dans les meilleures conditions pour mener à bien ce travail. Je remercie également Mme Sandra Hassan pour son aide dans mes démarches administratives et pour son accueil.

Je ne remercierai jamais assez M. Philippe Blanc, directeur de recherche au centre, pour sa disponibilité à toute épreuve, et pour son incroyable bienveillance. La curiosité scientifique qui l'anime, le volume et la profondeur de ses connaissances, ajoutées à la générosité dont il fait preuve au quotidien, en font un exemple pour moi. Un grand merci également à M. Benoît Gschwind pour m'avoir partagé sa science et sa passion de la programmation avec pédagogie, et pour toute l'assistance qu'il m'a gentiment offerte lorsque des difficultés techniques se sont présentées. J'adresse ensuite toute ma reconnaissance à M. Lucien Wald pour les enseignements sur le rayonnement solaire et les précieux conseils méthodologiques qu'il m'a apportés, ainsi que pour sa sympathie.

Sans oublier mes compagnons de bureau : Jean, Mireille, et Mélodie, qui m'ont, entre autres, permis de prendre un recul précieux sur mon travail. J'ai beaucoup appris à leurs côtés. Leur bonne humeur et leur écoute m'ont offert un cadre de travail des plus enthousiasmants.

Toute ma gratitude à tous les autres membres de l'équipe du Centre Observation, Impacts, Energie, pour l'accueil qui m'ont offert : Yves-Marie, Isabelle, Paula, Romain, Marc, Loïc et Marc.

Je souhaite également remercier l'EIVP, mon École, et Mme Nassopoulos, responsable des stages, pour avoir rendu ce stage possible. J'adresse des remerciements particuliers à Mme Cécile Haeck, ma tutrice de stage, pour sa disponibilité et pour les corrections qu'elle m'a proposées au cours du stage ; ainsi qu'à M. Chachoua, enseignant-chercheur à l'EIVP, qui a accepté de se porter président du jury de ce travail de fin d'études.

Enfin, j'adresse un grand merci à Stéphane, mon oncle, et Céline, ma tante, pour m'avoir accueilli sur la Côte d'Azur et rendu mille services pendant ces 6 mois. Merci à ma sœur pour avoir apporté ses corrections à mon abstract, et à mes parents pour leur relecture et leurs précieux encouragements.

Table des matières

Table des figures

Liste des tableaux

Introduction

Pour étudier la façon dont notre environnement évolue, on peut utiliser des séries temporelles de mesures de grandeurs physiques - comme la température au sol, la vitesse du vent, etc. - effectuées localement. On parle de mesures *in situ*. Ces mesures fournissent des données numériques. Ce sont des données d'observations de la Terre. Les Technologies de l'Information et de la Communication (TIC) permettent de partager ces données numériques, afin qu'un plus grand nombre d'individus puissent les exploiter. L'Open Geospatial Consortium (OGC) propose un grand nombre de standards pour l'encodage et la mise à disposition de données d'observation de la Terre. Parmi ces standards, le standard Sensor Observation Service (SOS) permet l'encodage et la mise à disposition sous forme de service Web de mesures *in situ*. Des serveurs SOS utilisent des solutions d'implémentation du standard SOS pour permettre le stockage et l'encodage de séries temporelles de mesures *in situ*. Le service SOS rend également possible la visualisation et le téléchargement de ces séries temporelles. Du point de vue du développement informatique, il est alors possible de créer des interfaces numériques, appelés clients, qui communiquent avec les serveurs SOS, afin de permettre à des utilisateurs de visualiser puis télécharger les séries temporelles de mesures *in situ* stockées sur ces serveurs. Les données de mesures *in situ* incluent la localisation des stations où elles ont été effectuées. Les clients développés utilisent cette information géospatiale pour afficher la carte des stations de mesures associées à un serveur SOS. Théoriquement, de telles cartes peuvent également être chargées dans des Systèmes d'Information Géographique (SIG). Dans le cadre de sa contribution à l'initiative Global Earth Observation System of Systems (GEOSS), qui s'inscrit dans son travail de dissémination des données d'observations de la Terre, le Centre Observation, Impacts, Energie (O.I.E.) de MINES ParisTech s'est renseigné sur l'existence d'un tel client dans le SIG Open Source QuantumGIS (QGIS). N'ayant pas trouvé de client satisfaisant, le Centre O.I.E. s'est interrogé sur l'existence d'une demande quant à l'accès, via un client dans QGIS, à des données de séries temporelles de mesures *in situ* stockées dans des serveurs SOS. Une telle demande existe. Ce stage a contribué à y répondre par le développement d'une extension (un plugin) QGIS. Il s'est agi de répondre à la problématique technique suivante : « *Comment développer une extension QGIS pour permettre l'accès, la visualisation, et le téléchargement des données de mesures in situ stockées dans des serveurs d'implémentation du standard SOS de l'OGC ?* ». Le développement de cette extension QGIS a été effectué en langage Python. En premier lieu, la démonstration a été apportée que les fonctionnalités attendues de l'extension peuvent être générées en langage Python. Ensuite, une extension QGIS a été développée. La librairie Python PyQt a été utilisée pour le développement d'une interface graphique, et l'API QGIS a permis de faire le lien entre cette interface et les fonctions Python créées pour donner accès aux fonctionnalités de l'extension. L'extension développée, baptisée « SOS 2.0 Client », a ensuite été publiée dans le dépôt officiel des extensions Python de QGIS, et l'intégralité du code source a été mise à la disposition de tous, afin d'encourager les contributions à l'amélioration du plugin QGIS publié et l'apport d'une réponse technique à la demande initiale.

1 Contexte général

1.1 Présentation du Centre Observation, Impacts, Énergie (O.I.E.)

Situé dans la technopole de Sophia-Antipolis, le Centre Observation, Impacts, Énergie (O.I.E.) se définit comme "une équipe de recherche commune MINES ParisTech – ARMINES, [comptant une quinzaine de chercheurs et doctorants], dont les activités se situent au carrefour de l'énergie, de l'environnement et de l'observation de la Terre." [1] Au centre O.I.E. sont étudiées les ressources en énergie renouvelable et les impacts environnementaux liés à leur exploitation. Les travaux de recherche menés au sein du centre s'appuient sur des disciplines de sciences fondamentales (mathématiques, physique,...) et appliquées (métrologie, sciences de l'environnement,...). Afin de rendre accessibles les résultats de ses travaux à la communauté scientifique, aux industriels, aux instances décisionnelles, ainsi qu'au grand public, le centre s'appuie sur les Technologies de l'Information et de la Communication (TIC). Concrètement, cela se traduit par le fait que "les bases de données et [des] services Web [associés] constituent pour l'équipe un des principaux vecteurs de la dissémination de ses connaissances." [1]

Les activités du Centre "Observation, Impacts, Énergie" de MINES ParisTech contribuent à la valorisation et au développement des énergies nouvelles et renouvelables. Son expérience et son expertise dans le domaine de l'observation de la Terre - historiquement, le Centre O.I.E. a pris la succession du Centre de Télédétection de MINES ParisTech - lui permet d'investiguer des problématiques de recherche intégrant "les aspects temporels et spatiaux (i.e. géographiques)" [1] à haute résolution. Pour traiter de ces aspects au mieux, le centre développe des outils à même de "représenter mathématiquement, au moyen de l'observation, de la modélisation et de la décision, [la] réalité géographique" [2] qui intervient lors de l'évaluation de la ressource en énergie renouvelable et lors de l'évaluation des impacts environnementaux liés à l'exploitation de cette ressource pour la filière de l'énergie.

Les activités du centre sont donc structurées autour de deux thématiques [1] :

1. L'évaluation de la ressource énergétique ;
2. L'évaluation des impacts environnementaux

Évaluation de la ressource énergétique

Le Centre O.I.E. a acquis une expertise particulière dans l'estimation de la ressource solaire. Les technologies de cartographie satellite à haute résolution du potentiel solaire exploitées au centre, ainsi que les modèles qu'il développe pour décrire le rayonnement solaire, aboutissent à la réalisation

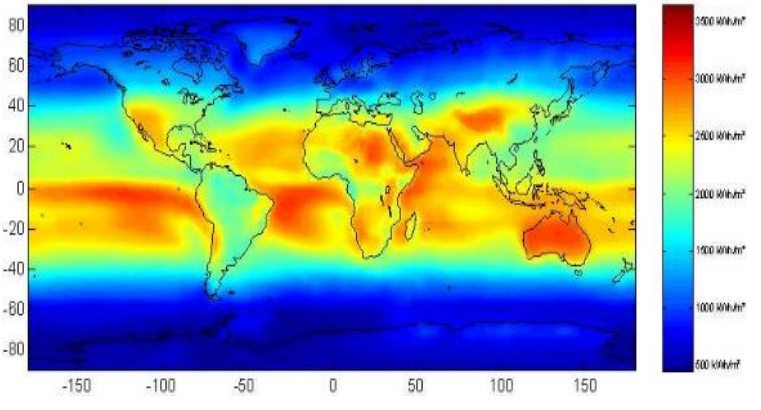


Figure 1-B. Espinar, P. Blanc, L. Wald
Distribution spatiale du rayonnement annuel
(© MINES ParisTech, 2006). [29]

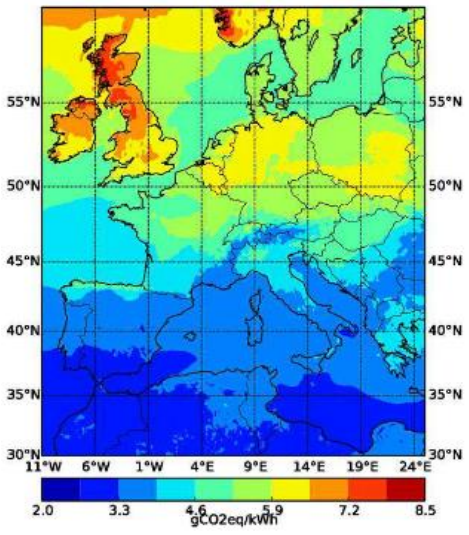


Figure 2 -C. Marini, I. Blanc, P. Sinha et al. Carte d'empreinte carbone pour évaluer les impacts environnementaux de la filière photovoltaïque. [30]

d'atlas solaires et de cadastres solaires [2]. Le centre mène aussi des projets d'estimation de la ressource pour le domaine de l'éolien [2] et les énergies marines [3].

Évaluation des impacts environnementaux

Le Centre O.I.E. travaille également à l'évaluation des impacts environnementaux issus de l'exploitation des énergies renouvelables. Il cherche à fournir des méthodologies fiables d'évaluation, notamment en favorisant "l'approche Analyse de Cycle de Vie (ACV)", et "en intégrant une meilleure prise en compte des aspects spatio-temporels" [2]. Ces recherches concernent un large éventail de filières énergétiques qui comprend notamment "le photovoltaïque, la filière solaire thermique, l'éolien et la géothermie." [2]

Diffusion numérique des connaissances

Comme expliqué plus haut, le centre a choisi d'exploiter les TIC pour la gestion et la diffusion de la connaissance résultant de ses travaux. En particulier, il a développé une Infrastructure de Données Spatiales (IDS), accessible via le service Web webservice-energy.org, opérationnelle depuis 2008, dans le but d'assembler, documenter, diffuser et promouvoir l'utilisation de données d'observation de la Terre au service du développement des énergies renouvelables. Ouverte, standard et interopérable, elle suit les recommandations du Global Earth Observation System of Systems (GEOSS). D'après l'Association Francophone des Utilisateurs de Logiciels Libres (AFUL), l'interopérabilité désigne la « capacité que possède un produit ou un système, dont les interfaces sont intégralement connus, à fonctionner avec d'autres produits ou systèmes existants ou futurs, et ce sans restriction d'accès ou de mise en œuvre ». Cette IDS met à disposition un ensemble de services Web qui sont interopérables - ils sont conçus pour rendre possible leurs interactions avec d'autres applications informatiques existantes ou bien à venir - puisqu'ils s'appuient et suivent les spécifications de standards prévus à cet effet. Cette infrastructure héberge différents services Web qui sont autant d'outils de découverte et de visualisation des résultats des travaux du centre. L'un de ces services Web permet d'accéder via une application sur le Web à des données géolocalisées de mesures *in situ* en lien avec les énergies renouvelables. L'objet de ce stage est de fournir un autre type d'accès à ces mêmes données, comme expliqué dans la section suivante.

1.2 Sujet du stage

1.2.1 Présentation du sujet

L'ensemble des filières industrielles concernées par l'exploitation de l'énergie solaire pour la production d'électricité voit dans l'acquisition d'une connaissance fine des caractéristiques du rayonnement solaire à l'échelle locale un levier d'action important. Les mesures *in situ*, réalisées par des capteurs au sol, représentent un élément clé pour acquérir cette connaissance. Bien que des mesures *in situ* du rayonnement sont potentiellement disponibles, leur accès est trop souvent limité, en partie par des problèmes de propriété intellectuelle, mais aussi par un certain manque de coordination quant à leur mise à disposition dans des formats, des infrastructures et des réseaux qui puissent permettre leur accès au plus grand nombre. L'Open Geospatial Consortium (OGC) propose un grand nombre de standards pour l'encodage et la mise à disposition de données d'observation de la Terre [4]. Parmi ces standards, le standard Sensor Observation Service (SOS) permet l'encodage et la mise à disposition sous forme de service Web de mesures *in situ*. Le Centre O.I.E. s'appuie sur la solution d'implémentation du standard SOS que représente la plateforme 52°North, pour permettre le stockage, l'encodage, la visualisation et le téléchargement de mesures *in situ* du rayonnement solaire via une interface accessible sur Internet. Ces deux dernières fonctionnalités sont disponibles via un client Web.

L'objectif de ce stage est de fournir un accès complémentaire à ces mêmes mesures, en intégrant un client similaire au sein d'un Système d'Information Géographique (SIG). Plus précisément, le choix du centre s'est porté sur le développement d'une extension (un plugin) sur le SIG Open Source QuantumGIS (QGIS). Nous tenterons donc de répondre à la problématique suivante :

« *Comment développer une extension QGIS pour permettre l'accès, la visualisation, et le téléchargement des données de mesures in situ stockées dans des serveurs d'implémentation du standard SOS de l'OGC ?* »

1.2.2 Lien avec l'approche des TIC adoptée par le Centre O.I.E.

Le Centre O.I.E. a choisi de développer une IDS, webservice-energy.org, ouverte, standard, et interopérable ; pour la dissémination et la communication de ses travaux. Elle est ouverte au sens où l'accès par des personnes et/ou des ordinateurs aux ressources et données qu'elle héberge – accès qui s'effectue via Internet - est totalement libre. Le développement de l'extension QGIS s'inscrit dans le prolongement de cette approche, de même que le choix des outils et des technologies sur lesquelles elle s'appuiera.

Dans la continuité, ce stage vise au développement d'un outil de diffusion de données encodées selon le standard SOS de l'OGC. Le Centre O.I.E. est, depuis 2011 [5], membre de l'OGC, dont la mission première est de "coordonner le développement de standards ouverts pour l'encodage des données géospatiales" [6].

L'OGC s'est également rapproché [6] de l'Open Source Geospatial Foundation (OSGeo) - "dont la mission est de soutenir et de promouvoir le développement collaboratif des technologies et des données Open Source géospatiales"

[6] - en vue de faire converger les activités des deux organisations. Le développement Open Source garantit, à travers le libre accès à l'intégralité du code source, et de par la nature des licences (libres) sous lesquelles sont développés les programmes, la liberté d'exécuter ces derniers librement, d'en étudier et modifier le fonctionnement, de redistribuer ces programmes, et de distribuer librement à autrui des copies de versions modifiées [7]. Aussi, le Centre O.I.E. est sensible à l'intérêt de favoriser des outils Open source. Ce stage participe au développement d'un tel outil. Il doit permettre l'accès à des données de mesures *in situ* depuis un logiciel Open Source : QGIS. Au cours du développement, l'accès à des données stockées par le serveur SOS interne au Centre O.I.E. sera testé en priorité. Or ce serveur utilise une solution d'implémentation Open Source du standard SOS, comme nous le détaillerons en **1.3.2**.

1.2.3 Nature de l'outil logiciel

En se plaçant du point de vue de l'utilisateur, le plugin QGIS qui est l'objet de ce stage sera un outil logiciel à la fois horizontal et vertical :

- Vertical si l'on envisage les usages très ciblés qu'en feront des membres du Centre O.I.E. : la visualisation des données de mesures *in situ* pour les énergies renouvelables, et en particulier les mesures du rayonnement solaire. La petite taille de la communauté des utilisateurs du standard SOS pour le domaine des énergies renouvelables nous permet de la qualifier de "niche". Ce terme est encore davantage approprié pour qualifier la fraction de cette communauté qui utilise QGIS et qui est susceptible d'utiliser le plugin.
- Horizontal puisque, grâce à l'utilisation des standards et à l'interopérabilité en découlant, le plugin répondra aux besoins d'une communauté bien plus large. Il doit permettre d'afficher, de visualiser et de télécharger dans QGIS des données de mesures *in situ* encodées selon le standard SOS, et hébergées au sein d'un serveur SOS. Or, les données répondant à de tels critères ne se limitent pas au domaine des énergies renouvelables. Le standard SOS permet d'encoder toutes sortes de mesures *in situ* qui s'agrègent sous la forme de séries temporelles. La société hollandaise Geonovum utilise par exemple le standard SOS pour diffuser des données sur la qualité de l'air et la qualité de l'eau.

1.3 Diffusion des données de mesures *in situ*

1.3.1 Application au domaine du rayonnement solaire

Une connaissance précise du rayonnement surfacique (en anglais *surface solar irradiance*, SSI) est indispensable à la promotion de la filière de l'énergie solaire. Le rayonnement solaire surfacique, exprimé en $W.m^{-2}$, varie spatialement sur la surface de la Terre, et également au cours du temps. La connaissance du rayonnement est *de facto* soumise à des résolutions spatiales et temporelles.

Au sein de la communauté scientifique, les chercheurs travaillant à une meilleure connaissance de la ressource en énergie solaire tentent de décrire le rayonnement à la surface de la Terre le plus fidèlement possible. Trois moyens d'observation de la Terre sont utilisés afin de quantifier le rayonnement solaire surfacique [8] :

1. Les capteurs pyranométriques *in situ*. Ils fournissent des mesures du rayonnement solaire surfacique qui sont prises au sol en un point, et intégrées sur une durée variable. Cette dernière vaut généralement 1 minute, 10 minutes ou bien 1h. Des trois outils d'estimation du rayonnement, il s'agit de celui qui fournit la mesure la plus précise. Cette mesure fait donc office de référence et sert d'étalon pour les deux autres méthodes de quantification du rayonnement. On utilise généralement ces capteurs pour prendre des mesures successives, à pas de temps régulier, générant ainsi des séries temporelles de mesures *in situ*.

2. Le traitement de données provenant d'images satellite réalisées par des satellites géostationnaires, qui aboutit à la production de bases de données du rayonnement solaire. Le capteur du satellite mesure le flux lumineux réfléchi par la surface terrestre et l'atmosphère dans le domaine du visible et de l'infrarouge en utilisant la réflectance lumineuse bidirectionnelle du sol. On en déduit ensuite le rayonnement solaire en surface de la Terre. Le traitement des images satellite permet d'obtenir une grille d'échantillonnage de haute résolution temporelle (1 image reçue toutes les 15 minutes pour le satellite européen Météosat) et de résolution spatiale moyenne (3km de côté pour un pixel à l'équateur avec Météosat) couvrant toute la surface terrestre, régulière dans l'espace et dans le temps [9]. Malheureusement, malgré les avancées techniques permettant de la réduire progressivement, la résolution spatiale des images satellite reste limitée, du fait de la distance séparant le satellite du sol (environ 36 000 km pour Météosat), ce qui signifie que l'estimation fournit une valeur unique sur toute cette surface. En un point donné de cette dernière, l'erreur entre le rayonnement réel et son estimation peut être importante, l'ombre portée de la couverture nuageuse aussi bien que l'effet d'ombrage issu de la nature du terrain pouvant générer une grande variabilité naturelle du rayonnement sur la zone. Notons que les valeurs obtenues par ces méthodes sont calibrées à partir de mesures *in situ*.

3. Les modèles météorologiques numériques, obtenus via la méthode des éléments finis. Ici, un élément est semblable à un prisme carré droit d'une surface de l'ordre de 20km x 20km typiquement, et de hauteur variable selon les strates de l'atmosphère considérées. Ces modèles intègrent des données provenant des deux outils précédents, et, le plus souvent, des mesures *in situ*. Ils permettent de décrire, avec une incertitude connue, le rayonnement sur l'ensemble de la surface terrestre, et sur une période temporelle de plusieurs décennies.

Les chercheurs ont besoin d'accéder au plus grand nombre de mesures *in situ* possible dans l'espace et dans le temps pour étalonner les modèles météorologiques numériques d'estimation et de prévision du rayonnement, ainsi que sa cartographie obtenue à partir d'images satellites. Les stations qui diffusent la plupart des données de mesure *in situ* sont des stations météorologiques. Malheureusement, leur densité est faible et très inégale sur le territoire, si bien que la variabilité spatiale du rayonnement ne peut être restituée par leurs mesures [8]. Par ailleurs, les pyranomètres, qui produisent les mesures *in situ*, sont très coûteux à entretenir, ce qui freine le développement des stations [8].

S'ajoute à cela la couverture temporelle incomplète des jeux de données de mesure *in situ* partagés par les stations météorologiques [8].

Les industriels, tout comme les décideurs publics du secteur de l'énergie, ont, eux-aussi, besoin de connaître précisément le rayonnement solaire surfacique pour mener à bien des projets d'installation de centrales à énergie solaire - centrale photovoltaïque ou solaire thermique à concentration - mais aussi pour assurer le bon déroulement de l'exploitation de ces infrastructures. Prenons l'exemple d'une centrale photovoltaïque. Lors des études d'avant-projet, l'emplacement qui accueillera ces infrastructures est d'abord déterminé. Les responsables du projet d'implantation font d'abord appel à un SIG, pour proposer un certain nombre d'emplacements potentiels. Ils superposent des cartes d'occupation des sols dans le but d'éliminer des zones proscrites, la carte des postes électriques - afin de s'affranchir de zones qui rendraient le raccordement électrique de la centrale trop coûteux - et des cartes du rayonnement.

Une fois ce repérage régional effectué, les responsables du projet doivent déterminer l'emplacement de l'installation de manière plus précise. Ils espèrent trouver le site recevant le meilleur rayonnement en vue d'obtenir la meilleure production électrique possible dans la durée. Lors de leur travail d'estimation du rayonnement, ils font face aux limitations énoncées précédemment : incapacité des données satellites à rendre compte de la variabilité spatiale naturelle du rayonnement, et trop grande distance aux sources de mesures *in situ* disponibles pour pouvoir négliger les biais systématiques en cas d'intégration de ces mesures aux modèles. Pour pallier ces problèmes, les gestionnaires des centrales à énergie solaire existantes ont installé leurs propres capteurs pyranométriques. Pour affiner le maillage du réseau de stations de mesure, il a récemment été envisagé de diffuser les données produites par ces capteurs. Si la volonté des industriels de partager leurs données ne semble pas constituer un obstacle *a priori* [8], il est nécessaire de leur fournir des outils permettant ce partage. Le plugin QGIS qui doit être développé au cours de ce stage s'inscrit dans le prolongement d'un travail du Centre O.I.E. participant de cette démarche. Encore récemment, le prix de l'électricité en sortie des centrales à énergie solaire garantissait une certaine rentabilité économique, ce qui permettait aux développeurs d'un projet de ne pas tenir compte de ces biais jusqu'au moment où ils démarchaient les banques pour le financement du projet. Désormais, la tendance d'évolution du prix de l'électricité en sortie des centrales à énergie solaire est à la baisse. Les industriels entreprennent des estimations précises de la production en amont du projet pour éviter une surévaluation de la production d'électricité qui mettrait en péril le retour sur investissement. Pouvoir accéder aux données de mesures *in situ* récoltées par les stations alentours déjà en fonctionnement les aiderait à mieux sélectionner le site d'installation afin d'assurer la bonne santé financière du projet.

Une fois le site d'implantation choisi, les industriels doivent financer le projet. L'essentiel des dépenses pour un tel projet est concentré à son démarrage : la récolte de l'énergie depuis sa source est gratuite, mais les infrastructures sont coûteuses, à l'inverse de ce que l'on observe avec les projets de centrales à charbon, par exemple. Généralement, les industriels font appel à un emprunt bancaire pour financer les infrastructures, lequel emprunt sera remboursé grâce

à la vente de l'électricité produite. Afin d'évaluer l'intérêt d'une participation au financement d'un projet de centrale solaire, les banquiers tiennent compte de la durée de vie des centrales - qui s'étale fréquemment sur plus de 20 années - et ils évaluent la solvabilité de la dette à partir d'une estimation des rentrées financières du projet à échéances régulières. En tenant compte de nombreux aléas (pannes et baisses de rendement des capteurs, déconnexions du réseau électrique, aléas climatiques et météorologiques, etc.) et en se basant sur des données décrivant le rayonnement sur le site d'installation, elles évaluent le risque d'un financement du projet. En fonction de celui-ci, les banques peuvent proposer des taux d'intérêts différents. Pour un responsable de projet d'installation d'une centrale solaire négociant un prêt bancaire, disposer des données de mesures *in situ* du rayonnement récoltées dans les centrales déjà en fonctionnement, et situées à proximité du site choisi pour le projet, peut donc être avantageux. Malheureusement, les industriels se heurtent à "un manque d'un cadre commun qui, organise au sein d'une infrastructure numérique interopérable, la collecte, le stockage, le traitement et la diffusion des données de mesures *in situ*" [8]. Le Centre O.I.E. a récemment œuvré à la création d'un tel cadre permettant une plus grande diffusion des données de mesures *in situ*. Le développement du plugin QGIS s'insère dans la continuité de ce travail.

