Projet de programmation

CONCEPTION ET DÉVELOPPEMENT D'UNE APPLICATION WEB DE VISUALISATION DE CARTES MÉTABOLIQUES

Réalisé par

Marine ALVES DE BARROS - Mathieu BOLTEAU
Mélanie CARRIAT - Solweig HENNECHART

Master 1 mention Bioinformatique - Université de Bordeaux

Client : Gencovery - *Adama OUATTARA*

Sommaire

Introduction

Le métabolisme est l'ensemble des interactions moléculaires se produisant dans l'organisme. On parle alors de métabolites (molécules jouant le rôle de substrats ou de produits), d'enzymes (molécules qui catalysent une réaction) et de réactions (qui transforment un substrat en produit). La représentation de ce dernier est possible grâce à des modèles en bioinformatique.
Il existe plusieurs types de modélisations du métabolisme. Le premier basé sur l'interaction des molécules présentes dans l'organisme et utilisant la théorie des graphes pour analyser la nature des interaction grâce à la topologie. Le second, basé sur les relations stœchiométriques utilisant l'analyse des flux pour interpréter une nouvelle fois les interactions entre les molécules et les quantifier. Le dernier, quant à lui, se base sur le mécanisme réactionnel dont les paramètres cinétiques sont déterminés expérimentalement.
Le premier type de modélisation utilisant le concept de la théorie des graphes est celui se rapprochant le plus du résultat souhaité dans ce projet.
En effet, les graphes sont une modélisation mathématique capable de simplifier l'analyse d'un réseau métabolique complexe par le nombre de composants mis en jeu et par sa taille.
Les cartes du métabolisme ont donc un intérêt certain en biologie notamment grâce à cette notion de visualisation, et c'est pourquoi de nombreux logiciels comme Cytoscape ou Reactome fleurissent et sont en perpétuel développement. Cependant, ces logiciels ne prennent pas la notion d'espace tridimensionnel en compte. En effet, les représentations sont souvent en deux dimensions, et manquent d'informations pour certaines. C'est pourquoi, pour satisfaire aux besoins de la biologie, le développement d'un outil permettant la visualisation en trois dimensions de cartes métaboliques a été décidé dans le cadre d'un projet proposé par l'entreprise Gencovery. Il sera alors possible de visualiser des grands jeux de données stockés dans des fichiers de types JSON (*JavaScript Object Notation*[1]) en trois dimensions (3D). Cette application sera développée sur une interface web pour une meilleure facilité de visualisation, rapidité, et prise en main. Ce choix de développement se base sur une accessibilité, et une gestion optimale de l'évolution de l'outil.
Dans un premier temps, une analyse du sujet et des outils existants sera réalisée, puis la conception de l'outil sera évoquée. Enfin, la réalisation de ce dernier sera présentée, en évoquant d'éventuelles perspectives d'amélioration.
L'intégralité de l'application web implémentée durant le projet est disponible à l'adresse suivante : `https://github.com/mablt/M1_PDP_Project`.

1. Notation d'objet JavaScript

Chapitre 1

Analyse

1.1 Contexte

Le développement d'une application web permettant la visualisation en trois dimensions de cartes métaboliques a été proposé par Monsieur OUATTARA Adama, co-fondateur de l'entreprise Gencovery. Cette entreprise, basée à Villeurbanne, a pour but de mieux tirer parti des nombreuses données générées en recherches biotechnologiques et pharmaceutiques grâce aux "jumeaux numériques". Un jumeau numérique est une technologie s'adressant aux biologistes permettant la représentation numérique d'objets ou processus naturels biologiques.

1.2 Objectifs du projet

L'application à développer est un **outil de visualisation multidimensionnelle de flux et réactions métaboliques** à partir de l'importation de fichiers contenant des données biologiques réelles, disponibles sur des bases de données telles que **BiGG** [1] ou **BioModels** [2]. L'objectif pour l'entreprise est de mettre en place un service de visualisation (des interactions de flux métaboliques) au sein d'un cellule vivante, le tout sur le web pour une accessibilité optimale de l'outil, ainsi que de son évolution.

