Université de Toulouse

MASTER 2 GEOMATIQUE

« **Sc**I**e**nces **G**éomatiques en environne**M**ent et **A**ménagement » (**SIGMA**)

http://sigma.univ-toulouse.fr

RAPPORT DE STAGE

Développement Web de la liaison BIM-SIG sur les outils ESRI

Source : Autodesk

DUFLOS Arthur

Magellium

Maître de stage : Guillaume Olive
Enseignant-référent : David Sheeren

Septembre 2020

Résumé

Dans le cadre de mon projet de fin d'étude, j'ai eu l'opportunité de travailler au sein de l'entreprise Magellium sur le développement Web autour de la liaison BIM-SIG avec les outils ESRI. Au cours de ce stage j'ai pu aborder deux grandes thématiques que sont le BIM (Building Information Modelling) et la cartographie en ligne. Le BIM consiste à représenter les bâtiments en trois dimensions sous la forme de maquette numérique. Il est très utilisé dans le secteur du bâtiment, tant par les architectes que par les maitres d'œuvres. Grâce à la collaboration entre Magellium et Tisséo, j'ai eu l'occasion de travailler sur des données réelles mais aussi d'avoir des cas d'utilisations. La première partie de mon stage a permis de convertir les données issues du monde du BIM en format de données typiques de la géomatique propriétaire, la GéoDataBase. Une fois ces données converties, elles ont été stockées sur un serveur afin de pouvoir être utilisé sur la solution de cartographie en ligne d'ESRI : le Web AppBuilder. La seconde partie a alors consisté à développer des outils permettant d'améliorer l'expérience utilisateur mais également de répondre aux besoins formulés lors de nos échanges avec Tisséo. Ces exemples de besoins ont été le point de départ aux outils que nous avons créés tels qu'un outil de requête attributaire, ou encore la visualisation de bâtiments étage par étage. Les outils développés sont une première piste permettant de montrer l'intérêt que peuvent représenter les SIG pour le monde du BIM.

Abstract

For the last four months, I had the opportunity to work within the Magellium company on web development around the BIM-GIS link with ESRI tools. This internship dealed with two theme : BIM (Building Information Modeling), and web mapping. BIM consists in a three dimensional representation of buildings as a digital model. It is widely used in the building sector, both by architects and by contractors. Thanks to the collaboration between Magellium and Tisséo, I had the opportunity to work on real data but also to have use cases. The first part of my internship allowed to convert data from the world of BIM into data format typical of proprietary geomatics, the GeoDataBase. Once this data was converted, it was stored on a server so that it could be used on ESRI's web mapping solution : the Web AppBuilder. The second part then consisted in the development of tools to improve the user experience but also to propose solutions to the needs expressed during our discussions with Tisséo. These sample needs were the starting point for the tools we created, such as an attribute query tool, or even the visualization of buildings floor by floor. The tools developed are a first avenue for showing the interest that GIS can represent for the world of BIM.

Glossaire

API : *Application Programming Interface*. Ensemble de fonctions et classes appelables et facilitant le développement de fonctionnalités

ArcGIS Pro : Logiciel de SIG de l'éditeur ESRI, permettant la visualisation de données 2D et 3D et de leur appliquer des géotraitements

Arcpy : Librairie python de fonctions SIG développée par ESRI, et disponible dans ses logiciels

As-Built : Ensemble de mesures effectuées à la fin d'un projet, souvent différentes de celles annoncées dans les plans initiaux

Attribut : Paramètre décrivant un objet, commun aux objets d'une même classe en programmation orientée objet

BIM : *Building Information Modelling*. Maquette numérique très utilisée dans le secteur du bâtiment mais désigne également les technologies qui y sont associées

CAO : Conception assistée par ordinateur

CPU : *Central Processing Unit*. Processeur principal chargé de l'exécution des tâches

Experience Builder : Outil de cartographie en ligne développé par ESRI, disponible depuis 2020 en version Bêta

Fonction asynchrone : Fonction dont l'exécution ne se fait qu'une fois l'exécution des autres fonctions terminée

GDB : *GeoDataBase.* Dossier système de fichiers communs, rassemblant des jeux de données géographiques de différents types

Géosignet : Emplacement géographique à sauvegarder pour une prochaine utilisation

GPU : *Graphics Processing Units*. Processeur assurant le calcul des fonctions d'affichage

JSON : *JavaScript Object Notation*. Format de données textuelles issues de la notation des objets en Javascript

Level of Details : Niveau de détail, Echelle permettant de décrire la précision et la qualité d'une maquette BIM

Revit : Logiciel de CAO utilisé pour la représentation 3D de bâtiments, et format de données issu de ce logiciel

Toolbox : Ensemble de script permettant d'accomplir des traitements dans les logiciels SIG d'ESRI

Web AppBuilder : Solution de cartographie en ligne développé par ESRI, reposant sur des objets stockés sur un serveur ArcGIS Server

Web mapping : Cartographie en ligne

WebMercator : Projection popularisée par Google Maps depuis 2005, la norme pour les applications de cartographie en ligne

Widget : Application permettant l'accès à des services. Dans le cas du Web AppBuilder, les widgets permettent les géotraitements, ou l'amélioration de la visualisation par exemple

Table des matières

Introduction 1

I- La liaison BIM-SIG, une relation intéressante mais qui reste à étoffer 3

A- Présentation de Magellium 3

B- Le BIM qu'est-ce c'est ? 4

C- Quels atouts apportés par les SIG ? 7

II- Le passage des données Revit à un format de la géomatique propriétaire 10

A- Présentation des objectifs 10

B- La bibliothèque ArcPy 10

C- Conversion des données 11

D- Les difficultés rencontrées par un développeur junior sur une librairie python propriétaire 18

III- Une facilité d'utilisation apportée par le Web AppBuilder et la cartographie en ligne 20

A – Présentation des objectifs 20

B – Les outils ESRI pour la cartographie en ligne 21

C – Le développement de Widgets pour améliorer l'expérience utilisateur 23

Conclusion et bilan de mon parcours 33

Table des Figures

Figure 1 : Représentation des niveaux de détails des maquettes, (Source personnelle) **5**

Figure 2 : Outils de géoréférencement d'un modèle Revit **8**

Figure 3 : Segmentation des objets d'ArcGIS Pro **11**

Figure 4 : Synthèse des traitements appliqués au fichier Revit **17**

Figure 5 : Vue du widget SlidesMag **24**

Figure 6 : Cycle de vie d'un Widget **25**

Figure 7 : Vue de la page de réglages du Widget QueryMag **28**

Figure 8 : Résultats d'une requête du widget QueryMag **29**

Figure 9 : Filtre par étage et interface utilisateur du widget FilterMag **31**

Table des Codes

Code 1 : Retranscription du système de coordonnées en chaine de caractères **8**

Code 2 : Conversion du fichier Revit en GDB **13**

Code 3 : Gestion de la symbologie des couches **15**

Introduction

Pendant les quatre derniers mois, j'ai eu l'opportunité de réaliser mon projet de fin d'études au sein de l'entreprise Magellium à Toulouse. Cette entreprise édite des solutions logicielles sur des thématiques diverses de la géomatique. Ce stage portait sur le développement Web autour de la liaison BIM-SIG sur les outils ESRI.

Le BIM (*Building Information Modelling*) est une technologie très utilisée dans le monde du bâtiment. Il permet de rendre les informations disponibles pour un grand nombre d'acteurs de ce secteur. Il s'agit d'une maquette numérique en trois dimensions, voire quatre avec la temporalité du projet. Les utilisations du BIM sont en plein développement, de nombreuses entités souhaitant l'utiliser à des fins de communications, de gestion de biens immobiliers, ou de gestion de projet.

Les systèmes d'informations géographiques et le BIM convergent tous deux vers une entité commune, le bâtiment. Les SIG partent d'entités beaucoup plus vastes, comme une ville ou une région, le BIM part du bâtiment et représente des objets aussi petits que les serrures d'une porte. Partant de ce constat, ESRI, la société éditrice de logiciels de systèmes d'informations géographiques, et Autodesk, société reconnue pour ces logiciels de la conception assistée par ordinateur (CAO) ont passé des accords permettant d'intégrer mutuellement leurs formats de données dans les logiciels de chacun. Ces accords ont pour but de permettre à des gestionnaires d'infrastructure de pouvoir gérer leur parc immobilier d'une manière globale, tout en pouvant récupérer des informations très précises. Il est possible d'utiliser les données issues du BIM dans des solutions de cartographie en ligne comme le Web AppBuilder. Ces accords sont assez récents, de nouvelles fonctionnalités sont régulièrement implémentées mais pour des entreprises comme Magellium, cela représente une opportunité de développer leurs propres fonctionnalités.

Mon stage s'est inscrit dans ce contexte de nouveautés. Il s'agissait de proposer des fonctionnalités qui ne sont pas proposées par les outils d'ESRI pour améliorer l'expérience utilisateur du consommateur de BIM, et ainsi démontrer les atouts que pouvaient apporter les SIG au monde du BIM. Un partenariat est né entre Magellium et Tisséo, qui a recours à des maquettes BIM dans le cadre de ses activités. Ce partenariat a généré des discussions autour des besoins de Tisséo sur le BIM mais a aussi permis de développer les nouvelles fonctionnalités sur des données réelles.

Afin de développer de nouvelles fonctionnalités sur les outils de la cartographie en ligne, il était également nécessaire de traiter les données BIM afin de mieux les intégrer aux logiciels SIG d'ESRI. Après avoir décrit l'entreprise Magellium et la thématique du BIM, ces développements feront l'objet de la deuxième partie de ce rapport. Enfin, dans un dernier temps, nous verrons les différents développements de fonctionnalités que j'ai pu aborder sur les widgets du Web AppBuilder.

I- La liaison BIM-SIG, une relation intéressante mais qui reste à étoffer

A- Présentation de Magellium

Magellium est une entreprise créée en 2003 à Toulouse dont l'objectif initial était de proposer des solutions aux problèmes d'acteurs comme le CNES, l'IGN, ou SpotImage. Depuis, l'entreprise n'a cessé de se développer en intégrant, entre autres, une branche dédiée aux demandes de la défense. Aujourd'hui l'entreprise compte près de 200 collaborateurs et possède une antenne à Paris. Le chiffre d'affaire de Magellium s'élevait à 17M d'euros en 2018. Environ 15% du chiffre d'affaire est réinvesti dans des projets de recherche et développement.

L'expertise de Magellium est reconnue dans des domaines variés tels que l'ingénierie logicielle des systèmes d'informations géographiques, l'intelligence artificielle et la robotique, le développement d'outils de production de données géographiques ou encore le traitement d'image. Une partie de ses activités est dédiée à l'export, notamment vers l'Europe et le Moyen-Orient.