1.3.2 Diffusion des données de mesures *in situ* par le Centre O.I.E.

Le Centre O.I.E. a contribué à répondre au problème de la diffusion des données de mesures *in situ* pour le domaine des énergies renouvelables, dans le cadre du projet européen (Horizon2020) ConnectinGEO, financé par la Commission Européenne. Un des objectifs de ce projet est de "faire en sorte que les données [de mesures *in situ*] puissent s'échanger de la manière la plus respectueuse des standards internationaux et du concept d'interopérabilité" [5].

Comme indiqué plus haut lors de sa présentation, le Centre O.I.E. a également développé un client Web pour diffuser les données géolocalisées de mesures *in situ* liées aux énergies renouvelables. La pertinence de l'utilisation de cette plateforme et de ce client Web pour le stockage, l'accès, la visualisation, et le téléchargement de mesures *in situ* par les industriels a été démontrée à travers la collaboration avec la PME Solaïs dans le cadre du programme AVENE PME (Avenir Énergie PME) financé par les Investissements d'Avenir. Une vidéo, contenant notamment le témoignage de Solaïs, présente plus en détail ce projet.

1.4 Développement d'un plugin QGIS

1.4.1 Justification du choix de QGIS

Le choix du logiciel QGIS comme SIG se fonde sur ses atouts comparatifs. Parce que la communauté qui œuvre à son développement est très active et qu'il est devenu très complet, QGIS est le SIG Open Source le plus répandu. QGIS est gratuit et peut être téléchargé librement, ce qui est un autre avantage pour la communauté ayant besoin d'outils pour superposer des cartes de la ressource en énergie renouvelable et des séries temporelles de mesures *in situ*, et souhaiterait disposer des fonctionnalités du plugin dans un SIG.

Un autre avantage de QGIS est qu'il permet la mise à disposition des nouvelles fonctionnalités par le biais d'un outil conçu pour être particulièrement accessible aux développeurs. Cet outil est le *plugin*, une extension développée indépendamment du SIG que l'on soumet à la communauté des utilisateurs de QGIS. Les bonnes pratiques pour le développement d'une telle extension sont détaillées au sein d'une documentation officielle très complète.

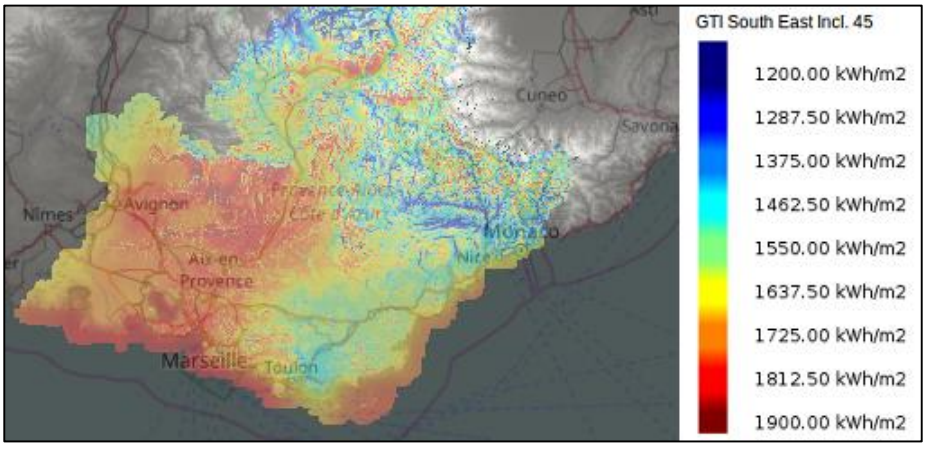


Figure 3 -Visualisation sous QGIS de l'atlas solaire de la région Provence-Alpes-Côte d'Azur
Provenance des données : Atlas solaire : Centre O.I.E. (couche WMS), topographie : NASA, fond de carte : OpenStreetMap

Enfin, l'usage de ce plugin, combiné aux fonctionnalités de base de QGIS, offrira un accès simultané à des cartes et à des séries temporelles de mesures *in situ*. Le choix de QGIS comme SIG se révèle pertinent lorsque l'accès à de telles ressources cartographiques s'effectue non pas depuis une machine locale, mais à distance, par l'intermédiaire d'un service Web. Citons deux services Web pour illustrer cette fonctionnalité : Web Mapping Service (WMS) et Web Feature Service (WFS). Le premier permet de récupérer en flux continu des données raster géoréférencées restituées sous forme d'images de cartes envoyées par le serveur WMS [10]. A titre d'exemple, c'est via un service WMS mis à disposition par le Centre O.I.E. qu'a pu être visualisé dans QGIS l'atlas solaire de la région Provence-Alpes-Côte d'Azur qu'il a produit, comme le montre la **figure 3**. À la différence d'un WMS, un Web Feature Service permet d'obtenir l'envoi par un serveur WFS des couches vecteurs d'objets géographiques ainsi que les attributs les caractérisant.

1.4.2 Demande

Nous venons de le voir, au sein de la communauté des utilisateurs du standard SOS pour l'encodage et la mise à disposition de données de mesures *in situ*, un certain nombre de membres ont besoin d'accéder simultanément à ces données et à des cartes. C'est en particulier le cas des utilisateurs travaillant dans le domaine des énergies renouvelables, pour lesquels ce double accès permet de tirer profit des aspects temporels et géospatiaux de l'information à disposition. À l'heure actuelle, il n'existe pas d'outils satisfaisants pour l'accès aux données de mesures *in situ* stockées sur un serveur SOS depuis un SIG. C'est à ce besoin que doit répondre le plugin dont le développement est l'objet de ce stage.

Plusieurs organismes de recherche et entreprises, avec lesquels nous avons eu l'occasion d'échanger, et qui utilisent le standard SOS pour rendre accessibles des données géolocalisées de mesures *in situ* via un service Web, ont explicitement manifesté un intérêt à l'égard du plugin QGIS. Il s'agit des organismes suivants :

- 52°North, regroupement d'entreprises et de laboratoires publics de recherche, fondé en Allemagne ;
- BRGM, Bureau de Recherches Géologiques et Minières, organisme public de recherche français ;
- Geonovum, l'organisme national hollandais en charge d'établir la dissémination des données géospatiales au sein du secteur public [11] ;
- NIWA, National Institute of Water and Atmospheric Research of New Zealand, institut national public de recherche néozélandais ;
- SUPSI, University of Applied Sciences and Arts of Southern Switzerland, Earth Science Institute, Division of Geomatics, laboratoire public de recherche suisse ;
- University of Geneva, enviroSPACE lab., laboratoire public de recherche suisse ;
- University of Santiago de Compostela, CiTIUS, laboratoire public de recherche espagnol.

1.4.3 Fonctionnalités attendues du plugin

Chaque mesure *in situ* encodée selon le standard SOS se voit doter d'attributs parmi lesquels la station de provenance, le capteur ayant généré la mesure, ainsi que la grandeur physique mesurée. Nous détaillerons plus longuement la structure des données encodées selon le standard SOS dans la partie **2.1**. Remarquons qu'un serveur SOS donné peut abriter des données provenant de différentes stations de mesure.

A partir de cette structure de données, et en suivant les recommandations de l'OGC et de son initiative Sensor Web Enablement (SWE), le Centre O.I.E. a identifié les fonctionnalités utilisateurs suivantes pour le client que nous avons évoqué plus haut [8] :

- Sélectionner un serveur SOS ;
- Géolocaliser une station parmi celles qui se situent dans une *bounding box* adaptée au niveau de zoom d'une couche cartographique ;
- Sélectionner un jeu de données correspondant à une série temporelles de mesures *in situ* à partir des attributs structurant de la mesure ;
- Accéder aux métadonnées portant sur le serveur SOS sélectionné en entrée par l'utilisateur ;
- Visualiser la série temporelle sous forme de tableau ou bien de graphique ;
- Télécharger le jeu de données de mesures *in situ* sur sa propre machine.

Ces mêmes fonctionnalités sont attendues pour le plugin. Elles visent d'abord à présenter à l'utilisateur l'ensemble des stations et des jeux de données à sa disposition, puis à lui permettre d'en sélectionner, et d'évaluer

l'intérêt de la donnée à l'aide d'outils de visualisation. A l'issue de cette phase d'évaluation, il pourra télécharger la donnée le cas échéant.

1.4.4 Démarche de développement

La démarche de développement de l'extension QGIS adoptée pour ce stage est basée sur l'échange et le partage. Comme expliqué en **2.4.2**, nous avons identifié des institutions intéressées par cet outil, voire même par une contribution à son développement. Au cours du stage, nous les avons tenues au courant de notre avancement par email dans le but de recueillir leurs recommandations. Nous avons effectué ce même travail de communication auprès de la liste de diffusion (email) du forum Sensor Web Community 52°North, qui rassemble de nombreux utilisateurs de la solution développée par 52°North pour l'implémentation de services intégrés au "Sensor Web" (l'Internet des capteurs). SOS est l'un de ces services.

Nous avons aussi souhaité échanger avec l'ensemble des développeurs qui souhaiteraient contribuer à l'amélioration du plugin. C'est pourquoi nous avons mis la totalité du code source du plugin à la disposition de tous. Il a été distribué au sein d'un dossier GitHub. Nous avons ainsi suivi le cadre de travail d'un développement Open Source, et les attentes de l'OSGeo quant à la soumission d'une nouvelle extension QGIS à la communauté [12].

2 Développement d'un script en Python permettant de récupérer, visualiser et télécharger les données de mesures *in situ*

Le plugin permettra de visualiser et télécharger des données hébergées par des serveurs SOS. L'interface de service Web que définit le standard SOS permet la communication à de tels serveurs via le protocole HTTP. Ce protocole est du type client-serveur. Cela signifie que pour récupérer les données d'un serveur SOS, une machine, qualifiée de client, lui envoie des requêtes. Le serveur est continuellement "à l'écoute" de telles requêtes. Lorsqu'il reçoit une requête, il l'intègre, la traite, et retourne une réponse au client. Le standard SOS, qui définit la structure d'encodage en XML des données hébergées sur le serveur, définit également la forme du contenu en XML des différentes requêtes et réponses. Les extensions QGIS peuvent être développées en langage C++ ou bien en langage Python. La documentation officielle de QGIS souligne que le choix de Python garantit une plus grande simplicité de distribution et de développement [13], ce pourquoi nous avons opté pour ce langage. Avant de créer le plugin à proprement parler, nous avons développé un script Python intégrant l'ensemble des fonctionnalités du plugin à partir de la connexion à un serveur SOS. Ce script sera à la base du fichier principal dans le code source du plugin.

2.1 Schéma d'utilisation envisagé pour le plugin

Pour comprendre les paramètres d'entrées et les résultats retournés par le plugin tel que définit en **1.4.3**, il faut d'abord s'attarder quelques instants sur certains aspects du fonctionnement du service SOS comme service Web, et sur la structure d'encodage des données définie par le standard. Les explications qui suivent sont reprises de l'excellente présentation du modèle standard Observation&Measurement (O&M), mise en ligne par 52°North.

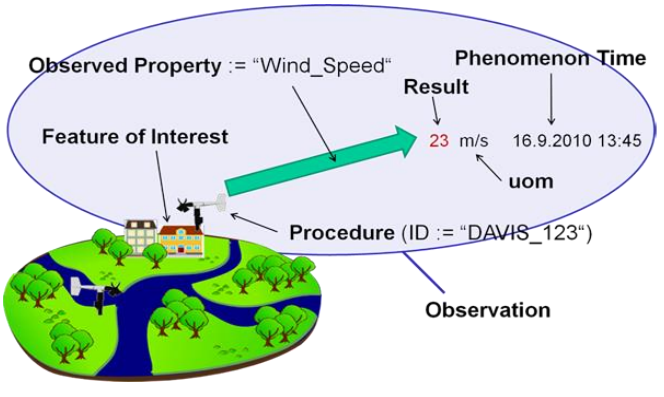


Figure 4 -Open Geospatial Consortium
Illustration du modèle Observation&Measurement (O&M)

Un serveur SOS abrite des données d'observations encodées selon les prescriptions du standard SOS. L'Open Geospatial Consortium (OGC), à l'origine du standard SOS, et du schéma O&M [14] sur lequel ce dernier s'appuie, assimile une mesure à une observation particulière d'un phénomène, à laquelle on associe une valeur numérique. En tant qu'observation, cette mesure est encodée selon le modèle conceptuel de données défini par le standard O&M, dont on peut extraire 6 composantes-clés, comme présenté sur la figure de l'OGC (**figure 4**).

- La source de la mesure est décrite par l'entité *procedure*, qui renvoie généralement à un capteur, mais aussi parfois à un procédé numérique ou à une étape de traitement ultérieur à la mesure en tant que telle ;
- La grandeur physique qui est l'objet de la mesure est encodée via l'entité *observedProperty ;*
- L'objet matériel géolocalisé d'où provient la mesure se voit attribuer l'entité *featureOfInterest*. Il s'agit généralement d'une station d'observation de la grandeur physique mesurée. La localisation associée à la mesure est donc donnée par l'information spatiale et géométrique encodée comme attribut de l'entité *featureOfInterest* ;
- La mesure est décrite par trois entités :
    - une valeur (*result*),
    - une unité (*uom*),
    - une date correspondant à la date à laquelle la mesure a été effectuée (*phenomenon Time* ou *samplingTime*).

Pour résumer, nous pouvons dire que le modèle O&M permet d'encoder le résultat (*result*) de la mesure d'une grandeur physique (*observedProperty*) dans une unité donnée (*uom*). Cette mesure est effectuée à un instant donné (*samplingTime*) par un capteur (*procedure*) situé au niveau d'un emplacement géolocalisé (*featureOfInterest*).

Pour interroger un serveur SOS, des requêtes bien précises sont utilisées selon les schémas du standard SOS fournis par l'OGC. Lors de notre développement, deux des trois requêtes, auxquelles le serveur SOS doit obligatoirement être en mesure de répondre après son implémentation, vont intervenir : la requête *GetCapabilities*, qui permet d'obtenir des informations générales sur le fournisseur du serveur, son contenu et la structure de son contenu, et la requête *GetObservation*, qui permet de récupérer des données d'observations, dans notre cas les données de mesures *in situ*.

En observant à nouveau le schéma d'encodage de la mesure de la **figure 4**, on comprend que lancer une requête de type *GetObservation* suppose d'avoir au préalable fixé 4 paramètres :

- la station (*featureOfInterest*)
- le capteur (*procedure*),
- la grandeur physique mesurée (*observedProperty*)
- la plage de temps (*samplingTime*) pendant laquelle les mesures ont été effectuées.

En réponse à ce constat, j'ai élaboré un premier schéma de principe de l'utilisation du plugin, présenté ci-dessous en **figure 5**. Il permettra de fournir les différentes fonctionnalités attendues du plugin.

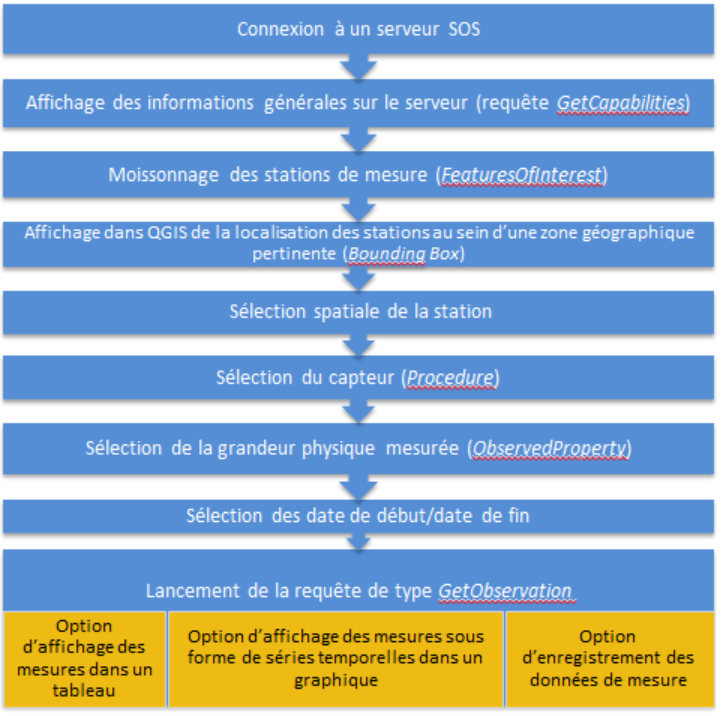


Figure 5 -Principe de fonctionnement (workflow) du plugin

2.2 Récupération des données à l'aide de requêtes XML

Le principe de fonctionnement de la **figure 5** nous montre que le script en Python devra être capable de générer des requêtes de type *GetCapabilities* et *GetObservation*. Avant de créer ce script, il m'a fallu me familiariser avec la notion de service Web et la communication avec un serveur SOS. Pour cela, j'ai effectué des requêtes de type *GetCapabilities* et *GetObservation* en envoyant à plusieurs serveurs SOS des requêtes HTTP Post - codées en langage XML - depuis le terminal de ma machine.

Des exemples de telles requêtes, utilisées pour récupérer des données du serveur déployé par le Centre O.I.E., sont disponibles en **annexe 1**. En envoyant des requêtes de type *GetObservation* à ce dernier serveur, j'ai constaté des messages d'erreur dans les réponses. Les données de mesures *in situ* ne pouvaient pas être récupérées. La réponse retournée par le serveur m'a permis de comprendre qu'il s'agissait d'une erreur dans l'encodage de l'unité associée aux mesures dans la base de données PostgreSQL. Pour permettre le bon calibrage du plugin sur un serveur interne au centre, non entaché de ces erreurs, nous avons décidé que j'installerais mon propre serveur SOS que j'alimenterais de données de mesures *in situ.*

2.3 Création d'un "bac-à-sable" pour le développement

Un délai était nécessaire pour corriger ce problème d'encodage des données stockées dans le serveur SOS du Centre O.I.E. puisque l'ingénieur de l'équipe disposant des compétences techniques pour le résoudre avait un agenda très chargé. Il a donc fallu trouver une solution à moyen terme. En attendant que l'erreur soit corrigée, le serveur né du déploiement de la plateforme 52°North sur ma machine a été utilisé. Puisqu'il s'agit d'un serveur local, j'ai eu accès aux journaux d'enregistrements (ou *log files* en anglais). Ces fichiers contiennent l'enregistrement de tous les évènements affectant le serveur, notamment la réception de requêtes, les opérations de traitement, puis les envois de réponses par le serveur. Cela m'a permis de suivre la façon dont le serveur répond lorsque je lui ai envoyé des requêtes afin d'accéder aux données qu'il héberge. Ainsi, lorsque des erreurs sont apparues, j'ai été en mesure d'identifier leurs origines plus facilement, puis de modifier le code source en conséquence.

2.3.1 Installation d'un serveur Tomcat

L'installation de la plateforme 52°North nécessite de déployer au préalable un serveur Apache Tomcat. Il s'agit d'un serveur d'applications Web Java Open Source. Ce serveur va exécuter les programmes Java de la plateforme, lesquelles génèrent des communications avec d'autres serveurs via le réseau Internet, selon le protocole HTTP. On dit qu'il se comporte comme un serveur HTTP. Le Web manager du serveur Tomcat que j'ai installé, présenté en **figure 6**, permet de déployer puis gérer les applications Web. On peut noter la présence de "52n-sos-webapp" dans les applications déployées. Il s'agit de l'instance de la plateforme 52°North que j'ai installée sur ma machine.

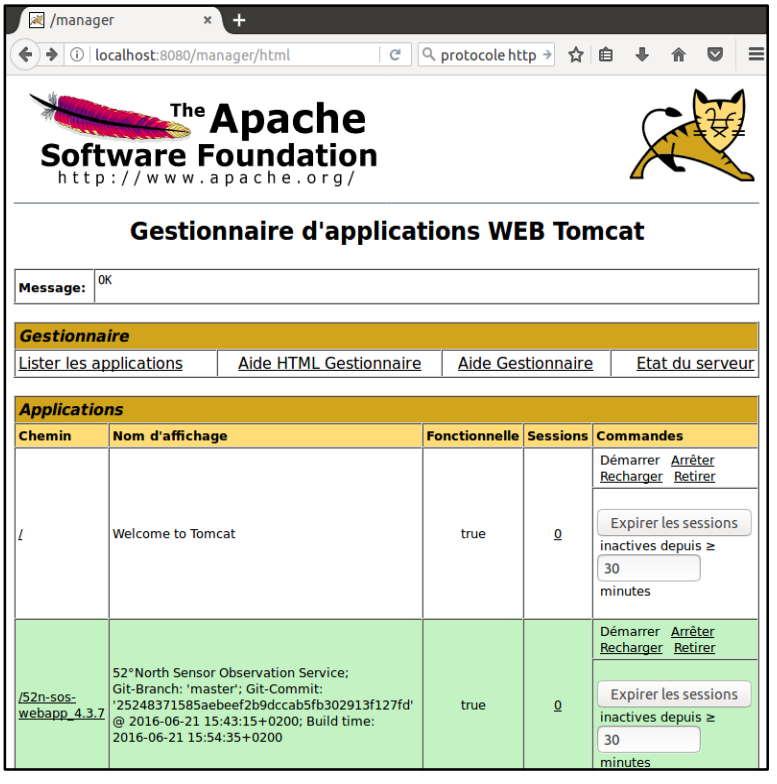


Figure 6 -Web manager du serveur Tomcat installé depuis la machine

2.3.2 Création d'une base de données PostgreSQL

La plateforme 52°North utilise le système de gestion de bases de données PostgreSQL, et plus précisément, son extension PostGIS comme système de gestion de bases de données spatiales. PostgreSQL est un système de gestion de bases de données relationnel-objet (l'information est représentée sous forme d'objets) Open Source très populaire, dont le développement a débuté il y a plus de 15 années. PostGIS est une extension de PostgreSQL qui permet d'inclure l'information géographique ou spatiale dans ses bases de données. Comme nous l'avons vu en **2.1**, le standard SOS identifie une mesure *in situ* à une observation du standard Observation&Measurement. La mesure encodée intègre alors de l'information géographique à travers la classe *featureOfInterest* qui permet de localiser la station de mesure. Pour stocker cette information géographique dans la base de données, la solution d'implémentation du standard SOS 52°North s'appuie sur le type de données 'geometry' de PostGIS. En pratique, la table *feature_of_interest* contient une colonne 'geom' destinée à stocker l'information géographique sur la localisation de la station. Le tutoriel recommandé par 52°North m'a permis de créer la base PostGIS exigée pour l'implémentation de la plateforme. La visualisation de cette base à l'aide de l'outil pgAdminIII – un outil graphique pour l'administration et l'édition des bases de données PostgreSQL - présentée en **figure 7**, permet de retrouver l'ensemble des tables du modèle de données (52°North Sensor Observation Service 4.x database model) utilisées par cette même plateforme.

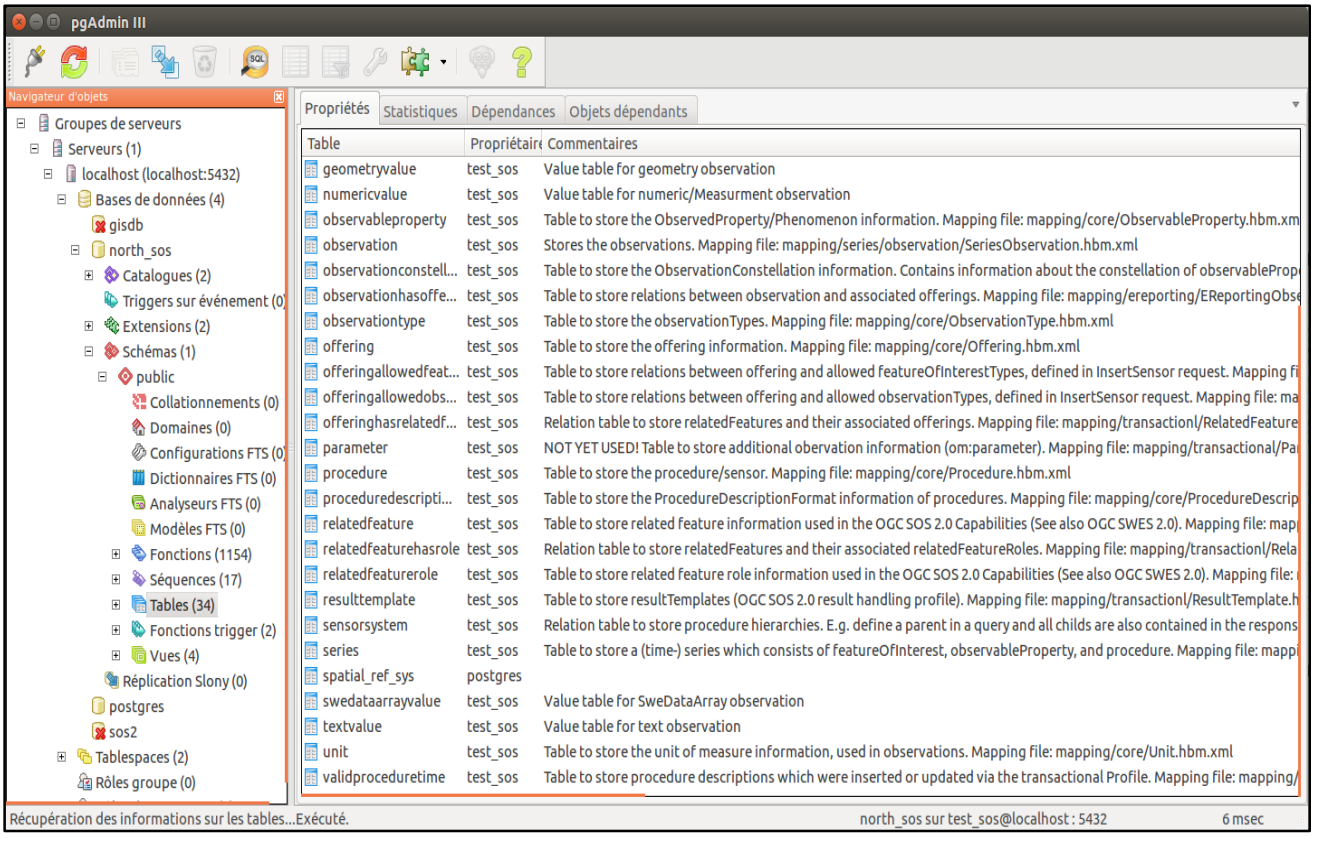


Figure 7 -Visualisation via pgAdmin des tables de la base PostGIS

2.3.3 Installation de la plateforme 52°North et ajout de données dans la base

Une fois le serveur Tomcat installé et la base de données PostGIS créée, la plateforme 52°North a pu être déployée selon les prescriptions de la documentation officielle. La **figure 8** montre le menu de configuration auquel j'ai pu accéder en tant qu'administrateur de la plateforme déployée par notre serveur Apache Tomcat. Il permet d'activer ou de désactiver les opérations d'interaction avec le serveur SOS stipulées par le standard.

