



Ecole Polytechnique de l'Université de Tours

Département Informatique

64 avenue Jean Portalis

37200 Tours, France

Tél. +33 (0)2 47 36 14 14

[polytech.univ-tours.fr](http://polytech.univ-tours.fr)

**Projet Recherche & Développement**

**2020-2021**

# **Modélisation, visualisation et comparaison de connectomes à l'aide de graphes**



**POLYTECH<sup>®</sup>**  
**TOURS**

**Entreprise**

**Polytech**



**Tuteur entreprise**

**Jean-Yves RAMEL**

**Étudiant**

**Jean-Baptiste HUYGHE (DI5)**

**Tuteur académique**

**Jean-Yves RAMEL**

# Liste des intervenants

## Entreprise

Polytech  
64 avenue Jean Portalis  
37200 Tours, France  
[polytech.univ-tours.fr](http://polytech.univ-tours.fr)



Nom	Email	Qualité
Jean-Baptiste HUYGHE	<a href="mailto:jean-baptiste.huyghe@etu.univ-tours.fr">jean-baptiste.huyghe@etu.univ-tours.fr</a>	Étudiant DI5
Jean-Yves RAMEL	<a href="mailto:jean-yves.ramel@univ-tours.fr">jean-yves.ramel@univ-tours.fr</a>	Tuteur académique, Département Informatique
Jean-Yves RAMEL	<a href="mailto:jean-yves.ramel@univ-tours.fr">jean-yves.ramel@univ-tours.fr</a>	Tuteur entreprise



# Avertissement

Ce document a été rédigé par Jean-Baptiste HUYGHE susnommé l'auteur.

L'entreprise Polytech est représentée par Jean-Yves RAMEL susnommé le tuteur entreprise.

L'Ecole Polytechnique de l'Université de Tours est représentée par Jean-Yves RAMEL susnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assume l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



# Pour citer ce document

Jean-Baptiste HUYGHE, *Modélisation, visualisation et comparaison de connectomes à l'aide de graphes* : , Projet Recherche & Développement, Ecole Polytechnique de l'Université de Tours, Tours, France, 2020-2021.

```
@mastersthesis{
  author={HUYGHE, Jean-Baptiste},
  title={Modélisation, visualisation et comparaison de connectomes à l'aide de graphes: },
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université de Tours},
  address={Tours, France},
  year={2020-2021}
}
```



# Table des matières

Liste des intervenants	<b>a</b>
Avertissement	<b>b</b>
Pour citer ce document	<b>c</b>
Table des matières	<b>i</b>
Table des figures	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
1 Acteurs, enjeux et contexte .....	1
2 Objectifs .....	1
3 Hypothèses .....	1
<b>2 Description générale</b>	<b>2</b>
1 Environnement du projet .....	2
2 Caractéristiques des utilisateurs .....	3
3 Fonctionnalités du système .....	3
4 Structure générale du système .....	4
<b>3 Etat de l'art</b>	<b>5</b>
1 Choix des outils .....	5
2 Structure du projet .....	5
2.1 Partie interface graphique .....	6
2.2 Partie librairie .....	6
FileManager .....	7
Connectome .....	7

	Visualization .....	8
	Comparison.....	8
3	Formats de fichiers.....	8
3.1	Fichiers d'Atlas.....	8
	.txt .....	8
	.csv .....	9
	.node.....	9
	.xml .....	9
3.2	Fichiers de Matrices d'adjacences.....	10
	.txt .....	10
	.edge .....	10
3.3	Fichiers de graphes .....	10
	.graphml .....	10
<b>4</b>	<b>Analyse et conception</b>	<b>13</b>
1	Analyse .....	13
1.1	Spécifications fonctionnelles .....	13
	Imports et exports.....	14
	Comparaisons.....	14
	Visualisation .....	14
	Interface graphique.....	14
1.2	Spécifications non fonctionnelles .....	14
2	Modélisation proposée.....	14
<b>5</b>	<b>Mise en oeuvre</b>	<b>15</b>
1	Outils et librairie utilisés.....	15
2	Éléments d'implémentation, choix techniques .....	15
	Paragraphe 1.....	16
3	Analyse des résultats, évaluation, qualité .....	16
4	Principales IHM.....	16
4.1	IHM 1 .....	16
<b>6</b>	<b>Bilan et conclusion</b>	<b>17</b>
1	Bilan du semestre 9 .....	17
1.1	Tâches effectuées au S9.....	17
1.2	Tâches en retard au S9 .....	17
1.3	Tâches à faire au S10.....	18
2	Bilan du semestre 10.....	18
3	Bilan sur la qualité .....	18
4	Bilan auto-critique.....	18

<b>Annexes</b>	<b>19</b>
<b>A Planification, gestion de projet</b>	<b>20</b>
1 Evolution du projet .....	20
<b>B Description des interfaces</b>	<b>21</b>
1 Interfaces homme/machines .....	21
2 Interfaces homme/machine.....	23
<b>C Cahier de Specification</b>	<b>24</b>
1 Spécifications Fonctionnelles .....	24
1.1 Description de la fonction 1 : importConnectomeData.....	24
Identification de la fonction : importConnectomeData .....	24
Présentation :.....	24
Priorité : primordiale.....	24
Entrée :.....	24
Sortie :.....	24
Description : .....	24
Extension facultative : importConnectomeDataAndImages .....	25
1.2 Description de la fonction 2 : visualizeConnectome.....	25
Identification de la fonction : visualizeConnectome .....	25
Présentation :.....	25
Priorité : primordiale.....	25
Entrée :.....	25
Sortie :.....	25
Description : .....	25
Extension facultative : visualizeBrainConnectome.....	25
Extension facultative : visualizeBrainSectionConnectome .....	25
Extension facultative : visualizeCustomConnectome .....	25
1.3 Description de la fonction 3 : compareGraphs .....	26
Identification de la fonction : compareGraphs .....	26
Présentation :.....	26
Priorité : primordiale.....	26
Entrée :.....	26
Sortie :.....	26
Description : .....	26
1.4 Description de la fonction 4 : graphEditDistance.....	26
Identification de la fonction : graphEditDistance .....	26
Présentation :.....	26
Priorité : primordiale.....	26

	Entrée : .....	26
	Sortie : .....	26
	Description : .....	26
1.5	Description de la fonction 5 : graphFindByPattern .....	27
	Identification de la fonction : graphFindByPattern .....	27
	Présentation : .....	27
	Priorité : facultative .....	27
	Entrée : .....	27
	Sortie : .....	27
	Description : .....	27
1.6	Description de la fonction 6 : saveLog .....	27
	Identification de la fonction : saveLog .....	27
	Présentation : .....	27
	Priorité : facultative .....	27
	Entrée : .....	27
	Sortie : .....	28
	Description : .....	28
1.7	Description de la fonction 7 : selectConnectome .....	28
	Identification de la fonction : selectConnectome .....	28
	Présentation : .....	28
	Priorité : primordiale .....	28
	Entrée : .....	28
	Sortie : .....	28
	Description : .....	28
1.8	Description de la fonction 8 : editParameters .....	28
	Identification de la fonction : editParameters .....	28
	Présentation : .....	28
	Priorité : facultative .....	28
	Description : .....	28
	Extension facultative : saveAndLoadParameters .....	28
2	Spécifications non fonctionnelles .....	29
2.1	Contraintes de développement et conception .....	29
2.2	Contraintes de fonctionnement et d'exploitation .....	29
2.2.1	Performances .....	29
2.2.2	Capacités .....	29
2.2.3	Contrôlabilité .....	29
2.2.4	Sécurité .....	29
2.2.5	Intégrité .....	29
2.3	Maintenance et évolution du système .....	30



<b>D</b>	<b>Cahier du développeur</b>	<b>31</b>
1	Introduction .....	31
2	Diagrammes architecturaux et UML .....	31
3	Descriptions détaillées de données exploitées .....	31
4	Descriptions détaillées des classes, modules, réalisations .....	31
<b>E</b>	<b>Document d'installation</b>	<b>32</b>
<b>F</b>	<b>Document d'utilisation</b>	<b>33</b>
<b>G</b>	<b>Cahier de test</b>	<b>34</b>
1	Tests unitaires .....	34
2	Tests d'intégration .....	34
<b>H</b>	<b>Bibliographie</b>	<b>35</b>
<b>I</b>	<b>Glossaire</b>	<b>36</b>
<b>J</b>	<b>Acronymes</b>	<b>37</b>



# Table des figures

<b>2</b>	<b>Description générale</b>	
2.1	Structure simplifiée du projet.....	3
2.2	Structure simplifiée du projet.....	4
<b>3</b>	<b>Etat de l'art</b>	
3.1	Diagramme de classes de l'existant.....	6
3.2	Exemple d'un volume de cerveau issu du rapport de PRD de Clément Condette ...	7
<b>A</b>	<b>Planification, gestion de projet</b>	
A.1	Le diagramme de Gantt prévisionnel.....	20
<b>B</b>	<b>Description des interfaces</b>	
B.1	Maquette de l'interface proposée pour importer un connectome.....	22
B.2	Proposition de maquette de l'interface principale .....	23

# 1

## Introduction

### 1 Acteurs, enjeux et contexte

Ce projet est un projet informatique qui regroupe deux domaines, le domaine du médical et celui de l'informatique. Plus précisément il s'agit d'un projet qui relie les graphes et les connectomes dans un projet informatique. Un connectome est une représentation des échanges entre les neurones réunis dans différentes zones du cerveau. De par leur nature, les connectomes peuvent être représentés par des graphes. Il s'agit d'un terrain peu exploré et mon tuteur et client monsieur Jean-Yves RAMEL souhaite étudier. Après des échanges, le Centre Hospitalier Régional Universitaire de Tours a montré son intérêt pour ces recherches et collabore en fournissant notamment des fichiers contenant des connectomes. Le projet présenté ici a deux buts à terme, permettre au CHRU de Tours d'analyser et d'étudier des connectomes à partir de fichier dans un logiciel et de l'autre de tester et proposer une suite d'outils (une API) open source, utilisable par des chercheurs et informaticiens, pour étudier les utilisations possibles des graphes leurs algorithmes sur les connectomes.