Les solutions SIG proposées peuvent à la fois reposer sur des technologies du monde libre ou sur les logiciels ESRI. Magellium possédant une expertise dans ces deux mondes, le choix de la technologie peut se faire entièrement en fonction du fonctionnement du client et de ses besoins.

Les produits proposés par Magellium sont développés pour des clients très variés qui n'ont pas forcément de notions ou de compétences en géomatiques, l'entreprise peut donc proposer une formation pour maitriser les solutions clé-en-main, ce qui requiert en plus de compétences techniques des qualités pédagogiques.

Quelques projets de Magellium portant sur les domaines de la cartographie en ligne ou des technologies 3D :

- Architecture SIG Mutualisée

  Ce projet a été mené pour le compte d'IDFM (Ile-De-France Mobilité) sur 2 ans. Il a pour but de proposer des applications Web permettant aux utilisateurs (internes) de publier ou de rechercher des cartes et d'avoir accès à toutes les données métiers présentes sur le

serveur en plus de proposer des fonctionnalités génériques telles que le calcul d'itinéraires. Cette application repose sur les services ESRI pour la gestion des serveurs.

- Maquette numérique Grand Paris Express

  La société du Grand Paris a confié à un consortium composé de Magellium – Systra – Vectuel ainsi que leur sous-traitant la réalisation de la Maquette Numérique et son administration. La réussite de ce projet reposait sur 3 services opérationnels :

  - L'interopérabilité des données, assurée par les formats propres au BIM
  - Le maquettage, une production de la maquette précise de l'existant et du bâti à l'état de projet
  - L'infrastructure système, permettant la diffusion des données à partir d'un entrepôt de données 3D et un entrepôt de données 2D

Au travers de ses différents projets, Magellium a pu proposer des solutions de cartographie en ligne mais a aussi pu aborder les thématiques du BIM, des technologies qui sont encore très peu utilisées dans le monde de la géomatique. L'intérêt de l'entreprise pour ce domaine s'est développé et des sujets de stages ont alors été déposés :

- La mise en place d'une relation BIM-SIG sur les outils du monde libre
- Le développement Web de la relation BIM-SIG sur les outils ESRI

B- Le BIM qu'est-ce c'est ?

Historiquement, le Building Information Modeling (BIM) ou Bâti Immobilier Modélisé depuis Janvier 2019 (JORF, 2019), est né d'un besoin des acteurs du bâtiment de faciliter les échanges autour de ce domaine. Le BIM est un processus permettant la visualisation de modèle 3D. Il s'agit d'une maquette numérique qui présente les mêmes caractéristiques qu'une maquette mais auxquelles peuvent s'ajouter des éléments propres au monde informatique telles que des tables de bases de données. En 1995, commence la création de ce qui aboutira aux IFC, développé par buildingSMART International (buildingSMART, 2020). Cette alliance internationale à but non lucratif est spécialisée dans l'interopérabilité des données. Elle produit des outils libres et open-sources.

La précision du modèle est définie par le LOD, le Level Of Details (M. J. Sani et A. Rahman, 2018) sur 5 niveaux, du LOD0 au LOD4. Comme le montre la Figure 1, inspirée par les travaux de M. J. Sani et A. Rahman, le niveau 0 ne représente que l'emprise au sol en deux dimensions, le niveau 4 quant à lui est le plus détaillé et contient les murs intérieurs ainsi que des éléments du mobilier. Des niveaux de détails intermédiaires ont été définis afin de pouvoir mieux apprécier la qualité de la maquette.

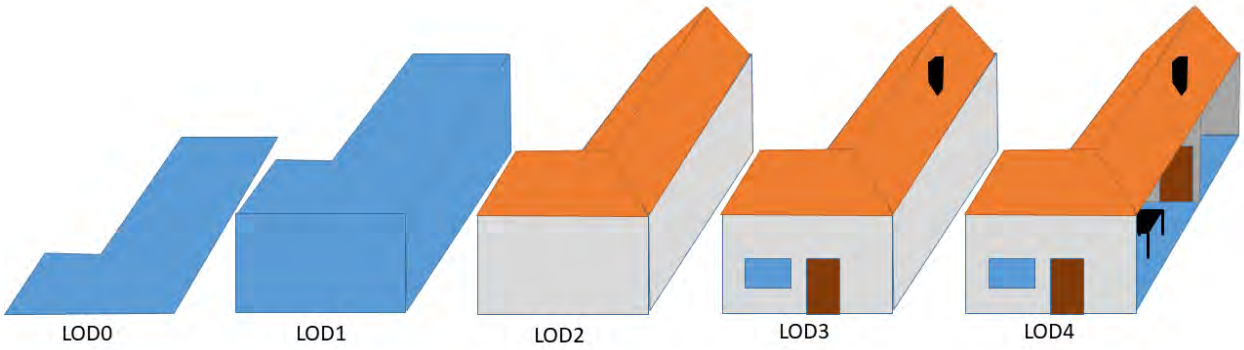

Figure 1 : Représentation des niveaux de détails des maquettes, (Source personnelle)

Le standard IFC :

L'Industry Foundation Classes est un format de fichier standardisé reposant sur la norme ISO 16739. Il est écrit en langage Express. Le modèle IFC définit une hiérarchie des objets, une fenêtre, par exemple, est placé dans le référentiel d'un mur, ce même mur dans une pièce, et ainsi de suite jusqu'aux bâtiments qui sont placés au sein d'un site. Cette arborescence permet de faciliter la multiplication d'éléments répétitifs ainsi l'objet n'est pas dupliqué mais un lien est créé dans l'arbre IFC pour le retrouver et le placer aux différentes positions auxquelles il appartient.

Des formats propriétaires

En plus des fichiers IFC, les maquettes BIM peuvent être enregistrées dans des formats propriétaires à partir de logiciels dédiés à la modélisation assistée par ordinateur. Autodesk, un éditeur de logiciel de conception assisté par ordinateur, leader dans son domaine, propose

plusieurs logiciels permettant de créer ou d'éditer des maquettes BIM. Chez cet éditeur on peut citer les logiciels BIM360 et Revit. Ces logiciels ont aussi pour but de faciliter la connexion entre les différents corps de métiers qui seraient susceptibles d'intervenir sur le projet décrit par la maquette. Ces fichiers ne peuvent être lus par tous les logiciels, mais ainsi que nous le verrons plus tard, les accords ESRI-Autodesk permettent leur intégration, plus particulièrement des données au format Revit, dans un logiciel de SIG. Le logiciel Revit, à la différence de certains logiciels d'autres éditeurs a la particularité d'enregistrer toutes les informations dans un seul fichier au format *.RVT*. Dans ce logiciel, un bâtiment est découpé en grande famille dont la symbologie est standardisé.

Quelles utilisations ?

Le BIM est à l'origine pensé pour favoriser le travail des acteurs du bâtiment. Dans ce secteur, et à la différence des autres secteurs de l'industrie, il est rare que les entreprises passent par des phases de prototype, et le BIM peut servir dans certains cas en tant que prototype. Il permet de faciliter leurs échanges. La possibilité pour les équipes en charge de la maitrise d'ouvrage de voir les plans de l'architecte sous différentes vues et d'avoir directement les informations sur la nature et le coût des matériaux diminuent les temps de latence qui peuvent être nombreux sur ce secteur. Les derniers outils utilisés pour créer et éditer des modèles BIM permettent d'ajouter une dimension temporelle au projet, il est donc possible de mieux visualiser l'avancée des travaux en le comparant à la maquette numérique. De plus, elle permet de centraliser les mesures qui pourraient être prises par les différents corps de métiers et artisans, qui prennent parfois plusieurs fois la même mesure mais avec un écart pouvant être source d'erreurs lors de la construction mais aussi dans les plans As-built (tel que construit).

Le BIM ne perd pas pour autant son utilité une fois les chantiers terminés puisqu'il peut être employé comme outil d'aide à la décision dans le suivi des infrastructures. Des applications métiers peuvent être créées afin de faciliter ces suivis comme le balisage des éléments électriques ou, le suivi de l'état des conduites d'air conditionnées, l'utilisateur a facilement accès à leurs positions et peut aussi enregistrer dans les tables les caractéristiques qu'il a vérifiées.

Enfin, dans un autre but que pour une utilisation technique, le BIM peut être envisagé comme un outil pédagogique et de communication, à la manière d'une maquette mais avec laquelle les interactions sont plus nombreuses et plus facile, chaque utilisateur pouvant avoir sa propre copie

et y ajouter ses éléments ou les modifier. Certains musées proposent ces maquettes BIM pour transmettre leurs informations de manière interactive, il est par exemple possible de mettre en évidence les évolutions architecturales qu'a subies un bâtiment, un quartier ou une ville au cours de l'histoire.

Tisséo

Ayant déjà collaboré ensemble, une relation de confiance s'est établie entre Tisséo et Magellium. Suite à des discussions sur le domaine du BIM, il est ressorti que les deux organismes étaient intéressés par cette thématique, et que Magellium pourrait être amené à répondre à un besoin de Tisséo sur le BIM. En tant que gestionnaire du réseau de transports en commun de Toulouse et donc de biens immobiliers, Tisséo gère à la fois des projets de constructions et le suivi de ces infrastructures. Pour certaines de ces infrastructures, il existe des fichiers BIM qui les modélisent, nous y retrouvons à la fois les parties architecturales et structurelles mais aussi les objets techniques tels que les alarmes incendies, les prises d'air ou les lampes. Les différentes voies des transports en commun de Tisséo sont également recréées sous la forme d'objet SIG. Le BIM représente une opportunité pour Tisséo d'améliorer sa connaissance des infrastructures dont elle a la charge. Cela pourrait permettre à l'état de projet de visualiser les emplacements de panneaux d'indications sur le long du parcours voyageur dans ces stations. Au cours des discussions avec le service SIG de Tisséo, nous avons pu obtenir des données BIM réelles issues d'opérations de sous-traitance, mais aussi de cas concrets sur les besoins des utilisateurs.

C- Quels atouts apportés par les SIG ?

Fin 2017, ESRI et Autodesk, ont passé un accord afin de favoriser l'intégration des objets de l'un vers l'autre. ESRI France parle de convergence BIM-SIG.

Les fichiers Revit peuvent donc être visualisés dans le logiciel ArcGIS Pro. Le modèle Revit est structuré en groupe de couches, un groupe correspondant au modèle, dans lequel nous retrouvons cinq sous-groupes qui constituent les grandes familles de fonctionnalités (architectural, structural, mechanical, piping, et electrical). Les tables attributaires des différentes couches du fichier sont accessibles mais non éditables. Les modèles peuvent être géoréférencés. Lors de la construction du fichier Revit, le dessinateur CAO peut choisir un système de coordonnées locales (propre au

projet) ou un système de coordonnées géographiques. Dans le premier cas, le positionnement est plus complexe (D. Gunther-Dilinger, 2016), l'utilisateur a accès à une série d'outil lui permettant de placer le modèle selon les axes X, Y et Z mais aussi de l'orienter et de lui appliquer un coefficient d'agrandissement. Ces différents outils sont illustrés sur la Figure 2.