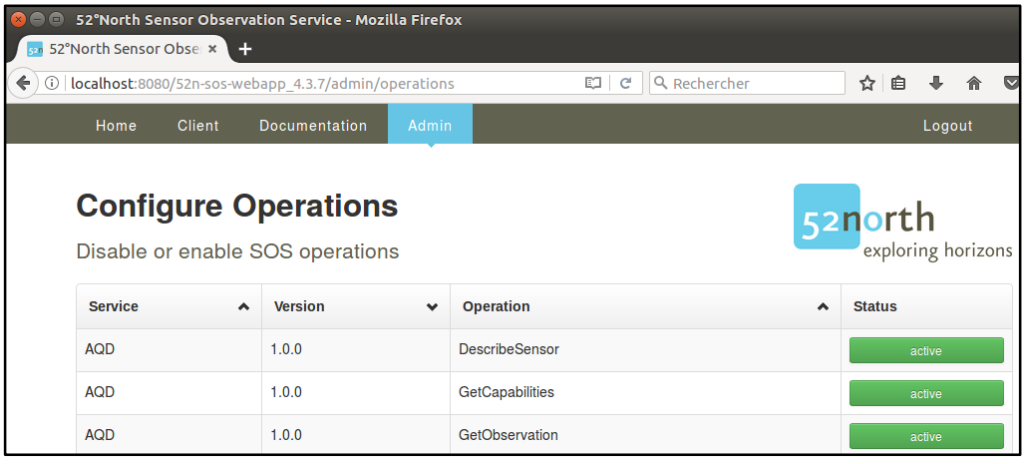


Figure 8 -Menu de configuration des opérations SOS

Pour me permettre de générer des requêtes du type *GetObservation* à destination de mon propre serveur SOS, il m'a fallu ajouter des mesures *in situ* dans la base de données PostGIS. Pour ce faire, j'ai utilisé un outil mis à disposition par 52°North. La **figure 9** permet de visualiser la table *feature_of_interest* de cette base de données, une fois les données de mesures *in situ* ajoutées.

Édition des données - localhost (localhost:5432) - north_sos - public.featureofinterest

Pas de limite

	featureof [PK] bigin	hibern charac	featu bigin	identil charac	code bigir	name character varying(255)	co bi	d cl	geor geor	descriptionxml text
1	1	T	2	http:/	1		1			
2	2	T	2	http:/	1	con terra	1		0101	
3	3	T	2	http:/	1	ESRI	1		0101	
4	4	T	2	http:/	1	Kisters	1		0101	
5	5	T	2	http:/	1	con terra	1		0101	
6	6	T	2	http:/	1	TU-Dresden	1		0101	
7	7	T	2	http:/	1	Hochschule Bochum	1		0101	
8	8	T	2	http:/	1	ITC	1		0101	
9	9	T	2	http:/	1	DLZ-IT	1		0101	
10	10	T	2	http:/	1	Heiden	1		0101	
11	11	T	2	http:/	1	Münster/FE101	1		0101	
12	12	T	2	http:/	1	Portland	1		0101	
13	13	T	2	http:/	1	TODO	1		0101	
14	14	T	1	Propan	1	Propan_Station	1		0101	<ns2:SF_SpatialSamplingFeature xmlns:n
15	15	T	1	Water	1	Water_Station	1		0101	<ns2:SF_SpatialSamplingFeature xmlns:n
16	16	T	1	Krypto	1	Krypton_Station	1		0101	<ns2:SF_SpatialSamplingFeature xmlns:n

Figure 9 -Visualisation via pgAdmin de la table *feature_of_interest* de la base de données PostGIS

2.4 Le script en Python

2.4.1 Choix techniques

Un premier plugin QGIS, présentant les mêmes fonctionnalités que celui que nous cherchons à développer, était déjà disponible lorsque ce stage a débuté. Ce plugin n'a néanmoins jamais fonctionné lorsque nous l'avons testé au Centre O.I.E. Il s'agit d'un outil Open Source, dont l'ensemble est accessible au sein d'un dossier GitHub. Après avoir passé quelques jours sans parvenir à comprendre comment ce plugin récupérait les données de mesures d'un serveur SOS, j'ai proposé à mes encadrants d'abandonner le projet de correction de ce plugin, et d'essayer d'en développer un en repartant de zéro. Cette proposition repose sur le constat classique de la difficulté de reprendre un code écrit par un développeur plus expérimenté, comme c'était le cas ici. Mes encadrants ont accepté ma proposition, et j'ai donc développé une extension QGIS entièrement nouvelle.

Au préalable, il fallait s'assurer qu'il était possible d'offrir les fonctionnalités attendues de l'extension à l'aide d'un code écrit en langage Python. J'ai donc développé un script Python offrant les fonctionnalités attendues du plugin QGIS. Pour cela, j'ai choisi d'utiliser la librairie OWSLib, conçue afin de simplifier la programmation de clients pour l'utilisation des services Web de l'Open Geospatial Consortium [15]. La librairie OWSLib permet notamment de générer l'envoi de requêtes du type *GetObservation* via l'appel d'une de ses fonctions, d'en réceptionner la réponse et d'en extraire les mesures. L'utilisation de cette librairie a donc simplifié le développement en fournissant les moyens d'automatiser les opérations d'écriture et de lecture de requêtes dont le contenu doit respecter un encodage complexe défini par le standard SOS.

La contrepartie de cette utilisation a été l'apparition d'une contrainte forte : l'obligation de travailler avec la version 2.0 du standard SOS pour réaliser des requêtes du type *GetObservation*. Il s'agit de la dernière version du standard, qui apporte un certain nombre d'améliorations par rapport à la version 1.0 [16]. En conséquence, si le serveur interrogé ne présente qu'une compatibilité SOS 1.0, la librairie OWSLib ne permet pas de récupérer les séries temporelles de mesures *in situ* qu'il héberge. Ce qui est fréquemment le cas, car implémenter une nouvelle version du standard pour encoder et stocker les données d'un serveur exige de mobiliser des moyens techniques onéreux et d'investir dans des opérations chronophages. La plateforme développée par 52°North fournit des instances SOS compatibles avec la version 2.0 du standard SOS, comme par exemple celle du serveur du Centre O.I.E., *insitu.webservice-energy.org*. Nous avons donc choisi d'écarter cette limitation pour l'heure, sachant que la libraire OWSLib est Open Source. Son code source pourra donc potentiellement être modifié à l'avenir pour permettre également l'envoi de requêtes du type *GetObservation* à des serveurs SOS 1.0.

2.4.2 Fonctionnement du script et résultats

Le script Python permettant de récupérer, visualiser, et télécharger des séries temporelles de mesures *in situ* stockées dans des serveurs SOS 2.0 est disponible en **[G1]** (voir la liste des liens GitHub en fin de document).

Une fois l'URL de service du serveur SOS sélectionné saisi en entrée du plugin, celui-ci devra en tout premier lieu afficher des informations générales sur le serveur. La fonction getCapabilitiesSOS200 retourne ces informations dans la console Python à partir de la saisie d'une URL appropriée en entrée.

L'utilisateur du plugin devra fournir plusieurs paramètres d'entrée avant de récupérer la série temporelle qui l'intéresse. Depuis l'interface graphique, ces paramètres seront sélectionnés à partir de menus déroulants. Lorsqu'on utilise le script Python que j'ai développé, le choix des paramètres 'station', 'capteur' et 'grandeur mesurée' se fait par la sélection d'un élément d'une liste. L'affichage de l'ensemble des valeurs possibles pour chacun des paramètres a lieu dans la console Python. A l'aide de la librairie OWSLib, je suis parvenu à créer des fonctions Python générant cet affichage pour chacun des paramètres.

L'utilisateur souhaite ensuite connaître la localisation des différentes stations où ont été prises les mesures *in situ* du serveur SOS. La fonction printStations permet d'afficher dans la console Python l'ensemble des localisations à partir de la saisie de l'URL de service d'un serveur SOS. La fonction retourne un quadruplet de coordonnées géographiques : long1, lat1, long2, lat2 (correspondant respectivement aux longitudes et latitudes des deux points d'une paire de sommets opposés d'un rectangle) qui forment une boîte de délimitation (ou *bounding box*). La localisation de la station est ainsi encodée au sein du standard SOS. En pratique, la boîte de délimitation est bien souvent réduite à un point. On le constate en observant la **figure 10,** qui présente le contenu de la console Python résultant de l'appel de la fonction printStations, avec pour paramètre d'entrée l'URL de service du serveur du Centre O.I.E., *insitu.webservice-energy.org*.

```
>>> >>> List of station locations for SOS server http://insitu.webservice-
energy.org/52n-sos-webapp/sos  <<< <<<



>> station [0]:  (71.5, 30.169999999999998, 71.5, 30.169999999999998)
>> station [1]:  (28.8156, 47.0014, 28.8156, 47.0014)
>> station [2]:  (4.9225, 45.7786, 4.9225, 45.7786)
>> station [3]:  (27.99, 27.06, 27.99, 27.06)
>> station [4]:  (5.182, 43.52, 5.182, 43.52)
>> station [5]:  (74.24, 31.69, 74.24, 31.69)
>> station [6]:  (72.98, 33.64, 72.98, 33.64)
>> station [7]:  (32.78, 23.970000000000002, 32.78, 23.970000000000002)
>> station [8]:  (4.927, 51.972, 4.927, 51.972)
>> station [9]:  (71.82, 29.330000000000002, 71.82, 29.330000000000002)
>> station [10]:  (8.1659, 53.1523, 8.1659, 53.1523)
>> station [11]:  (7.199, 43.671, 7.199, 43.671)
>> station [12]:  (5.811157, 43.251327, 5.811157, 43.251327)
>> station [13]:  (5.768, 50.91, 5.768, 50.91)
>> station [14]:  (5.843, 43.857, 5.843, 43.857)
>> station [15]:  (6.1386, 46.1386, 6.1386, 46.1386)
>> station [16]:  (31.282999999999998, 30.083, 31.282999999999998, 30.083)
```

Figure 10 -Réponse de la console Python à l'appel de la fonction 'printStations'

Une fois la station sélectionnée par l'utilisateur, celui-ci attend que le plugin affiche l'ensemble des capteurs de la station. Le fonctionnement de la librairie OWSLib nous conduit cependant à remplacer l'étape de sélection du capteur (*procedure*) par une étape de sélection d'une *offering*. Une *offering* correspond à un groupement d'observations générées par le même système de mesure [17]. Dans le cadre de la version 2.0 du standard, une *offering* ne peut être associée qu'à un seul capteur [16], de sorte qu'il m'a été possible d'effectuer la substitution de l'attribut capteur par l'attribut *offering* sans modifier la logique du processus de sélection des paramètres de la requête *GetObservation*. La fonction printOfferings déclenche l'affichage par la console Python de toutes les *offerings* d'une station de mesures à partir de la donnée de l'URL de service du serveur SOS choisi et de l'index de la station parmi la liste des stations

retournée par printStations. La **figure 11** présente ce qui a été affiché après sélection d'une station située au Caire et dont les données de mesures sont hébergées par le serveur *insitu.webservice-energy.org*.

```
>>> >>> Offering list for station located at (31.282999
999999998, 30.083, 31.282999999999998, 30.083)  <<< <<<



>> offering [0]:  CAIRO-CLS-DHI
>> offering [1]:  CAIRO-CLS-DNI
>> offering [2]:  CAIRO-CLS-GHI
>> offering [3]:  CAIRO-DHI-sensor-QC1
>> offering [4]:  CAIRO-DHI-sensor-QC1_Completed
>> offering [5]:  CAIRO-DHI-sensor-QCfull
>> offering [6]:  CAIRO-DNI-sensor-QC1
>> offering [7]:  CAIRO-DNI-sensor-QC1_Completed
>> offering [8]:  CAIRO-DNI-sensor-QCfull
>> offering [9]:  CAIRO-DryBulbT-sensor-QC1
>> offering [10]:  CAIRO-GHI-sensor-QC1
>> offering [11]:  CAIRO-GHI-sensor-QC1_Completed
>> offering [12]:  CAIRO-GHI-sensor-QCfull
>> offering [13]:  CAIRO-Pressure-sensor-QC1
>> offering [14]:  CAIRO-RH-sensor-QC1
>> offering [15]:  CAIRO-SAA
>> offering [16]:  CAIRO-SZA
>> offering [17]:  CAIRO-TOA-HI
>> offering [18]:  CAIRO-TOA-NI
>> offering [19]:  CAIRO-WD-sensor-QC1
>> offering [20]:  CAIRO-WS-sensor-QC1
>> offering [21]:  CAIRO-WetBulbT-sensor-QC1
```

Figure 11-Réponse de la console Python à l'appel de la fonction 'printOfferings'

En observant à nouveau la **figure 5**, on peut constater que les étapes qui suivent correspondent aux sélections successives par l'utilisateur de la grandeur physique mesurée et des dates de début et de fin de la série temporelle à récupérer, en accord avec le schéma de fonctionnement du plugin que j'ai adopté. Pour permettre ces sélections à partir de l'exécution de notre script, les fonctions printObservedProperties et printTimeBeginEndPositions ont été créées. Elles permettent d'afficher dans la console Python les informations dont l'utilisateur du plugin aura besoin lors de ces deux étapes. Remarquons que les attributs *BeginPosition* et *EndPosition* fournies par la fonction printTimeBeginEndPositions donnent seulement un encadrement de la période temporelle que devra sélectionner l'utilisateur.

L'ensemble des paramètres ayant été choisis, l'utilisateur peut récupérer la série temporelle à l'aide de la fonction getSeriesSOS200. Elle présente les paramètres d'entrée suivants :

1. sos_url : l'URL de service du serveur SOS choisi
2. station_number : l'index de la station sélectionnée au sein de la liste des stations
3. offering_number : l'index de l'*offering* choisie parmi la liste des offerings de la station sélectionnée
4. property_number : l'index de la grandeur physique mesurée parmi la liste des grandeurs disponibles pour l'*offering* choisie
5. starting_time_string : une chaîne de caractères contenant la date de début pour la série temporelle à récupérer, encodée au format ISO 8601. Par exemple, '2004-01-01 00 :00 :00+00 :00' est une date valide.
6. ending_time_string : une chaîne de caractères contenant la date de fin pour la série temporelle à récupérer, également encodée au format ISO 8601.

Une fois la série temporelle récupérée, le plugin devra offrir plusieurs fonctionnalités à l'utilisateur :

– Une visualisation sous forme de tableau. La fonction arraySeries de notre script Python permet cette visualisation en générant l'affichage de la série temporelle par la console Python, comme le montre la **figure 12**.

```
2004-01-01 00:29:00+00:00    0
2004-01-01 00:30:00+00:00    0
2004-01-01 00:31:00+00:00    0
                            ..
2004-01-01 23:31:00+00:00    0
2004-01-01 23:32:00+00:00    0
2004-01-01 23:33:00+00:00    0
2004-01-01 23:34:00+00:00    0
2004-01-01 23:35:00+00:00    0
2004-01-01 23:36:00+00:00    0
2004-01-01 23:37:00+00:00    0
2004-01-01 23:38:00+00:00    0
2004-01-01 23:39:00+00:00    0
2004-01-01 23:40:00+00:00    0
2004-01-01 23:41:00+00:00    0
2004-01-01 23:42:00+00:00    0
2004-01-01 23:43:00+00:00    0
2004-01-01 23:44:00+00:00    0
2004-01-01 23:45:00+00:00    0
2004-01-01 23:46:00+00:00    0
2004-01-01 23:47:00+00:00    0
2004-01-01 23:48:00+00:00    0
2004-01-01 23:49:00+00:00    0
2004-01-01 23:50:00+00:00    0
2004-01-01 23:51:00+00:00    0
2004-01-01 23:52:00+00:00    0
2004-01-01 23:53:00+00:00    0
2004-01-01 23:54:00+00:00    0
2004-01-01 23:55:00+00:00    0
2004-01-01 23:56:00+00:00    0
2004-01-01 23:57:00+00:00    0
2004-01-01 23:58:00+00:00    0
2004-01-01 23:59:00+00:00    0
2004-01-02 00:00:00+00:00    0
Name: Diffuse Horizontal Irradiance (W.m-2), dtype: float64
```

Figure 12 -Réponse de la console Python à l'appel de la fonction 'arraySeries'

– Une visualisation graphique de la série temporelle. J'ai conçu la fonction plotSeries à cet effet. Des exemples de graphiques obtenues à l'aide de cette fonction sont présentés en **figure 13** à **16**.

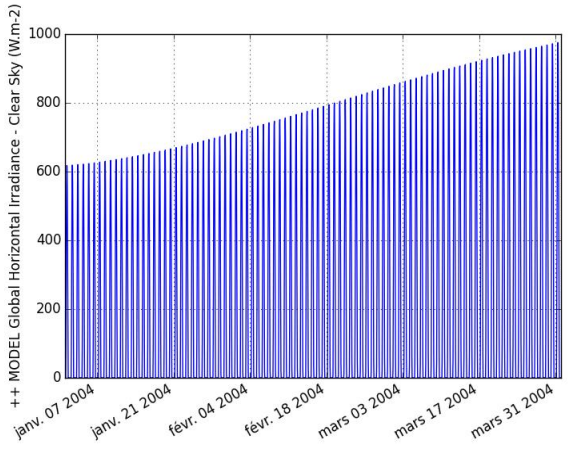


Figure 13 -Série temporelle de 3 mois de mesures par une station située au Caire du rayonnement horizontal global après correction par un modèle

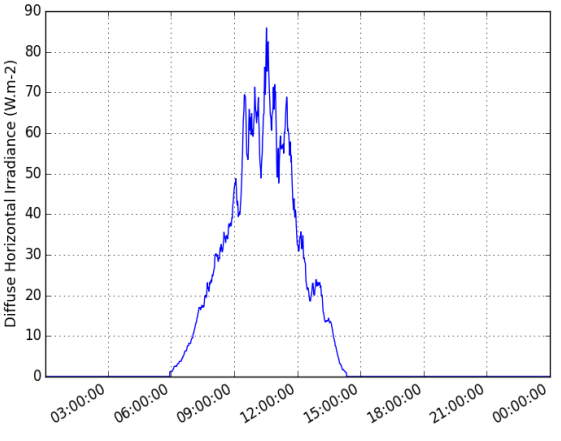


Figure 14 -Série temporelle d'une journée de mesures par une station située à Chisinau du rayonnement horizontal diffus

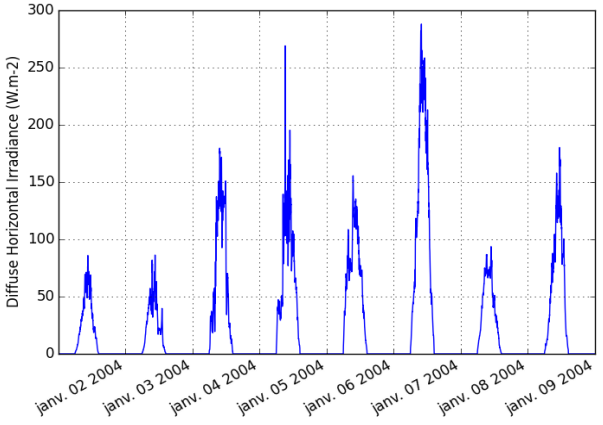


Figure 16 -Série temporelle d'une semaine de mesures par une station située à Chisinau du rayonnement horizontal diffus

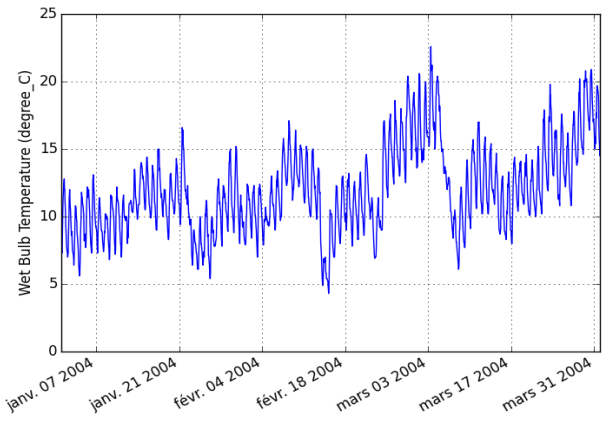


Figure 15 -Série temporelle de 3 mois de mesures par une station située au Caire de la température

– Le téléchargement de la série temporelle. La fonction exportSeries permet de récupérer un fichier au format CSV contenant la série sous la forme de deux colonnes : une colonne de temps, et une colonne de valeurs de la grandeur physique mesurée.

2.4.3 Limites et problèmes de performance

Au vu des résultats de plusieurs tests de récupération de séries temporelles de mesures *in situ* stockées sur le serveur *insitu.webservice-energy.org*, et sur d'autres serveurs SOS 2.0, il apparaît que le script développé présente plusieurs limites pouvant impacter négativement l'expérience utilisateur du plugin. Une grande part de ces limitations est liée au standard SOS lui-même, à ces implémentations, ainsi qu'aux limites de la librairie OWSLib : il est donc difficile d'envisager une solution à court terme qui ne nécessiterait pas de modifier des outils techniques développés en amont de ce script, réutilisés par ce dernier.

Un problème conséquent constaté lors de l'utilisation du script développé est celui de la lenteur du processus de récupération des séries temporelles. Il est certain que l'utilisateur du plugin souhaiterait idéalement obtenir les mêmes temps de réponse que lors de l'utilisation d'un client Web permettant de bénéficier des mêmes services, soit pas plus d'une minute dans le pire des cas. En observant la **table 1**, on peut constater que les ordres de grandeur des temps de récupération de séries temporelles de mesures *in situ* stockées sur le serveur *insitu.webservice-energy.org* dépassent fréquemment la minute. Autrement dit, il est impossible de récupérer, à partir de ce script, des séries temporelles d'un mois de mesures, ou plus, dans les délais idéalement désirés.

Pas de temps des mesures	Longueur de la période temporelle	Durée de récupération de la série temporelle
1 min	1 jour	10s-15s
1 min	1 semaine	2 min
1 min	1 mois	10 min
10 min	1 jour	5s
10 min	1 semaine	10s
10 min	1 mois	30s
10 min	1 année	2 min
1 h	1 jour	5s
1 h	1 semaine	15s
1 h	1 mois	1 min
1 h	1 année	5 min

Tableau 1: Ordres de grandeur des temps de récupération par le script Python de séries temporelles de mesures *in situ* stockées sur le serveur *insitu.webservice-energy.org*

Pour trouver des solutions à ce problème, j'ai d'abord cherché à en identifier précisément la cause. Pour ce faire, j'ai explicité le résultat de plusieurs lignes de code du script disponible en **[G1]**, en utilisant la commande Python « *print* » afin que la console Python retourne ce résultat. Cela m'a permis d'identifier la ligne qui est à l'origine d'une attente plus longue lors du processus de récupération de la série temporelle. Il s'agit de la ligne 303 du script **[G1]** :

```
303                 response1 = sos1.get_observation(
304                         offerings=offerings,
305                         responseFormat=omFormat,
306                         observedProperties=observedProperties,
307                         timeout=600,
308                         namespaces=namespaces,
309                         eventTime=event_time
310                     )
```

Figure 17 -Ligne du script développé qui génère la requête "GetObservation"

La commande « get_observation » utilisée ici est issue de la librairie OWSLib. Elle génère une connexion HTTP pour faire parvenir la requête au serveur. Pour ce faire, elle s'appuie sur la librairie *requests* de Python, laquelle utilise elle-même la librairie *urllib3*. Ayant connaissance de ces dépendances, j'ai pu accéder aux journaux d'enregistrements de la connexion HTTP à l'aide des lignes de code Python suivantes :

```
import logging
owslib_log = logging.getLogger('owslib')
# Add formatting and handlers as needed
owslib_log.setLevel(logging.DEBUG)


logging.basicConfig(level=logging.DEBUG)
requests_log = logging.getLogger("requests.packages.urllib3")
requests_log.setLevel(logging.DEBUG)
requests_log.propagate = True
```

Figure 18 -Lignes du script développé qui permettent de suivre la connexion HTTP au serveur SOS

Les informations suivantes sont alors retournées dans la console Python :

```
INFO:requests.packages.urllib3.connectionpool:Starting new HTTP connection (1):
insitu.webservice-energy.org
DEBUG:requests.packages.urllib3.connectionpool:"GET /52n-sos-webapp/service/kvp
?service=SOS&offering=MULTAN-WSG-Tier2-MSNUET-QC1&request=GetObservation&versio
n=2.0.0&responseFormat=http%3A%2F%2Fwww.opengis.net%2Fom%2F2.0&observedProperty
=Wind+Speed+Gust&namespaces=xmlns%28om%2Chttp%3A%2F%2Fwww.opengis.net%2Fom%2F2.
0%29&temporalFilter=om%3AphenomenonTime%2C2014-10-22T00%3A00%3A00%2B00%3A00%2F2
014-11-21T00%3A00%3A00%2B00%3A00 HTTP/1.1" 200 None
```

Figure 19 -suivi de connexion HTTP GET lors de l'envoi d'une requête "GetObservation"

J'ai alors compris que la connexion HTTP établie lors de l'appel de la commande « get_observation » de la librairie OWSLib est identique à celle que j'aurais établie en entrant manuellement, dans un navigateur Web, l'adresse URL indiquée par la **figure 19**, qui suit le modèle spécifié par le standard SOS. Une fois la commande appelée, il est impossible de contrôler le temps d'attente nécessaire pour récupérer les données de la série temporelle. En effet, des paramètres inhérents au réseau, comme par exemple le débit de la connexion Internet, et surtout le temps de traitement de la requête par le serveur jusqu'à l'envoi de la réponse, vont générer un temps d'attente pour l'utilisateur, sur lequel je n'ai aucun contrôle en tant que développeur du script. Si le temps de récupération de séries temporelles de mesures *in situ* stockées sur des serveurs SOS dépasse ce qu'un utilisateur attendrait lors d'une utilisation Web, il semble que ce sont les spécifications du standard, et surtout les implémentations du standard utilisées pour déployer des serveurs SOS, qui sont à l'origine de ces difficultés.