### 2 Objectifs

Dans ce contexte, monsieur Jean-Yves RAMEL a lancé un projet l'année dernière pour répondre à ce double objectif. Celui-ci a eu lieu dans le cadre d'un Projet de Recherche et Développement et a été réalisé par monsieur Clément CONDETTE. Le projet a bien avancé mais n'est pas terminé, il permet à l'heure actuelle d'importer des connectomes à partir de plusieurs types de fichiers, de choisir de calculer certaines informations et de visualiser les connectomes compatibles en 3 dimensions. L'objectif sera pour moi de reprendre l'existant, de l'analyser, de l'améliorer et d'ajouter de nouvelles fonctionnalités. Le but de ce document sera de vous présenter en détail ce qui devra être fait.

### 3 Hypothèses

Le projet manipule des graphes à partir de connectomes. Nous supposons que les connectomes fournis comportent des données spatiales pour chaque sommet du graphe car sans cette information, leur visualisation en 3D ne sera pas possible. Nous supposons également que les connectomes sont stockés dans des fichiers avec des formats compatibles avec le projet existant.

# 2

## Description générale

### 1 Environnement du projet

Le projet est la reprise et la continuité d'un projet réalisé l'année dernière en Projet de Recherche et Développement par Clément Condette. Il s'agit d'un projet réalisé en python sur l'IDE PyCharm. En plus des fonctionnalités de base du langage, utilise des librairies Python extérieur com NetworkX. Le projet à été réalisé sur le système d'exploitation Windows et devra y rester compatible car les utilisateurs de la partie logiciel graphique l'utilise. Le projet doit également être compatible avec GNU/Linux qui pourra être utilisé par les utilisateurs de l'API.

Le projet existant est divisé en différentes parties qui travaillent ensemble, elles sont bien divisées dont je conserverai la structure actuelle. Cependant, pour répondre au besoin, je vais devoir agir sur quelques-unes d'entre elles. Dans le plan ci-dessous, les cases bleues correspondent aux parties que je devrais modifier pour répondre aux besoins. La case verte correspond aux fonctionnalités facultatives.

type=figure

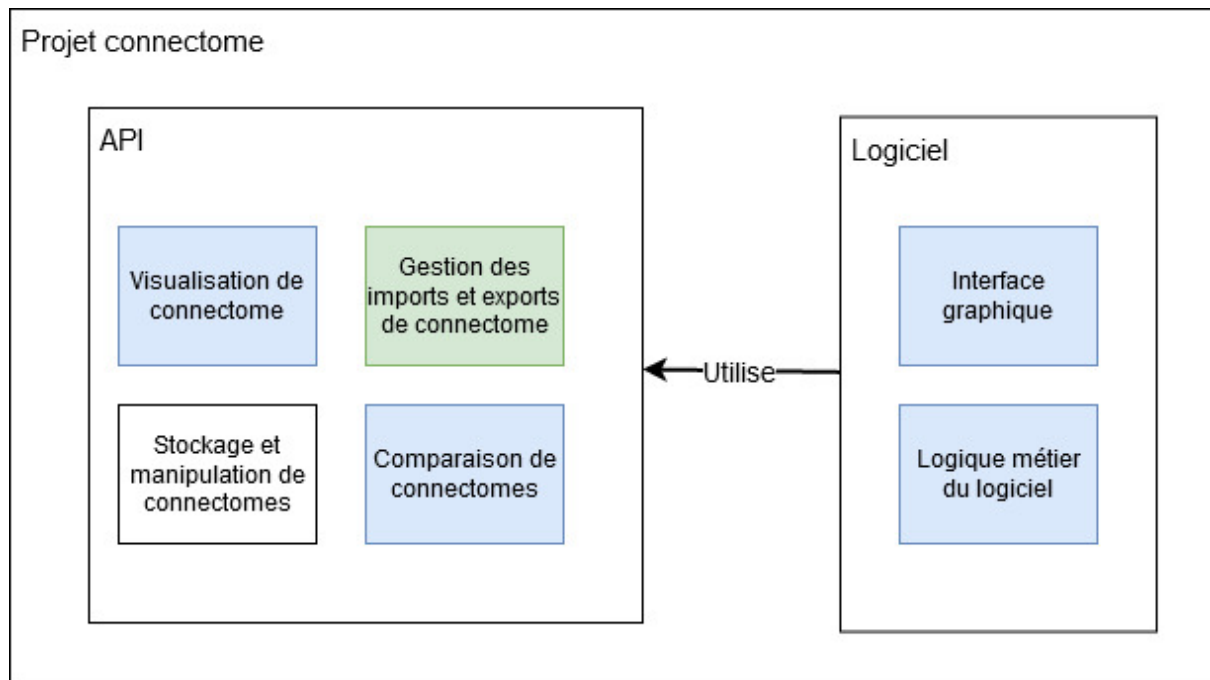


Figure 2.1 – Structure simplifiée du projet

## 2 Caractéristiques des utilisateurs

Comme il y a deux parties à ce projet, le logiciel et l'API, il y a deux profils d'utilisateurs bien distincts.

Le premier profil d'utilisateurs est celui de ceux qui utiliseront la partie logiciel dotée d'une interface graphique. Il s'agit de personnel médical, cela peut comprendre des médecins, des chercheurs sur les connectomes, des infirmières, etc. Ce sont des personnes qui ont une connaissance poussée du domaine et des termes médicaux, mais pas de connaissances particulières en informatique. Cela implique que le logiciel pourra contenir des termes médicaux mais que l'interface graphique devra être simple à prendre en main, intuitive et ergonomique.

Le second profil d'utilisateurs est celui de ceux qui utiliseront l'API. Ce profil rassemble des chercheurs, développeurs, etc. qui possèdent une connaissance avancée en informatique. L'API doit pouvoir être comprise et intégrée facilement à d'autres projets, pour cela le projet devra être bien commenté et documenté.

## 3 Fonctionnalités du système

Cette partie présente dans les grandes lignes les fonctionnalités qui existaient déjà, celles que je vais devoir modifier ou ajouter. Pour cela, voici un diagramme de cas d'utilisation qui les résume.

type=figure

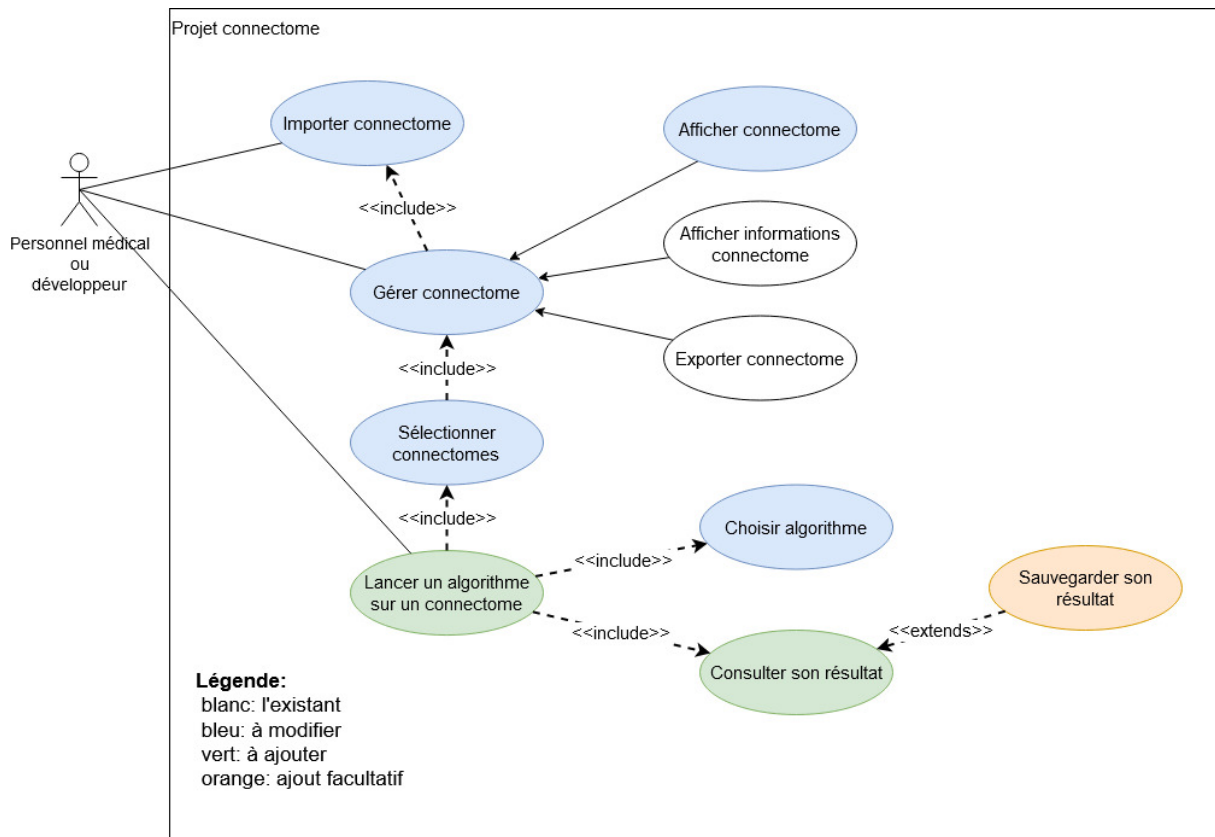


Figure 2.2 – Structure simplifiée du projet

Il est important de rappeler que les fonctionnalités existante du projet seront conservées et adaptées au besoin.