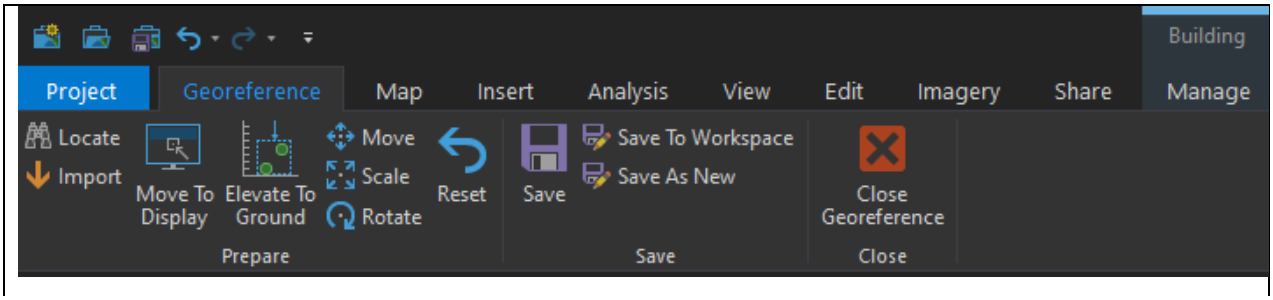

Figure 2 : Outils de géoréférencement d'un modèle Revit

Ce géoréférencement manuel est alors enregistré dans deux fichiers séparés. Le fichier au format WLD3 contient les coordonnées de deux sommets du pavé englobant la totalité du modèle, le fichier PRJ, quant à lui, contient une chaine de caractère qui définit tous les paramètres d'un système de coordonnées dans lequel le modèle a été géoréférencé. Ce type de chaine est souvent utilisé par le logiciel d'ESRI et permet notamment de définir le système de coordonnées (Code 1) dans les fonctions d'Arcpy qui en ont besoin en entrée. Il comprend entre autres le nom du système de coordonnées, son EPSG ou encore le méridien de référence utilisé.

```
PROJCS["RGF_1993_CC43",GEOGCS["GCS_RGF_1993",DATUM["D_RGF_1993",SPHERO
ID["GRS_1980",6378137.0,298.257222101]],PRIMEM["Greenwich",0.0],UNIT["
Degree",0.0174532925199433]],PROJECTION["Lambert_Conformal_Conic"],PAR
AMETER["False_Easting",1700000.0],PARAMETER["False_Northing",2200000.0
],PARAMETER["Central_Meridian",3.0],PARAMETER["Standard_Parallel_1",42
.25],PARAMETER["Standard_Parallel_2",43.75],PARAMETER["Latitude_Of_Ori
gin",43.0],UNIT["Meter",1.0],AUTHORITY["EPSG",3943]]
```

Code 1 : Retranscription du système de coordonnées en chaine de caractères

Dans le deuxième cas, il est facile de replacer le modèle Revit dans son contexte géographique, grâce à une simple opération de projection. A la suite de cette projection, seul le fichier PRJ est nécessaire pour replacer le modèle. Une fois le géoréférencement effectué, d'autres fonctionnalités ont été implémentées, en plus du module 3D Analyst qui permettait déjà de faire des opérations sur un environnement doté d'une troisième dimension. Ainsi les données Revit peuvent être converties dans des formats permettant par exemple l'édition des tables

attributaires des différentes couches, ou permettant le calcul de distance, d'aire, ou de volume. Ces fonctionnalités ne concernent pas seulement la suite Desktop, puisqu'il est également possible d'utiliser des données issues du BIM sur les solutions de cartographie en ligne d'ESRI. Ces solutions feront l'objet d'un développement dans la suite de ce rapport.

ArcGIS Pro permet donc de charger et visualiser des ensembles de bâtiments provenant de différents fichiers Revit. Ce qui rend possible le recours, pour un acteur de l'urbanisme par exemple, à plusieurs sous-traitants à la conception des maquettes virtuelles. A l'aide de simples cliqués-glissés, nous pouvons charger les modèles Revit géoréférencés dans l'environnement ESRI.

Ces accords sont toujours d'actualité et de nouvelles fonctionnalités sont régulièrement ajoutées au logiciel ArcGIS Pro. Cependant dans l'état actuel des choses, ces fonctionnalités sont limitées et l'utilisateur peut parfois se sentir bridé. L'édition des tables attributaires n'est par exemple possible qu'après une opération de conversion de données, opération qui peut prendre un certain temps en fonction de la complexité et de la taille des données. Pour une navigation dans un environnement composé de plusieurs modèles Revit, comprenant chacun un grand nombre d'objets, ArcGIS Pro nécessite un ordinateur avec non seulement un CPU (Processeur central) puissant mais aussi, à la différence de ArcMap un GPU (Processeur graphique) avec une grande mémoire partagée. ESRI recommande une configuration quad cœur, une mémoire vive de 8 Go, ainsi que 4 Go de mémoire graphique. L'ordinateur que j'ai utilisé au cours de ce stage possède une mémoire vive de 16 Go et une carte graphique présentant une mémoire de 8 Go pourtant j'ai pu constater des ralentissements, des diminutions de la fluidité voire même quelques arrêts d'exécutions lors de mes traitements ou de navigations.

II- Le passage des données Revit à un format de la géomatique propriétaire

A- Présentation des objectifs

Comme il a été vu précédemment, il est possible d'utiliser les objets stockés au format Revit dans le logiciel ArcGIS Pro, mais ces interactions sont limitées et un grand nombre d'opérations appartenant aux SIG ne sont pas possibles. ESRI propose des outils de conversion applicables sur les fichiers Revit, que ce soit vers du Shapefile ou vers une GeoDataBase (GDB), un dossier système de fichiers communs, rassemblant des jeux de données géographiques de différents types. Cette GeoDataBase peut être sous la forme d'un fichier dans un premier temps mais pourra à terme être une GeoDataBase d'entreprise, avec la mise en place d'un SGBD qui permettra une gestion des bases de données plus commune.

Pour ce faire, la création d'une *toolbox*, boite à outils, développée en langage Python a été privilégiée. Le logiciel d'ESRI a été conçu pour pouvoir être manipulé par ce langage mais possède également sa propre librairie. La bibliothèque ArcPy permet d'interagir à la fois avec les fichiers SIG mais aussi avec l'interface du logiciel ArcGIS Pro. Une fois ces conversions effectuées, il s'agissait de modifier l'organisation des objets dans le logiciel afin d'améliorer leur visualisation et la navigation dans la vue 3D. Enfin, une fois toutes ces opérations effectuées, les données seront intégrées au portail de Magellium reposant sur ArcGIS Server.

B- La bibliothèque ArcPy

Disponible depuis la version 10 du logiciel ArcMap, ArcPy est une bibliothèque développée par ESRI. De nombreuses opérations de SIG telles que l'intersection, ou l'interaction avec les tables de données par requêtes, ont été traduites et sont maintenant possible en python. Cette bibliothèque est utilisable directement dans une fenêtre de l'interface du logiciel mais aussi dans un script indépendant compilé dans l'IDE (panel d'outil dédié à la programmation) d'ESRI.

Elle repose sur une programmation orientée objet, et segmente les objets présents dans les logiciels d'ESRI comme on peut le voir sur la Figure 3. Ces segments vont du projet, l'objet

englobant tous les autres, jusqu'aux champs des tables attributaires des différentes couches. Chaque couche appartient à une carte en deux dimensions ou une scène pour les objets en trois dimensions.

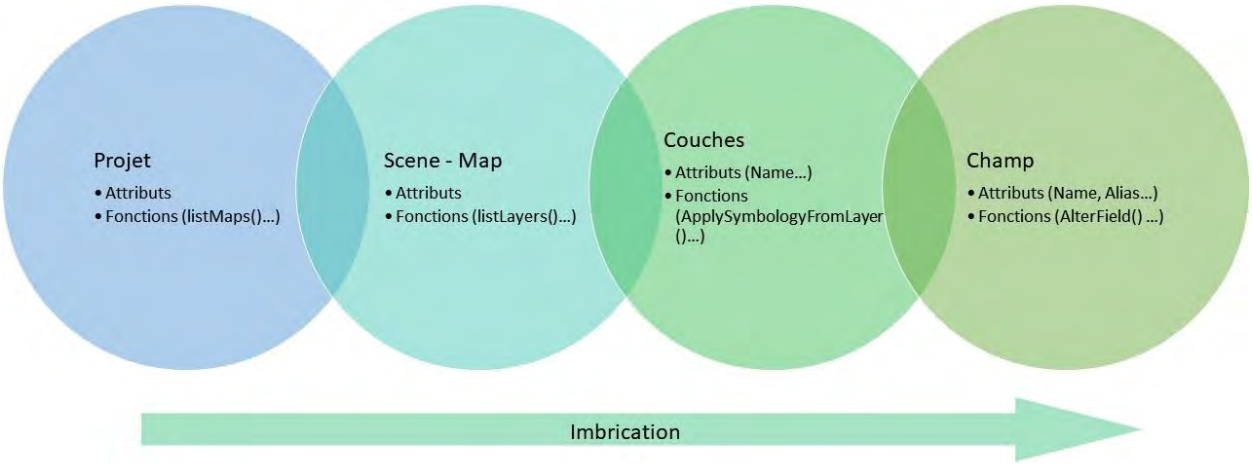

Figure 3 : Segmentation des objets d'ArcGIS Pro

La plupart des opérations et géotraitements disponibles sur l'interface ArcGIS a un équivalent python dont les paramètres sont des variables de types pouvant être appréhendés en langage python. Par exemple une couche en entrée peut être sous la forme d'une chaine de caractères indiquant l'emplacement de ce fichier sur le disque ou un objet de type Layer présent dans le projet.

C- Conversion des données

Les traitements décrits dans ce chapitre ont été développés sur des données mises à disposition par l'éditeur Autodesk pour les tutoriels de prise en main du logiciel Revit. Ces données ont été conçues pour être « parfaites » et ne pas poser de problème à l'utilisateur débutant. Je n'ai donc pas pu étudier la partie géoréférencement immédiatement. Une fois ces développements terminés nous avons reçu les données de Tisséo qui ont été produites par des bureaux d'études et qui présentent donc des particularités, voir même de petites erreurs. Nous nous sommes par exemple confrontés à une maquette volant au-dessus du Golfe de Gascogne. L'élévation était correcte, mais il y avait une erreur de latitudes et longitudes.