1.3 État de l'art

L'étude préliminaire des outils existants de visualisation d'interactions métaboliques est une phase capitale permettant de mieux définir les limites des outils actuels et les fonctionnalités nécessaires au développement d'un nouveau programme.

1.3.1 Modélisation de la visualisation des réseaux

Les systèmes biologiques peuvent être modélisés afin de mieux comprendre le métabolisme d'un organisme. L'étude de ce dernier peut se faire grâce à des modèles comme ceux servant à une analyse dynamique, ce qui est pertinent pour ce projet. Ils sont justifiés de "dynamiques" car ils permettent surtout l'étude des réseaux en tenant compte de valeurs quantitatives (métabolites, réactions, *etc.*) et de graphes. Cette notion de graphe est rencontrée lorsque le sujet de

la modélisation des voies métaboliques est évoqué.

Tout d'abord, il existe deux types de représentation de réseaux métaboliques [3], aussi appelés graphes (FIGURE 1.1), à savoir :

- les **graphes simples** :
  - le réseau de métabolites, où les noeuds représentent les molécules et les arrêtes représentent les réactions ;
  - le réseau de réactions, où les noeuds représentent les réactions et les arrêtes représentent les molécules. Ici, un métabolite est produit par une réaction et consommé par une autre. Il est produit et substrat en même temps ;
- les **graphes bipartis**, où les molécules et les réactions sont représentées par des noeuds et reliées entre elles par des arêtes.

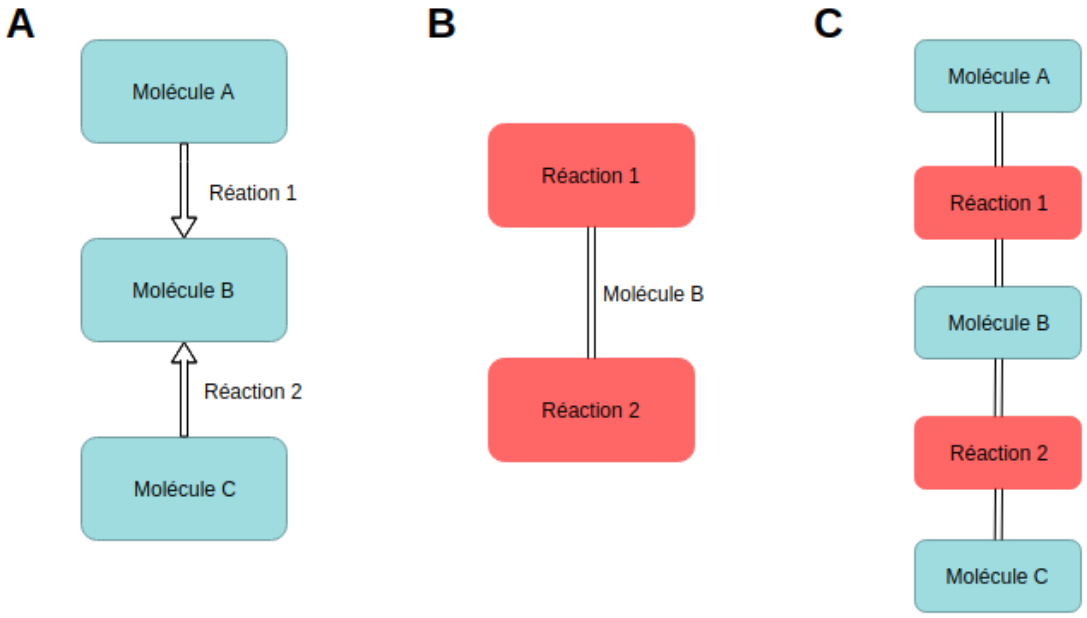
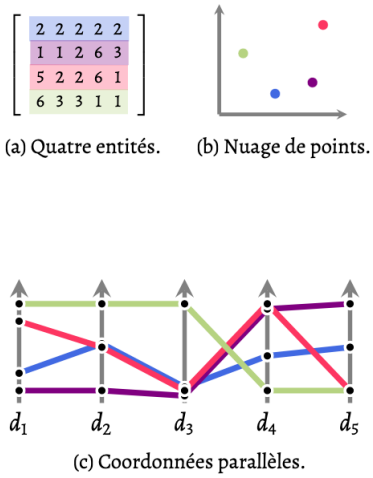