## 4 Structure générale du système

Le projet est donc composé de deux parties, la partie API et la partie logiciel graphique.

La partie API n'a pas d'interaction avec d'autres composants si ce n'est la librairie NetworkX pour manipuler des graphes. L'API est en elle même décomposée en 3 sous parties : une s'occuper des imports et exports de fichiers de connectomes, une autre du stockage et de la manipulation de connectomes, et enfin une dernière qui permet d'appliquer des algorithmes de comparaison de connectomes.

La partie logicielle graphique a des interactions avec l'API puisqu'elle en utilisera les fonctionnalités. La partie logicielle est donc presque exclusivement composée de composants pour l'affichage. A savoir les parties Vue et contrôleur du modèle MVC.

Il existe un diagramme de classe de l'existant que vous pouvez trouver en annexe. Cette conception est assez modulaire donc je pourrai facilement intervenir dessus en modifiant uniquement la partie concernée.

# 3

## Etat de l'art

Pour répondre à ce besoin d'analyse des connectomes, un projet a déjà été réalisé par Clément CONDETTE. Je vais dans cette partie analyser ce qui a été fait pour pouvoir l'utiliser et me baser dessus pour la suite du PRD.

### 1 Choix des outils

Tout d'abord le langage qui est utilisé est le Python car c'est un langage qui permet de développer assez rapidement une solution n'ayant pas particulièrement besoin de performance. De plus c'est un langage portable qui est utilisable quel que soit le système d'exploitation utilisé. Cependant, le principal point fort du Python est son nombre impressionnant de bibliothèque disponible et son emploi courant parmi les chercheurs.

### 2 Structure du projet

Lors de son projet Clément avait la même attente de développer d'un côté une librairie permettant de manipuler et comparer les graphes de connectomes et de l'autre une interface qui utilise la librairie pour permettre d'effectuer ces opérations et de visualiser des graphes en 3 dimensions.

Pour cela il a découpé son programme en différentes parties :

```
type=figure
```

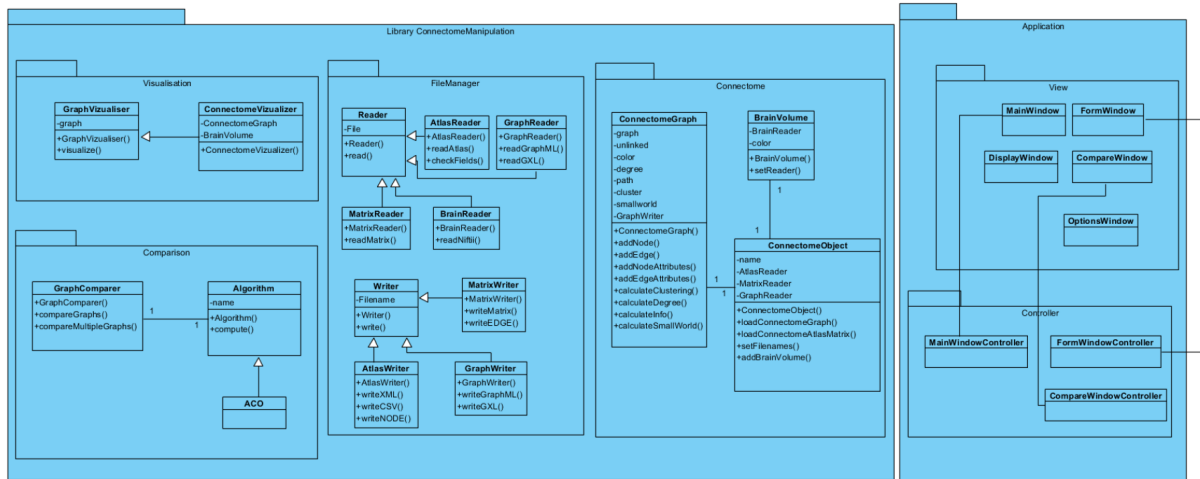


Figure 3.1 – Diagramme de classes de l'existant

Sur ce diagramme on peut voir la séparation entre les composants de la partie librairie à gauche et de la partie interface graphique à droite que je vais détailler.

## 2.1 Partie interface graphique

La partie interface graphique respecte le modèle architecturale MVC pour Modèle Vue Contrôleur. C'est un modèle de rangement des composants d'un programme par fonction. La «vue» va contenir tout ce qui concerne l'affichage graphique (les fenêtres, les boutons, le texte,...). Le modèle contient et permet la manipulation des éléments, dans notre cas la manipulation des graphes des connectomes. Enfin le "contrôleur" se charge de relier la "vue" et le "modèle" et contient la logique propre à l'interface graphique.

Toujours dans le diagramme de classe précédent on voit clairement les parties "view" ("vue" en anglais) et "Controller" ("contrôleur" en anglais). La partie "modèle" n'apparaît pas car il s'agit en fait de toute la partie librairie.

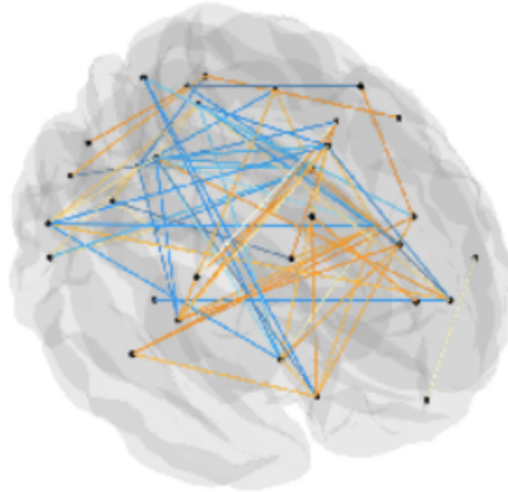
## 2.2 Partie librairie

La partie librairie a pour objectif de facilement être reprise pour un autre projet et elle comporte différents composants permettant de manipuler des connectomes. Elle se compose de quatre parties distinctes séparées en différents packages que je détaillerai ensuite :

- file manager
- connectome
- visualisation
- comparaison

Il est à noter qu'il existe dans presque tous ces packages des classes utilisées pour charger, relier et afficher un "volume de cerveau" avec le graphe d'un connectome. Il s'agit d'une représentation en 3 dimensions de la surface du cerveau duquel est extrait le connectome. Toutes les classes reliées à cette fonctionnalité comporte le mot "Brain" dans leur nom. En raison de la complexité technique de cette fonctionnalité, elle n'a pas été implémentée entièrement et n'est pas utilisée.





type=figure

**Figure 3.2** – *Exemple d'un volume de cerveau issu du rapport de PRD de Clément Condette*

### FileManager

Cette partie a pour fonction de gérer les fichiers. Cela regroupe les fonctions d'import des données nécessaires pour créer de nouveaux graphes de connectomes ainsi que l'import et d'export de graphes de connectomes.

Les fichiers qui contiennent les informations pour créer un graphe de connectome peuvent être de deux types : "atlas" ou "matrice d'adjacence".

Les fichiers atlas contiennent les des informations sur les sommets du graphe. Cela peut être le nom ou numéro d'un sommet, sa position,... Les fichiers d'atlas peuvent être au format .txt, .csv, .node ou .xml. Le code source permettant d'importer un atlas quel que soit son format se trouve dans le fichier et la classe AtlasReader.

Les fichiers "matrice d'adjacence" contiennent les arrêtes entre les sommets et leur poids. Il s'agit d'une matrice qui pour chaque sommet, donne le poids qui le sépare de chaque autre sommet (ou 0 s'ils ne sont pas relié). Les fichiers peuvent être au format .txt ou .edge. Le code source permettant d'importer une matrice d'adjacence quel que soit son format se trouve dans le fichier et la classe MatrixReader.

Les fichiers et classes GraphReader et GraphWriter permettent quand à eux d'importer et d'exporter des graphes de connectomes. Leur code actuel leur permet de gérer le format .graphml mais les classes ont été prévues pour permettre d'ajouter d'autres formats.

Dans le package FileManager sont également présente les classes AtlasWriter et MatrixWriter mais qui n'ont pas été implémentée.