Outils Communs aux différents scripts

Dans le but de créer une *toolbox*, chaque script a été défini sur un fichier python différent. Cependant, certaines fonctionnalités sont requises dans les différents scripts. Un fichier *Common_services.py* a été créé et contient ces fonctions. Ces fonctions sont génériques, elles permettent par exemple de renvoyer la catégorie Revit à laquelle appartient une couche, ou encore la liste des couches d'une catégorie. Cela permet, si jamais la structure des couches Revit vient à être modifiée dans le futur, de n'avoir à modifier que ce fichier, les autres scripts récupéreront automatiquement les modifications.

La GeoDataBase (GDB)

Dans un premier temps, l'objectif étant de permettre l'augmentation des interactions avec les modèles BIM, il s'agissait de convertir le fichier Revit en un fichier propre aux SIG, et plus particulièrement aux logiciels d'ESRI, la GeoDataBase. La taille de la GeoDataBase varie de la GDB fichier mono-utilisateur jusqu'à une GDB d'entreprise à laquelle de nombreux utilisateurs peuvent avoir accès et la modifier. Nous sommes partis sur une GDB personnelle pour commencer, stockée sous la forme d'un fichier sur mon poste. Mais avec plus de temps, ou si nous avions axés le développement sur la mise à disposition de données BIM pour plusieurs utilisateurs, nous aurions pu stocker ces données sur une GDB d'entreprise.

La GDB peut être créée en utilisant une fonction d'Arcpy, l'utilisateur renseigne l'emplacement, le nom, ainsi que la version. Nous avons pris la décision de créer une GDB par fichier Revit, cela présente deux avantages : la structure interne de la GDB est simplifiée, nous ne rajoutons pas une couche dans l'arborescence pour accéder à une maquette spécifique, et nous pouvons voir directement les maquettes qui ont été converties dans l'explorateur de fichier Windows. En effet même si nous pouvons ouvrir une GDB avec celui-ci, les fichiers qui se trouvent à l'intérieur sont cryptés et nous ne pouvons pas décoder le contenu sans l'ouvrir par les logiciels ESRI.

Le script parcourt tous les fichiers présents dans le dossier entré par l'utilisateur. Pour chaque fichier au format Revit, une GDB fichier est créé portant le nom du fichier. L'algorithme a été adapté pour les données de Tisséo. En effet leurs maquettes BIM sont définies selon le système de coordonnées Lambert 93 en CC43, la GDB est donc par défaut dans ce système de coordonnées.

Ainsi, nos données passent du point zéro du référentiel, à leur emplacement réel. La conversion est rendue possible grâce à la fonction *BIMFileToGDB* du module conversion. Le fichier est converti en jeu de classes d'entités (Feature Datasets) et enregistré dans une GDB (Code 2). Le fichier Revit est construit de manière à ce que toutes les couches soient créées même si la maquette ne possède pas d'éléments d'une de ces couches. Cette dernière est alors vide mais tout de même présente dans le fichier. Ce n'est pas le cas de la GDB, la conversion supprime ces couches vides.

```
PRJ="PROJCS … IsHighPrecision"
folder = arcpy.GetParameterAsText(0)
revit = [f for f in listdir(folder)]
aliasFile = r"C:\Users\ARDUFL\Desktop\02-03\Alias.xlsx"
workbook=xlrd.open_workbook(aliasFile)
columnSheet = workbook.sheet_by_name("Colonnes")
for rvt in revit:
    if rvt.endswith('.rvt'):  #Permet de mettre les WLD3 sans risquer d'erreur lors
de la conversion
        print(rvt)
        name_rvt=rvt[0:-4]    #Le nom uniquement, sans l'extension
        arcpy.management.CreateFileGDB(folder,name_rvt ,"CURRENT")
        GDB = folder + "\\" + name_rvt+'.gdb'#Réxupération de la GDB du fichier RVT
en cours de traitement
        arcpy.conversion.BIMFileToGeodatabase(folder+"\\"+rvt,GDB,name_rvt, PRJ, '')
        #Conversion du fichier RVT
        print('{} a été ajouté à la GDB : {}'.format(name_rvt,GDB))
        arcpy.env.workspace=GDB #Définition de l'espace de travail pour parcourir les
Datasets
        datasets=arcpy.ListDatasets(feature_type='feature') #Normalement on aura un
datasets par GDB (1 Revit = 1 GDB)
        for ds in datasets:
            for fc in arcpy.ListFeatureClasses(feature_dataset=ds): #Toutes les
couches sont devenues des FeatureClasses
                fieldList = arcpy.ListFields(fc)
                for field in fieldList:
                    for i in range(columnSheet.nrows):
                            if columnSheet.cell_value(i,0) == field.name and
                             columnSheet.cell_value(i,1) != "" and
                             columnSheet.cell_value(i,1)!= field.aliasName :

                                arcpy.management.AlterField(fc,field.name,columnShee
                                t.cell_value(i,0), columnSheet.cell_value(i,1))
                                print(fc, '/', field.name)
```

Code 2 : Conversion du fichier Revit en GDB

Afin de rendre la GDB plus simple à utiliser pour les utilisateurs francophones, les alias des champs de chaque table de la GDB ont été traduits en français à partir d'un fichier Excel. Cette partie du script allonge un peu le temps de traitement puisqu'il faut parcourir tous les champs de toutes les tables et à chaque fois tout le fichier d'Alias. Avec le recul, je pense qu'il aurait été possible de limiter le nombre de parcours du fichier d'Alias à un seul passage en créant un dictionnaire avant de parcourir les tables.

La symbologie

Une fois la GDB créée, et les différentes classes d'entités (FeatureClasses) chargées dans ArcGIS Pro, il est apparu que la maquette ne possédait plus de symbologie correcte, toutes les couches apparaissent blanches, sans structuration des données entre elles, les catégories du fichier Revit avaient disparu. Revit a standardisé ses couches, groupement, et symbologies, ainsi même si une maquette ne présente pas un certain type d'objet, la couche se trouve quand même dans le fichier de même que son appartenance à un groupe et sa symbologie. En partant de ce constat, j'ai créé des couches contenant le style de chaque couche dans un dossier qui sert de base de données des symbologies. ESRI, à la différence de QGIS, n'a pas choisi de permettre d'enregistrer le style dans un fichier spécifique. Au lieu de ça, le style peut être enregistré dans des fichiers au format .lyrx, ces fichiers ne sont pas dédiés au style. Ils contiennent les informations de symbologies ainsi que l'emplacement où le logiciel peut aller chercher les informations géographiques. Il est donc possible de charger des données en chargeant des fichiers lyrx sur les logiciels ESRI.

Le choix de partir sur des fichiers au format lyrx pour stocker les symbologies a été amené par l'existence de la fonction *ApplySymbologyFromLayer*, cette fonction permet d'appliquer le style d'un fichier lyrx sur une couche chargée dans ArcGIS Pro. L'algorithme développé parcourt chaque couche chargée dans ArcGIS Pro et lui applique le style de son équivalent au format lyrx. Le temps d'exécution était excessivement long pour une simple application de style, sur les GDB présentant le plus grand nombre de couches, il dépassait les cinq minutes. Dans la documentation de la bibliothèque Arcpy, il apparait que les couches ont un attribut *symbology*,(Code 3), j'ai donc défini l'attribut des couches à modifier en fonction de leur équivalent lyrx, préalablement chargé sur le projet et ajouté en dernière couche du projet pour le retrouver plus facilement. Cette modification a permis de réduire le temps d'exécution à moins d'une minute. Plusieurs fois au cours de mes développements, le temps de traitements des fonctions Arcpy se sont révélés supérieurs à une solution de contournement. Essayer de comprendre le fonctionnement des fonctions Arcpy a représenté une part importante de mon travail.

```
mapNumber=0
project = 'CURRENT'
#project =
r'C:\Users\ARDUFL\Documents\ArcGIS\Projects\MyProject6\Project6.aprx'
aprx = arcpy.mp.ArcGISProject(project)
```

```
map = aprx.listMaps()[mapNumber]
for lyr in map.listLayers():
    cat = common_services.get_categorie(lyr.name)#Utilisation de la catégorie
de la couche pour aller dans la bonne arborescence
    if cat != False:
        lyrFile=arcpy.mp.LayerFile(r"C:\Users\ARDUFL\Documents\ArcGIS\Projec
        ts\layerx\\"+cat+"\\"+lyr.name+".lyrx")
        map.addLayer(lyrFile,"BOTTOM")
        symlist= map.listLayers(lyr.name)
        symlist[0].symbology = symlist[-1].symbology
        map.removeLayer(symlist[-1])
    else :
        print('{} ne correspond pas à une couche RVT'.format(lyr.name))
aprx.save()
```

Code 3 : Gestion de la symbologie des couches

Le groupement des couches

Le groupement des couches permet d'avoir une meilleure visualisation des données, il est plus facile de voir les couches métier par métier. Les informations de structuration des couches ayant disparues lors du passage à la GDB, il était important de les recréer sur le projet ArcGIS Pro, mais aussi d'avoir un moyen de charger rapidement ces groupes sur un autre projet. Le développement de ce script s'est développé en se basant sur la fonction *addLayerToGroup*. Le principe du script est donc de parcourir toutes les couches et de les ajouter au groupe auquel elles appartiennent.

La première particularité rencontrée a été l'impossibilité de créer un groupe de couches via le module Arcpy. Il s'agissait donc créer une couche qui allait être le groupe dans lequel nous ajouterons les différentes couches du projet. Mais Arcpy ne permet pas encore de créer de couche vide non plus. Etant toujours en développement, la fonctionnalité pouvait être ajoutée mais dans la version que j'ai utilisée elle n'était pas implémentée. Un fichier au format lyrx a donc été créé à la main et placé dans le même répertoire que la base de données des symbologies. De nombreuses bibliothèques python permettent d'interagir avec des fichiers, le fichier lyrx vide est copié dans le dossier où se trouve la GDB pour chaque catégorie ainsi que le fichier Revit. Le fichier ainsi dupliqué est ajouté au projet, et renommé selon la catégorie puis la fonction *addLayerToGroup* peut enfin être appliquée.

Ensuite il est apparu que la fonction *addLayerToGroup* ne déplaçait pas les couches mais les copiait dans le groupe. Il était donc nécessaire de supprimer la couche une fois celle-ci dupliquée. Il s'agissait alors de trouver un moyen de différencier quelle couche était à supprimer puisque les

deux étaient sensiblement les mêmes, l'appartenance à un groupe n'est pas une information qui peut être retrouvée grâce à Arcpy. Lorsque nous listons les couches d'une scène à l'aide de *listLayers*, toutes les couches sont au même niveau, nous aurions pu nous attendre à avoir une liste de liste pour ainsi différencier les groupes mais ce n'est pas le cas. Lors du parcours des couches, une fois la couche ajoutée au groupe, la couche est renommée de manière à pouvoir retrouver quelles couches doivent être supprimées.