FIGURE 1.1 – Schéma représentant les différents types de réseaux métaboliques

A. Réseau de métabolites. B. Réseau de réactions. C. Réseau biparti.

En soi un graphe n'est rien de plus qu'une modélisation mathématique d'objets. Il se compose de noeuds et d'arêtes, que l'on choisit d'associer à des molécules ou/et des réactions, pour constituer un réseau métabolique. Il est donc nécessaire de choisir la meilleure représentation parmi celles existantes.

Une autre possibilité de modélisation est celle des coordonnées parallèles. C'est un système défini comme multidimensionnel permettant une représentation de données qui sont donc multivariées, par la présence d'une polyligne qui coupe autant d'axes parallèles qu'il y a de dimensions définies. Et ce, en donnant une position correspondant à son attribut pour la dimension coupée en question [4]. Cette notion assez complexe est illustrée par l'exemple suivant (FIGURE 1.2).



(a) Quatre entités. (b) Nuage de points.

(c) Coordonnées parallèles.

FIGURE 1.2 – Exemple montrant plusieurs représentations de données multivariées dont les coordonnées parallèles [4]

Pour représenter des données sur plusieurs dimensions, il faut travailler avec une matrice ou un tableau (a), dont les lignes représentent des entités de données et les colonnes leurs dimensions ou attributs. La modélisation des données se fait, à partir de cette matrice, et est représentée en nuage de points (b) ou coordonnées parallèles (c). Dans cette figure qui regroupe différentes représentations, une entité de données se définit par une couleur spécifique (rose, violet, bleu ou vert). Sur le nuage de points (b), seules deux ou trois dimensions sont visibles, il n'y a alors qu'un point pour chaque entité. Cependant, pour les coordonnées parallèles (c), chaque entité est représentée par une ligne qui se brise au niveau des axes à la valeur de leur attribut. Sur l'axe d1, quatre entités de différentes couleurs sont représentées, bleues, violettes, roses et vertes qui sont respectivement attribuées aux valeurs 2, 1, 5 et 6.

1.3.2 Annotation des réseaux en biologie et librairies de développement pour la visualisation

Annotation des réseaux en biologie

Le génome est souvent décrit comme le support d'informations de la cellule ou d'un organisme. Son étude permet de comprendre son origine, son fonctionnement, voir même son évolution future. Il est donc primordial de pouvoir annoter ces informations dans une voie métabolique afin de permettre une analyse plus évidente du fonctionnement moléculaire et cellulaire de l'organisme étudié. Les éléments à annoter seront donc fonctionnels (gènes, protéines, ARN, *etc.*), non fonctionnels (pseudo-gènes) et non codants (introns). Cela se fait grâce à des banques de données regroupant des données expérimentales et bibliographiques (NCBI [5], PubMed [6], UniProt [7]).

Il existe plusieurs niveaux d'annotations :

- l'annotation syntaxique : elle permet d'identifier des éléments biologiques d'intérêt sur la séquence d'ADN comme des signaux, des gènes d'ARN ou de protéines. Ceci permet d'avoir un aperçu de l'organisation des gènes et de leur localisation par exemple ;
- l'annotation fonctionnelle : avec laquelle on peut attribuer une fonction aux gènes identifiés dans le premier niveau d'annotation ;
- l'annotation relationnelle : qui décrit la relation potentielle entre les éléments identifiés

plus tôt, et de là leur possible interaction les uns avec les autres (réseaux métaboliques).

Lors de la modélisation de voies métaboliques, un problème se pose concernant la représentation des éléments impliqués dans le métabolisme cellulaire. En effet, une protéine peut par exemple se retrouver avec différentes dénominations selon si la recherche a été effectuée dans le domaine de la biologie ou biochimie, ou encore une catalyse imagée par un symbole dans un logiciel de visualisation et qui sera différent dans un autre. Pour pallier à cela et contrôler la multiplicité des termes, des ontologies (ensemble de termes et concepts représentant un champ d'informations par des métadonnées ou un domaine) ont été définies. Deux principales sont répertoriées : celle créée avec avec le consortium *Gene Ontology*[1] (GO) [8], et une autre dénommée *Systems Biology Ontology*[2] (SBO) [9].