Ce package est accessible mais n'est pas à utiliser directement mais plutôt par les méthodes proposées dans le package Connectome.

### Connectome

Ce package contient deux classes utilisée et utilisable pour manipuler les données d'un connectome : ConnectomeGraph et ConnectomeObject.

La classe ConnectomeGraph contient le graphe en lui même, ses informations et permet d'y effectuer des calculs et opérations comme ajouter des sommets, calculer le degré des sommets du graphe,...

La classe `ConnectomeObject` utilise la classe `ConnectomeGraphe` et permet d'initialiser, de gérer les import et exports de graphes ainsi que des opérations plus lourde qui modifient profondément le graphe. Cela peut être par exemple recharger le graphe, en faire un sous graphe.

### Visualization

Ce package a pour objectif de permettre a visualisation en 3 dimensions d'un graphe de connectome. Cela ce fait avec la méthode `visualizeConnectomeGraph` de la classe `ConnectomeVisualizer`.

La librairie graphique `Plotly` a été choisie pour permettre la visualisation d'un graphe en 3D. Cependant, il s'agit d'une librairie web car elle retourne une page web. Donc pour pouvoir utiliser et afficher le résultat de `Plotly`, la librairie `Pywebview` a été utilisé car elle permet d'afficher dans un programme une page web.

Le choix d'utiliser une librairie web fonctionne mais ne plaît pas au client qui demande à la changer par une technologie logiciel plus légère.

### Comparison

La partie comparaison doit permettre de comparer deux graphes de connectomes à l'aide de différents algorithmes. Pour cela, chaque algorithme que l'on souhaite ajouter doit hériter de la classe `Algorithm` qui possède une méthode à redéfinir dans les classes filles : `compute` qui prend en paramètre une liste de graphe. Ce choix de modélisation et d'implémentation permet d'ajouter de nouveaux algorithmes rapidement et qui deviennent directement utilisable.

Pour choisir d'utiliser un algorithme, il faut utiliser la classe `GraphComparer` et donner l'algorithme que l'on souhaite utiliser en paramètre de sont constructeur. Une fois que cela est fait, il reste à appeler la méthode `compareGraphs()` en lui donnant une liste de graphes en paramètre.

Dans le projet deux algorithmes sont présent : l'algorithme `AntColonyOptimization` qui ne fait rien (car il n'a pas encore été implémenté) et l'algorithme `SymmetricComparer` qui retourne la différence symétrique de deux graphes.

## 3 Formats de fichiers

Dans cette partie je vais détailler les types de formats utilisable dans ce projet. Comme énoncé dans la partie `FileManager`, il y a trois types de fichiers : atlas, matrice d'adjacences, graphe quie peuvent être dans différents formats.

### 3.1 Fichiers d'Atlas

#### .txt

Les fichiers d'atlas `.txt` sont les plus simples et contiennent deux informations par ligne : un numéro de sommet et son nom séparé par une tabulation.

En voici un exemple (extrait du fichier `liste_regions_valides.txt` présent dans le dossier `Tests_files`)

```

1 59   Right-Substantia-Nigra
2 60   Right-VentralDC
3 63   Right-choroid-plexus
4 11101   ctx_lh_G_and_S_frontomargin
5 11102   ctx_lh_G_and_S_occipital_inf
6 11103   ctx_lh_G_and_S_paracentral

```

### .csv

Les fichiers d'atlas .csv contiennent plus d'informations. Ils commencent par une ligne d'entête comportant une suite d'attribus séparés par des virgules. Les lignes suivantes constituent les valeurs de ces attributs, toujours séparé par des virgules.

En voici un exemple (extrait du fichier msdl\_rois\_labels.csv présent dans le dossier Tests\_files)

```
1 x,y,z,name,net name
2 53.47,-6.49,27.52,R Aud,Aud
3 -55.52,-43.77,10.08,L TPJ,Language
4 -48.66,25.11,5.7,Broca,Language
5 -3.39,17.19,63.52, Sup Front S,Language
6 54.42,-29.5,-2.72, R TPJ,Language
```

### .node

Les fichiers .node sont comme les fichiers .csv mais sans la ligne d'entête et en remplaçant les virgules par des tabulations.

En voici un exemple (extrait du fichier Node\_Brodmann82.node présent dans le dossier Tests\_files)

```
1 -18.278 22.911 55.919 1 1 8L
2 16.811 23.184 56.088 1 1 8R
3 -24.594 37.320 42.974 1 1 9L
4 22.509 37.266 43.076 1 1 9R
5 -16.038 59.012 9.959 1 1 10L
6 13.505 59.285 10.292 1 1 10R
```

### .xml

Les fichiers .xml sont plus complexes car ils contiennent plus d'informations. Ils se présentent sous cette forme (une partie des informations non utilisées pour ce projet ne sont pas présent ici) :

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <atlas version="1.0">
3   <header>
4     <name>Nom de l atlas </name>
5     <shortname>Nom court de l alias </shortname>
6   </header>
7   <data>
8     <label attribut1="valeur1" attribut2="valeur2" ...>Nom du premier sommet</label>
9     <label attribut1="valeur1" attribut2="valeur2" ...>Nom du second sommet</label>
10    ...
11  </data>
12 </atlas>
```

En voici un exemple (extrait du fichier HarvardOxford.xml présent dans le dossier Tests\_files)

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <atlas version="1.0">
3   <header>
4     <name>Harvard-Oxford Subcortical Structural Atlas</name>
```

```

5      <shortname>HOSPA</shortname>
6      <type>Probabalistic </type>
7  </header>
8  <data>
9  <label index="0" x="60" y="37" z="52">Left Cerebral White Matter</label>
10 <label index="1" x="70" y="58" z="40">Left Cerebral Cortex </label>
11 <label index="2" x="56" y="42" z="41">Left Lateral Ventrical</label>
12 </data>
13 </atlas>

```

### 3.2 Fichiers de Matrices d'adjacences

Les fichiers de matrices d'adjacences sont des matrice de dimension  $n^2$  où  $n$  est le nombre de sommets du graphe.

#### .txt

Les fichier de matrices d'adjacences au format .txt comportent les valeurs de la matrice séparées par des tabulations ou des virgules. Chaque retour à la ligne correspond à une autre ligne de la matrice. Il est à noter que dans ce format, une ligne vide est présente entre chaque ligne de la matrice.

En voici un exemple (extrait du fichier msdl\_matrix.txt présent dans le dossier Tests\_files)

```

1  -0.03227121000987019      0.2830037221596288      -0.40749976092707
2  -0.03227121000987019      0.0          0.928380713644892      0.3435322840245371
3  0.2830037221596288      0.928380713644892      0.0          0.19998103828042363

```

#### .edge

Les fichier de matrices d'adjacences au format .edge sont similaire au format .txt sauf qu'il n'y a pas une ligne sur deux vide.

En voici un exemple (extrait du fichier Edge\_Brodmann82.edge présent dans le dossier Tests\_files)

```

1  0.85      0.86      0.91      0.69      0.42
2  0.56      0.93      0.53      0.76      0.43
3  0.93      0.98      0.11      0.43      0.12
4  0.70      0.86      0.83      0.66      0.02
5  0.58      0.79      0.34      0.11      0.29

```

### 3.3 Fichiers de graphes

#### .graphml

Ce format est le plus complexe car il contient tout un graphe. En effet c'est ce format qui est utilisé pour exporter ou importer un graphe de connectome complet. En voici le format :

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <graphml xmlns="http://graphml.graphdrawing.org/xmlns"
3  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

```

```

4      xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
5      http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
6      <key attr.name="Nom de 1 attribut 1" attr.type="type attribut 1"
7      for="edge" id="code attribut1" />
8      <key attr.name="Nom de 1 attribut 2" attr.type="type attribut 2"
9      for="edge" id="code attribut2" />
10     ...
11     <key attr.name="Nom de 1 attribut A_id" attr.type="type attribut A"
12     for="node" id="code attribut A" />
13     <key attr.name="Nom de 1 attribut B" attr.type="type attribut B"
14     for="node" id="code attribut B" />
15     ...
16     <graph edgedefault="undirected">
17       <node id="code sommet 1">
18         <data key="code attribut1">valeur attribut 1 pour sommet 1</data>
19         <data key="code attribut1">valeur attribut 2 pour sommet 1</data>
20         ...
21       </node>
22       <node id="code sommet 2">
23         <data key="code attribut1">valeur attribut 1 pour sommet 2</data>
24         <data key="code attribut1">valeur attribut 2 pour sommet 2</data>
25         ...
26       </node>
27       ...
28       <edge source="code sommet 1" target="code sommet 2">
29         <data key="code attribut A">valeur attribut A pour cette arrête</data>
30         <data key="code attribut B">valeur attribut A pour cette arrête</data>
31         ...
32       </edge>
33       <edge source="code sommet 2" target="code sommet 1">
34         ...
35       </edge>
36       ...
37     </graph>
38 </graphml>

```

En voici un exemple (extrait du fichier Edge\_Brodmann82.edge présent dans le dossier Tests\_files)

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <graphml xmlns="http://graphml.graphdrawing.org/xmlns"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
5   http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
6   <key attr.name="number_of_fibers" attr.type="double" for="edge" id="d11" />
7   <key attr.name="FA_mean" attr.type="double" for="edge" id="d10" />
8   <key attr.name="fiber_length_mean" attr.type="double" for="edge" id="d9" />
9   <key attr.name="dn_correspondence_id" attr.type="int" for="node" id="d8" />
10  <key attr.name="dn_hemisphere" attr.type="string" for="node" id="d7" />
11  <key attr.name="dn_name" attr.type="string" for="node" id="d6" />
12  <key attr.name="dn_fsname" attr.type="string" for="node" id="d5" />
13  <key attr.name="dn_region" attr.type="string" for="node" id="d4" />