Nous avons alors nos cinq groupes métiers tels que présents dans les modèles Revit. Ils peuvent être vides si aucune couche n'appartenant à ce groupe n'est présente dans la GDB. Nous réitérons l'opération d'ajout des couches à un groupe mais en prenant cette fois les sous-groupes et en les ajoutant au groupe portant le nom du fichier Revit. Ensuite ce groupe est enregistré sous la forme d'un fichier lyrx, qui contient les symbologies, la structuration des couches ainsi que les emplacements des données de chaque couche. Cela permet de charger rapidement la maquette sur un autre projet sans avoir à réutiliser les différents scripts. L'avantage du lyrx est qu'il est assez souple en cas de déplacement des données dans l'arborescence Windows ou même sur un autre poste, si les données sont présentes sur l'autre poste, les liens se recréeront.

Récapitulatif des traitements

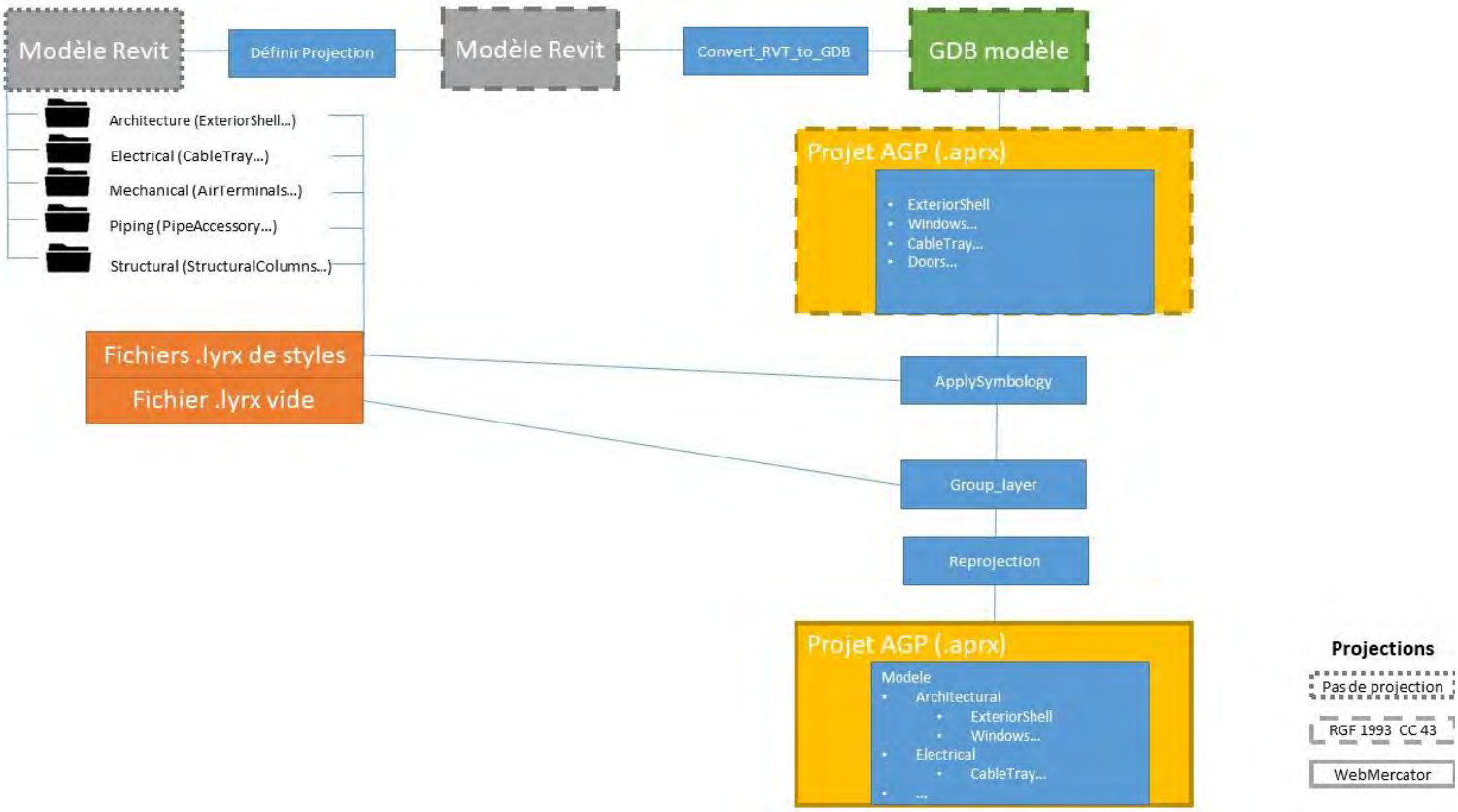

Figure 4 : Synthèse des traitements appliqués au fichier Revit (Source personnelle)

La Figure 4 ci-dessus retrace les différents traitements qui permettent de passer d'un modèle Revit non géoréférencé à un modèle BIM structuré, avec les bonnes symbologies et stockés dans une GDB.

Afin de faciliter l'utilisation de ces données sous la forme de cartographie en ligne, les couches sont reprojetées en WebMercator qui est le système de coordonnées privilégié par les solutions de WebMapping d'ESRI.

D- Les difficultés rencontrées par un développeur junior sur une librairie python propriétaire

Cette partie du stage consacrée au développement python d'outils permettant la conversion de données Revit a représenté environ quatre semaines sur la durée de mon stage. Ayant eu l'occasion au cours de la spécialité Agrogéomatique de prendre en main la bibliothèque Arcpy, et le développement python de manière plus global, j'ai rapidement eu une idée de la manière dont les différents problèmes posés pouvaient être résolus. La partie recherche bibliographique consistait à trouver les fonctions Arcpy qui pouvait répondre à mes besoins de la manière la plus précise possible. La documentation Arcpy disponible sur le site d'ESRI est bien construite et nous pouvons facilement retrouver les fonctions dans l'arborescence des classes de la librairie.

Cependant comme pour les suites Desktop, ESRI établit une migration de certains de ces outils d'une classe à une autre, ainsi, dans la version que j'ai utilisée, certains outils étaient présents deux fois dans la librairie. En s'intéressant aux identifiants de ces fonctions, il ressort que certaines d'entre elles ont des duplicatas avec le même identifiant, ce qui sous-entend que la fonctionnalité ne sera pas modifiée au cours de la migration, mais uniquement la manière de les appeler dans l'arborescence. Il s'agit alors de vérifier lequel des moyens d'appeler la fonction tend à être obsolète. D'autres ont, en revanche, des identifiants différents mais en apparence les mêmes fonctionnalités, cette particularité m'a poussé à éviter l'utilisation de ces fonctions dans la mesure du possible, afin que les différents scripts soient encore performants dans les prochaines versions d'Arcpy.

De même, au fil de mes développements, j'ai réalisé que certaines fonctions avaient des fonctionnalités cachées ou qui ne correspondaient pas à leurs intitulés. C'est le cas par exemple de *ApplySymbologyFromLayer,* qui dans certains cas applique non seulement le style mais aussi la projection de la couche source de style à la couche cible. Ne l'ayant pas réalisé tout de suite, cela a posé problème lors de la mise en ligne de ces couches sur le portail, la solution de web mapping d'ESRI ne pouvait pas charger de fond de carte en plus de la couche 3D car il y avait une incompatibilité de système de coordonnées. C'est ce qui explique que systématiquement les différentes couches sont reprojetées en WebMercator avant leur mise en ligne sur le portail. Une autre fonctionnalité, non explicitée sur sa fiche du site d'ESRI, est le fait que la fonction Arcpy projetant les objets dans un système de coordonnées, convertit également les données Revit en

GDB. Cette fois la particularité s'est révélée plutôt avantageuse puisqu'elle permet de réduire le temps de traitement en convertissant immédiatement les données Revit lors de leur projection. Cependant, si cette fonctionnalité disparait dans les prochaines versions cela risque de ne pas être renseigné.

D'une manière générale, travailler à l'aide de cette bibliothèque s'est révélé être un challenge tant dans sa prise en main que dans les manières de parvenir aux objectifs en la contournant. Il est parfois surprenant de voir qu'une opération qui ne semble pas demander de ressources ou de temps de traitements excessifs nous fasse attendre plusieurs dizaines de minutes. La première version de la Toolbox n'utilisant que les fonctions présentes dans Arcpy convertissaient les fichiers Revit en environ trente ou quarante minutes. Avec les contournements la Toolbox convertit les fichiers en quinze minutes. Dans un futur stage, il pourrait être intéressant d'améliorer la souplesse du script python en permettant à l'utilisateur de modifier plus de paramètres, de créer une véritable interface graphique pour la toolbox, et un véritable travail d'optimisation du code est nécessaire.

III- Une facilité d'utilisation apportée par le Web AppBuilder et la cartographie en ligne

A – Présentation des objectifs

Ces dernières années, avec le développement et la montée en puissance des navigateurs internet, il existe une forte tendance à la mise en place de solutions de cartographie en ligne. Ces services peuvent reposer sur des logiques libres comme OpenStreetMap ou propriétaires comme GoogleMaps et BingMaps. La cartographie en ligne ou Webmapping tend à se généraliser, de nombreux sites y ont recours pour diverses applications. Nous pouvons prendre pour exemple Amazon ou autres sociétés livrant à domicile comme DPD ou Uber Eats qui permettent au client de visualiser où se trouve le livreur par rapport à la destination, ou encore les plans d'accès de structures accueillant du public.

Ces solutions reposent sur trois composantes : un client, un serveur ainsi que les données. ESRI a développé les trois composants de sa solution. ArcGIS Entreprise est un portail que peut mettre en place une entreprise et sur lequel sont stockées toutes ses données géographiques. ArcGIS Pro permet de mettre en ligne sur le portail les couches qu'il intègre sous différents formats : en tant que couches d'entités, groupe de couches, couche Web (WebLayer) ou encore de partager la scène entière en tant que WebScene. ArcGIS Server permet de gérer ces couches et assure les mêmes fonctions que les autres serveurs comme GeoServer ou MapServer. Enfin ESRI a mis en place le Web AppBuilder pour assurer la partie Client de sa solution, ainsi que plus récemment l'Experience Builder.