La GO se caractérise par des concepts décrivant les gènes et leur(s) produit(s) :
- un nom de terme ;
- un numéro d'accession unique (préfixe "GO") ;
- une ontologie (les processus biologiques, les composants cellulaires ou les fonctions cellulaires) ;
- une définition ;
- des commentaires.

De plus, elle propose une base de données d'annotations du nom de GOA (*Gene Ontology Annotation*) [10] [11], sur les produits de gènes avec ses ontologies, à travers laquelle on peut naviguer à l'aide d'un moteur de recherche spécialisé, intitulé QuickGo [12].
La SBO, quant à elle, regroupe des concepts autour de la biologie des systèmes [13] [14] [15]. En plus des termes formalisant les gènes et les produits, elle intègre la possibilité d'annotation pour des modèles mathématiques systémiques, ce pourquoi elle est généralement utilisée . La SBO est utile pour les fichiers SBML (*System Biology Markup Language*[3]) (Section 1.3.3), et plus précisément pour les modèles mathématiques inclus dans ces fichiers, car elle permet d'annoter ces derniers. Son organisation se fait en sept vocabulaires différents :
- rôle des entités dans une réaction (ex. : substrat) ;
- paramètres quantitatifs (ex. : constante de Michaëlis) ;
- classification des expressions mathématiques du système (ex. : loi de masse) ;
- *framework*[4] de modélisation (logique, discret ou continu) ;
- nature de l'entité (ex. : macromolécule) ;
- type d'interactions ;
- type de représentation des méta-données du modèle.

Après avoir vu comment l'annotation des systèmes biologiques est contrôlée, il est intéressant d'aborder la question de la notation graphique de ces derniers.

En effet, les représentations de réseaux métaboliques ont également besoin d'être standardisées. Pour cela, il existe des notations graphiques comme KEGGmaps ou SBGN (*Systems Biology Graphical Notation*[5]) [15]. Cependant, certains logiciels de visualisation comme KEGG et Reactome ont fini par utiliser la notation SBGN, alors que CellDesigner l'utilisait déjà [16] [17]. Tous les symboles utilisés dans ce langage sont associés à un terme SBO. Ceci permet par exemple de générer des cartes métaboliques SBGN à partir de modèles SBML. Le SBGN est un langage gra-

1. Ontologie génétique
2. Ontologie de la biologie des systèmes
3. Langage de balisage de la biologie des systèmes
4. Infrastructure logicielle
5. Notation graphique de systèmes biologiques

phique qui vise à standardiser la représentation graphique. Il se compose de trois sous-ensembles de langages permettant de réaliser un type de réseau qui est appelé *map*[6] : *Process Description*[7] (SBGN-PD), *Activity Flow*[8] (SBGN-AF) et *Entity Relationship*[9] (SBGN-ER) (FIGURE 1.3).

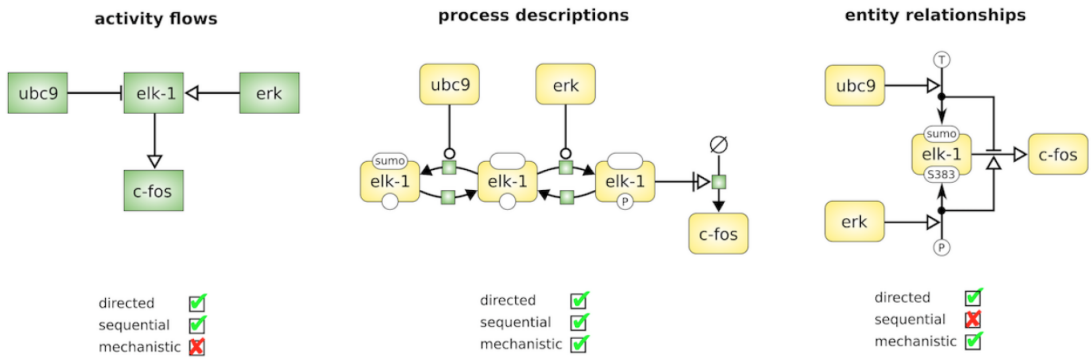


FIGURE 1.3 – Exemple de représentation des langages SBGN-PD, SBGN-AF et SBGN-ER. [18]