```

```

14 <key attr.name="dn_correspondence_id" attr.type="string" for="node" id="d3" />
15 <key attr.name="dn_position_z" attr.type="double" for="node" id="d2" />
16 <key attr.name="dn_position_y" attr.type="double" for="node" id="d1" />
17 <key attr.name="dn_position_x" attr.type="double" for="node" id="d0" />
18 <graph edgedefault="undirected">
19   <node id="1">
20     <data key="d0">35.2333637192</data>
21     <data key="d1">83.3792160438</data>
22     <data key="d2">14.9257064722</data>
23     <data key="d3">1</data>
24     <data key="d4">cortical</data>
25     <data key="d5">lateralorbitofrontal</data>
26     <data key="d6">rh.lateralorbitofrontal</data>
27     <data key="d7">right</data>
28   </node>
29   <node id="2">
30     <data key="d0">25.3272138229</data>
31     <data key="d1">86.7602591793</data>
32     <data key="d2">17.1976241901</data>
33     <data key="d3">2</data>
34     <data key="d4">cortical</data>
35     <data key="d5">parsorbitalis</data>
36     <data key="d6">rh.parsorbitalis</data>
37     <data key="d7">right</data>
38   </node>
39   ...
40   <edge source="1" target="34">
41     <data key="d9">15.6748124361</data>
42     <data key="d10">0.21499947831</data>
43     <data key="d11">372.0</data>
44   </edge>
45   <edge source="1" target="36">
46     <data key="d9">29.9328789711</data>
47     <data key="d10">0.139346929267</data>
48     <data key="d11">25.25</data>
49   </edge>
50   <edge source="1" target="37">
51     <data key="d9">17.8043575287</data>
52     <data key="d10">0.176173517481</data>
53     <data key="d11">119.375</data>
54   </edge>
55   ...
56 </graph>
57 </graphml>

```

# 4

## Analyse et conception

### 1 Analyse

L'analyse est une partie importante dans la réalisation de tout projet. C'est lors de celle-ci que l'on détermine précisément ce qui est attendu par le client. Cela comprend la liste détaillée de toutes les contraintes imposées et la liste des fonctionnalités désirées. C'est à partir de cette analyse qu'est rédigé le cahier des spécifications. Le cahier des spécifications est un document contractuel qui répertorie toutes ces attentes et qui est validé par le client. Le cahier des spécifications comprend deux grandes parties :

- les spécifications fonctionnelles
- les spécifications non fonctionnelles

Les spécifications fonctionnelles contiennent toute la liste des fonctionnalités attendues et détaillées. Pour chaque fonctionnalité est indiquée : son nom, sa priorité (primordiale, secondaire, facultative), ses entrées, ses sorties et sa description.

Les spécifications non fonctionnelles décrivent les autres contraintes qui ne concernent pas les fonctionnalités. Il peut s'agir par exemple de contraintes de développement, de conception, de fonctionnement,...

Ces spécifications sont assez longues, en conséquence vous trouverez ici une version résumée et simplifiée. Si vous souhaitez consulter la version complète du cahier des spécifications, vous pourrez la trouver en annexe.

#### 1.1 Spécifications fonctionnelles

Les spécifications fonctionnelles pour ce projet s'articulent autour de 4 axes :

- les imports et exports de connectomes
- les comparaisons de connectomes
- la visualisation de connectome
- l'interface graphique

### Imports et exports

L'utilisateur doit pouvoir créer un connectome à partir d'une liste des régions anatomiques (atlas) et de plusieurs valeurs pour les échanges entre ces régions (matrices d'adjacences). En conséquence, l'import et l'export de connectomes doit pouvoir prendre en compte ce changement.

Cette partie comprend également une fonctionnalité facultative qui doit permettre l'import de coupes 3D du cerveau.

### Comparaisons

L'utilisateur peut choisir un algorithme pour comparer deux graphes de connectomes et consulter son résultat. Un algorithme de mesure de similarité entre deux graphes doit également être ajouté.

Un autre algorithme doit être ajouté de manière facultative : un algorithme de recherche de pattern dans un graphe.

### Visualisation

Le graphe d'un connectome doit pouvoir être visualisé en 3D. Cela comprend une rotation ce celui-ci, une gestion du zoom, une sélection d'un sommet ou d'une arrête du graphe pour en consulter ses informations et de pouvoir changer la couleur des sommets et des arrêtes.

Il y a aussi des fonctionnalités facultatives dans cette partie : une visualisation en plus de coupes 2D du cerveau, une visualisation en 3D de coupes 2D du cerveau sur le graphe et une mise en évidence de certains sommets.

### Interface graphique

L'interface graphique devra être modifier pour être plus ergonomique, pouvoir sélectionner plusieurs connectomes, de pouvoir effectuer des comparaisons de connectomes, configurer les paramètres donnés aux algorithmes de comparaison, de consulter les résultats de ces algorithmes.

Deux fonctionnalités facultatives sont présente dans cette partie : l'export de résultat d'algorithme de comparaison dans un fichier et pouvoir gérer les patterns de l'algorithme facultatif de détection de pattern dans un graphe.

## 1.2 Spécifications non fonctionnelles

La spécification non fonctionnelles imposent d'abord des contraintes sur les technologies utilisées. Le langage de programmation Python devra être utilisé pour reprendre et poursuivre le projet existant. Une liste de librairie doit également être utilisée alors que d'autre utilisant des technologies web ont été banni. Le projet devra être compatible sur les systèmes d'exploitations Windows et GNU/Linux. Enfin, ces spécifications définissent les attentes pour les rendus à la fin du projet.

## 2 Modélisation proposée

Inclure ici une description du système à développer pouvant notamment inclure les principaux diagrammes UML non détaillés et démontrant sa faisabilité durant la phase de mise en oeuvre.

Les modes de validation prévus pour les différents éléments à produire pourront être précisés ici.



# 5

## Mise en oeuvre

Description de vos productions et de leurs modes de réalisation.  
(résumé du cahier de développement inséré en ANNEXE)

### 1 Outils et librairie utilisés

### 2 Éléments d'implémentation, choix techniques

Exemples d'utilisation du glossaire et acronymes : **Unified Modeling Language (UML)** and **Framework**

Voici un exemple de citations [2] [1].

Paragraphe 1

### 3 Analyse des résultats, évaluation, qualité

### 4 Principales IHM

#### 4.1 IHM 1

Résumé des principaux elements présent dans le Guide de l'utilisateur avec d'éventuels compléments d'information sur leur mode de mise en oeuvre.

# 6

## Bilan et conclusion

### 1 Bilan du semestre 9

Le travail réalisé au cours de ce semestre a en très grande partie été la recherche. En effet, reprendre un projet existant nécessite de prendre connaissance et de comprendre son contexte et ses enjeux. Une fois que cela est fait, il faut analyser l'existant pour savoir où et comment agir ainsi que se documenter sur différentes solutions qui pourront être utilisées pour mener à bien ce projet.

Cette première partie de projet axée sur la recherche, l'analyse et la conception n'est pas une partie qui est très stimulante, d'autant plus lors des périodes de confinement. Cependant elle est d'une importance capitale pour la seconde phase du projet qui est le développement. En effet, tout le travail qui a été fait permet d'avoir une vision globale du projet et de savoir quoi faire. Tout ce qui va être fait ensuite sera basé sur cette première partie et donne ainsi un cap à suivre.

#### 1.1 Tâches effectuées au S9

- Découverte du sujet et de l'environnement
- Test du programme existant
- Étude de l'existant
- Échanges avec Clément CONDETTE à propos du projet
- État de l'art des solutions de visualisation de connectome
- Analyse de l'intégration des nouvelles fonctionnalités
- Rédaction du cahier des spécifications
- Planification prévisionnelle des tâches
- Rédaction de la partie S9 du rapport

#### 1.2 Tâches en retard au S9

Un retard a été pris au niveau de l'état de l'art pour trouver une API de visualisation de connectome en 3D. Cette recherche comprend l'analyse de solutions trouvées sur internet ainsi que d'une liste de propositions de mon tuteur. En conséquence cette tâche sera la première qui devra être faite à la suite du rendu de ce rapport et à la soutenance du S9. Ce choix d'API devra être fait et validé par mon tuteur.

### 1.3 Tâches à faire au S10

Les tâches à réalisées au S10 seront en grande parie du développement et du test. Elles sont presque toutes issues du cahier des spécifications. Vous pouvez trouvez mon planning prévisionnel pour réaliser ces tâches en annexes.