A la suite des traitements effectués sur les données Revit, ces données ont pu être mises en ligne sur le Portal de Magellium. Un des atouts primordiaux des accords entre ESRI et Autodesk est la possibilité d'intégrer des données issues du monde du BIM dans les solutions de cartographie en ligne d'ESRI. Cependant, le catalogue des widgets disponibles sur la cartographie en ligne en trois dimensions n'est pas aussi développé qu'en deux dimensions, d'une palette d'environ cinquante widgets natifs en version 2.16 du Web AppBuilder, il n'y a que dix widgets natifs disponibles pour la 3D. Afin d'améliorer l'expérience de l'utilisateur souhaitant utiliser les BIM en tant que service en ligne, il est nécessaire de développer de nouveaux widgets facilitant la prise en main de ce service. Une des premières idées émises a été l'amélioration du widget de géosignet, afin de

permettre à l'utilisateur d'en créer lui-même. Le partenariat développé entre Magellium et Tisséo a permis, en plus d'avoir accès à des fichiers Revit réels, de générer des discussions sur les besoins d'un utilisateur et cela nous a fourni d'autres exemples de widgets qu'il pouvait être intéressant de développer.

B – Les outils ESRI pour la cartographie en ligne

La cartographie en ligne, ou *web mapping* est devenue une part non négligeable de la géomatique, autant pour la communication que pour la production de données. Afin d'être présent sur ces parts du marché, ESRI a rapidement mis en place des solutions permettant de créer des applications de Web mapping. Comme nous l'avons vu, ces solutions reposent sur une partie serveur, gérée par ArcGIS Server et le stockage des données qui se fait sur un portail interne de l'entreprise qui l'a mis en place. La création des applications Web est permise grâce au Web AppBuilder. Le Web AppBuilder est conçu pour permettre à des utilisateurs n'ayant pas forcément de compétences poussées en développement en ligne de créer des applications efficaces. L'administrateur peut choisir si l'application est dédiée à la visualisation de données en 2D ou en 3D, la scène stockée sur le portail sur laquelle l'application se basera, le thème, les couleurs puis quels outils (widgets) les utilisateurs auront à leur disposition.

Ces widgets permettent de réaliser des opérations de géotraitements avec par exemple la possibilité de mettre en place des zones tampons, de faire des intersections, mais aussi de faire des requêtes sur les tables attributaires. Outre la manipulation des données, il est aussi possible de modifier les aspects plus esthétiques en changeant la luminosité de la carte, ou encore de partager la carte via les réseaux. Une fois ces widgets choisis, l'administrateur peut lancer l'application en plaçant le dossier dans lequel s'est créé l'application sur un serveur. Le Web AppBuilder étant pensé comme une application intuitive, nous pouvons penser que les utilisateurs souhaitant y apporter des modifications n'y seront pas autorisés. Pourtant, ESRI a mis en place une version pour les développeurs, une grande partie du code est donc accessible, et des tutoriels sont disponibles sur le site d'ESRI pour prendre en main le développement sur cette solution, ces tutoriels permettent de voir la création d'un widget, et celle d'un thème.

Le Web AppBuilder repose sur les langages HTML, CSS pour la structure de la page et des widgets, et surtout sur le langage Javascript, et plus particulièrement deux librairies : l'API ESRI Javascript, et DOJO.

API ESRI Javascript

Cette API permet la visualisation des données cartographiques à la manière de la segmentation des objets sur les suites Desktop, on peut créer les objets des différents segments allant de la scène à un champ. Cette bibliothèque est propriétaire, et le code source de chaque objet et fonction de l'API n'est pas disponible mais la documentation en ligne permet de prendre en main les fonctionnalités de l'API plus facilement que celle de Arcpy. Une difficulté majeure dans son utilisation réside dans le fait que plusieurs versions sont utilisées simultanément dans le Web AppBuilder. En effet si les widgets destinés à des applications 3D utilisent la dernière version du Web AppBuilder en 4.X, les widgets d'applications 2D sont conçus sur la version 3.X. Toutes les fonctionnalités 3D de l'API sont inexistantes en version 3 cependant les pages de réglages des widgets 3D sont sous la version 3 alors même que le widget une fois lancé tournera sur la version 4. Je ne connaissais pas cette particularité lors du premier développement d'un widget nécessitant une page de réglages et cela a représenté une perte de temps, il a fallu comprendre d'où provenaient les erreurs.

DOJO

Le fonctionnement en widget est dû à l'utilisation de la librairie DOJO. Cette API développée depuis 2004 est en partie portée par IBM. Elle permet le fonctionnement en widget du Web AppBuilder, la classe permettant la création d'un widget est en effet importée de DOJO. DOJO est toujours en développement actif. L'importation d'objets issus de DOJO, a facilité ma prise en main du développement Web, puisqu'il repose sur des techniques assez communes avec le développement sous python, ce qui représente une approche assez différente du fonctionnement par prototypage de Javascript. Le fait de travailler avec des techniques familières était en quelque sorte rassurant et m'a permis d'être plus à l'aise avec le développement de widgets.

La prise en main du Web AppBuilder

A la suite du développement python, j'ai suivi les différents tutoriels afin de comprendre le fonctionnement du Web AppBuilder. Cette prise en main s'est révélée complexe, n'ayant pas beaucoup de connaissances sur le développement en ligne. Pendant le confinement, lorsque les stages de Magellium étaient gelés j'ai suivi des cours axés sur le développement en ligne et plus particulièrement en Javascript, pourtant il y a un écart entre les connaissances que j'ai acquises sur le langage Javascript et les compétences requises pour pouvoir développer une application efficace. Cette partie du stage a été assez difficile à la fois sur le plan technique mais aussi sur celui de la motivation. Les tutoriels m'ont permis de comprendre le fonctionnement et le cycle de vie d'un widget.

D'un autre côté, ESRI met en place depuis cette année une nouvelle solution pour la cartographie en ligne, l'Experience Builder en version Bêta. Cette solution repose sur des technologies plus récentes que DOJO telle que Angular et est encore en développement, le stage n'avait donc pas pour but de l'aborder. Des changements importants dans le fonctionnement peuvent survenir et les développements qui auraient pu aboutir n'auraient aucune assurance d'être des solutions pérennes. De plus toutes les fonctionnalités ne sont pas encore en place sur l'Experience Builder. ESRI assure que l'Experience Builder n'a pas pour vocation de remplacer le Web AppBuilder dans l'immédiat au moins. Mais, il est important de se demander vers quelles solutions se tourner en fonction de la durée de son projet. J'ai tout de même eu recours à l'Experience Builder afin de mieux prendre en main l'API Javascript ESRI qui peut sembler plus accessible puisqu'on peut l'utiliser sur des applications Web comme Codepen.io ou encore la Sandbox d'ESRI, nos tests et essais sont alors directement visible sans même recharger la page.

C – Le développement de widgets pour améliorer l'expérience utilisateur

Afin d'assurer le développement de différents widgets, j'ai mis en place quelques dispositions pour faciliter les tests et le bon déroulement de la mission. Ces widgets sont testés sur trois navigateurs : Mozilla Firefox, Chrome, et Edge. Le développement des scripts s'est fait sur l'environnement de développement « *Atom* » permettant d'ajouter des plug-ins utiles tels qu'une mise en page automatique selon le langage, la modification automatisée du nom des variables, ou la

visualisation des couleurs selon leurs encodages. J'ai versionné chaque avancée que je faisais sur un widget afin de ne pas perdre cet avancement en cas d'erreur. Les widgets ne nécessitant pas de couches d'objet pour fonctionner ont été développés sur une application vide dédiée au développement par ESRI, les autres sur une application créée à cet effet contenant un des bâtiments de Tisséo.

1- SlidesMag

Le Widget

La première idée de widget pouvant apporter une facilité d'utilisation était la création d'un outil permettant d'ajouter des Géosignets, lesquels seraient utilisés pour naviguer d'une vue à une autre en cliquant dessus. Cela permet de faciliter la navigation d'une zone technique à une autre. Ce widget est basé sur celui déjà présent nativement dans le Web AppBuilder. Il ajoute un bouton dans la barre des géosignets permettant d'ajouter un géosignet à cette liste. On peut également donner un titre à ce géosignet, via un popup. Si jamais, l'un d'eux n'a plus d'utilité, il est alors possible de le supprimer de la liste. Enfin ces différents géosignets peuvent se retrouver d'une session à l'autre. La Figure 5 présente l'interface utilisateur de ce widget.

Figure 5 : Vue du widget SlidesMag

Son développement

Il s'agit du premier widget que j'ai développé. Dans un premier temps, je suis parti du widget Demo, servant de base au développement de widget dans les tutos, en effet certains composants sont obligatoires pour le fonctionnement d'un widget et celui-ci les contient tous. J'ai pris en main des objets variés dans cette première version afin de comprendre comment fonctionne l'objet de type View et comment le manipuler. Dans cette première version, le widget présentait deux boutons, le premier pour ajouter une vue dans un tableau Javascript, et le second qui permettait de passer d'une vue à l'autre successivement. Ce n'était évidemment pas un widget recevable mais il m'a permis de vraiment comprendre le cycle de vie d'un widget, qui est illustré sur la Figure 6. En effet, j'ai pu par exemple vérifier que les éléments HTML du widget n'étaient pas encore créés avant la fin du fonctionnement de *postCreate*, j'ai donc placé toutes mes modifications de code HTML par des codes Javascript dans la fonction *startup*. Cette fonction est appelée une fois les éléments HTML du widget créés.

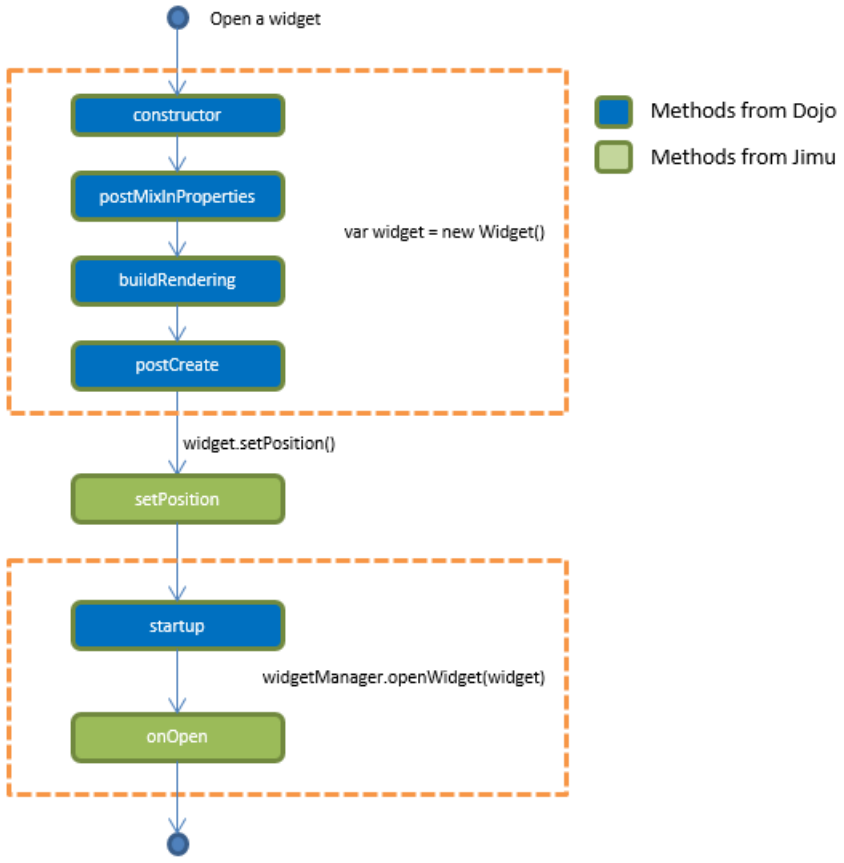

Figure 6 : Cycle de vie d'un Widget (Source ESRI : https://developers.arcgis.com/web-AppBuilder/guide/widget-life-cycle.htm)

Une fois la compréhension de la navigation d'une vue à l'autre terminée, nous avons décidé d'adapter le widget de Géosignet déjà existant sur le Web AppBuilder, qui ne permet que de visualiser et utiliser les géosignets définis au moment de la création de la scène sur le portail, en y ajoutant les fonctionnalités d'ajout et de suppression de nouveaux géosignets.