Un même système biologique est représenté par les différents langages composant le SBGN. Le langage "Activity Flows" (AF) montre le flux d'informations entre les entités biochimiques dans un réseau. Ça ne prend pas en compte l'état de transition des entités et est particulièrement pratique pour la représentation des effets de perturbations qu'elles soient génétiques ou environnementales. Le langage "Process Description" (PD), quant à lui, montre les interactions biochimiques d'un réseau dans le temps. Il peut être utilisé pour montrer toutes les interactions moléculaires d'entités biochimiques dans un réseau, avec une entité pouvant apparaître plusieurs fois dans le même diagramme. Enfin, le "Entity Relationships" (ER) représente les relations à laquelle une entité participe. Les relations peuvent être vues comme des règles décrivant l'influence des entités (noeuds) sur d'autres relations.

Le langage SBGN expose l'importance de la représentation graphique des éléments dans un graphe. Une réflexion à ce sujet sera nécessaire afin de choisir le mode de représentation des noeuds et des liens avec les librairies qui seront utilisées.

Librairies (spécialisées et non spécialisées) de développement pour la visualisation

Plusieurs librairies existent pour la visualisation. Certaines sont spécialisées dans la visualisation de réseaux, d'autres non. Il existe un certain nombre de librairies permettant la visualisation non spécialisée. De ce fait, seulement deux librairies seront abordées, à savoir Three.js et D3.js.

Three.js est une librairie en libre accès permettant de créer des scènes 3D dans un navigateur web. Elle utilise WebGL pour modéliser la 3D qui, utilisé comme base, est un système de bas niveau permettant de dessiner seulement des points, des lignes et formes basiques telles qu'un triangle. Cette librairie va donc permettre l'affichage des dessins 3D.

D3.js quant à elle, permet l'affichage de données sous une forme graphique et dynamique en 2D. Cette librairie, libre d'accès, utilise des fonctions pré-construites de JavaScript pour sélectionner des éléments, créer des objets, les styliser, ou y ajouter des transitions. L'idée principale de D3.js

6. Carte
7. Description du processus
8. Flux d'activités
9. relation d'entité

est de lier les données au DOM (*Document Object Model*[10]), et d'appliquer des transformations, basées sur les données, au document.

Concernant les librairies spécialisées pour la visualisation de réseaux, deux librairies seront évoquées : Cytoscape.js et 3D Force Graph.

Cytoscape.js est une librairie en libre accès permettant la visualisation de réseaux en 2D à partir d'un fichier JSON. Elle représente un réseau de métabolites. L'édition du réseaux est possible pour l'utilisateur. Cependant, les formats JSON utilisés pour l'import des données, sont différents de ceux que l'on retrouve dans les bases de données telles que BioModels.

3D Force Graph est une librairie de visualisation de réseaux en 3D. Elle utilise Three.js et donc WebGL pour l'affichage 3D, ce qui permet de faire de la visualisation 3D de tous types. Il est possible avec 3D Force Graph de créer des graphes composés de noeuds et d'arêtes. De plus, cette librairie, codée en JavaScript, peut être utilisée avec la technologie web. Également, elle est libre d'accès et de modifications. Il est aussi possible d'importer un fichier JSON avec cette technologie.