### 2 Bilan du semestre 10

Bilan global => respect du cahier des charges (fai/ a faire)

### 3 Bilan sur la qualité

### 4 Bilan auto-critique

# Annexes

# A

# Planification, gestion de projet

## 1 Evolution du projet

Le diagramme de Gantt Initial pour la planification de ce projet .

type=figure

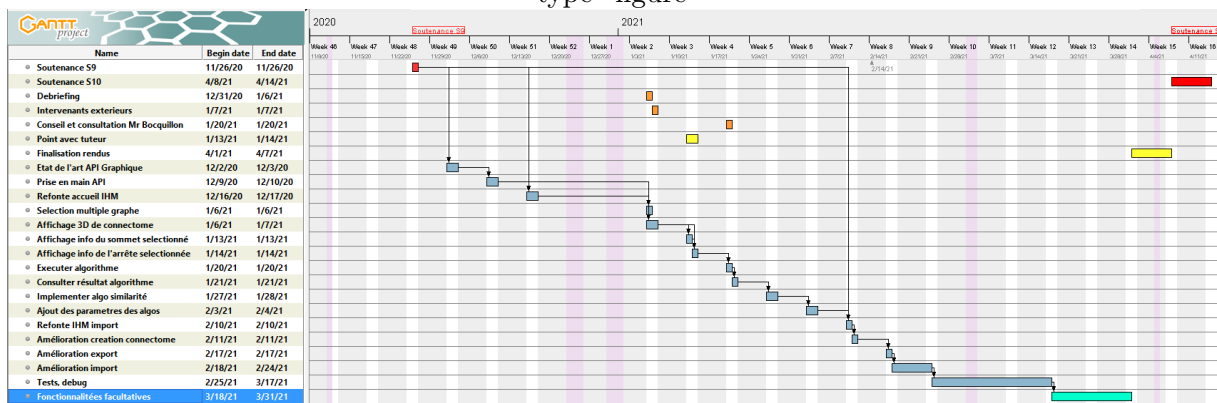


Figure A.1 – Le diagramme de Gantt prévisionnel

# B

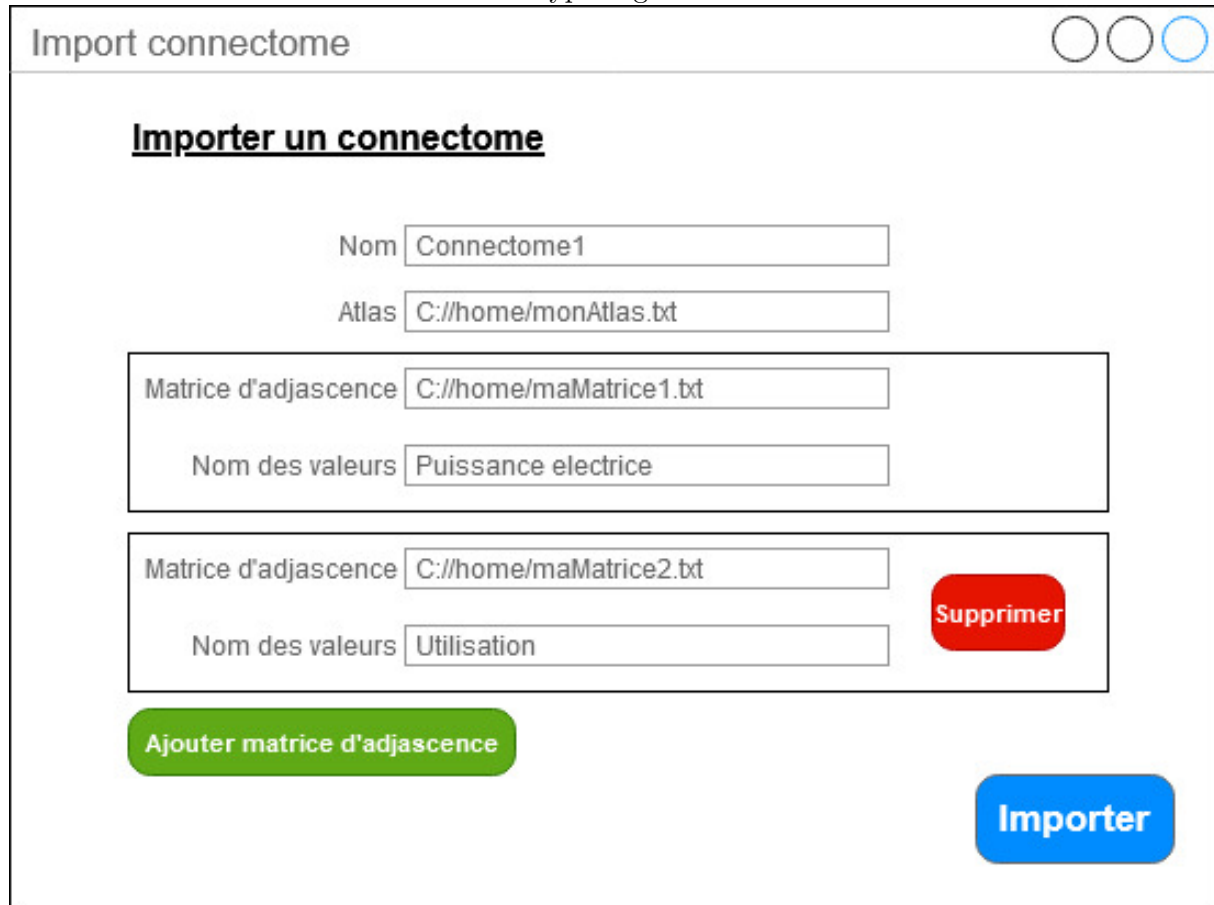
## Description des interfaces

### 1 Interfaces homme/machines

Dans la partie logiciel graphique, l'ergonomie est importante pour que la prise en main soit simple et intuitive. En conséquence, les mêmes types fonctionnalités doivent se trouver proche. Elle doit respecter les conventions des logiciels usuels (selection, menus,...). Les procédures et tâches doivent être guidées, pour l'import des fichiers notamment. Les messages d'erreurs doivent être simples à comprendre.

La partie API doit reprendre le même principe concernant la prise en main, les messages d'erreurs. Vous trouverez ci dessous les propositions des maquettes pour les interfaces graphiques attendues.

type=figure



Import connectome

**Importer un connectome**

Nom

Atlas

Matrice d'adjascence

Nom des valeurs

Matrice d'adjascence

Nom des valeurs

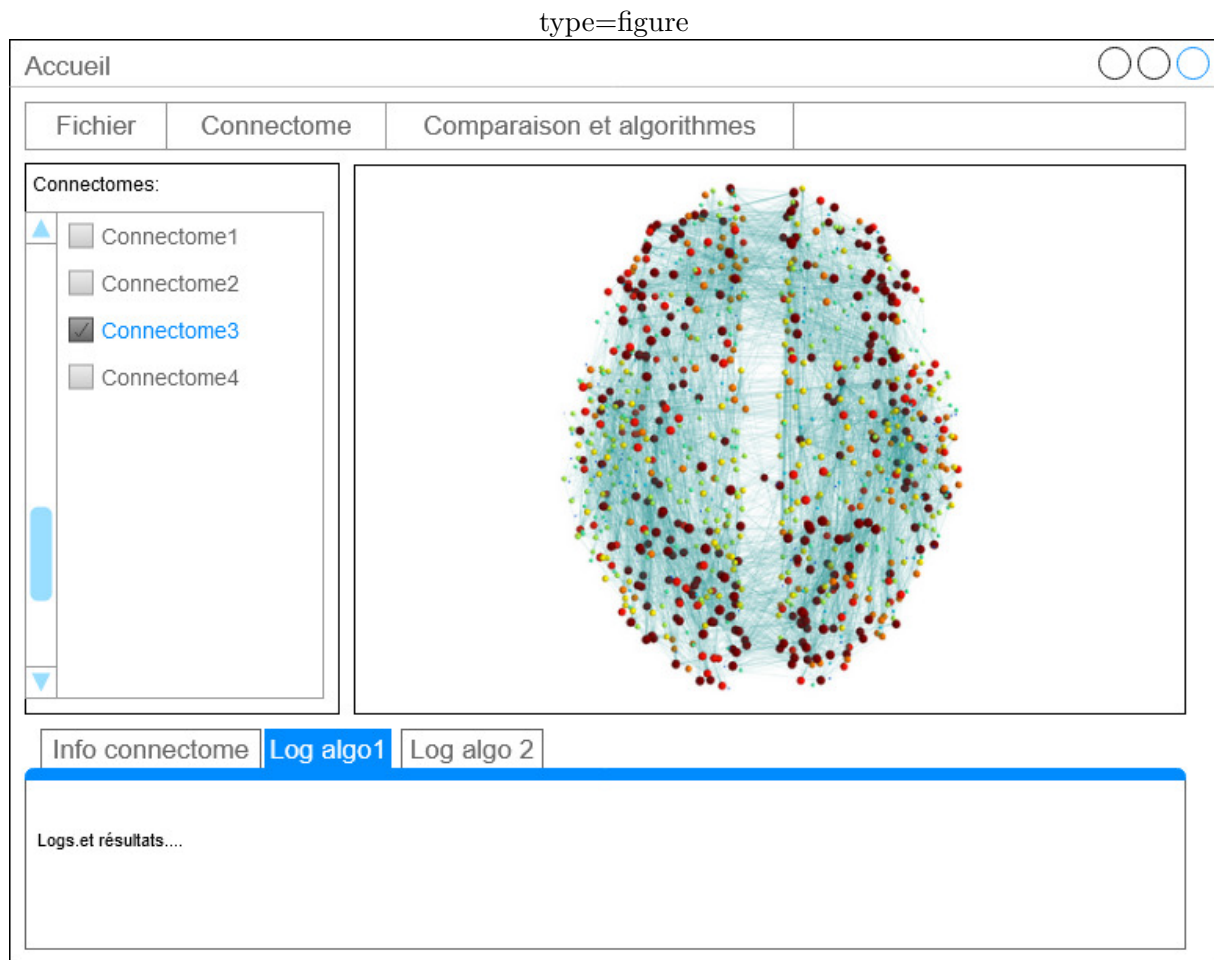
**Supprimer**

**Ajouter matrice d'adjascence**

**Importer**

Figure B.1 – Maquette de l'interface proposée pour importer un connectome





**Figure B.2** – Proposition de maquette de l'interface principale

## 2 Interfaces homme/machine

Le projet aura besoin de droits d'utilisateur suffisants sur le système d'exploitation sur lequel il est exécuté. Cela comprend de pouvoir lire les fichiers à importer ainsi que les droits d'écriture à l'endroit désiré pour l'export.