Le plugin développé doit prioritairement être capable de récupérer des séries temporelles de mesures *in situ* stockées sur le serveur *insitu.webservice-energy.org*. Lorsqu'ils avaient développé un client Web pour permettre l'accès aux données de ce serveur, mes collègues du Centre O.I.E. avaient déjà rencontré ce type de limitations. D'après eux, elles sont imputées à la grande "verbosité" du format XML utilisé pour encoder la réponse du serveur SOS aux requêtes de type *GetObservation*, ainsi qu'à la taille considérable de la base de données de mesures *in situ* du serveur, due au modèle de données adopté par 52°North dans sa solution d'implémentation du standard SOS. Pour

contourner le problème, l'interface de programmation (*Application Programming Interface* en anglais, très souvent abrégé en API) de type REST API de 52°North avait été utilisée [18]. Ici, nous n'avons pas recours à cette solution, puisque l'API est sensible à des mises à jour, et que nous ne souhaitons pas prendre le risque que l'extension QGIS devienne inutilisable suite à des modifications d'un outil externe, comme l'est cette API.

Afin de remédier au problème de performance, j'ai essayé de paramétrer le format utilisé pour encoder la réponse du serveur SOS aux requêtes de type *GetObservation*, en substituant au "verbeux" format XML le bien plus léger format JSON (JavaScript Object Notation). Une comparaison visuelle des deux formats peut être effectuée à partir de l'**annexe 2**. Les versions ultérieures à 4.0.0 de la plateforme 52°North permettent l'usage du format JSON comme format d'encodage des requêtes envoyées au serveur et des réponses retournées par celui-ci [19]. J'ai donc cherché à tirer profit de la verbosité moindre du format JSON afin de réduire le temps d'attente de l'utilisateur lors de l'appel de la commande « get_observation » de la librairie OWSLib par mon script Python. Pour ce faire, je me suis inspiré du code Python présenté sur une page Web mise en ligne par l'Institut des Sciences Marines du Conseil National de la Recherche italien, qui suggère de passer le paramètre « responseFormat = 'application/json' ». Cette tentative a débouché sur un message d'erreur dans la Console Python, présenté en **annexe 3**. Je ne suis pas parvenu à corriger cette erreur, qui provient de la librairie OWSLib, et je me suis alors tourné vers un collègue du Centre, ingénieur et docteur en informatique, qui possède une plus grande expérience que moi dans le domaine du développement Python. Après s'être penché sur mon problème, il m'a indiqué qu'à sa connaissance, on ne peut pas utiliser ce paramétrage du format de réponse en JSON avec l'implémentation de la plateforme 52°North. A plus long terme, contacter les développeurs de la librairie OWSLib pour leur signaler les difficultés rencontrées pourrait permettre d'aboutir à une utilisation robuste du format JSON, et à un gain de performance significatif pour le plugin présenté dans la section suivante.

Enfin, une solution de dernier recours pour régler le problème de performance pourrait être de tester l'utilisation de la librairie Pyoos comme alternative à la librairie OWSLib. Du fait des délais impartis pour ce stage, j'ai choisi de continuer le développement à partir de la librairie OWSLib. J'ai néanmoins identifié une page Web d'un blog, créé par l'océanographe Rich Signell, qui fournit des éléments de code permettant de passer d'une librairie à l'autre.

3 Création d'un plugin QGIS

3.1 Cadre de développement et outils utilisés

Le script Python présenté dans la section précédente a servi de socle à la création d'un plugin QGIS, baptisé SOS 2.0 Client. Pour développer ce plugin, j'ai eu recours (abondamment) à la documentation officielle QGIS, et plus précisément au PyQGIS Developer Cookbook. Il s'agit du guide incontournable pour un développeur qui entreprend la création d'un nouveau plugin.

En termes de développement, le noyau de QGIS repose sur des librairies Qt. Qt est une API de programmation orientée objet, développée en C++, utilisée pour le développement d'applications multiplateformes et d'interfaces graphiques. Le développement d'une telle interface est indispensable au fonctionnement du plugin, puisqu'il doit permettre d'échanger avec l'utilisateur. Cette interface graphique doit s'insérer dans celle de QGIS. J'ai donc utilisé Qt pour le développement d'une interface graphique, et plus précisément, puisque le plugin est écrit en langage Python, la librairie qui fait le lien entre Python et Qt, à savoir PyQt. Là encore, j'ai utilisé la documentation officielle Qt à maintes reprises. L'utilisation de PyQt présente une certaine difficulté technique, puisqu'elle fait intervenir la notion de programmation événementielle, que j'ai découverte au cours de ce stage. Il s'agit d'un paradigme de programmation dans lequel des *événements*, comme par exemple le fait qu'un utilisateur clique sur un bouton, déclenche des réponses de l'outil logiciel – par exemple l'ouverture d'une fenêtre par l'extension QGIS développée – que le développeur doit implémenter judicieusement. Dans le cadre de développement fixé par Qt, un événement génère un *signal*. Ce signal est connecté à une fonction Python (un *slot*) qui déclenche une réponse du logiciel.

Le deuxième outil technique qu'il m'a fallu apprendre à utiliser pour développer un plugin QGIS en langage Python n'est autre que l'API QGIS, qui représente en quelque sorte le registre de toutes les commandes Python propres à QGIS. Cette API dispose, elle aussi, de sa propre documentation. Mentionnons un exemple d'utilisation de l'API QGIS pour le développement de mon plugin : la classe QgsVectorLayer qui m'a permis d'automatiser l'affichage de la

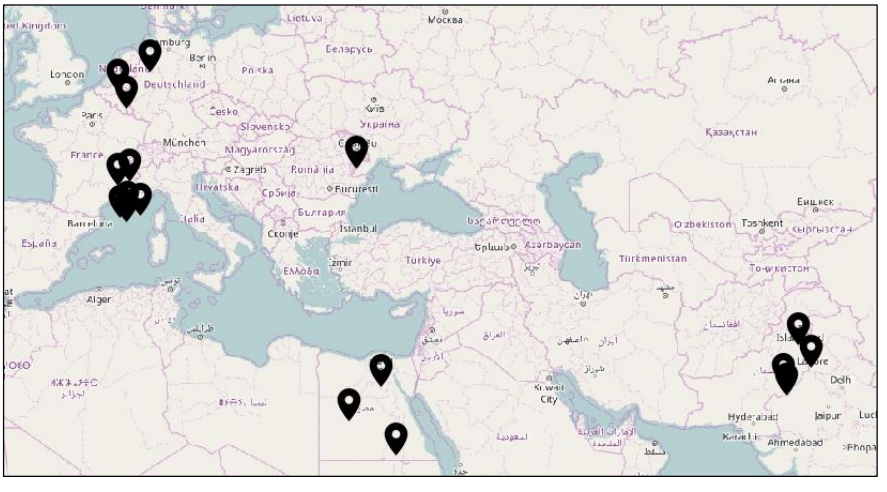


Figure 20 -Couche vecteur 'Features of interest' générée par mon plugin pour le serveur *insitu.webservice-energy.org*
(données du fond de cartes : OpenStreetMap)

localisation des différentes stations de mesure où ont été enregistrées les mesures *in situ* stockées par un serveur SOS 2.0 donné.

3.2 Fonctionnement du plugin

Le fonctionnement du plugin suit le schéma présenté en **2.1** (figure 5). Le plugin SOS 2.0 Client permet à l'utilisateur de saisir les informations relatives au paramétrage des requêtes de type *GetCapabilities* et *GetObservation*, telles qu'elles sont définies par le standard SOS 2.0. Le plugin intègre ces informations et envoie les requêtes à un serveur SOS 2.0 sélectionné par l'utilisateur. Ce plugin est donc un client. En vue de récupérer des informations générales sur ce serveur – grâce à une requête du type *GetCapabilities* du standard SOS – puis des séries temporelles de mesures in situ – grâce à une requête du type *GetObservation*, l'utilisateur est donc amené à :

1. Sélectionner les paramètres des requêtes
2. Faire appel à des fonctionnalités du plugin qui interviennent ultérieurement à cette sélection

Pour ces deux processus, l'interaction avec l'utilisateur s'effectue via l'interface graphique suivante :

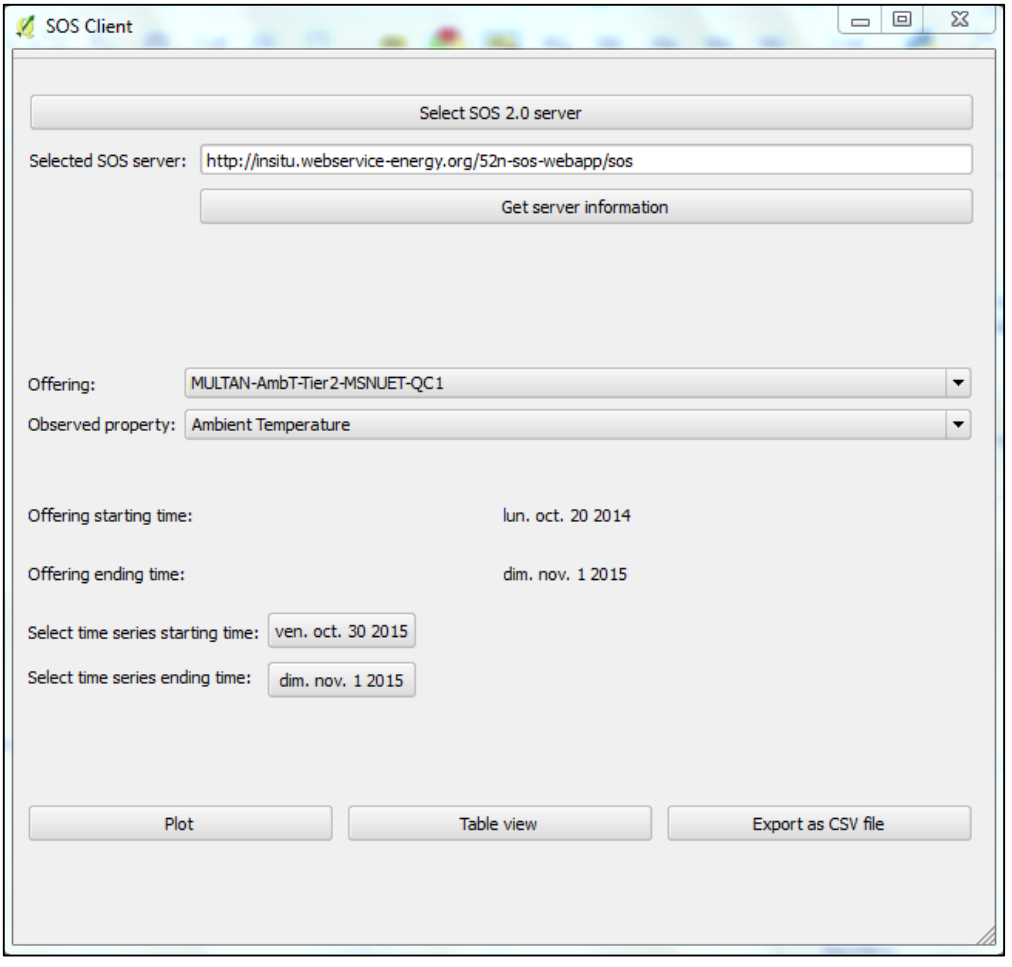


Figure 21 -Fenêtre principale de l'interface utilisateur du plugin SOS 2.0 Client chargée dans QGIS depuis un système d'exploitation Windows

3.2.1 Sélection des paramètres des requêtes par l'utilisateur

En adéquation avec le schéma de la **figure 5**, l'utilisateur doit sélectionner un certain nombre de paramètres pour récupérer des séries temporelles de mesures *in situ*. Dans le script Python de la **partie 2**, il s'agissait des paramètres d'entrée de la fonction getSeriesSOS200. L'interface graphique du plugin permet désormais à l'utilisateur d'effectuer la sélection à partir d'une série d'actions. Ces actions sont résumées dans le **tableau 2**.

Étape du schéma de principe de fonctionnement présenté en **2.1**	Paramètre de la fonction getSeriesSOS200 du script Python présenté en **2.4**	Actions de l'utilisateur permettant la sélection du paramètre depuis le plugin
Connexion à un serveur SOS	1. sos_url : l'URL de service du serveur SOS choisi	Cliquer sur le bouton 'Select SOS 2.0 server'. Une boîte de dialogue apparaît. Taper l'URL de service dans la ligne de saisie de cette boîte de dialogue
Sélection spatiale de la station de mesure	2. station_number : l'index de la station sélectionnée au sein de la liste des stations	Cliquer sur l'outil de sélection dans la barre d'outils de QGIS. Sélectionner spatialement une station de la couche vecteur 'Features of interest'.
Sélection du capteur	3. offering_number : l'index de l'offering choisie parmi la liste des offerings de la station sélectionnée	Sélectionner une offering de la liste déroulante du même nom
Sélection de la grandeur physique mesurée	4. property_number : l'index de la grandeur physique mesurée parmi la liste des grandeurs disponibles pour l'offering choisie	Sélectionner une grandeur physique de la liste déroulante nommée 'Observed property'
Sélection de la date de début de la série temporelle	5. starting_time_string : une chaîne de caractères contenant la date de début pour la série temporelle à récupérer, encodée au format ISO 8601.	Cliquer sur le bouton à droite de la mention 'Select time series starting time'. Un calendrier interactif s'ouvre. Sélectionner une date.
Sélection de la date de fin de la série temporelle	6. ending_time_string : une chaîne de caractères contenant la date de fin pour la série temporelle à récupérer, également encodée au format ISO 8601.	Cliquer sur le bouton à droite de la mention 'Select time series ending time'. Un calendrier interactif s'ouvre. Sélectionner une date.

Tableau 2 : Sélection des paramètres des requêtes : du schéma de principe au plugin QGIS

3.2.2 Accès aux fonctionnalités principales du plugin

Les fonctionnalités du plugin ont été présentées en **1.4.3**. Les fonctions donnant accès à ces fonctionnalités, développées au sein du script Python de la **partie 2**, ont été « reliées » à l'interface graphique du plugin. L'utilisateur peut alors faire appel à ces fonctionnalités à partir d'une nouvelle série d'actions. Ces actions sont reportées dans le **tableau 3**, à la page suivante.

Étape du principe de fonctionnement présenté en **2.1**	Fonction du script Python correspondante	Actions de l'utilisateur permettant l'accès à la fonctionnalité depuis le plugin
Affichage des informations générales sur le serveur	1. Fonction getCapabilitiesSOS200	Cliquer sur le bouton 'Get server information' après sélection du serveur.
Affichage de la série temporelle de mesures dans un tableau	2. Fonction arraySeries	Cliquer sur le bouton 'Table View' après sélection des différents paramètres de la requête *GetObservation*.
Affichage de la série temporelle de mesures dans un graphique	3. Fonction plotSeries	Cliquer sur le bouton 'Plot' après sélection des différents paramètres de la requête *GetObservation*.
Téléchargement de la série temporelle de mesures	4. Fonction exportSeries	Cliquer sur le bouton 'Export as CSV file' après sélection des différents paramètres de la requête *GetObservation*.

Tableau 3 : Accès aux fonctionnalités principales : du schéma de principe au plugin QGIS

3.3 Résultats

3.3.1 Exemple de résultats obtenus lors de l'utilisation des fonctionnalités utilisateurs du plugin

En **1.4.3**, nous avions identifié les fonctionnalités utilisateurs attendues du plugin QGIS. Listons-les à nouveau en parcourant quelques résultats qui témoignent des réponses apportées par le plugin SOS 2.0 Client :

– *Sélectionner un serveur SOS* :

Lorsque l'on clique sur le bouton 'Select SOS 2.0 server', une boîte de dialogue apparaît. Elle permet de sélectionner un serveur SOS 2.0 en entrant dans une ligne de saisie l'URL de service du serveur choisi. Ici, nous choisissons le serveur du Centre O.I.E.

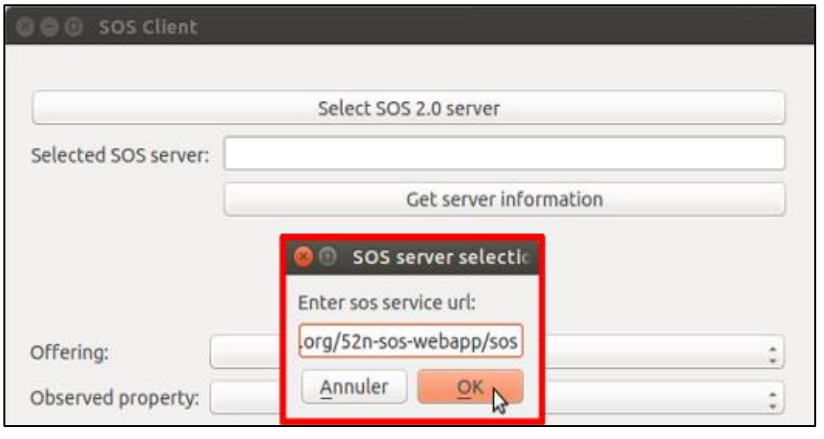


Figure 22 -Sélection d'un serveur SOS 2.0 depuis l'interface utilisateur du plugin

– *Géolocaliser une station parmi celles qui se situent dans une bounding box adaptée au niveau de zoom d'une couche cartographique* :

Après avoir validé le choix du serveur, l'utilisateur peut constater qu'une couche vecteur intitulée 'Features of interest' est ajoutée à sa session QGIS courante. Elle indique la localisation des différentes stations de

mesure dont les données de mesures sont stockées par le serveur choisi. Le niveau de zoom s'accorde automatiquement avec l'étendue géographique de cette couche. L'utilisateur peut ensuite cliquer sur l'outil de sélection dans la barre d'outils de QGIS avant de sélectionner spatialement une station.

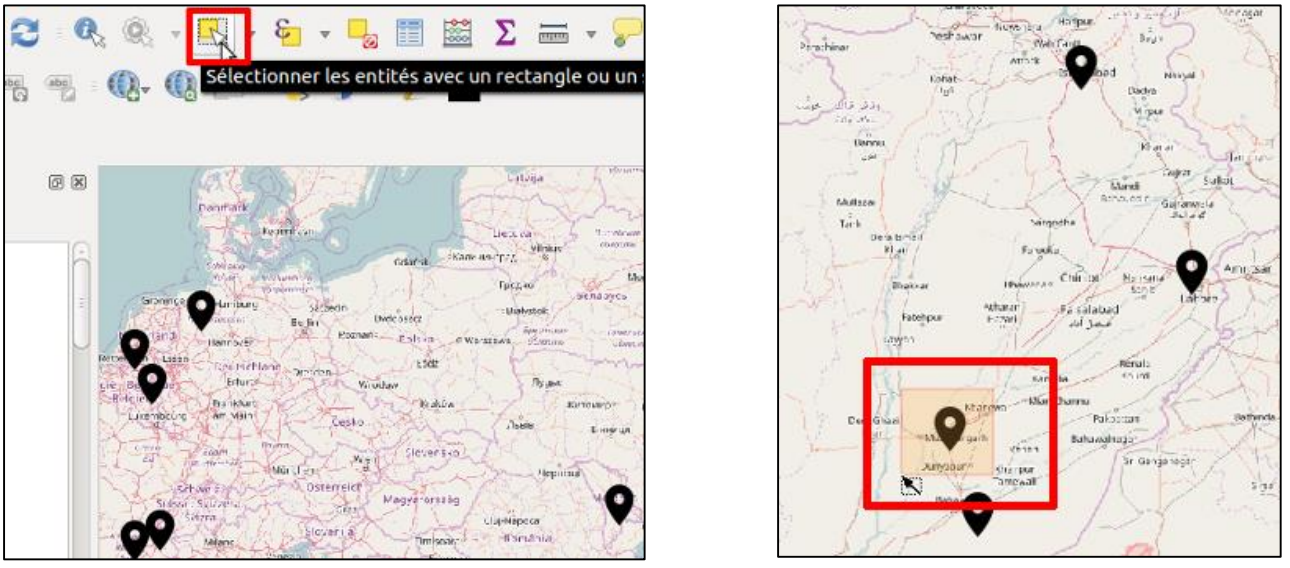


Figure 23 -Sélection d'une station de mesure depuis l'interface utilisateur du plugin

– *Sélectionner un jeu de données correspondant à une série temporelle de mesures in situ à partir des attributs structurant de la mesure* :

Une fois la station sélectionnée, des paramètres par défaut sont affichés dans l'interface utilisateur du plugin. L'utilisateur peut changer ces paramètres à sa guise. Les éventuelles relations de dépendances entre paramètres sont intégrées dans les structures développées en langage Python pour le plugin. L'utilisateur peut ainsi changer, sans contrainte d'ordre et autant de fois qu'il le souhaite, l'*offering*, la grandeur physique mesurée, puis les dates de début et de fin de la série temporelle de son choix.

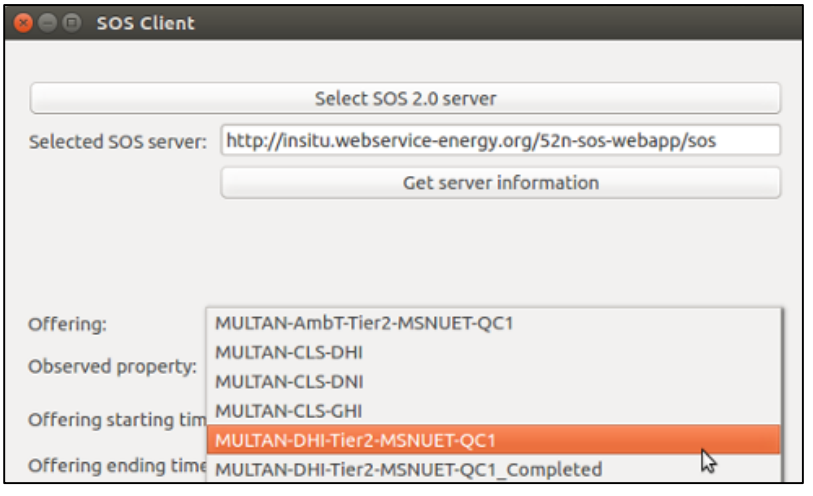


Figure 24 -Sélection d'une *offering* depuis l'interface utilisateur du plugin

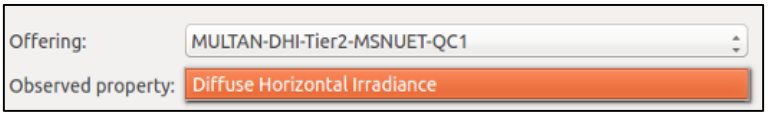


Figure 26 -Sélection de la grandeur physique mesurée depuis l'interface utilisateur du plugin

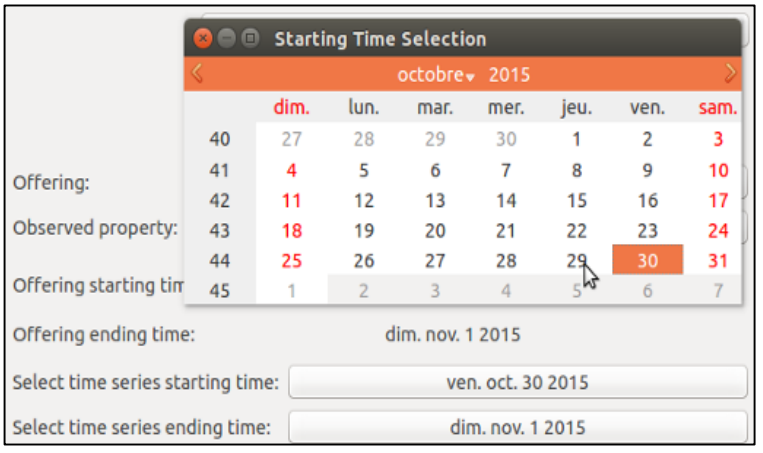


Figure 26 -Sélection de la date de début de la série temporelle depuis l'interface utilisateur du plugin

– *Visualiser la série temporelle sous forme de tableau ou bien de graphique* :

Après que l'utilisateur ait sélectionné les différents paramètres de la requête *GetObservation*, il peut envoyer la requête afin de récupérer les données puis d'afficher la série temporelle de mesures dans un tableau, en cliquant sur le bouton 'Table View'. Pour afficher la série temporelle dans un graphique cette fois-ci, l'utilisateur dispose du bouton 'Plot'.