Une phase supplémentaire a donc été ajoutée avant de passer à l'implémentation de nouvelles fonctionnalités, la compréhension du code d'ESRI. Le fonctionnement est le suivant :

- Récupération des slides existantes sur le portail contenues dans un tableau JS
- Création de la div qui contiendra l'ensemble des géosignets
- Création des div qui contiennent chacun des géosignets
- Création de l'action en cas de clic sur un des géosignets

Les géosignets sont des objets de classe Slide dans ce widget. J'ai donc recréé un objet de ce type lors du clic de l'utilisateur pour ajouter un géosignet. Un objet slide est composé d'un objet contenant les informations sur la caméra (position x, y, z, orientation), une liste des couches visibles, ainsi qu'une miniature (thumbnail). Pour les couches visibles, il s'agissait de récupérer les couches qui étaient affichées lors du clic mais aussi de ne pas oublier la couche d'élévation du sol si elle était présente. La création de la miniature m'a permis d'aborder le domaine des fonctions asynchrones, il existe la fonction *takeScreenshot* dans l'API ESRI mais cette fonction renvoie une promesse, objet représentant l'état d'une fonction asynchrone. Suite à des discussions avec mon maitre de stage, j'ai pu appréhender ces concepts et réussir à intégrer la fonction dans mon code. La balise « img » qui contient la miniature est créée en même temps que les autres éléments mais la valeur de sa source est définie à l'intérieur de la promesse. Virtuellement cela fait que pendant un moment, l'image n'apparait pas mais pour l'utilisateur le script est suffisamment rapide pour que l'image apparaisse tout de suite. Cet objet est également ajouté au tableau qui contient toutes les slides.

Afin que le placement du bouton d'ajout soit cohérent, j'ai placé le bouton sur la barre des géosignets. Les géosignets, une fois ajoutés, sont placés avant le bouton d'ajout. En utilisant les différentes fonctions javascript disponibles pour manipuler les documents HTML, nous pouvons placer de nouveaux éléments dans les Nœuds, Enfants ou Parents, ici les slides sont ajoutées avant le Nœud contenant le bouton d'ajout. Il est également possible d'ajouter un titre. Ce titre apparait en transparence au-dessus de la miniature au passage de la souris.

Afin de pouvoir supprimer les géosignets, un bouton a été placé au-dessus de chaque image. Il permettait dans les premières versions de développement de supprimer les éléments HTML afin que le géosignet ne soit plus accessible par l'utilisateur. Cependant il était toujours présent dans la liste des slides de la scène.

Ayant modifié le tableau qui contenait les différentes slides, je m'attendais à ce que les ajouts de slides se retrouvent d'une session à l'autre. Or, lorsque l'on recharge la page on ne retrouvait que les géosignets qui avaient étaient définis dans le portail et non ceux créés depuis l'application. Les interactions avec la scène, dont l'appel de le tableau contenant les géosignets, se faisaient via l'objet *this.sceneview*. Après des recherches il est apparu que nous pouvions également appeler les objets au travers de *_sceneview*. Les différents objets de la scène étaient dupliqués et placé dans le *this* qui correspond à l'objet du widget, mais les objets de la scène sont réellement accessibles via le deuxième objet, *_sceneview*. J'ai donc changé l'appel à la liste des géosignets par les objets authentiques, non leur duplicata. Ensuite à l'aide d'une fonction de l'API, la scène a été sauvegardé avec ces différents attributs modifiés. Cette fois ci les géosignets ajoutés se retrouvaient d'une session à l'autre. Mais la suppression n'affectait dans cette version que la partie HTML et non la liste des géosignets de la scène, nous retrouvions les géosignets supprimés dans les autres sessions. Il s'agissait donc de retrouver quel élément de la liste des géosignets devait être supprimer et cela ne pouvait se faire via leur indice dans la liste, si une suppression avait déjà eu lieu dans les premières positions de la liste, le décalage se propagerait sur toute la liste. Je suis donc passé par un attribut qui me permettait de différencier chacune des slides afin de savoir laquelle avait été retirée.

Afin de résumer le travail sur ce widget, je pense que d'une manière générale, le script peut être amélioré tant dans sa légèreté que dans son efficacité. Je suis par exemple exclusivement resté dans la fonction *startup* au lieu de définir des fonctions en dehors de cette fonction et ne faire seulement des appels. De cette manière je n'aurais pas eu à dupliquer le code pour permettre à l'utilisateur de valider la saisie du nom du géosignet en appuyant sur la touche « Entrer » en plus du clic sur le bouton. Il pourrait également être intéressant de se pencher sur le cas où plusieurs utilisateurs de l'application ajouteraient ou supprimeraient des géosignets et la manière dont résoudre les conflits que cela engendrera.

2- QueryMag

Le widget

A la suite des discussions avec Tisséo, le développement d'un widget permettant de générer des requêtes attributaires sur les couches présentes sur l'application s'est révélé pertinent. Cela peut avoir pour application de faciliter les inventaires ou de rapidement savoir où sont positionnées les portes coupe-feu par exemple. Les requêtes sont définies lors de la création de l'application par l'administrateur. Elles sont paramétrées dans la page des réglages du widget, au moment où l'on ajoute le widget à l'application. L'administrateur peut choisir la couche sur laquelle il souhaite que la requête porte. Ensuite le popup de la requête apparait. Un formulaire permet de définir le champ auquel nous nous intéressons, l'opérateur que nous souhaitons lui appliquer ainsi que le paramètre de comparaison. Plusieurs requêtes peuvent être ajoutées et sont listées dans un tableau de la page de réglages. Les requêtes sont définies lors de la création de l'application par l'administrateur. Une fois cette application lancée, l'utilisateur peut exécuter les requêtes et accéder aux résultats mais ne pourra pas en ajouter ou les modifier. Le paramétrage d'une requête est illustré dans la Figure 7.

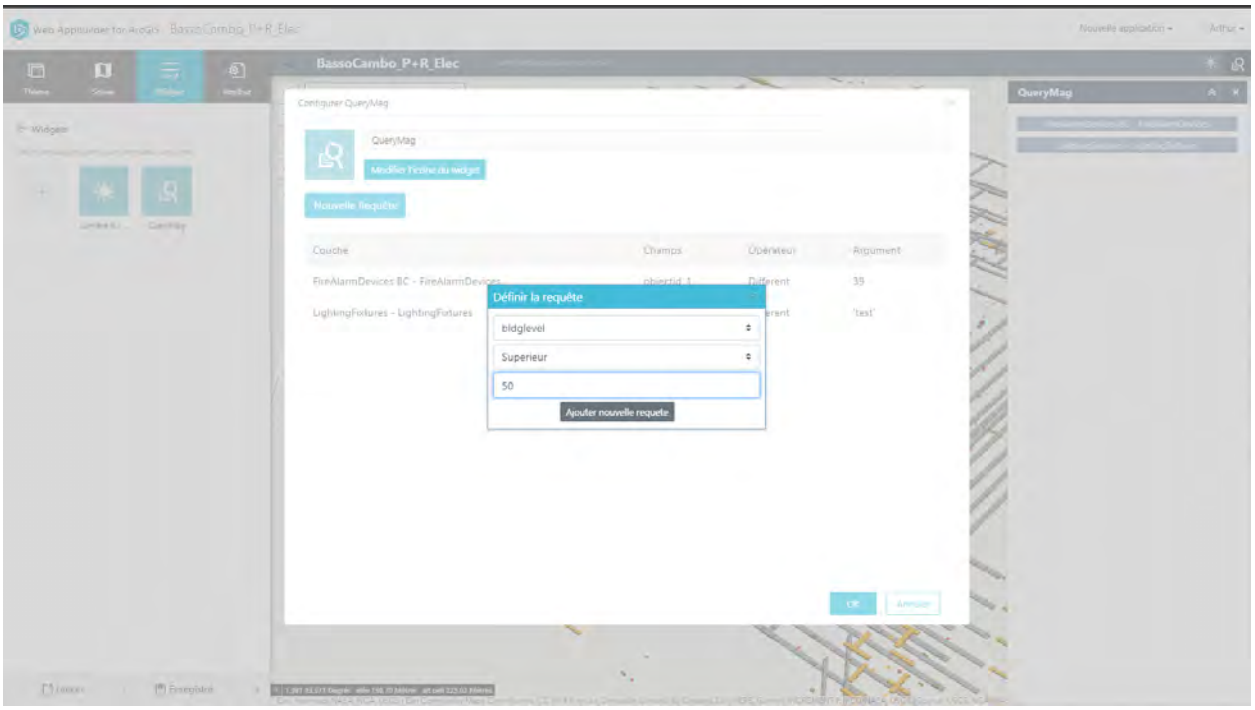

Figure 7 : Vue de la page de réglages du Widget QueryMag

L'utilisateur peut cliquer sur les boutons de chaque requête pour les lancer. Le contenu du widget change alors pour afficher les résultats répondant à la requête sous la forme d'un tableau, comme le montre la Figure 8. Afin d'améliorer la visibilité des résultats, chaque objet correspondant est surligné en bleu, et l'utilisateur peut zoomer sur chacun d'entre eux en cliquant sur les lignes du tableau.