Ce n'est pas tout à fait une interface logiciel / logiciel puisque cela se fait au niveau de l'implémentation, mais le logiciel graphique utilisera l'API, qui elle-même utilisera des bibliothèques.



# Cahier de Specification

## 1 Spécifications Fonctionnelles

### 1.1 Description de la fonction 1 : importConnectomeData

**Identification de la fonction : importConnectomeData**

**Présentation :**

Fonction permettant d'importer les données concernant un connectome dans l'application.

**Priorité : primordiale**

**Entrée :**

- Une chaîne de caractères représentant le titre du connectome.
- Un fichier texte contenant l'atlas (la liste des régions anatomiques) utilisé pour construire le connectome.
- Une liste de fichiers texte contenant une matrice d'adjacence (les liens et informations entre ces régions) du connectome.
- Une liste de chaîne de caractères représentant le nom de chaque matrice d'adjacence.

**Sortie :**

- Un connectome sous forme de graphe.

**Description :**

Cette fonction va stocker la liste des régions anatomiques d'un connectome et leurs échanges dans des variables pour pouvoir construire un graphe. L'utilisateur va pouvoir lors de l'import définir un nom pour le connectome qui servira pour les affichages et la construction du graphe. Les fichiers d'entrées doivent contenir à minima les données pour construire un graphe binaire et non orienté, avec des labels pour les sommets. L'utilisateur pourra à l'import choisir un titre pour le graphe modélisé à partir de ce connectome. Il pourra importer autant de matrice d'adjacence qu'il le souhaite et y donner un nom aux valeurs contenues.

Les matrices d'adjacence doivent être de même dimensions que l'atlas, sinon une erreur est retournée.

**Extension facultative : importConnectomeDataAndImages**

Cette fonction va stocker la liste des régions anatomiques d'un connectome et leurs échanges dans des variables pour pouvoir construire un graphe. L'utilisateur va pouvoir lors de l'import définir un nom pour le connectome qui servira pour les affichages et la construction du graphe. Les fichiers d'entrées doivent contenir à minima les données pour construire un graphe binaire et non orienté, avec des labels pour les sommets. L'utilisateur pourra à l'import choisir un titre pour le graphe modélisé à partir de ce connectome. Il pourra importer autant de matrice d'adjacence qu'il le souhaite et y donner un nom aux valeurs contenues.

**1.2 Description de la fonction 2 : visualizeConnectome****Identification de la fonction : visualizeConnectome****Présentation :**

Fonction permettant d'afficher un graphe issu d'un connectome en 3 dimensions.

**Priorité : primordiale****Entrée :**

- Un connectome
- Une liste de paramètres

**Sortie :**

- Pas de sortie, l'effet est uniquement graphique

**Description :**

Cette fonction va permettre à l'utilisateur de visualiser en 3D une représentation d'un connectome. Au minimum, l'utilisateur pourra tourner la représentation, changer le zoom de la visualisation. L'utilisateur pourra également sélectionner un sommet ou une arête du graphe du connectome et visualiser les informations sur sa sélection. La liste de paramètres permet de modifier l'affichage.

Cette fonction ne devra pas utiliser d'API utilisant de technologie web. La librairie utilisée devra permettre d'afficher des images issus de fichiers ".nii" même si cela n'est pas implémenté.

**Extension facultative : visualizeBrainConnectome**

Cette fonction facultative reprend exactement les mêmes éléments mais qui doit permettre en plus d'afficher avec le connectome ses images 3D.

**Extension facultative : visualizeBrainSectionConnectome**

Cette fonction facultative doit permettre d'afficher avec le connectome des coupes 2D de la visualisation 3D.

**Extension facultative : visualizeCustomConnectome**

Cette fonction facultative doit permettre de modifier la taille et la couleur des noeuds et arêtes en fonction de la valeur de leurs attributs

### 1.3 Description de la fonction 3 : compareGraphs

#### Identification de la fonction : compareGraphs

##### Présentation :

Fonction permettant de comparer deux graphes.

##### Priorité : primordiale

##### Entrée :

- Deux graphes
- Une liste de paramètres de l'algorithme de comparaison

##### Sortie :

- Le résultat de l'algorithme.

##### Description :

Cette fonction est une reprise de l'existant qui devra être adaptée et améliorée.

Elle permet à l'utilisateur de choisir deux graphes de connectome et de les comparer avec un algorithme choisi en fonction des paramètres fournis. Les résultats de la comparaison sont retournés par la fonction. Ils sont ensuite consultables dans une zone de texte dédiée du logiciel, ou dans la console pour l'API.

### 1.4 Description de la fonction 4 : graphEditDistance

#### Identification de la fonction : graphEditDistance

##### Présentation :

Fonction permettant de mesurer la similarité entre deux graphes en utilisant leurs distances.

##### Priorité : primordiale

##### Entrée :

- Deux graphes
- Une liste de paramètres de l'algorithme de comparaison

##### Sortie :

- Le résultat de la comparaison

##### Description :

Cette fonction permet à l'utilisateur de mesurer la similarité en utilisant la distance entre deux graphes. Cette fonction doit faire appel à la méthode `algorithms.similarity.graph_edit_distance` de la librairie Python NetworkX. Le résultat est retourné par la fonction. Il est ensuite affiché sous forme de chaîne de caractère dans un espace dédié du logiciel, ou dans la console pour l'API.

## 1.5 Description de la fonction 5 : graphFindByPattern

### Identification de la fonction : graphFindByPattern

#### Présentation :

Fonction permettant de trouver un pattern particulier dans un graph donné.

#### Priorité : facultative

#### Entrée :

- Un graph
- Un pattern
- Une liste de paramètres pour l'algorithme

#### Sortie :

- Le résultat de la recherche

#### Description :

Cette fonction permet à l'utilisateur de chercher un pattern particulier dans un graph du connectome sélectionné. Cela nécessite de pouvoir choisir un pattern. Le résultat est retourné puis affiché sous forme de chaîne de caractères dans un espace dédié du logiciel, ou dans la console pour l'API. Au niveau de l'implémentation côté API, un pattern est un graph, comme un connectome.

Une variante de cette fonctionnalité alternative est d'afficher en plus le résultat en mettant en évidence le pattern trouvé sur la visualisation 3D.

Cette fonction peut également contenir des sous fonctions, elles aussi facultatives, permettant la gestion des patterns particuliers. Cela comprend :

- l'ajout de patterns
- la suppression de patterns
- la sélection de sommets sur la visualisation pour en créer un pattern
- l'import de patterns au format ".graphml"
- l'export de patterns au format ".graphml"

Les fonctionnalités d'ajout et de création de patterns demandent à l'utilisateur un nom pour le nouveau pattern.

## 1.6 Description de la fonction 6 : saveLog

### Identification de la fonction : saveLog

#### Présentation :

Fonction permettant de sauvegarder les résultats d'un algorithme dans un fichier.

#### Priorité : facultative

#### Entrée :

- Une chaîne de caractères représentant la résultat d'un algorithme de comparaison
- Le chemin où enregistrer le fichier

**Sortie :**

- Un fichier contenant la chaîne de caractères

**Description :**

Cette fonction permet à l'utilisateur de sauvegarder le résultat d'un algorithme présent dans la zone des logs dans un fichier. L'utilisateur du logiciel se voit proposer une interface pour choisir où sera sauvegarder les logs ainsi que le nom qu'il souhaite donner au fichier.

## 1.7 Description de la fonction 7 : selectConnectome

**Identification de la fonction : selectConnectome****Présentation :**

Fonction permettant de sélectionner un ou plusieurs connectomes.

**Priorité : primordiale****Entrée :**

- Une liste de connectome

**Sortie :**

- Une liste de connectome sélectionnée

**Description :**

Cette fonction permet à l'utilisateur de choisir un ou plusieurs connectomes parmi ceux chargés dans le logiciel. Cette fonction est nécessaire pour pouvoir envoyer une liste de connectome à l'algorithme choisi.

## 1.8 Description de la fonction 8 : editParameters

**Identification de la fonction : editParameters****Présentation :**

Fonction permettant de modifier les paramètres fournis aux algorithmes pour l'interface graphique.