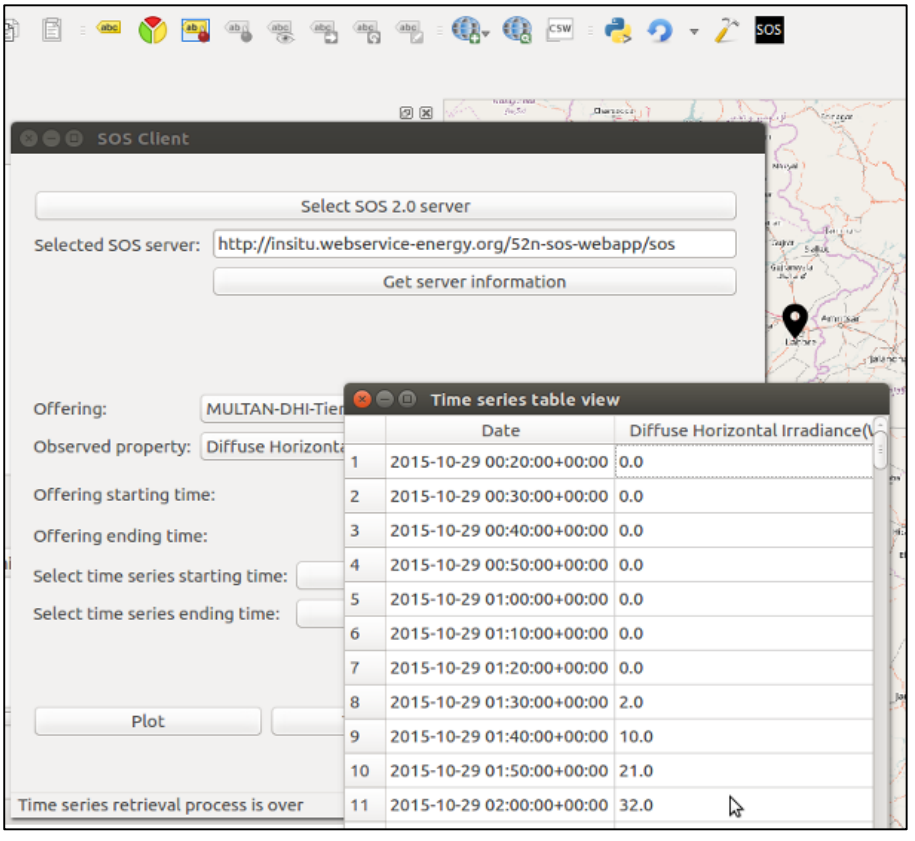


Figure 27 -Affichage de la série temporelle de mesures sous forme de tableau depuis l'interface utilisateur du plugin

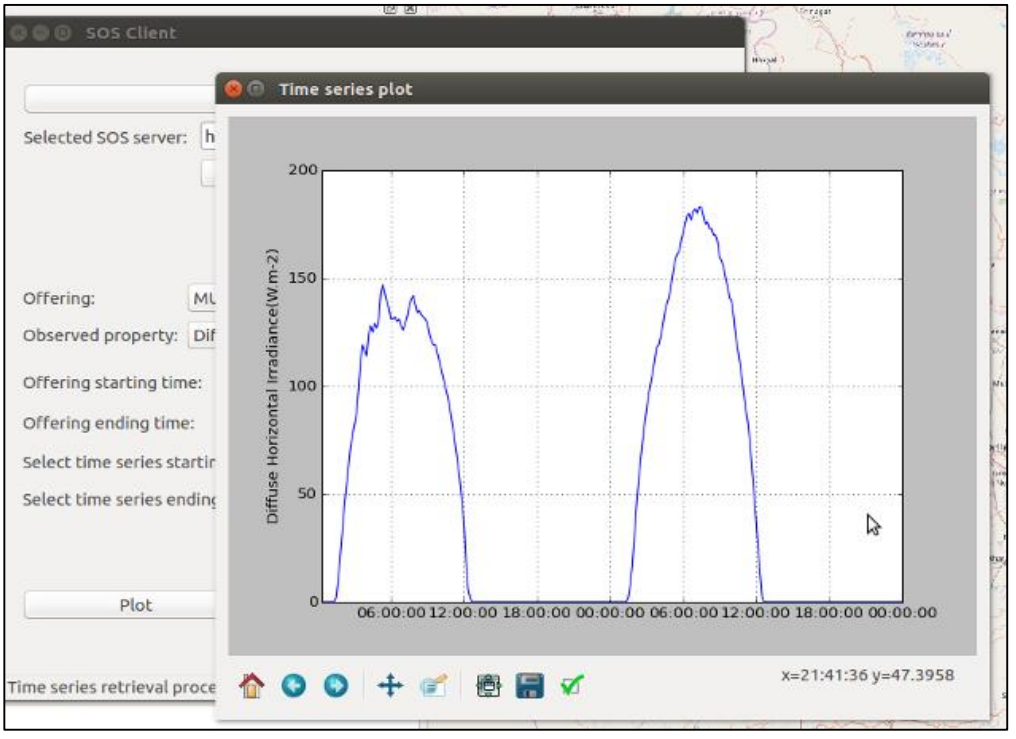


Figure 28 -Affichage de la série temporelle de mesures sous forme de graphique depuis l'interface utilisateur du plugin

– *Télécharger le jeu de données de mesures in situ sur sa propre machine* :

Enfin, l'utilisateur peut également cliquer sur le bouton intitulé 'Export as CSV file', après sélection des différents paramètres de la requête *GetObservation*, dans le but de télécharger les données de la série temporelle de mesures.

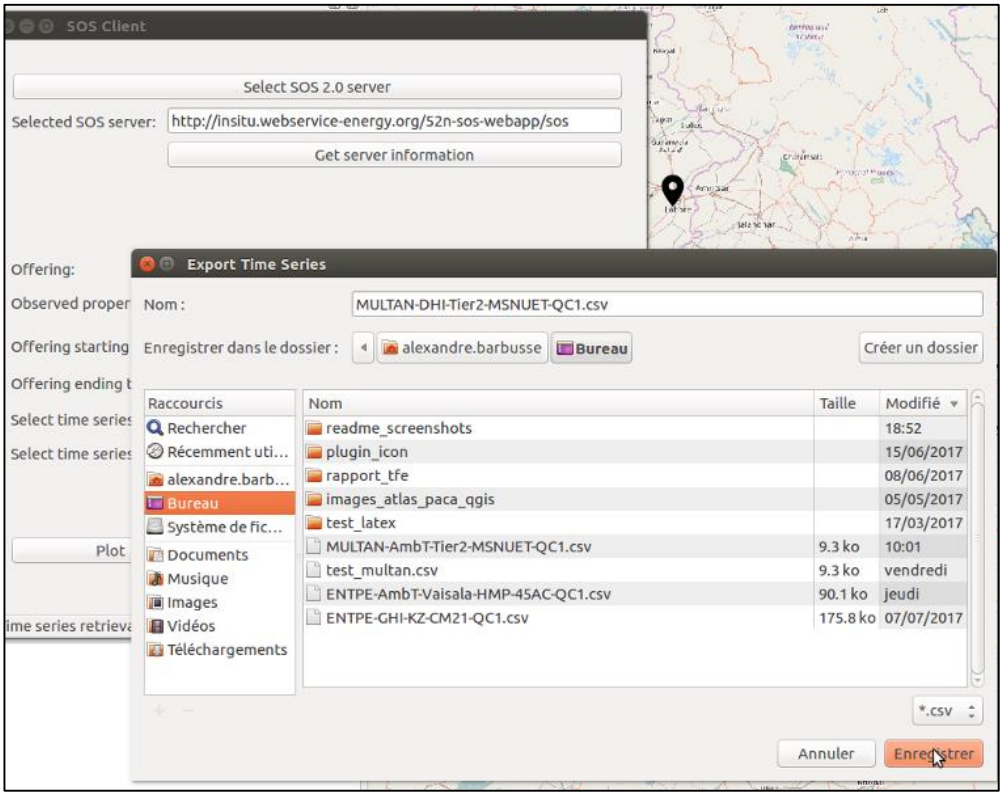


Figure 29 -Export en fichier CSV de la série temporelle de mesures depuis l'interface utilisateur du plugin

– *Accéder aux métadonnées portant sur le serveur SOS sélectionné en entrée par l'utilisateur* :

Pour cela, il suffit à l'utilisateur de cliquer sur le bouton intitulé 'Get server information', ce qui génère l'ouverture d'une fenêtre comprenant les métadonnées.

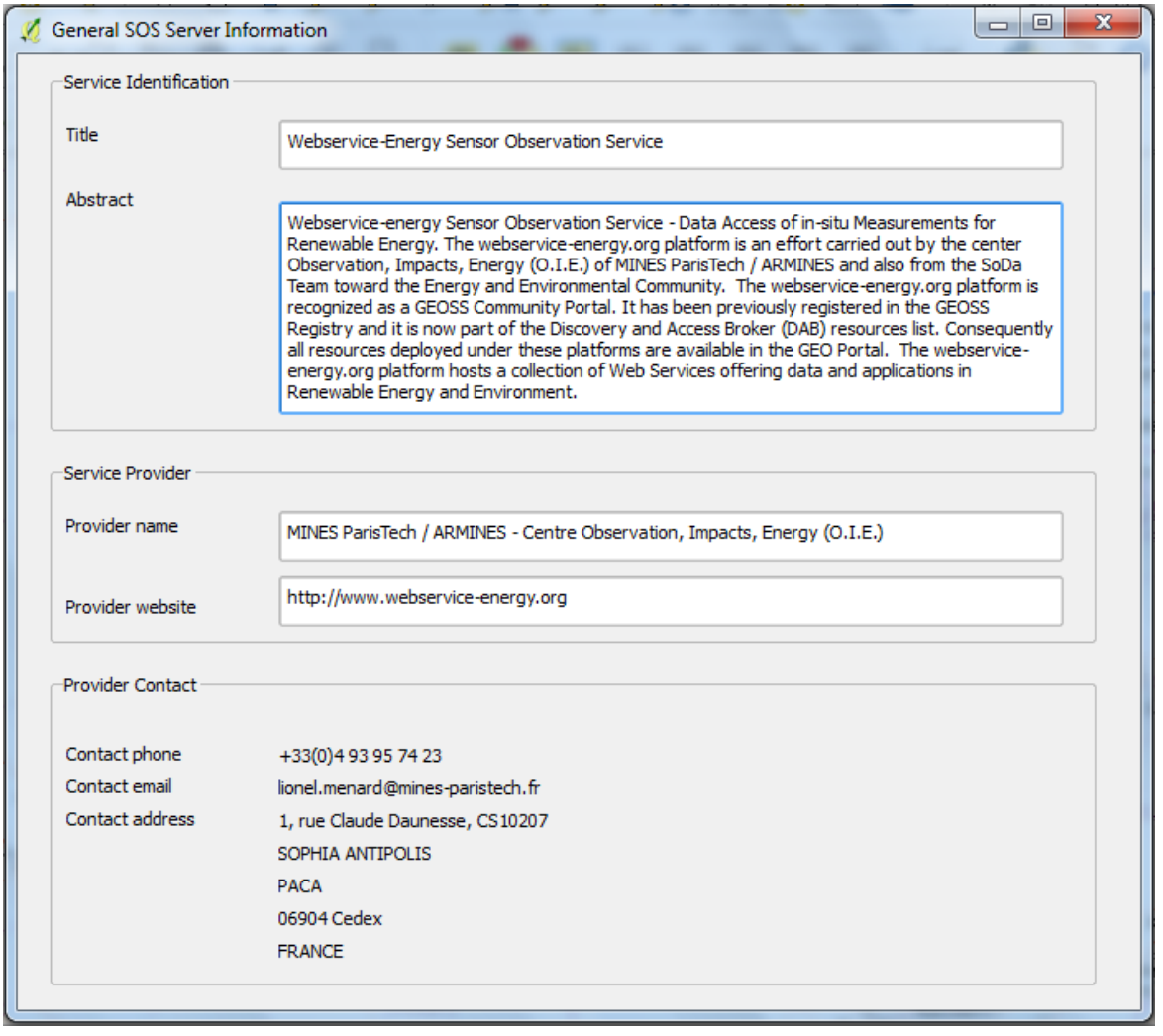


Figure 30 -Fenêtre d'affichage des métadonnées portant sur le serveur SOS sélectionné en entrée par l'utilisateur

3.3.2 Robustesse et gestion des problèmes rencontrés

Dans la section **1.2.2**, j'ai indiqué que le serveur SOS du Centre O.I.E., *insitu.webservice-energy.org*, serait testé en priorité au cours du développement du plugin. La section précédente suggère que des résultats encourageants ont été obtenus lors de l'utilisation du plugin SOS 2.0 Client pour l'accès aux données de ce serveur. Au cours du développement, je me suis intéressé aux résultats obtenus avec d'autres serveurs en vue de corriger d'éventuelles erreurs. Il s'agissait d'améliorer au mieux la robustesse de mon plugin. La liste des serveurs utilisés pour cette phase de test est présentée à la page suivante.

URL de service du serveur SOS 2.0	Organisme à l'origine du serveur
http://insitu.webservice-energy.org/52n-sos-webapp/sos	MINES ParisTech / ARMINES - Centre Observation, Impacts, Energie (O.I.E.)
http://sensor.webservice-energy.org/52nSOSv3.5.0/sos	MINES ParisTech / ARMINES - Centre Observation, Impacts, Energie (O.I.E.)
http://localhost:8080/52n-sos-webapp_4.3.7/sos	Serveur personnel
http://geo.irceline.be/sos/service	Cellule Interrégionale de l'Environnement (CELINE - IRCEL) belge
http://vesk.ve.ismar.cnr.it/observations/sos/kvp	Institut des Sciences Marines du Conseil National de la Recherche italien
http://geonodenodc.ogs.trieste.it/observations/sos/kvp	Institut National d'Océanographie et de Géophysique Expérimentale italien
http://nodc.ogs.trieste.it/sos/service	Institut National d'Océanographie et de Géophysique Expérimentale italien
http://sk.ise.cnr.it/observations/sos/kvp	Institut pour l'Étude des Écosystèmes du Conseil National de la Recherche italien
http://173.167.227.68:8080/52n-sos-webapp-437/service	Clermont County, Ohio
http://52.6.7.23/52n-sos-webapp/service	United States Environmental Protection Agency

Tableau 4 : Liste des serveurs SOS 2.0 testés lors du développement du plugin SOS 2.0 Client

J'avais évoqué en **2.4.3** les difficultés soulevées par la lenteur du processus de récupération des séries temporelles déclenché par l'envoi d'une requête de type *GetObservation* au serveur SOS sélectionné par l'utilisateur. Lors du développement du plugin SOS 2.0 Client, des dispositions ont été prises pour que l'utilisateur :

1. soit informé qu'un temps d'attente conséquent pourrait être nécessaire. Lorsqu'il lance la requête, une boîte de dialogue s'ouvre. Elle contient un message d'information prévu dans ce but.

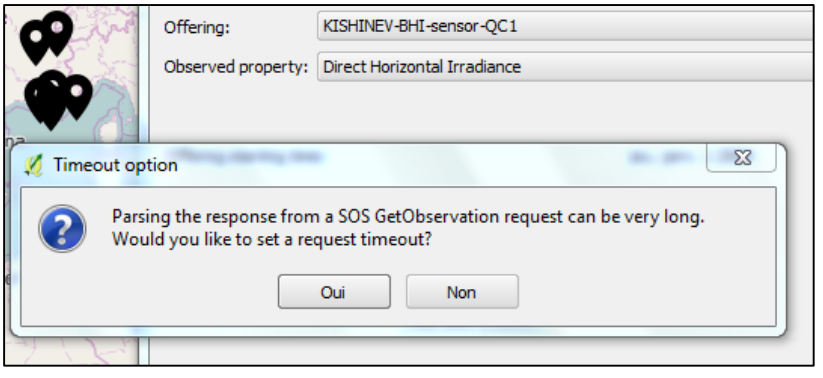


Figure 31 -Message d'information relatif au délai d'attente nécessaire pour la récupération de la série temporelle

2. puisse choisir de fixer un délai d'attente maximal (*timeout*). Pour cela il suffit à l'utilisateur de répondre positivement à la question qui clôt le message d'information, puis d'entrer la durée de son choix en secondes.

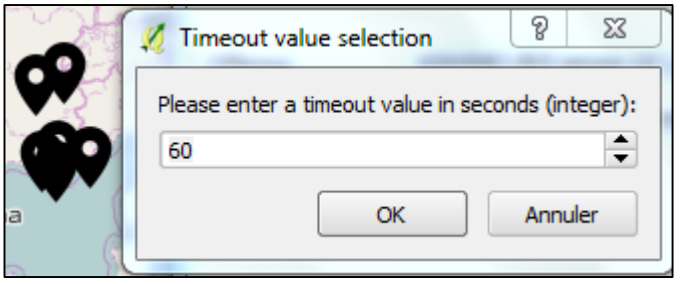


Figure 32 -Sélection d'un délai d'attente maximal (timeout)

3. soit informé lorsque le délai d'attente maximal qu'il a fixé a été dépassé. Lorsque cela se produit, un message d'information est émis. Malheureusement, l'étape de récupération de la série temporelle est alors avortée, et l'utilisateur doit choisir s'il souhaite attendre plus longuement, ou s'il préfère abandonner le projet de récupération de la série temporelle.

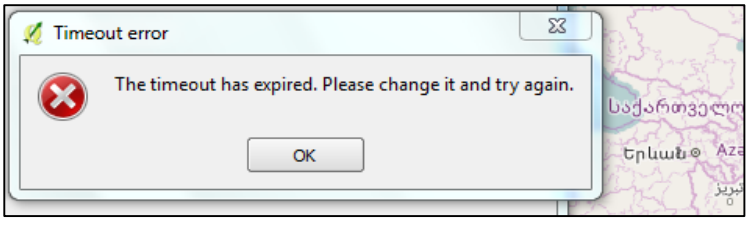


Figure 33 -Message d'erreur signalant que le délai d'attente maximal configuré a été dépassé

Ensuite, la récupération de l'information géographique portant sur la localisation des stations de mesure peut également être la source de problèmes. En temps normal, l'utilisateur voit apparaître un message d'information après avoir sélectionné le serveur SOS 2.0. Il l'informe qu'une couche vecteur contenant la localisation des différentes stations de mesure a été ajoutée à sa session QGIS, puis lui indique la marche à suivre afin de récupérer une série temporelle de mesures.

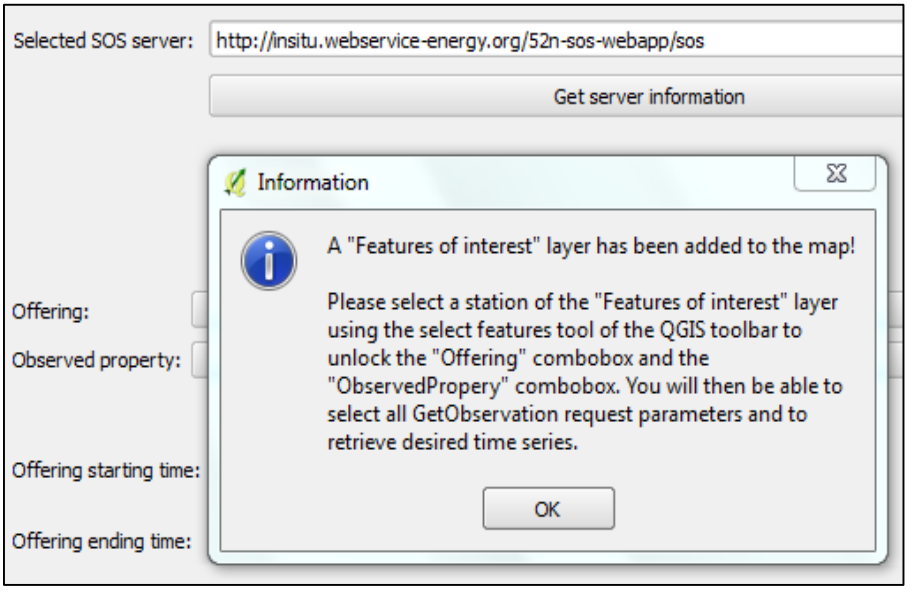


Figure 34 -Message d'information consécutif à la bonne récupération de l'information géographique sur la localisation des stations de mesure

Malheureusement, il se peut que les données (stockées dans le serveur SOS) sur la localisation de la station soient « mal encodées », au sens où l'appel de la commande *.bbox* de la librairie OWSLib pour une *offering* donnée retourne l'objet *None* ou bien encore l'objet *[ ]*, au lieu du quadruplet de coordonnées géographiques attendu (long1, lat1, long2, lat2). Rappelons que c'était ce quadruplet qui était retourné par la fonction printStations du script Python présenté en **2.4**. Ainsi, j'ai pu utiliser le résultat affiché par une console Python lors de l'appel de cette fonction pour expliciter (cf. **figure 35**) le problème constaté dans le cas où l'utilisateur choisi un serveur SOS tel que celui de la Cellule Interrégionale de l'Environnement (CELINE - IRCEL) belge, par exemple.

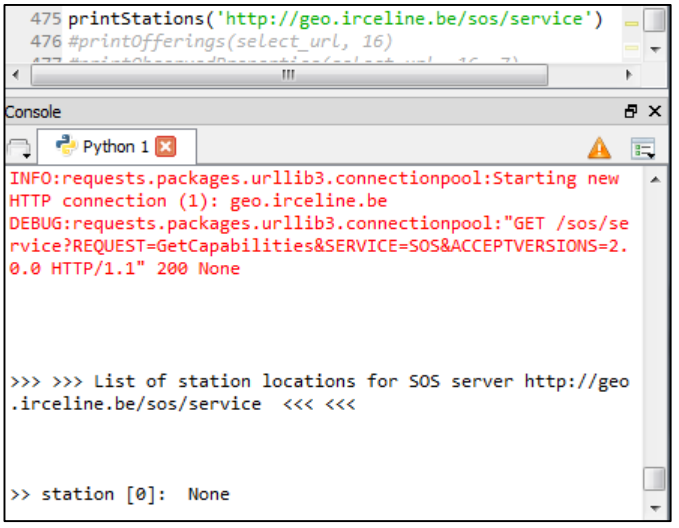


Figure 35 -Résultat de l'appel de la fonction printStations pour le serveur de la Cellule Interrégionale de l'Environnement belge

Dans un tel cas, la localisation des stations de mesure ne peut être affichée. Après avoir sélectionné le serveur SOS 2.0, l'utilisateur voit apparaître un message d'avertissement, présenté en **figure 36**. Au cours du développement, j'ai fait en sorte que l'utilisateur puisse tout de même récupérer les données d'une série temporelle de mesures *in situ*. Ainsi, le message d'avertissement ci-contre comporte des instructions qui indiquent à l'utilisateur la marche à suivre dans cette perspective. Il lui suffit de choisir une offering dans la liste déroulante associée, et de suivre la procédure usuelle de paramétrage de la requête *GetObservation*. La **figure 37** permet de constater qu'une série temporelle de mesures peut être récupérée même si la

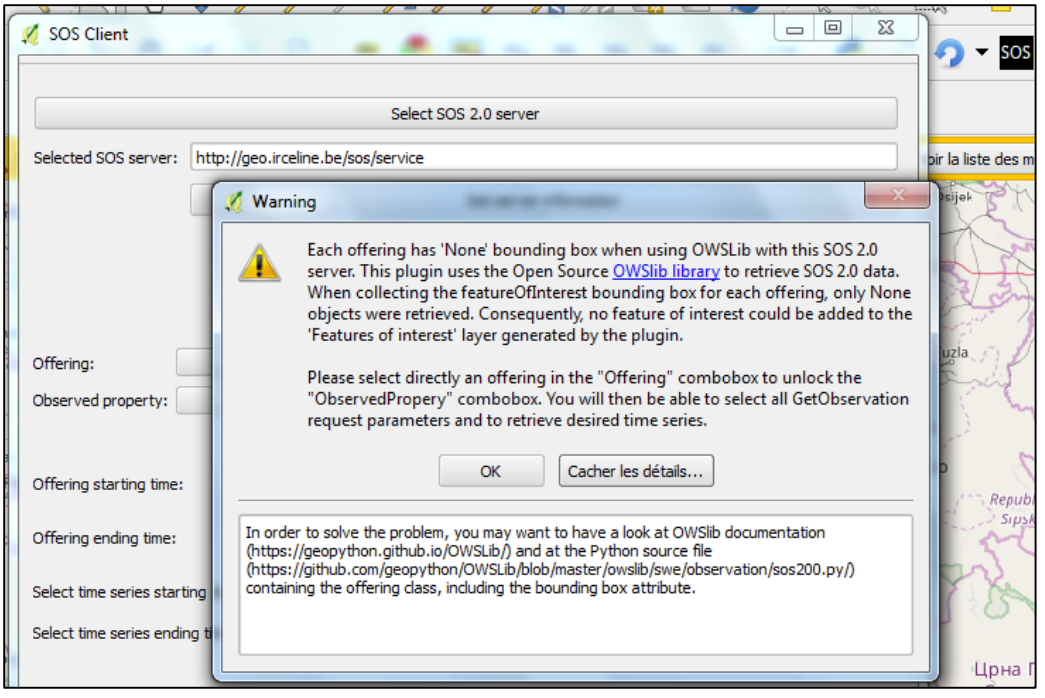


Figure 36 -Message d'avertissement signalant à l'utilisateur que la localisation des stations de mesure ne peut être affichée, sur l'exemple du serveur de la Cellule Interrégionale de l'Environnement belge

localisation des stations n'a pas pu être affichée. Néanmoins, l'intérêt pour l'utilisateur demeure limité, puisqu'il ne sait pas où a été effectuée la mesure, sauf si le titre de l'*offering* en fait mention.