1.3.3 Formats de données en biologie systémique et logiciels de visualisation

Formats de données en biologie systémique

Deux principaux formats sont utilisés pour stocker les données en biologie systémique : BioPAX et SBML.
BioPAX (*Biological Pathway Exchange*[11]) est langage standard qui permet de faciliter les communications entre divers logiciels. Il établit un standard en terme de représentation des informations de réseaux métaboliques et signalétiques, d'interactions moléculaires et génétiques et de régulation génétique [19]. Il est basé sur le langage OWL (*Web Ontology Language*[12]), qui est en quelques sortes un standard pour définir des ontologies web structurées, en informatique.
SBML est un langage de balises qui est basé sur du XML (*Extensible Markup Language*[13]). Le document est découpé en éléments structurés hiérarchiquement. Chacun des éléments marqué par des balises, permettant ainsi de définir son type [20]. Ce langage est couramment utilisé dans les bases de données de réseaux métaboliques.
Un autre format, apparu plus récemment, est de plus en plus utilisé : le **JSON**. Ce dernier, lisible aisément par l'humain, reprend l'idée de hiérarchie présente dans le SBML. Néanmoins, l'utilisation de balises a été supprimée au profit de données textuelles dérivées de la notation des objets du langage JavaScript. Sa structure est une table de hachage(structure "clé/valeur").

Logiciels de visualisation

Cinq outils de visualisation ont été étudiés : deux de visualisation web (Reactome [21] et BioCyc [22]), et trois logiciels *open source*[14], téléchargeables (Path Visio [23], Cytoscape [24] et CellDesigner [25]). Leurs principales informations sont reportées dans le Tableau 1.1.

10. Modèle d'objet de document
11. Échange de voies biologiques
12. Langage d'ontologie web
13. Langage de balisage extensible
14. Libre d'accès

TABLE 1.1 – Récapitulatif des caractéristiques des différents outils de visualisation existants

Nom	**Caractéristiques**	**Avantages**	**Inconvénients**
Reactome	- En ligne, libre d'accès et de commercialisation - Dernière mise-à-jour : 16 Mars 2020 - Réseau métabolique - Visualisation 2D - Zoom avant/arrière - Déplacement sur le réseau	- Données présentes vérifiées - Informations complètes lors du clic sur un élément - Export d'une partie d'un réseau en SBML	- Réseaux disponibles pour seulement 16 espèces - Pas de chargement de fichiers possibles - Mouvement des noeuds impossible - Création/suppression de noeuds/arrête impossible
PathVisio	- En libre accès, libre de droits et de commercialisation - Dernière mise-à-jour : 22 Janvier 2018 - Réseau métabolique - Visualisation 2D	- Ajout de plugins possible - Visualisation quantitative possible - Lien vers des bases de données externes	- Zoom avant/arrière impossible - Déplacement impossible - Import de fichier SBML impossible - Visualisation d'interactions simples
Cytoscape	- En libre accès, libre de droits - Dernière mise-à-jour : - Visualisation 2D - Réseau biparti - Zoom avant/arrière - Déplacement sur le réseau - Librairie JavaScript pour des applications web	- Chargement de fichiers SBML possible - Sélection d'une zone à visualiser possible - Personnalisation possible des noeuds - Recherche d'un terme et affichage du noeud recherché en couleur possible - Informations lors du clic sur un élément - Ajout/Suppression de noeud ou d'arrête possibles - Enregistrement des modifications des noeuds possible (au format .cys)	- Chargement de fichiers JSON impossible
BioCyc	- En libre accès - Dernière mise-à-jour : 18 Décembre 2019 - Réseau métabolique - Visualisation 2D - Zoom avant/arrière	- Surlignage d'élément possible - Données présentes vérifiées	- Chargement de fichiers impossibles - Création/suppression de noeuds/arrête impossible

Cell Designer	- En libre accès, libre de droits - Dernière mise-à-jour : 20 Mai 2019 - Réseau métabolique - Visualisation 2D - Zoom avant/arrière - Déplacement sur le réseau	- Lien vers des bases de données externes (chargement de fichiers) - Chargement de fichiers SBML possible - Personnalisation possible - Mouvement des noeuds possible - Informations lors du clic sur un élément - Enregistrement des modifications des noeuds possible (au même format SBML) - Ajout/Suppression de noeud ou d'arrête possibles	- Chargement de fichiers JSON impossible

En somme, l'ensemble des outils étudiés permettent une visualisation 2D. Certaines fonctionnalités sont intéressantes, telles que le zoom ainsi que la possibilité de modifier le réseau (mouvement, suppression, ajout de noeuds et d'arêtes). Des outils proposent le chargement du réseau à partir d'un fichier, ce qui est demandé par le client. L'utilisation de Cytoscape, qui possède les fonctionnalités souhaitées pourrait être choisie. Or, ce dernier ne permet pas de visualisation 3D, importante pour le client. De plus, ce type de logiciel est désormais peu adapté aux technologies modernes. En effet, ces dernières se dirigent plutôt vers des applications web.