**Priorité : facultative****Description :**

Cette fonction permet à l'utilisateur de modifier dans une interface graphique les paramètres fournis à chaque algorithme de comparaison.

**Extension facultative : saveAndLoadParameters**

Cette fonction facultative permet de charger et de sauvegarder les paramètres des différents algorithmes dans un fichier afin de sauvegarder les modifications de paramètre entre chaque utilisation du logiciel.

## 2 Spécifications non fonctionnelles

### 2.1 Contraintes de développement et conception

Les choix de mon prédécesseur ainsi que les attentes de monsieur RAMEL m'imposent certaines contraintes de développement et de conception. La première concerne l'utilisation du langage de programmation qui sera le python et de l'API graphique base Tkinter et la librairie NetworkX. Le système d'exploitation n'est pas imposé mais le projet devra être compatible sur GNU/Linux et Windows. Le choix de la librairie de visualisation en 3 dimensions d'un connectome ne doit pas utiliser de technologie web.

Le système de version Git devra être utilisé et présenter tous les codes et documents attendus. Le dépôt GitHub devra être utilisé et accessible par monsieur Jean-Yves RAMEL. Le rendu final se fera sur le dépôt GitHub.

### 2.2 Contraintes de fonctionnement et d'exploitation

#### 2.2.1 Performances

Ce projet n'a pas de contraintes particulières concernant les performances ou les temps d'exécution si ce n'est s'exécuter dans un temps raisonnable pour que cela soit confortable pour l'utilisateur. Pour les algorithmes à implémenter, ils n'ont pas non plus d'attente concernant leurs performances / complexité mais il serait préférable d'essayer de prendre en compte cet élément.

#### 2.2.2 Capacités

Il n'y a pas de contrainte de capacité particulière si ce n'est les capacités de stockage ou de calculs de la machine. Il faut cependant garder à l'esprit que plus le connectome à traiter sera gros, plus cela demandera de temps de calcul pour réaliser l'opération désirée.

#### 2.2.3 Contrôlabilité

L'exécution des algorithmes doit afficher un retour dans un espace dédié de l'interface graphique, dans la console pour l'API.

#### 2.2.4 Sécurité

Le projet ne doit pas répondre à un niveau de confidentialité. Les données les plus sensibles sont les informations stockées dans les fichiers de connectomes et dont la responsabilité revient à l'utilisateur.

#### 2.2.5 Intégrité

Il n'y a pas de contrainte d'intégrité, le programme chargé en mémoire vive les données importées et travaille toujours sur cette mémoire. Toute fermeture (volontaire ou non) du logiciel ou de l'API sans effectuer d'export ne sauvegarde aucune donnée.

## 2.3 Maintenance et évolution du système

Les questions de maintenance et d'évolutivité sont très importantes pour ce projet et ce pour plusieurs raisons. La première est que le but final de la partie API est de devenir open source. La seconde est que ce projet pourra être repris par quelqu'un d'autre pour y effectuer différentes actions : la maintenance, l'amélioration et l'ajout de nouvelles fonctionnalités.

En conséquence, le code et le projet devront être bien documentés et testés afin de faciliter toute maintenance ou évolution.

En plus du code source du projet, les tests, la documentation, le cahier des spécifications, et un rapport devront être fournis.



# D

## Cahier du développeur

1 Introduction

2 Diagrammes architecturaux et UML

3 Descriptions détaillées de données exploitées

4 Descriptions détaillées des classes, modules, réalisations



# Document d'installation

Ce document regroupe toutes les informations nécessaires pour l'installation du projet sur les machines, ainsi que pour sa mise en production.

A blue square containing a white capital letter 'F'.

# Document d'utilisation

Instructions d'utilisation



# Cahier de test

Les tests visent à garantir l'exactitude, l'intégrité, la sécurité et les performances du logiciel.

## 1 Tests unitaires

blablabla

ID	IDENTIFICATION OF COMPONENT
1	Afficher toutes les missions de l'utilisateur identifié
DESCRIPTION OF THE TEST (granularity, scenario, values, actions)	
Action :blablabla. blblbla.	
EXPECTED RESULTS	
Cas 1 : blblablaa. Cas 2 : blablabla.	
OBTAINED RESULTS	
blablabla.	

## 2 Tests d'intégration

blablabla

# H

## Bibliographie

- [1] Joseph REDMON, Santosh Kumar DIVVALA, Ross B. GIRSHICK et Ali FARHADI. « You Only Look Once : Unified, Real-Time Object Detection ». In : *CoRR* abs/1506.02640 (2015). arXiv : [1506.02640](https://arxiv.org/abs/1506.02640). URL : <http://arxiv.org/abs/1506.02640>.
- [2] Joseph REDMON et Ali FARHADI. « YOLOv3 : An Incremental Improvement ». In : *CoRR* abs/1804.02767 (2018). arXiv : [1804.02767](https://arxiv.org/abs/1804.02767). URL : <http://arxiv.org/abs/1804.02767>.

# I

## Glossaire

**Framework** Ensemble d'outils et de composants logiciels organisés conformément à un plan d'architecture et des patterns. 16

# J

## Acronymes

**UML** Unified Modeling Language. 16

Jean-Baptiste HUYGHE

Encadrement : Jean-Yves RAMEL



En collaboration avec Polytech

## Objectifs

- point 1
- point2
- point 3



LABORATOIRE D'INFORMATIQUE FONDAMENTALE ET APPLIQUÉE DE TOURS

## Mise en œuvre

Blablabla



LABORATOIRE D'INFORMATIQUE FONDAMENTALE ET APPLIQUÉE DE TOURS

## Résultats attendus

blablabla



LABORATOIRE D'INFORMATIQUE FONDAMENTALE ET APPLIQUÉE DE TOURS





# Modélisation, visualisation et comparaison de connectomes à l'aide de graphespolytech@subtitle:

Jean-Baptiste HUYGHE

Encadrement : Jean-Yves RAMEL



En collaboration avec Polytech

## Objectifs

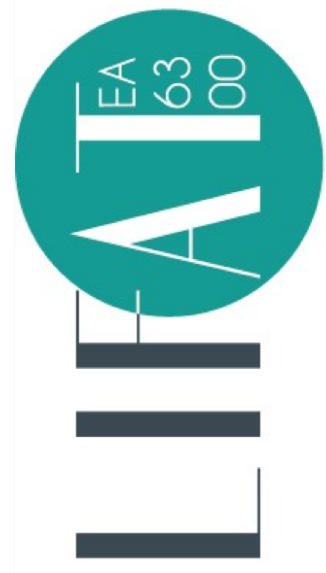
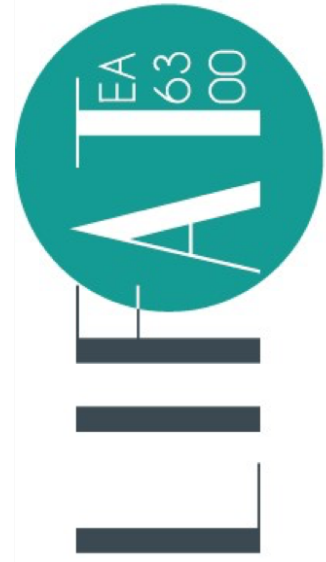
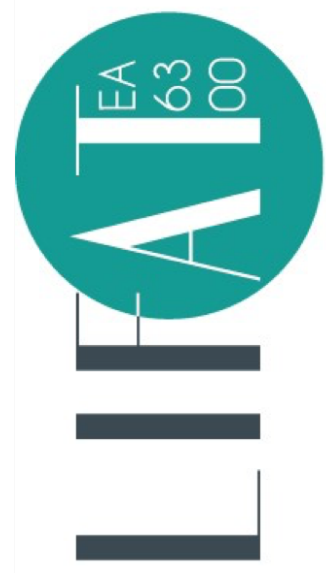
- point 1
- point2
- point 3

## Mise en œuvre

Blablablabla

## Résultats attendus

blablabla



LABORATOIRE D'INFORMATIQUE FONDAMENTALE ET APPLIQUÉE DE TOURS

LABORATOIRE D'INFORMATIQUE FONDAMENTALE ET APPLIQUÉE DE TOURS

LABORATOIRE D'INFORMATIQUE FONDAMENTALE ET APPLIQUÉE DE TOURS



# Modélisation, visualisation et comparaison de connectomes à l'aide de graphes

## Résumé

Le sujet de Projet de Recherche et Développement est la reprise d'un projet existant de visualisation de connectome. Plus précisément, il s'agit d'une API et d'un logiciel graphique de manipulation, de comparaison et de visualisation de connectomes. Mon objectif est de reprendre ce projet afin d'y ajouter des fonctionnalités et revoir son ergonomie.

## Mots-clés

connectome, visualisation, graphe, python

## Abstract

The Research and Development Project topic is the resumption of an existing connectome visualization project. More specifically, it is an API and graphical software for handling, comparing and viewing connectomes. My goal is to take over this project in order to add features and review its ergonomics.

## Keywords

connectome, visualisation, graph, python

**Entreprise**

Polytech



**Tuteur entreprise**

Jean-Yves RAMEL

**Étudiant**

Jean-Baptiste HUYGHE (DI5)

**Tuteur académique**

Jean-Yves RAMEL