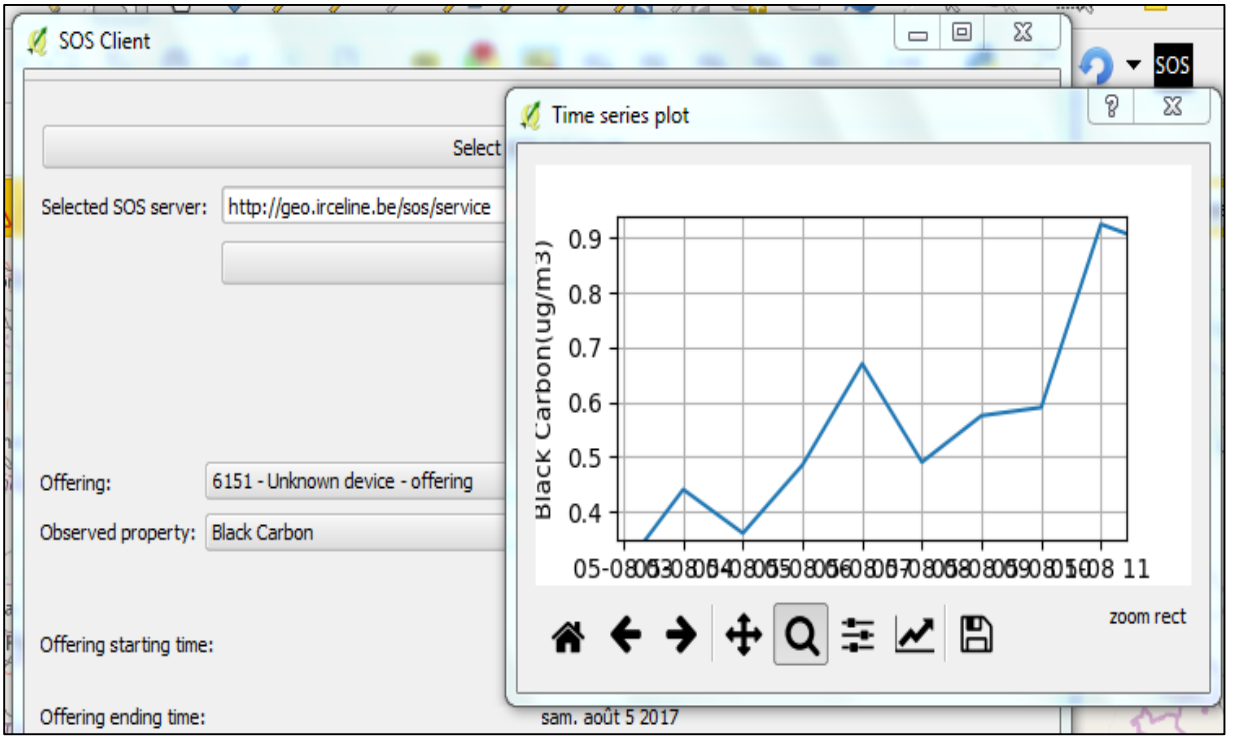


Figure 37 -Affichage sous forme de graphique d'une série temporelle de mesures *in situ*, sur l'exemple du serveur de la Cellule Interrégionale de l'Environnement belge

Par ailleurs, lorsque l'implémentation du standard SOS 2.0 est défectueuse ou présente des défauts de robustesse, il est possible qu'aucune réponse à la requête *GetObservation* envoyée par le plugin ne puisse être récupérée, par exemple si la connexion HTTP avec le serveur ne peut être établie. Lorsque cela se produit, l'exécution par Python du programme de mon plugin génère une exception. Il s'agit d'une erreur qui conduit à

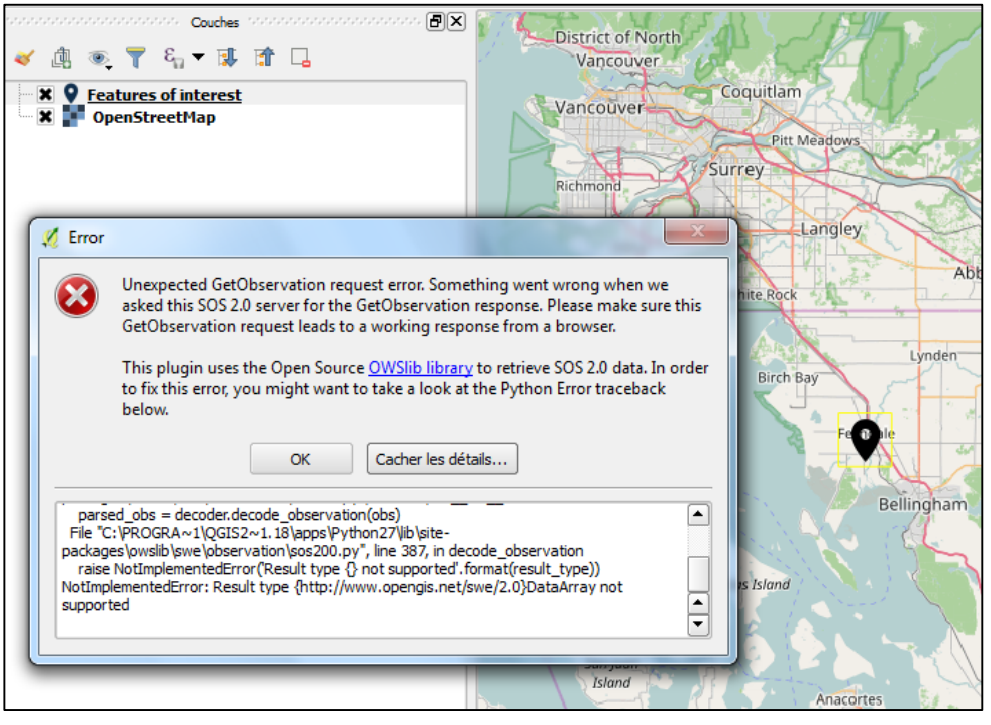


Figure 38 -Message d'erreur signalant à l'utilisateur que la réponse du serveur à sa requête n'a pas pu être récupérée, sur l'exemple du serveur de l'United States Environmental Protection Agency

l'interruption de la tentative de récupération de la série temporelle. En tant que développeur du plugin, je n'ai pas les moyens de corriger ces erreurs qui proviennent des serveurs interrogés. En revanche, j'ai pu utiliser la librairie *traceback* de Python pour collecter des informations sur la source de l'erreur, de façon à les retourner à l'utilisateur. Dans la pratique, l'utilisateur voit apparaître un message d'erreur, comme le montre la **figure 38**. Il peut alors consulter une liste d'informations, qu'on désigne en informatique sous le nom de trace d'appels (ou *stack trace* en anglais), en cliquant sur le bouton « Montrer les détails » situé sous l'énoncé du message d'erreur. Cette trace d'appels contient la description complète de la *pile d'appels* du programme au moment où l'exception a été générée. Il s'agit d'une liste de noms de fichiers, de classes, et de fonctions, assortis du numéro de la ligne du programme qui génère leur appel. Elle indique très précisément la position de l'élément qui déclenche l'apparition de l'erreur lors de la communication entre le plugin et le serveur interrogé. Dans l'éventualité où l'utilisateur disposerait d'une connaissance avancée du standard SOS et de ses implémentations, il pourrait alors, s'il le souhaitait, tenter de corriger le problème à la source, et ainsi contribuer à l'amélioration des implémentations du standard SOS.

Enfin, lorsque l'implémentation du standard SOS 2.0 par le serveur présente certains défauts, ou bien que la librairie OWSLib atteint ses limites, il est possible qu'une réponse à la requête *GetObservation* envoyée par le plugin soit récupérée avec succès, mais que celle-ci ne soit pas valide au sens où elle ne permet pas la bonne extraction, par la librairie OWSLib, des données de la série temporelle. Lors du développement, j'ai pu utiliser la réponse XML à la requête pour remonter à la source de ces problèmes, mais il m'était impossible de les corriger, puisque l'erreur provient ou bien de l'implémentation du standard - donc du serveur interrogé – ou bien de la librairie OWSLib. Néanmoins, des dispositions ont été prises pour que l'utilisateur :

1. soit informé du problème par un message d'avertissement.

2. puisse consulter la réponse XML à la requête *GetObservation* afin de remonter au problème d'implémentation du standard pour ce serveur SOS 2.0, s'il souhaite en corriger les défauts. A cet effet, il peut cliquer sur le bouton « Montrer les détails ».

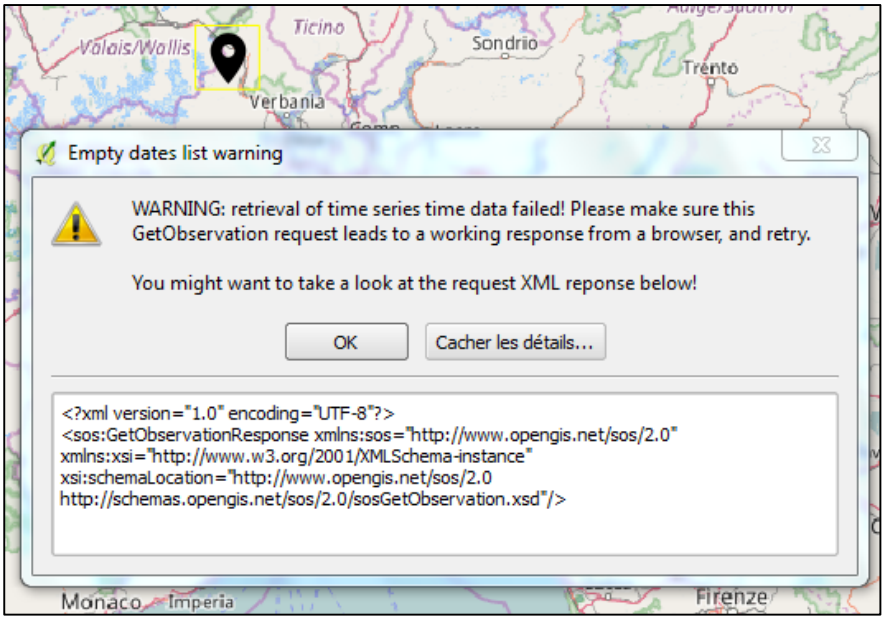


Figure 39 -Message d'avertissement signalant à l'utilisateur que la réponse du serveur à sa requête n'est pas valide, sur l'exemple du serveur de l'Institut pour l'Étude des Écosystèmes italien

3.3.3 Résultats du test du plugin et limites constatées

J'ai testé l'utilisation de mon plugin sur un échantillon de 10 serveurs afin d'évaluer sa robustesse. A partir des problèmes constatés dans la section précédente, j'ai pu identifier des étapes « critiques » du déroulement. L'issue de ces différentes étapes conditionne la possibilité d'affichage dans QGIS de :

1. la localisation des stations de mesures ;
2. la série temporelle sélectionnée par l'utilisateur.

En utilisant la réponse XML à la requête *GetObservation* envoyée au serveur, ou bien la trace d'appels des exceptions retournées par Python (selon la nature de l'éventuelle erreur constatée), j'ai également pu déterminer la source de l'erreur à corriger. Les résultats obtenus sont présentés dans le tableau ci-dessous.

URL de service du serveur SOS 2.0	Affichage des stations	Réponse à la requête récupérée	Réponse à la requête exploitable	Affichage de la série temporelle	Source de l'erreur à corriger
http://insitu.webservice-energy.org/52n-sos-webapp/sos	**Oui**	Oui	Oui	**Oui**	-
http://sensor.webservice-energy.org/52nSOSv3.5.0/sos	**Oui**	Non	-	**Non**	Implémentation /Serveur OWSLib
http://localhost:8080/52n-sos-webapp_4.3.7/sos	**Oui**	Oui	Oui	**Oui**	-
http://geo.irceline.be/sos/service	**Non**	Oui	Oui	**Oui**	Implémentation /Serveur OWSLib
http://vesk.ve.ismar.cnr.it/observations/sos/kvp	**Non**	Non	-	**Non**	Implémentation /Serveur
http://geonodenodc.ogs.trieste.it/observations/sos/kvp	**Oui**	Oui	Oui	**Oui**	-
http://nodc.ogs.trieste.it/sos/service	**Oui**	Oui	Non	**Non**	Implémentation /Serveur
http://sk.ise.cnr.it/observations/sos/kvp	**Oui**	Oui	Non	**Non**	Implémentation /Serveur
http://173.167.227.68:8080/52n-sos-webapp-437/service	**Oui**	Non	-	**Non**	OWSLib
http://52.6.7.23/52n-sos-webapp/service	**Oui**	Non	-	**Non**	OWSLib

Tableau 5 : Récapitulatif des résultats obtenus pour différents serveurs lors du test du plugin SOS 2.0 Client

Malgré les dispositions prises pour améliorer la robustesse du plugin, l'affichage de la série temporelle par ce dernier n'a été possible que pour 4 des 10 serveurs testés. *A priori*, les erreurs constatées ne peuvent pas être corrigées en modifiant le code source du plugin, puisqu'elles proviennent des limites de la librairie OWSLib, ou bien d'erreurs d'implémentation du standard dans les différents serveurs.

La librairie OWSLib est une librairie Open Source, et ses capacités s'améliorent à mesure que de nouveaux problèmes sont identifiés lors de son utilisation. Du fait des délais impartis pour ce stage, et des limites de mes connaissances du développement Python, je ne me suis pas consacré au perfectionnement de cette librairie. A l'avenir, les développeurs Python souhaitant contribuer à l'amélioration de ce plugin pourront, s'ils le désirent, s'y investir.

Pour une fraction non négligeable des serveurs interrogés, la communication entre le serveur et le plugin n'aboutit pas, du fait de défauts dans l'implémentation du standard SOS 2.0. Ces défauts pourraient provenir d'une erreur dans le déploiement des serveurs concernés, ou encore, peut-être, de limitations intrinsèques aux solutions d'implémentations du standard SOS 2.0. L'irruption de tels problèmes n'est pas à exclure, puisque le standard SOS est un standard récent – sa création date de 2006 [20] – et que plusieurs années de perfectionnement sont nécessaires avant l'obtention d'implémentations parfaitement robustes d'un tel standard. A l'avenir, les défauts d'implémentation constatés pourraient être corrigés, et de meilleurs résultats lors de l'utilisation de ce plugin pourraient être obtenus.

De surcroît, le code source du plugin n'est pas figé, son développement n'étant pas achevé. Dans l'état actuel, le plugin fait office de démonstration technique pour une communauté de développeurs intéressés par la possibilité de disposer de ses fonctionnalités dans QGIS, et pour les différents partenaires présentés en **1.4.2**. Nous reviendrons plus longuement sur ces points dans la **partie 5**.

4 Déploiement des outils développés

4.1 Mise à disposition du script Python sur GitHub

Pour permettre l'accès au script Python présenté dans la **partie 2,** aux personnes intéressées par la procédure - automatisée en langage Python - de collecte, de visualisation, puis de téléchargement dans un fichier .csv, de séries temporelles de mesures *in situ* encodées selon le standard SOS 2.0 ; j'ai déposé le code source de ce script sur la plateforme GitHub. Le lien [G1] permet de remonter à la page Web du dépôt. GitHub est une plateforme de stockage et de gestion de codes sources de logiciels, dont les fonctionnalités s'appuie sur le logiciel de gestion de versions Git. GitHub est très utilisé pour le développement de projets informatiques Open Source. GitHub propose de nombreuses fonctionnalités pour le développement collaboratif comme, entre autres, le partage de versions, la suggestion de nouvelles fonctionnalités, le suivi des bugs, la gestion de la répartition des tâches entre collaborateurs, etc. Comptant plusieurs dizaines de millions d'utilisateurs investis dans le développement informatique, GitHub est parfois présenté comme « le Facebook pour développeurs ». Le code est donc mis à la disposition de tous en libre accès. Il s'intègre ainsi dans l'approche des TIC adoptée par le centre O.I.E. qui privilégie le développement et l'utilisation d'outils ouverts.

4.2 Publication du plugin SOS 2.0 Client

Afin que les utilisateurs de QGIS puissent installer l'extension SOS 2.0 Client, je l'ai téléversée dans le dépôt officiel des extensions Python, en suivant les instructions de la section de la documentation officielle de QGIS consacrée à la publication d'une extension. Il est notamment demandé d' « inclure un dépôt du code ». Le code source de l'extension a donc été déposé sur GitHub. Le lien [G2] renvoie à la page Web du dépôt. Le code source a été commenté avec soin de manière à faciliter sa compréhension et sa réutilisation par d'autres développeurs Python. Les fichiers sources écrits en langage Python suivent les conventions et respectent les recommandations relatives à la forme du code fournies par le PEP 8 -- Style Guide for Python Code de la documentation officielle de Python.

Lors de la publication de l'extension, j'ai également été amené à choisir une licence. L'Association Francophone des Utilisateurs de Logiciels Libres (AFUL) présente l'intérêt d'une licence :

« *La loi protège automatiquement et implicitement toutes les œuvres de l'esprit par le droit d'auteur (ou le copyright pour les pays de droit anglo-saxon). Cette protection est, dans le cas du droit d'auteur, au bénéfice de l'auteur et contre toute forme d'usage par des tiers.*

La mise à disposition d'une œuvre, logiciel ou non, notamment sur l'Internet, doit donc être un acte volontaire et explicite. Cet acte s'exprime par l'adjonction d'une licence, qui est un contrat type proposé aux acquéreurs de l'œuvre (gratuit ou payant) indiquant les droits qui leurs sont concédés, et les obligations éventuelles qui leur sont imposées en échange. » [21]

Dans le même temps, une page Web intitulée *Licensing requirements for QGIS plugins*, et issue du blog officiel de QGIS, signale que : « il est nécessaire que tous les plugins distribués via http://plugins.qgis.org [qui n'est autre que l'adresse du dépôt officiel des extensions QGIS] [...] se conforment aux prescriptions de la licence GPL version 2 ou une version plus récente ». La communauté de développeurs derrière le projet de QGIS demande que le développeur d'une extension, comme ici le plugin SOS 2.0 Client, distribue cette dernière sous une licence compatible avec la licence GPL version 2 (ou une version plus récente), pour que l'extension puisse être téléversée dans le dépôt officiel des extensions Python. La licence GPL (*General Public License*) est une licence libre au sens de la Free Software Foundation, ce qui signifie qu'elle garantit ces 4 libertés essentielles à l'utilisateur [7] :

1. la liberté d'exécuter le programme (liberté 0) ;
2. la liberté d'étudier le fonctionnement du programme, et de le modifier (liberté 1) ;
3. la liberté de redistribuer des copies (liberté 2) ;
4. la liberté de distribuer aux autres des copies de vos versions modifiées (liberté 3).

QGIS est distribué sous la licence GPL version 2 (ou une version plus récente). La communauté de développeurs à l'origine de QGIS souhaite donc suivre les termes de cette licence. Lorsqu'une extension est ajoutée à QGIS, comme par exemple le plugin SOS 2.0 Client, les deux programmes que sont QGIS d'une part, et le plugin QGIS de l'autre, sont combinés dans une « création plus vaste ». D'après la Free Software Foundation, du point de vue de la propriété intellectuelle :

« *Pour combiner deux programmes (ou des parties importantes de ceux-ci) dans une « création plus vaste », vous devez avoir la permission d'utiliser les deux programmes de cette manière. Si les licences des deux programmes le permettent, elles sont compatibles. S'il n'y a aucun moyen de satisfaire les deux licences en même temps, elles sont incompatibles.* » [22]

La Free Software Foundation fournie une liste de licences compatibles avec la licence GPL. Pour la publication du plugin dans le dépôt officiel des plugins QGIS, nous avons donc choisi une licence de la liste : la licence BSD modifiée, ou licence BSD à trois clauses. Le texte de cette licence est reporté en **annexe 4**. Cette licence garantit l'ensemble des 4 libertés essentielles présentées plus haut, mais, à la différence de la licence GPL, elle ne pose aucune condition sur la nature des licences des programmes qui réutilisent le code source, et sur les droits conférés aux utilisateurs de ces programmes. On dit qu'il s'agit d'une licence *permissive*, là où la licence GPL est une licence *protectrice* relativement aux droits des utilisateurs. Nous avons choisi une licence permissive car, bien que nous soyons sensibles aux principes du développement Open Source, nous ne souhaitons pas empêcher la réutilisation du code source dans un programme propriétaire à vocation commerciale. En revanche, nous attachons une importance à à la protection du nom des auteurs du code source, et au fait qu'il soit fait mention des auteurs en cas de réutilisation du code source dans un autre programme. En lisant l'annexe 4, on pourra constater que c'est en fait précisément ce que la licence BSD modifiée protège, ce qui explique que nous l'ayons choisie.

Dans les instructions de la section de la documentation officielle de QGIS consacrée à la publication d'une extension, il est également demandé qu'une documentation minimale accompagne le dépôt du code source du plugin. J'ai donc ajouté au dossier GitHub contenant le code source ([G2]) une documentation qui :

- présente le plugin et ses fonctionnalités ;
- fournit les instructions relatives à son installation et à son utilisation ;
- contient le journal des modifications associées aux différentes versions publiées du plugin ;
- fournit les métadonnées du projet, dont notamment les informations de contact ;
- fournit des instructions permettant de contribuer au développement de l'extension en partageant une nouvelle version.

En suivant les instructions de la section de la documentation officielle de QGIS, j'ai pu soumettre le plugin SOS 2.0 Client pour approbation de son ajout au dépôt officiel des extensions Python. L'ajout a été approuvé par un membre de l'équipe responsable de la gestion du dépôt. Désormais, le plugin SOS 2.0 Client peut donc être téléchargé puis installé depuis une session QGIS via Extension ‣ Installer/Gérer les extensions. Un guide d'installation plus détaillé est fourni en **annexe 5**.

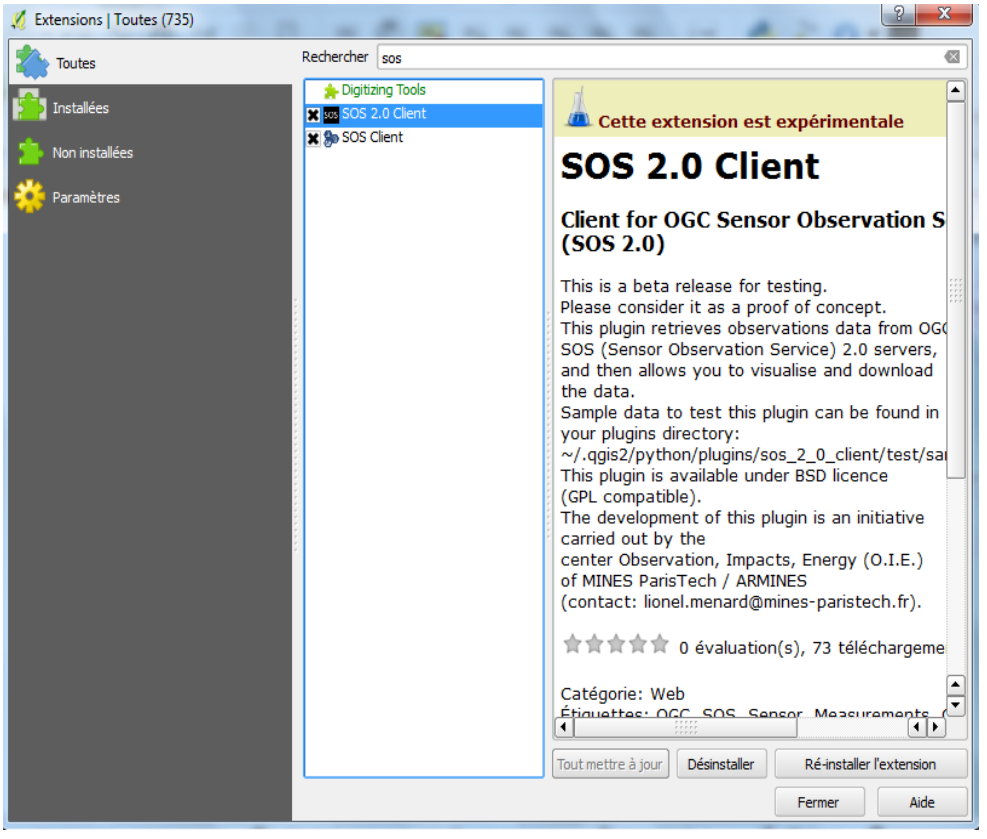


Figure 40 -Installation du plugin depuis le dépôt officiel dans QGIS

Il possède également sa propre page Web en tant qu'extension du dépôt officiel, comme en atteste la figure ci-dessous.

Figure 41 -Page Web du plugin SOS 2.0 Client comme extension QGIS officielle

Une fois le plugin ajouté au dépôt officiel des extensions QGIS, nous avons informé de sa disponibilité les collaborateurs des organismes partenaires intéressés par cette extension (présentés en **1.4.2**), ainsi que les membres de la liste de diffusion par email du forum Sensor Web Community 52°North ; puis nous avons invité l'ensemble des personnes intéressées à utiliser ce plugin. Au moment où cette ligne a été écrite, le plugin, disponible depuis le 26 Juillet 2017, a été téléchargé par **74 personnes**, comme l'indique la page du dépôt officiel.