Figure 8 : Résultats d'une requête du widget QueryMag

Son développement

Un tel widget existe en 2D, nous sommes donc initialement partis sur la conversion du widget 2D d'ESRI et l'appliquer à la 3D. Il s'est vite avéré que cette technique allait être inefficace. L'inexistence des objets 3D dans la version 3 de l'API nous obligeait à reprendre tout le code, en plus de faire les changements dans les manières d'appeler les différentes fonctions de l'API. La différence de version entre les réglages du widget et le widget a également représenté un problème. Dans le widget d'ESRI certains fichiers sont appelés à la fois dans les réglages et dans le widget, cela n'est plus possible en 3D. Nous avons donc choisi de construire les bases d'un widget de requêtage.

Dans un premier temps j'ai créé la page de réglages permettant de générer la requête. Elle repose sur différents popups successifs, le premier permet de choisir la couche dans la liste des couches présentes. Il était nécessaire de trouver un moyen de ne pouvoir choisir que les couches requêtables, c'est-à-dire sans les fonds de cartes et la couche d'élévation du sol. Cela a été permis grâce à un attribut qui n'est présent que si la couche possède des champs, ce qui n'est pas le cas sur les fonds de carte. Le deuxième popup dépend de la couche sélectionnée, il permet de choisir le champ sur lequel nous souhaitons établir notre requête, l'opérateur que nous souhaitons lui appliquer ainsi que le paramètre à entrer dans une balise input. Une fois que l'utilisateur a validé sa requête celle-ci est ajoutée à une liste sous forme d'objet, elle-même enregistrée dans un attribut du widget.

Sur le développement de ce widget, contrairement au widget SlidesMag, j'ai privilégié la définition de nouveaux éléments HTML en utilisant leur classe pour les distinguer plutôt que leur identifiant. Cela a apporté une manière différente d'aborder la récupération des éléments HTML. Outre le fait que l'on utilise la fonction *getElementsByClassName*, au lieu de *getElementById*, cela a supprimé les problèmes d'éléments ayant le même identifiant que j'avais résolu par l'utilisation d'un compteur.

En ce qui concerne le widget en lui-même, la liste d'objet qui est enregistrée dans les attributs est accessible dans le widget. Un bouton est créé pour chaque item de la liste de requête. Je me suis inspiré du widget d'ESRI pour définir ma requête, la requête est assurée par l'objet *Query* de l'API ESRI. Ces attributs permettent de gérer les principaux paramètres d'une requête SQL. La clause *where* est alors entièrement définie par les paramètres entrés par l'utilisateur lors des réglages. Une fois les paramètres de la Query définis, elle peut être lancée grâce à la fonction queryFeatures appliquée à l'objet de la couche à requêter. Cette fonction renvoie une promesse au sein de laquelle les résultats sont accessibles. Le contenu du widget est alors modifié pour y intégrer un tableau contenant les cinq premiers champs de ces objets. A cela j'ai ajouté une modification graphique de ces objets en les surlignant de la même manière que lorsqu'on sélectionne un objet sur ArcMap et ArcGIS Pro, et lorsqu'on clique sur une ligne du tableau, l'application zoome sur l'objet en question.

Une fois encore, le problème de passer les informations d'une session à l'autre s'est posé, mais cette fois il y avait une perte même au moment du lancement de l'application. Dans les tutoriels,

l'enregistrement des configurations se fait au niveau d'un fichier, config.json. Ce fichier contient un objet au format JSON qui contiendra les données à transférer. Dans le cas de ce widget, il contient un objet avec un seul attribut : la liste des requêtes. A l'origine cette liste est vide. La définition et la récupération des configurations se font par le biais de deux fonctions définies par ESRI. J'ai rencontré quelques problèmes lors de leur utilisation, notamment dans la manière et le moment de les appeler.

3- FilterBuilding

Le widget

Enfin, le dernier widget que j'ai développé a été inspiré par quelques présentations sur le BIM auxquelles j'ai pu assister (Webinaire ESRI, été 2020). La visualisation d'un bâtiment étage par étage, permet une navigation simplifiée au sein du bâtiment. L'interface est constituée d'une barre permettant de naviguer parmi les niveaux, elle va du niveau à la plus petite élévation jusqu'au niveau le plus haut. De plus, le nom du niveau auquel est défini le filtre est affiché s'il existe. La Figure 9 représente l'interface de ce widget.

Figure 9 : Filtre par étage et interface utilisateur du widget FilterMag

Son développement

Grâce au travail effectué sur le widget des requêtes, les développements sur le filtre ont été rendu plus faciles. Toutes les couches issues de fichiers Revit possèdent des champs renseignant le niveau et l'élévation de l'étage auquel appartiennent les objets de cette couche. A l'aide d'une requête, on peut récupérer les différentes valeurs de ces champs. Les valeurs du champ *bldglevel* sont de type numérique, sur les maquettes de Tisséo, elles s'échelonnent entre 0 et 10 par exemple. Ensuite, les valeurs minimales et maximales sont entrées en tant que paramètres d'une balise input de type range. Les couches possèdent un attribut *definitionExpression* qui définit quels objets apparaissent ou non. La valeur de cet attribut est définie par l'input du widget. Il y a également un bouton permettant de le réinitialiser.

Avec le recul, quelques points me paraissent à améliorer. Certaines couches n'apparaissent pas sur tous les niveaux, je n'ai pas eu le temps de résoudre cette problématique, mais il faudrait lister les niveaux de chaque couche et les trier dans le bon ordre afin d'avoir un filtre efficace sur tous les étages. Enfin l'affichage du nom de la couche peut être défaillant. Ce nom de la couche est régi par le champ *bldg_desc* mais il n'est pas forcément renseigné sur toutes les couches, comme sur la capture d'écran ci-dessus où nous pouvons voir *'undefined'*, il s'agirait d'ajouter un filtre afin de vérifier que le champ est bien renseigné.

Style CSS

J'ai privilégié le développement de nouvelles fonctionnalités avant de gérer le style CSS de ces widgets. Une fois ces trois widgets mis en place et fonctionnels, différents styles ont été appliqués aux éléments HTML que j'ai créés. Les tableaux du widget QueryMag ont reçu des paramètres CSS afin qu'ils soient plus lisibles, et que sur celui des résultats la ligne pointée change de couleur, donnant la sensation qu'elle est cliquable. Ces styles seront facilement modifiables par la personne qui reprendra le travail. Chaque classe HTML a son style se trouvant dans les fichiers *style.css*.

Conclusion et bilan de mon parcours

Après ces quatre mois au sein de l'entreprise Magellium, je retire de nombreux enseignements de cette expérience. La majorité des objectifs définis dans le cahier des charges de début de stage a trouvé une réponse. La conversion des données Revit en format de fichiers propre à la géomatique est permise à l'aide d'une Toolbox disponible dans ArcGIS Pro. Toutefois certains travaux sont encore nécessaires. Du point de vue du géoréférencement, l'automatisation n'est possible qu'à la condition que le modèle Revit soit positionné à son emplacement dans un système de coordonnées fixé depuis le logiciel Revit. En effet j'ai rencontré quelques problèmes sur certains fichiers Revit, une des maquettes BIM n'avaient pas les bonnes coordonnées, suite à une possible erreur du bureau d'étude. La seule solution aurait été de replacer le modèle à la main, ce qui n'est pas envisageable avec la précision requise par les utilisateurs de BIM. Une automatisation de la mise en ligne des données sur le portail pourrait également être ajoutée.

Ce travail sur la suite Desktop d'ESRI m'a permis de me poser des questions sur l'optimisation de mon code, la manière de présenter le code à un interlocuteur, mais aussi sur le fonctionnement d'une bibliothèque python propriétaire, ayant jusqu'à présent travaillé principalement avec des bibliothèques libres.

Le développement Web a représenté la majorité de mon temps de travail. Le cahier des charges de cette partie s'est construit au fil du stage et des discussions avec Tisséo et mon maitre de stage. Les widgets produits sont une première réponse à la future demande des consommateurs de BIM sur de la cartographie en ligne. Ils ont été définis d'après les besoins d'utilisateurs actuels et ont permis de montrer que la géomatique pouvait apporter des solutions à leurs problèmes. La prise en main des solutions ESRI pour la cartographie en ligne m'a demandé un certain temps. Il est difficile de comprendre le fonctionnement des différents outils nécessaires au développement de widget, et il ne faut pas hésiter à tester ce que l'on pense avoir compris. Je suis monté en compétence sur les solutions ESRI mais également en développement Web, les notions que j'ai mobilisées me seront utiles dans d'autres thématiques.

L'intégration au sein de l'entreprise Magellium est très bonne, les autres collaborateurs étant disponibles et à l'écoute, en plus de mon maitre de stage, si jamais je rencontrais un point

bloquant. Ce stage a renforcé ma motivation à l'idée de travailler dans le domaine du développement autant python que Web, et de l'appliquer au domaine de la géomatique.

Webographie

- BuildingSMART, Site officiel de l'organisation BuildingSmart (visité le 11 août 2020) : https://www.buildingsmart.org/
- D. Gunther-Dilinger, 2016, From BIM to GIS at the Smithsonian Institution
- DOJO, Documentation de la librairie DOJO : https://dojotoolkit.org/documentation/tutorials/1.9/templated/
- ESRI, 5 mai 2020, Wébinaire ESRI autour du BIM (16 Juillet 2020) : https://www.esrifrance.fr/video-ws-2020-bim-sig-transformation-numerique.aspx
- ESRI, Configuration Requise pour l'utilisation d'ArcGIS Pro (19 août 2020) https://pro.arcgis.com/fr/pro-app/get-started/arcgis-pro-system-requirements.htm
- ESRI, Références à l'API Javascript ESRI : https://developers.arcgis.com/javascript/latest/api-reference/
- GIS Stackexchange, Site d'échange autour du développement SIG : https://gis.stackexchange.com/questions/tagged
- Journal Officiel de la République Française n°0024 du 29 janvier 2019 texte n° 83 (11 août 2020): https://www.legifrance.gouv.fr/affichTexte.do;jsessionid=BC29AAFB90E7EAB36F24CA4115A369F1.tplgfr38s_3?cidTexte=JORFTEXT000038063061&dateTexte=&oldAction=rechJO&categorieLien=id&idJO=JORFCONT000038062550
- M. J. Sani et A. Rahman, 2018, GIS and BIM integration at data level a review
- Magellium, Architecture SIG Mutualisée (2 septembre 2020) https://www.magellium.com/fr/blog/portfolio-item/shared-gis-architecture/
- Magellium, Maquette Numérique GPE (2 septembre 2020) https://www.magellium.com/fr/blog/portfolio-item/maquette-numerique-gpe/
- Mozilla, Références des fonctions Javascripts : https://developer.mozilla.org/fr/docs/Web/API
- Stack Overflow, Site d'échange autour du développement : https://stackoverflow.com/