Figure 42 -Page Web du dépôt officiel des extensions QGIS

5 Bilan et perspectives

5.1 Bilan des outils développés

5.1.1 Script Python

Durant ce stage, un premier script Python a été développé indépendamment de l'environnement QGIS. Il exploite la librairie Open Source OWSLib, conçue afin d'encourager le développement de clients pour la communication avec des serveurs d'implémentation de différents standards de l'OGC, dont notamment le standard SOS. Il offre aux utilisateurs capables de le lire, de le comprendre et de l'exécuter, les possibilités suivantes:

- afficher, pour chaque station de mesure dont des données sont stockées sur un serveur SOS 2.0 sélectionné par l'utilisateur, les coordonnées, dans le système WGS84, d'une zone de délimitation indiquant la localisation de la station ;
- afficher la liste des *offerings* de la station de mesure sélectionnée par l'utilisateur ;
- afficher la liste des grandeurs physiques mesurées pour l'*offering* sélectionnée ;
- afficher les dates de début et fin de la série temporelle complète des mesures *in situ* associées à l'*offering* sélectionnée ;
- récupérer les données d'une série temporelle de mesures *in situ* du serveur, une fois l'ensemble des paramètres mentionnés précédemment fixés ;
- afficher cette série temporelle dans un tableau ;
- afficher cette série temporelle dans un graphique ;
- exporter la série temporelle dans un fichier CSV.

Ce script a été déposé sur la plateforme de développement GitHub (cf [G1]). Son code source est ouvert, à la disposition de tous, sans restriction de réutilisation.

Des limites ont été identifiées lors de l'utilisation de ce script :

- La librairie OWSLib ne permet de récupérer que les séries temporelles de mesures *in situ* provenant de serveurs d'implémentation de la version 2 du standard SOS (SOS 2.0). Pour accroître les capacités de ce script Python, l'extension de cette même fonctionnalité à la version 1 du standard serait souhaitable. Dans ce but, des échanges avec les développeurs à l'origine de la librairie seront probablement nécessaires. Un point de départ pourrait être la page GitHub de la librairie.
- Lors de nos tests, nous ne sommes pas parvenus à récupérer, avec ce script Python, des séries temporelles d'un mois de mesures, ou plus, dans des délais inférieurs à la minute. Ce problème de performance est

imputé à la grande "verbosité" du format XML utilisé pour encoder la réponse du serveur SOS aux requêtes de type *GetObservation*, ainsi qu'à des bases de données trop lourdes, résultant des modèles de données utilisés par les implémentations du standard SOS, qui ralentissent le traitement des requêtes par les serveurs interrogés. Pour répondre à cette problématique connue, l'OGC, ainsi que les développeurs à l'origine des solutions d'implémentation du standard, ont déjà mis en place des solutions reposant sur remplacement du format XML par le format JSON. De nouveaux efforts dans cette direction pourraient permettre de généraliser l'utilisation du format JSON au sein des implémentations du standard, puis de rendre possible - via l'utilisation de la librairie OWSLib - la récupération par ce script Python des séries temporelles de mesures encodées dans ce format. .

5.1.2 Plugin QGIS

Un plugin QGIS baptisé SOS 2.0 Client a été développé. Il a été conçu de manière à permettre aux utilisateurs l'ayant installé dans QGIS de :

- Sélectionner un serveur SOS 2.0 ;
- Afficher la localisation des différentes stations de mesure dont les données de mesures sont stockées sur le serveur sélectionné au sein d'une couche vecteur cartographique dans QGIS ;
- Sélectionner spatialement, dans QGIS, une de ces stations ;
- Sélectionner un jeu de données correspondant à une série temporelle de mesures *in situ* à partir des attributs structurant la mesure ;
- Accéder aux métadonnées portant sur le serveur SOS 2.0 sélectionné ;
- Visualiser la série temporelle sous forme de tableau ou bien de graphique ;
- Télécharger le jeu de données de mesures *in situ* sur sa propre machine, dans un fichier au format CSV.

Cette extension a été téléversée dans le dépôt officiel des extensions Python pour QGIS. Son ajout au dépôt a été approuvé par un membre de l'équipe QGIS responsable de la gestion du dépôt. Ce plugin peut donc être installé depuis une session QGIS, comme détaillé dans l'**annexe 5**. Au moment où cette ligne est rédigée, il a été installé **74 fois** depuis le 26 Juillet 2017, date à laquelle il a été mis en ligne. Son code source a été distribué via GitHub (cf [G2]), sous la licence BSD modifiée. Il est ouvert, à la disposition de tous, sans restriction de réutilisation, exceptée la protection du nom de ses auteurs.

La dernière version publiée du plugin SOS 2.0 Client présente de nombreuses limites. Il s'agit d'une pré-version publiée afin de récolter des retours des utilisateurs l'ayant testée. 10 serveurs SOS 2.0 ont été utilisés pour tester ses capacités et ses limites. L'affichage de la localisation des stations de mesures par le plugin a été possible pour **8**

serveurs parmi les 10 serveurs testés, et l'affichage d'une série temporelle de mesures *in situ* avec **4** d'entre eux. Le développement du plugin n'est donc pas achevé. En particulier, certaines limites ont pu être identifiées. Des orientations sont suggérées dans le **tableau 6** en vue de répondre à chacune de ces limites, et d'améliorer l'extension QGIS SOS 2.0 Client.

Limite	Perspectives d'amélioration suggérées
Utilisation limitée à des serveurs SOS 2.0	Ajouter à la librairie OWSLib la capacité de récupérer les séries temporelles de mesures *in situ* provenant de serveurs d'implémentation de la version 1 du standard SOS (SOS 1.0). Dans ce but, des échanges avec les développeurs de la librairie seront probablement nécessaires. Un point de départ pourrait être la page GitHub de la librairie.
Performance insuffisante pour la récupération de séries temporelles de plus d'un mois de mesures	- Généraliser l'utilisation du format JSON au sein des implémentations du standard. De nombreux échanges avec la communauté des utilisateurs du standard, l'OGC, ainsi que les développeurs de solutions d'implémentation, seront probablement nécessaires. - Assurer une plus grande robustesse de la capacité de récupération, par la librairie OWSLib, des séries temporelles de mesures encodées au format JSON. Cela devrait permettre de corriger l'erreur Python présentée en **annexe 3**.
Limitations d'OWSLib quant aux formats d'encodage des mesures (*observation result type*). Le format {http://www.opengis.net/swe/2.0}DataArray n'est pas supporté par la librairie, ce qui génère une exception Python du type « *NotImplemented error* ».	Améliorer la librairie OWSLib en ajoutant à la liste des formats disponibles pour l'encodage des mesures le format {http://www.opengis.net/swe/2.0}DataArray.
Ergonomie et design de l'interface utilisateur perfectibles	Continuer le développement Python de l'extension QGIS afin d'améliorer l'ergonomie et le design de l'interface utilisateur.

Tableau 6 : Limites constatées de l'extension QGIS SOS 2.0 Client et suggestions d'améliorations techniques associées

L'extension QGIS publiée présente donc plusieurs limitations d'utilisation, et le constat que l'affichage par le plugin d'une série temporelle de mesures *in situ* n'a été possible qu'avec **4 serveurs parmi les 10 serveurs testés** peut paraître décevant.

Néanmoins, il est important de remettre ce résultat de développement dans son contexte. Dans le cadre de sa contribution à l'initiative Global Earth Observation System of Systems (GEOSS), le Centre O.I.E. s'investit dans la dissémination de données d'Observation de la Terre, parmi lesquelles figurent des mesures *in situ*. Si le centre joue

donc rôle de fournisseur de données de mesures *in situ*, il s'intéresse également aux moyens de récupération de ces données à sa disposition. C'est ainsi que le Centre s'est penché sur l'existence d'un client pour le standard SOS dans le SIG QGIS. Aucun client satisfaisant n'ayant été trouvé, l'intérêt d'offrir un nouveau service a été sondé auprès de la communauté des utilisateurs du standard SOS. Un intérêt certain a été manifesté, comme nous l'avons vu en **1.4.2**. Le Centre a alors décidé d'apporter une réponse technique à cet intérêt sous la forme d'une contribution au développement d'une extension QGIS, un client pour le standard SOS offrant le service attendu. Cette contribution, objet du stage, a donc abouti à la publication de l'extension QGIS SOS 2.0 Client. Sachant qu'il s'agit d'une contribution à un développement Open Source, basé sur le partage avec plusieurs collaborateurs, ce plugin, bien que largement perfectible, et dont la version publiée n'est qu'une préversion destinée à être testée, joue le rôle de démonstration technique, mettant en lumière la possibilité d'obtenir une extension QGIS répondant à la demande du service. Le plugin SOS 2.0 Client permet d'initier la discussion en invitant les autres collaborateurs à s'investir dans la suite du développement s'ils le souhaitent. Ainsi, la publication du code source en libre accès sur la plateforme GitHub, ainsi que nos efforts dans la communication des résultats obtenus en fin de stage, pourront permettre d'aboutir à une version stable, parfaitement robuste et offrant l'ensemble des fonctionnalités attendues.

Le choix de l'utilisation de la librairie OWSLib a permis de simplifier considérablement le développement, en fournissant les moyens d'automatiser les opérations d'écriture et de lecture de requêtes dont le contenu doit respecter un encodage strict, défini par le standard SOS. En revanche, il a introduit des restrictions relatives aux serveurs SOS compatibles avec l'utilisation du plugin. A l'avenir, étendre les capacités de cette librairie Python sera très probablement un prérequis à l'amélioration du plugin. Grâce à l'interopérabilité du standard SOS, la grande flexibilité et la versatilité du langage Python, cela pourrait également ouvrir la voie à d'autres applications du standard SOS.

5.2 Bilan personnel

5.2.1 Apports et enseignements du stage sur le plan personnel

Avant de parvenir à développer puis publier l'extension QGIS SOS 2.0 Client, il m'a fallu considérablement progresser en termes de capacités de programmation orientée objet, puis apprendre à utiliser la librairie OWSLib, l'API Qt – et par extension la librairie Python PyQt – ainsi que l'API QGIS en Python. Pour l'étudiant d'une école généraliste en génie urbain que j'étais, intégrer simultanément les contraintes liées au standard SOS, et la spécificité de l'environnement QGIS pour le développement d'une extension, a donné lieu à des difficultés techniques. Ces difficultés se sont traduites par des retards vis-à-vis des jalons de développement fixé par le diagramme de Gantt initial du stage, élaboré en compagnie de mon maître de stage, et disponible en **annexe 6**. Ce diagramme était peut-être un peu ambitieux, puisqu'il fixait l'objectif de publier une version stable du plugin QGIS, objectif qui n'a pas pu être totalement atteint. Néanmoins, mon maître de stage a insisté à maintes reprises sur le fait que pour ce

stage, j'avais une obligation de moyens, et non de résultat. Il s'agissait d'apporter une contribution, une première tentative pour répondre à une demande technique inédite. Certes, j'ai parfois ressenti de la frustration devant les difficultés générées par des éléments externes sur lesquels je n'avais aucun contrôle, comme le temps de réponse des serveurs SOS, et j'ai quelques fois eu l'impression d'atteindre mes limites. Mais cet exercice technique s'est finalement révélé très formateur, précisément car j'ai eu besoin de faire preuve de discernement pour repousser ces limites techniques. Il m'a fallu décomposer des problèmes - à première vue complexes - en une suite de problèmes de programmation élémentaires. Assurément, il me faudra reproduire cette nouvelle gymnastique mentale à l'avenir, et avoir pu m'y confronter au court de ce stage est une réelle chance.

J'aurais peut-être pu aller un peu plus loin dans le développement du plugin QGIS SOS 2.0 Client si j'avais bénéficié, préalablement au stage, d'une formation plus avancée à la programmation orientée objet en langage Python. Aussi, je souhaiterais, dans l'intérêt des futurs étudiants de l'EIVP, inviter l'équipe pédagogique de l'École à considérer l'intérêt que pourrait représenter l'introduction d'un cours de Python au programme des études. Ce langage est flexible, versatile, et très accessible pour les débutants. Python est libre, gratuit, et simple à installer. Dans le domaine de la géomatique, le langage Python joue un rôle prépondérant. Par exemple, « *Python est le langage de script d'ArcGIS. ArcGIS inclut une API Python, ArcPy, qui* [..] *donne accès à tous les outils de géotraitement, ainsi qu'à des fonctions de script et à des modules spécialisés, qui* [...] *aident à automatiser les tâches SIG.* » [23]. En ce qui concerne QGIS, on peut relever cette indication de la documentation officielle :

« *Il y a de nombreuses façon d'utiliser les dépendances python de QGIS […] :*
- *lancer des commandes dans la console Python de QGIS*
- *créer et utiliser des plugins en Python*
- *créer des applications personnalisées basées sur l'API QGIS* » [24].

Plus généralement, Python pourrait être utilisé par les élèves de l'EIVP pour apporter des solutions techniques (notamment en modélisation) dans plusieurs domaines majeurs du cycle ingénieur, par exemple la construction, l'eau ou encore les transports.

5.2.1 Lien avec les perspectives professionnelles

Ce stage m'a permis d'acquérir de nouvelles compétences techniques de programmation. Lors de mon début de carrière comme ingénieur de la Mairie de Paris, elles pourront s'avérer utiles afin de résoudre des problèmes techniques liés à la conception logicielle, ou bien d'échanger avec des techniciens spécialisés dans le développement informatique. En particulier, si j'ai l'opportunité d'exercer une fonction liée au traitement de l'information géographique, je pourrai suggérer la création d'autres applications QGIS au service des habitants, ou bien, plus simplement, avoir recours aux fonctionnalités Python des SIG dans des projets urbains.

Ce stage m'a aussi montré certaines limites du développement Open Source dans la recherche et développement. La grande autonomie attendue des développeurs Open Source requiert une maitrise technique totale des outils de développement utilisés, que possèdent les ingénieurs spécialisés. Et il me semble qu'en l'absence de cette maitrise, de longues périodes d'apprentissage et de travail sans résultats probants immédiats peuvent être nécessaires. Cela étant, j'ai également découvert les avantages que procure ce type de développement, l'intérêt des échanges avec des organismes partenaires pour un développement basé sur le partage, et les enjeux derrière l'interopérabilité des systèmes d'information. Ces différentes notions peuvent aussi s'intégrer dans les politiques publiques en faveur d'une ville intelligente et durable, qui mobilisent les collectivités locales comme la Ville de Paris. J'ai le sentiment que je pourrai les aborder avec une plus grande clairvoyance à l'avenir. Aussi, ce stage a abouti à une contribution à l'élaboration d'une solution technique pour la dissémination de données de mesures *in situ*, qui sont des données publiques. Cette contribution va dans le sens de la « démarche d'ouverture des données publiques » et de la « politique globale d'innovation ouverte » adoptées par la Ville de Paris [25], ce qui pourrait me permettre de valoriser les acquis de ce stage à l'avenir.

Conclusion

Ce stage a permis d'apporter la démonstration technique qu'il est possible de communiquer avec un serveur d'implémentation du service SOS à partir d'un script Python afin de visualiser puis de télécharger des séries temporelles de mesures *in situ* ; et qu'il est possible de développer une extension Python pour QGIS offrant ces mêmes fonctionnalités aux utilisateurs du SIG. Pour répondre à la demande exprimée initialement, une extension QGIS baptisée SOS 2.0 Client a été développée. Elle a été téléversée dans le dépôt officiel des extensions Python de QGIS. Elle peut donc être installée depuis une session QGIS. Les codes sources du script Python et de l'extension QGIS ont été déposés sur une plateforme de partage de code informatique. Ils peuvent être réutilisés par tous, sans restriction, exceptée la protection du nom de ses auteurs. Les tests d'utilisation de ces outils ont montré que la récupération de séries temporelles de données de mesures *in situ* stockées sur des serveurs SOS se heurte à des limites de performance induites par la verbosité du format XML utilisé par le standard. La généralisation de l'utilisation du format JSON pour les communications client-serveur SOS pourrait peut-être améliorer leur performance. De plus, les outils développés au cours de ce stage utilisent la librairie OWSLib, qui, si elle permet d'automatiser l'envoi, la réception, et le traitement de la requête *GetObservation* définie par le standard, limite également l'utilisation des outils aux serveurs SOS 2.0, et montre parfois des limites de robustesse. Plus généralement, l'extension SOS 2.0 Client a été publiée comme une contribution pour apporter une réponse à la demande identifiée, une première démonstration technique que cette demande peut être satisfaite. Elle présente encore de nombreuses limitations d'utilisation. L'affichage des stations de mesure associées au serveur SOS spécifié en entrée a été obtenu pour 8 des 10 serveurs testés, là où la visualisation d'une série temporelle de données de mesures *in situ* n'a été possible qu'avec 4 de ces 10 serveurs. Le plus souvent, les erreurs constatées interviennent lors de l'envoie de la requête *GetObservation* au serveur SOS par une fonction de la librairie OWSLib. Elles sont généralement dues à des défauts d'implémentation du service. En effet, le standard SOS est récent, et les solutions d'implémentation de ce standard restent perfectibles. Des développeurs Python intéressés par ce projet Open Source pourront chercher à l'améliorer, en ajoutant des fonctionnalités à la librairie OWSLib. Grâce à l'interopérabilité du standard SOS, les apports de ces ajouts pourraient dépasser le simple cadre du développement de cette extension QGIS, et ainsi contribuer à améliorer encore davantage la dissémination des données d'observation de la Terre.

Références

[1] Centre Observation Impacts Énergie (O.I.E.) - MINES ParisTech, [En ligne]. Available: http://www.oie.mines-paristech.fr/Accueil/. [Accès le 10 Mars 2017].

[2] Centre Observation Impacts Énergie (O.I.E.) - MINES ParisTech, «De l'observation des ressources aux impacts environnementaux pour les filières énergétiques renouvelables,» [En ligne]. Available: http://www.oie.mines-paristech.fr/Recherche/Presentation-generale/. [Accès le 10 Mars 2017].

[3] J. DUBRANNA, T. RANCHIN, L. MENARD and B. GSCHWIND, "Production and Dissemination of Marine Renewable Energy Resource Information," in *11th European Wave and Tidal Energy Conference*, 2015.

[4] Open Geospatial Consortium, «OGC standards and supporting documents,» [En ligne]. Available: http://www.opengeospatial.org/standards/. [Accès le 14 Mars 2017].

[5] Institut Mines-Télécom, «Une plateforme ouverte pour agréger les données in situ,» [En ligne]. Available: https://blogrecherche.wp.imt.fr/2015/07/17/une-plateforme-ouverte-pour-agreger-les-donnees-in situ/. [Accès le 11 Mai 2017].

[6] Open Geospatial Consortium, «OGC and OSGeo Sign Memorandum of Understanding,» [En ligne]. Available: http://www.opengeospatial.org/pressroom/pressreleases/944. [Accès le 18 Mai 2017].

[7] Free Software Foundation, «Les quatre libertés essentielles,» [En ligne]. [Accès le 30 Mars 2017].

[8] L. MENARD, D. NUST, K.-M. NGO, P. BLANC, S. JIRKA, J. MASO, T. RANCHIN and L. WALD, "Interoperable exchange of surface solar irradiance observations : A challenge," *Energy Procedia*, vol. 76, pp. 113-120, 2015.

[9] R. PEREZ, R. SEALS and A. ZELENKA, "Comparing satellite remote sensing and ground network measurements for the production of site/time specific irradiance data," *Solar Energy*, vol. 60, no. 2, pp. 89-96, 1997.

[10] Rédacteurs de la documentation QGIS, «The nature and limitations of WMS,» [En ligne]. Available: https://docs.qgis.org/2.6/en/docs/training_manual/online_resources/wms.html. [Accès le 2017 Mai 2017].

[11] S. Hisham, «A country built by innovation: The Netherlands,» 30 Novembre 2015. [En ligne]. Available: https://www.geospatialworld.net/article/a-country-built-by-innovation-the-netherlands/. [Accès le 27 Juillet 2017].

[12] OSGeo, «Developing Python Plugins,» [En ligne]. Available: http://docs.qgis.org/2.18/en/docs/pyqgis_developer_cookbook/plugins.html#plugin-metadata. [Accès le 28 Juillet 2017].

[13] OSGeo, «PyQGIS Developer Cookbook: Introduction,» [En ligne]. Available: http://docs.qgis.org/2.18/en/docs/pyqgis_developer_cookbook/intro.html#python-plugins. [Accès le 28 Juillet 2017].

[14] S. J. COX, "OGC Implementation Specification 07-022r1: Observations and Measurements-Part 1-Observation schema. Open Geospatial Consortium," 2007.

[15] T. KRALIDIS, «OWSLib 0.14.0 documentation,» [En ligne]. Available: https://geopython.github.io/OWSLib/. [Accès le 12 Avril 2017].

[16] Open Geospatial Consortium, «OGC® Sensor Observation Service Interface Standard» 2012.

[17] OGC Network, «How to retrieve information about the observations and sensor metadata provided by a sos 2.0?,» [En ligne]. Available: http://www.ogcnetwork.net/sos_2_0/tutorial/capabilities. [Accès le 2 Juin 2017].

[18] 52°North, «Sensor Web Best Practices,» [En ligne]. Available: https://52north.github.io/sensor-web-tutorial/07_client-software.html. [Accès le 12 Avril 2017].

[19] 52°North , «Sensor Web - SOS - Download,» 12 Juin 2015. [En ligne]. Available: http://52north.org/communities/sensorweb/sos/download.html. [Accès le 28 Juillet 2017].

[20] A. NA et M. PRIEST, «OpenGIS sensor observation service implementation specification,» *Open Geospatial Consortium Implementation Specification*, vol. 91, 2006.

[21] Association Francophone des Utilisateurs de Logiciels Libres, «Licences libres,» 20 Février 2015. [En ligne]. Available: https://aful.org/ressources/licences-libres. [Accès le 31 Juillet 2017].

[22] Free Software Foundation, «Foire aux questions sur les licences GNU - Qu'entendez-vous par « deux licences compatibles » ?,» 2014. [En ligne]. Available: https://www.gnu.org/licenses/gpl-faq.html#WhatIsCompatible. [Accès le 31 Juillet 2017].

[23] ESRI, «ArcGIS Desktop - Python et géotraitement,» 2017. [En ligne]. Available: https://pro.arcgis.com/fr/pro-app/help/analysis/geoprocessing/basics/python-and-geoprocessing.htm. [Accès le 2 Août 2017].

[24] Communauté des développeurs QGIS, «Documentation de QGIS 2.0 - PyQGIS Cookbook - Introduction,» [En ligne]. Available: https://docs.qgis.org/2.0/fr/docs/pyqgis_developer_cookbook/intro.html. [Accès le 2 Août 2017].

[25] Mairie de Paris - Paris Data, «Paris ouvre ses données publiques,» [En ligne]. Available: https://opendata.paris.fr/page/lademarche/. [Accès le 2 Août 2017].

[26] K. SB, «Why is JSON more lightweight than XML?,» 10 Septemnre 2012. [En ligne]. Available: https://stackoverflow.com/questions/12346349/why-is-json-more-lightweight-than-xml. [Accès le 28 Juillet 2017].

[27] Association Francophone des Utilisateurs de Logiciels Libres, «Groupe de travail sur l'interopérabilité,» [En ligne]. Available: https://aful.org/gdt/interop. [Accès le 20 Mars 2017].

[28] W3C Working Group, «Web Services Glossary,» 11 Novembre 2004. [En ligne]. Available: https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice. [Accès le 17 Février 2017].

[29] B. ESPINAR, P. BLANC, L. WALD, B. GSCHWIND, L. MENARD, É. WEY, C. THOMAS and L. SABORET, "HelioClim-3: a near-real time and long-term surface solar irradiance database," in *Workshop on" Remote Sensing Measurements for Renewable Energy"*, 2012.

[30] C. MARINI, I. BLANC, P. SINHA and A. WADE, "A Prospective Mapping of Environmental Impacts of Large Scale Photovoltaic Ground Mounted Systems Based on the CdTe Technology at 2050 Time Horizon," in *29th European Photovoltaic Solar Energy Conference and Exhibition*, 2014.

Liens GitHub

Ci-dessous sont listés les liens GitHub renvoyant vers les codes sources évoqués dans le corps de ce rapport :

[G1] Python :

Script permettant de visualiser et télécharger des séries temporelles de données de mesures *in situ* stockées sur un serveur SOS 2.0

[G2] Python :

Code source du plugin QGIS SOS 2.0 Client

Annexes

Annexe 1 : Requêtes GetCapabilities et GetObservation

Code XML 1 : Requête GetCapabilities pour le serveur insitu.webservice-energy.org

```
<?xml version="1.0" encoding="UTF-8"?>
<GetCapabilities xmlns="http://www.opengis.net/sos/1.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/sos/1.0
http://schemas.opengis.net/sos/1.0.0/sosGetCapabilities.xsd" service="SOS"
updateSequence="">
<ows:AcceptVersions>
   <ows:Version>1.0.0</ows:Version>
</ows:AcceptVersions>
<ows:Sections>
   <ows:Section>OperationsMetadata</ows:Section>
   <ows:Section>ServiceIdentification</ows:Section>
   <ows:Section>Filter_Capabilities</ows:Section>
   <ows:Section>Contents</ows:Section>
</ows:Sections>
</GetCapabilities>
```

La réponse du serveur est trop longue pour qu'il soit pertinent de l'insérer dans ce rapport. Elle est accessible à l'adresse suivante :

http://insitu.webservice-energy.org/52n-sos-webapp/sos\%3FREQUEST=GetCapabilities&SERVICE=SOS&ACCEPTVERSIONS=1.0.0

Code XML 2 : Exemple de requête du type GetObservation pour le serveur insitu.webservice-energy.org

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
    xmlns:env="http://www.w3.org/2003/05/soap-envelope"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2003/05/soap-envelope
http://www.w3.org/2003/05/soap-envelope/soap-envelope.xsd">
    <env:Body>
        <sos:GetObservation
            xmlns:sos="http://www.opengis.net/sos/2.0"
            xmlns:fes="http://www.opengis.net/fes/2.0"
            xmlns:gml="http://www.opengis.net/gml/3.2"
            xmlns:swe="http://www.opengis.net/swe/2.0"
```

```
          xmlns:xlink="http://www.w3.org/1999/xlink"
          xmlns:swes="http://www.opengis.net/swes/2.0"  service="SOS"  version="2.0.0"
xsi:schemaLocation="http://www.opengis.net/sos/2.0
http://schemas.opengis.net/sos/2.0/sos.xsd">
            <!--identifier of an offering-->
            <sos:offering>KISHINEV-DHI-sensor-QC1</sos:offering>
            <!--identifier of a featureOfInterest-->
            <sos:featureOfInterest>KISHINEV</sos:featureOfInterest>
            <!--identifier of a responseFormat-->
            <sos:responseFormat>http://www.opengis.net/om/2.0</sos:responseFormat>
            <!--identifier of a temporalFilter-->
            <sos:temporalFilter>
               <fes:During>
                  <fes:ValueReference>phenomenonTime</fes:ValueReference>
                  <gml:TimePeriod gml:id="duringPosition">
                     <gml:beginPosition>2015-11-24T23:00:00</gml:beginPosition>
                     <gml:endPosition>2010-11-25T23:06:00</gml:endPosition>
                  </gml:TimePeriod>
               </fes:During>
            </sos:temporalFilter>
      </sos:GetObservation>
   </env:Body>
</env:Envelope>
```

La réponse du serveur est accessible à l'adresse suivante :

http://insitu.webservice-energy.org/52n-sos-webapp/sos?service=SOS&request=GetObservation&version=2.0.0&offering=KISHINEV-DHI-sensor-QC1&featureOfInterest=KISHINEV&temporalFilter=om\%3AphenomenonTime\%2C2015-11-24T23\%3A00\%3A00Z\%2F2015-11-25T23\%3A00\%3A00Z&responseFormat=http\%3A//www.opengis.net/om/2.0

Annexe 2 : Comparaison de la structure d'encodage du contenu de la requête GetObservation aux formats XML et JSON

Si les formats de données textuelles JSON (JavaScript Object Notation) et XML (EXtensible Markup Language) sont tous deux utilisés pour représenter des structures de données « en arbre » (arborescences), XML est bien plus verbeux. Dans une arborescence XML, chaque élément possède un nom (le nom du type de l'élément), et l'élément est entouré d'un couple de balises concordantes. Le format JSON utilise pour sa part une notation en « tableaux imbriqués », dérivée du langage de programmation JavaScript [26]. Sa structure est faite de listes ordonnées et de paires nom/valeur.

Ci-dessous, le contenu d'une requête du type GetObservation pour le serveur insitu.webservice-energy.org a été encodé en XML puis en JSON, ce qui permet au lecteur de constater la différence de verbosité entre les deux formats.

Code XML 3 : Exemple de requête du type GetObservation pour le serveur insitu.webservice-energy.org

```
<?xml version="1.0" encoding="UTF-8"?>
<sos:GetObservation service="SOS" version="2.0.0"
    xmlns:sos="http://www.opengis.net/sos/2.0"
    xmlns:fes="http://www.opengis.net/fes/2.0"
    xmlns:gml="http://www.opengis.net/gml/3.2"
    xmlns:swe="http://www.opengis.net/swe/2.0"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:swes="http://www.opengis.net/swes/2.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/sos/2.0
http://schemas.opengis.net/sos/2.0/sos.xsd">
    <!-- optional -->
    <!-- optional, multiple values possible -->
    <sos:procedure>http://www.52north.org/test/procedure/1</sos:procedure>
    <sos:procedure>http://www.52north.org/test/procedure/2</sos:procedure>
    <!-- optional, multiple values possible -->
    <sos:offering>http://www.52north.org/test/offering/1</sos:offering>
    <sos:offering>http://www.52north.org/test/offering/2</sos:offering>
    <!-- optional, multiple values possible -->

<sos:observedProperty>http://www.52north.org/test/observableProperty/1</sos:observedPr
operty>

<sos:observedProperty>http://www.52north.org/test/observableProperty/2</sos:observedPr
operty>
    <!-- optional -->
    <sos:temporalFilter>
        <fes:During>
            <fes:ValueReference>phenomenonTime</fes:ValueReference>
            <gml:TimePeriod gml:id="tp_1">
                <gml:beginPosition>2012-11-19T14:00:00+01:00</gml:beginPosition>
```

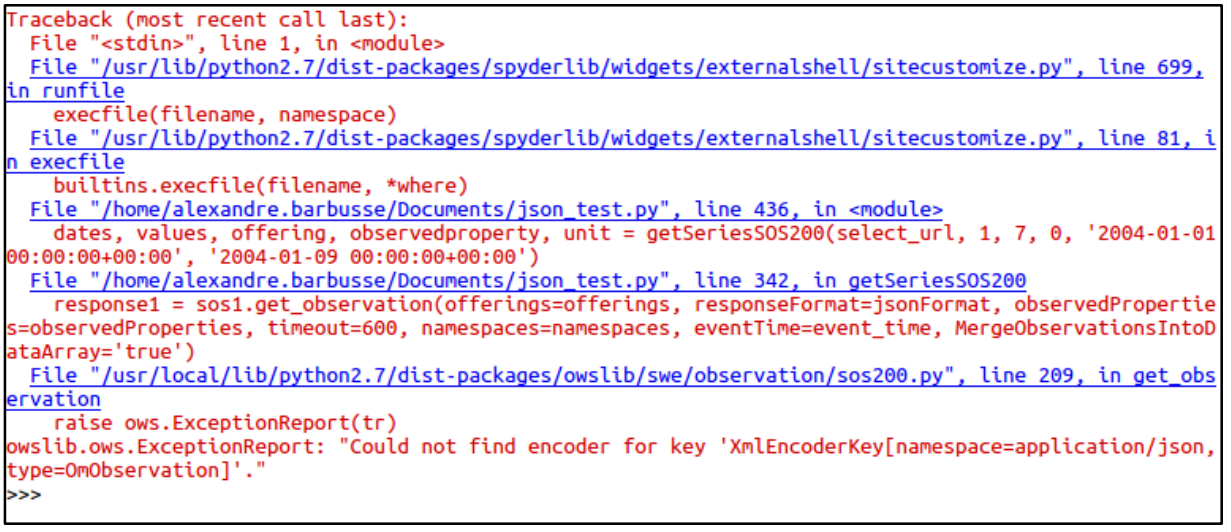
```
            <gml:endPosition>2012-11-19T14:05:00+01:00</gml:endPosition>
         </gml:TimePeriod>
      </fes:During>
   </sos:temporalFilter>

   <!-- optional, multiple values possible -->

<sos:featureOfInterest>http://www.52north.org/test/featureOfInterest/1</sos:featureOfI
nterest>

<sos:featureOfInterest>http://www.52north.org/test/featureOfInterest/2</sos:featureOfI
nterest>
   <!-- optional -->
   <sos:spatialFilter>
      <fes:BBOX>
         <fes:ValueReference>
            om:featureOfInterest/sams:SF_SpatialSamplingFeature/sams:shape
                </fes:ValueReference>
         <gml:Envelope srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
            <gml:lowerCorner>0 0</gml:lowerCorner>
            <gml:upperCorner>60 60</gml:upperCorner>
         </gml:Envelope>
      </fes:BBOX>
   </sos:spatialFilter>
   <!-- optional -->
   <sos:responseFormat>http://www.opengis.net/om/2.0</sos:responseFormat>
</sos:GetObservation>
```

Code JSON 1 : La même requête du type GetObservation au format JSON

```
{
  "request": "GetObservation",
  "service": "SOS",
  "version": "2.0.0",
  "procedure": [
    "http://www.52north.org/test/procedure/6",
    "http://www.52north.org/test/procedure/1"
  ],
  "offering": [
    "http://www.52north.org/test/offering/6",
    "http://www.52north.org/test/offering/1"
  ],
  "observedProperty": [
    "http://www.52north.org/test/observableProperty/1",
    "http://www.52north.org/test/observableProperty/6"
  ],
  "featureOfInterest": [
    "http://www.52north.org/test/featureOfInterest/6",
    "http://www.52north.org/test/featureOfInterest/1"
  ],
  "spatialFilter": {
    "bbox": {
```

```
      "ref": "om:featureOfInterest/sams:SF_SpatialSamplingFeature/sams:shape",
      "value": {
        "type": "Polygon",
        "coordinates": [
          [
            [
              0,
              0
            ],
            [
              60,
              60
            ]
          ]
        ]
      }
    }
  },
  "temporalFilter": [
    {
      "during": {
        "ref": "om:phenomenonTime",
        "value": [
          "2012-11-19T14:00:00+01:00",
          "2012-11-19T14:05:00+01:00"
        ]
      }
    },
    {
      "equals": {
        "ref": "om:phenomenonTime",
        "value": "2012-11-19T14:08:00+01:00"
      }
    }
  ]
}
```

Annexe 3 : Erreur Python rencontrée lors de l'utilisation du format JSON



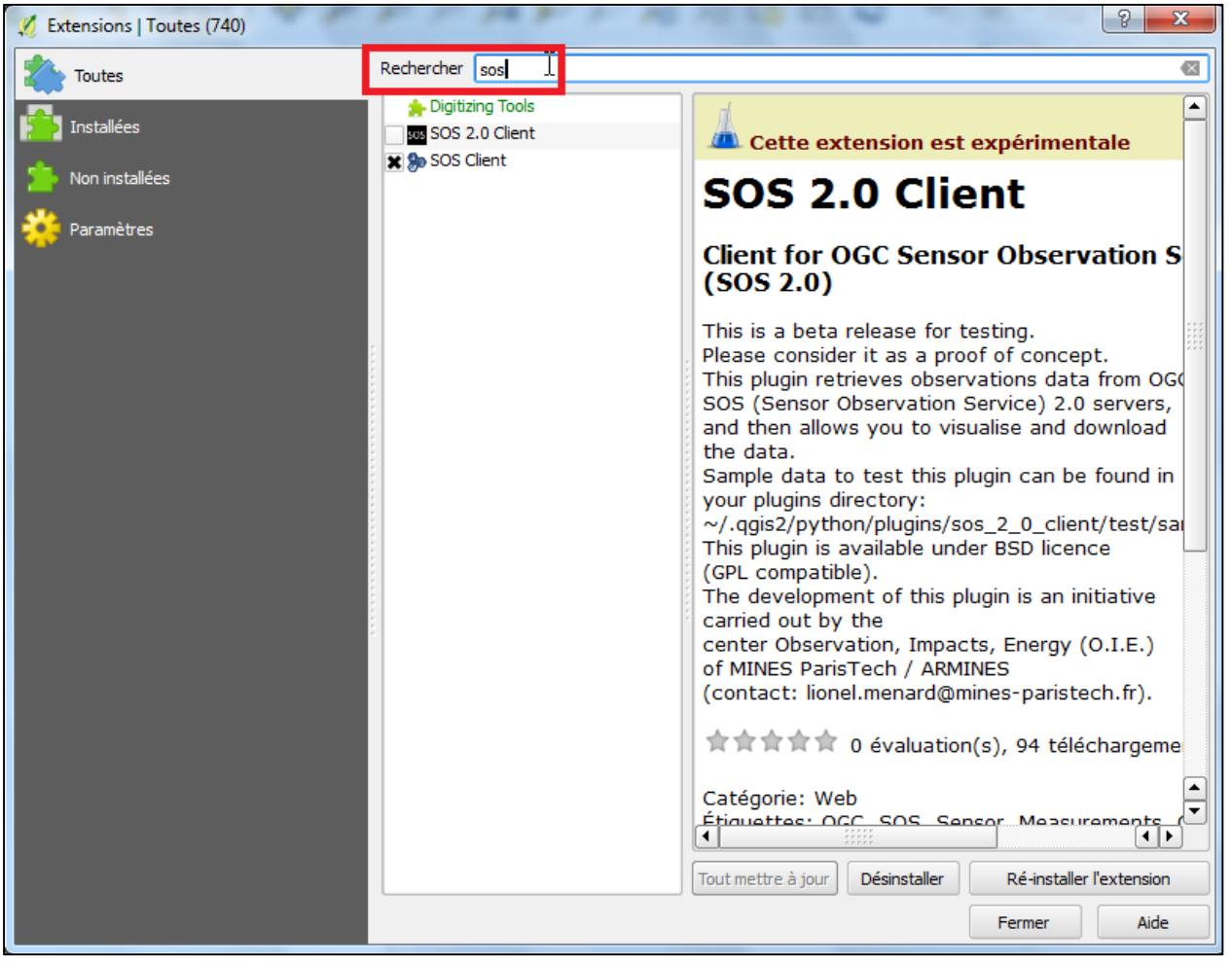
```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/lib/python2.7/dist-packages/spyderlib/widgets/externalshell/sitecustomize.py", line 699,
in runfile
    execfile(filename, namespace)
  File "/usr/lib/python2.7/dist-packages/spyderlib/widgets/externalshell/sitecustomize.py", line 81, i
n execfile
    builtins.execfile(filename, *where)
  File "/home/alexandre.barbusse/Documents/json_test.py", line 436, in <module>
    dates, values, offering, observedproperty, unit = getSeriesSOS200(select_url, 1, 7, 0, '2004-01-01
00:00:00+00:00', '2004-01-09 00:00:00+00:00')
  File "/home/alexandre.barbusse/Documents/json_test.py", line 342, in getSeriesSOS200
    response1 = sos1.get_observation(offerings=offerings, responseFormat=jsonFormat, observedPropertie
s=observedProperties, timeout=600, namespaces=namespaces, eventTime=event_time, MergeObservationsIntoD
ataArray='true')
  File "/usr/local/lib/python2.7/dist-packages/owslib/swe/observation/sos200.py", line 209, in get_obs
ervation
    raise ows.ExceptionReport(tr)
owslib.ows.ExceptionReport: "Could not find encoder for key 'XmlEncoderKey[namespace=application/json,
type=OmObservation]'."
>>>
```

Figure 43 -Message d'erreur présenté par la console Python lors de l'utilisation du format JSON dans le script [G1]

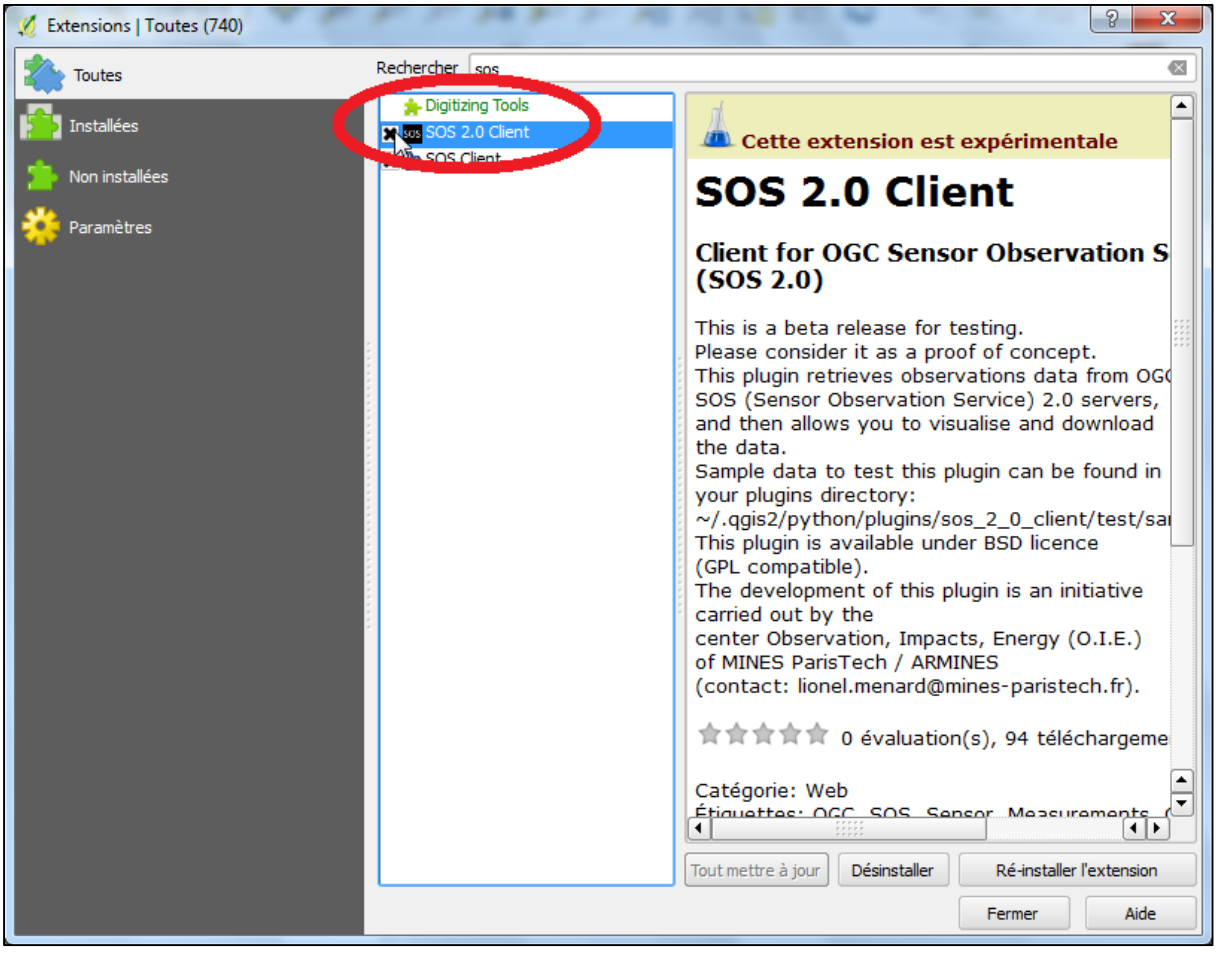
Annexe 4 : Texte de la licence BSD modifiée utilisée pour la distribution du plugin SOS 2.0 Client

Annexe 5 : Guide d'installation du plugin QGIS SOS 2.0 Client

1. Ouvrir QGIS.
2. Dans la barre de menu, cliquer sur Extension ‣ Installer/Gérer les extensions.
3. Dans la barre de recherche, taper "sos" pour trouver le plugin SOS 2.0 Client.



4. Cocher la case correspondant au plugin SOS 2.0 Client (voir illustration à la page suivante). Le plugin est maintenant installé.


Extensions | Toutes (740)
Toutes
Installées
Non installées
Paramètres
Rechercher
sos
Digitizing Tools
SOS 2.0 Client
SOS Client
Cette extension est expérimentale
SOS 2.0 Client
Client for OGC Sensor Observation S
(SOS 2.0)
This is a beta release for testing.
Please consider it as a proof of concept.
This plugin retrieves observations data from OG
SOS (Sensor Observation Service) 2.0 servers,
and then allows you to visualise and download
the data.
Sample data to test this plugin can be found in
your plugins directory:
~/.qgis2/python/plugins/sos_2_0_client/test/sa
This plugin is available under BSD licence
(GPL compatible).
The development of this plugin is an initiative
carried out by the
center Observation, Impacts, Energy (O.I.E.)
of MINES ParisTech / ARMINES
(contact: lionel.menard@mines-paristech.fr).
0 évaluation(s), 94 téléchargeme
Catégorie: Web
Tout mettre à jour
Désinstaller
Ré-installer l'extension
Fermer
Aide

Annexe 6 : Diagramme de Gantt du stage

Liste des tâches :

- T1 : Contexte et travail bibliographique
    - T1.1 : Contexte du laboratoire
    - T1.2 : Appropriation des enjeux
    - T1.3 : Compréhension des attentes du plugin
    - T1.4 : Rédaction de la première partie du rapport de stage, consacrée au contexte
- T2 : Production d'une version alpha du plugin QGIS
    - T2.1 : Découverte du cadre de travail et du référentiel de développement
    - T2.2 : Réalisation d'un plugin QGIS basique, prise de marque
    - T2.2bis : Réalisation d'un script Python offrant les mêmes fonctionnalités que le plugin QGIS
    - T2.2ter : Réalisation d'une interface graphique basée sur Qt préfigurant l'interface utilisateur du plugin QGIS
    - T2.3 : Réalisation d'un plugin QGIS simple permettant d'utiliser les fonctionnalités élémentaires d'un service de type SOS
    - T2.4 : Production de la version alpha du plugin
- T3 : De la version alpha à la version stable
    - T3.1 : Travail d'alpha-test en interne : corrections, élaboration d'une version bêta
    - T3.2 : Distribution de la version bêta à des collaborateurs ciblés
    - T3.3 : Suivi et corrections de la version bêta à partir des retours des bêta-testeurs, production de la version stable de l'extension QGIS
- T4 : Suite de la rédaction du rapport

Diagramme de Gantt initial

Tâches	Temps prévu (en jours)	Sem1 Février	Sem2 Fév	Sem3 Fév	Sem4 Fév	Sem5 Fév/ Mars	Sem6 Mar	Sem7 Mar	Sem8 Mar	Sem9 Mar/ Avril	Sem10 Avril
T1.1	5	III	II								
T1.2	11		III	IIIII	III						
T1.3	10				II	IIIII	III				
T1.4	5						II	III			
T2.1	5							II	III		
T2.2	5								II	III	
T2.3	5									II	III
T2.4	2										II
T3.1											
TOTAL	48										
DUREE	48										

Tâches	Temps prévu (en jours)	Sem11 Avr	Sem12 Avr	Sem13 Avr	Sem14 Mai	Sem15 Mai	Sem16 Mai	Sem17 Mai	Sem18 Mai/ Juin	Sem19 Juin	Sem20 Juin
T1.1											
T1.2											
T1.3											
T1.4											
T2.1											
T2.2											
T2.3											
T2.4	10	IIIII	IIIII								
T3.1	20			IIIII	IIIII	IIIII	IIIII				
T3.2	2							II			
T3.3	13							III	IIIII	IIIII	
T3.4	5										IIIII
T4											
TOTAL	50 (98)										
DUREE	50 (98)										

Tâches	Temps prévu (en jours)	Sem21 Juin	Sem22 Juin/ Juillet	Sem23 Juil	Sem24 Juil	Sem25 Juil	Sem26 Juil	Sem27 Juil/ Août
T1.1								
T1.2								
T1.3								
T1.4								
T2.1								
T2.2								
T2.3								
T2.4								
T3.1								
T3.2								
T3.3								
T3.4	10	IIIII	IIIII					
T4	21			IIIII	IIIII	IIIII	IIIII	I
TOTAL	31 (129)							
DUREE	31 (129)							

Diagramme de Gantt final

Tâches	Temps prévu (en jours)	Sem1 Février	Sem2 Fév	Sem3 Fév	Sem4 Fév	Sem5 Fév/ Mars	Sem6 Mar	Sem7 Mar	Sem8 Mar	Sem9 Mar/ Avril	Sem10 Avril
T1.1	5	III	II								
T1.2	11		III	IIIII	III						
T1.3	4				II	II					
T1.4	5					I	I	II		I	
T1.4bis	8						II	III	III		
T1.4ter	4						II		II		
T2.1	1										I
T2.1bis	6									IIII	II
T2.2	2					II					
T2.3	2										II
TOTAL	48										
DUREE	48										

Tâches	Temps prévu (en jours)	Sem11 Avr	Sem12 Avr	Sem13 Avr	Sem14 Mai	Sem15 Mai	Sem16 Mai	Sem17 Mai	Sem18 Mai/ Juin	Sem19 Juin	Sem20 Juin
T1.4bis											
T1.4ter											
T2.1	2	II									
T2.1bis											
T2.2											
T2.2bis	22		IIII	IIII	IIII	IIII	IIII	II			
T2.2ter	3									I	II
T2.3	5	II							III		
T2.4											
T4	18	I	I	I	I	I	I	III	II	II II	III
TOTAL	50 (98)										
DUREE	50 (98)										

Tâches	Temps prévu (en jours)	Sem21 Juin	Sem22 Juin/ Juillet	Sem23 Juil	Sem24 Juil	Sem25 Juil	Sem26 Juil	Sem27 Juil/ Août
T1.4ter								
T2.1								
T2.1bis								
T2.2								
T2.2bis								
T2.2ter	8	IIII	III					
T2.3	6		II	III				
T2.4	11			I	IIII	IIII		
T3.1	3						III	
T3.2								
T3.3								
T4	3						II	I
TOTAL	31 (129)							
DUREE	31 (129)							

Glossaire

Client :

Logiciel qui envoie des requêtes à un serveur et récupère les réponses à ces requêtes retournées par ce serveur.

***In situ* :**

Du latin *in situ* : « sur place », ici pour rendre compte du fait que les mesures sont réalisées par des capteurs au sol.

Interface de programmation (API) :

Un ensemble de ressources, similaire à un mode d'emploi, qui permet à un système informatique d'utiliser de façon automatique les fonctionnalités d'un autre système informatique.

Interopérable :

« Capacité que possède un produit ou un système, dont les interfaces sont intégralement connues, à fonctionner avec d'autres produits ou systèmes existants ou futurs et ce sans restriction d'accès ou de mise en œuvre ». [27]

Librairie :

Un ensemble de fichiers qui, une fois importé au sein d'un code informatique de plus haut niveau, permet d'exécuter des opérations complexes à programmer par le simple appel d'outils déjà implémentés, comme une fonction.

Ouvert :

Conçu ou configuré de manière à permettre le libre accès par des personnes et/ou des ordinateurs, sans aucune restriction.

Service Web :

« Un service web est un système logiciel conçu pour permettre une interaction interopérable de machine à machine via un réseau [(un sous-réseau du web)] ». [28]

Standard :

Dont le contenu est encodé dans des formats standardisés, et avec lequel la communication, l'interconnexion et l'échange sont structurés par des standards.