1.4 Cas concret : La glycolyse chez *Escherichia coli*

Dans le but d'illustrer les attendus, il a été décidé de s'intéresser au fichier JSON de la bactérie *Escherichia coli*. Ce fichier étant de taille assez importante, il a été réduit afin de conserver uniquement les métabolites et réactions impliqués dans la glycolyse, ce qui aboutit à un fichier composé de 21 métabolites et 12 réactions. Une représentation simple de ce qui est imaginé pour la représentation de ce fichier par la suite est représentée FIGURE 1.4.
Ce dernier est disponible sur le GitHub dans le dossier `test_data/`.
L'application devra donc permettre l'import de ce fichier, et afficher le contenu de ce dernier de façon claire et exploitable par l'utilisateur.
Cet exemple sera conservé tout au long de ce rapport.

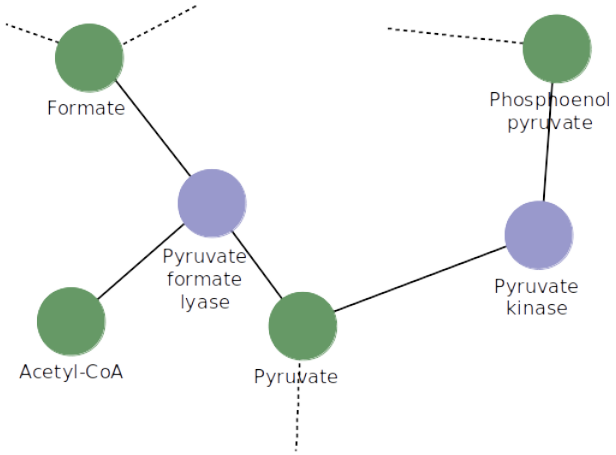


FIGURE 1.4 – Représentation simple du réseau métabolique de la glycolyse chez *Escherichia coli*.

Les disques représentent les métabolites (•) et les réactions(•). Les traits, représentent la liaison entre les réactions et les métabolites. Ceux en pointillés indiquent que le réseau métabolique se poursuit.

1.5 Contraintes de réalisation

L'application doit être développée en utilisant les langages de programmation JavaScript, HTML (*Hypertext Markup Language*[15]), et CSS (*Cascading Style Sheets*[16]). Elle doit permettre l'importation de fichiers au format JSON et la visualisation de leur contenu.
S'agissant d'un programme développé dans un cadre industriel, les modalités de licences des outils utilisés dans la conception doivent être clairement définies comme libres d'utilisation.

15. Langage de balisage hypertexte
16. Feuilles de style en cascade

Chapitre 2

Conception

2.1 Outils de développement

2.1.1 Langage

JavaScript

Le projet étant une application web, l'implémentation des fonctionnalités est demandée dans ce langage par le client, et ce, pour différentes raisons. Ce langage fut développé en 1995 par Brendan Eich qui s'est inspiré de plusieurs langages, mais en les simplifiant. Il est aujourd'hui largement utilisé pour le web car il est flexible et facile d'utilisation même pour des personnes peu habituées au développement, ce qui explique aussi pourquoi de nombreuses librairies additionnelles ont été développées, ajoutant encore à l'engouement de son utilisation. JavaScript est un langage "orienté-objet", qui permet de créer des objets se créant eux-mêmes grâce à la possibilité d'y inclure un constructeur, mais aussi ayant des propriétés propres. Certaines librairies de ce langage seront utiles, car elles incluent des fonctionnalités intéressantes pour le projet et sont analysées plus bas.

HTML

Ce langage fut développé au début des années 90 lors d'un projet collaboratif initié par Tim Berners-Lee et le Consortium World Wide Web (W3C). Il s'agit d'un langage de balisage majoritairement utilisé pour représenter des pages web. Il structure les informations (textuelles, *etc.*) à l'aide des différentes balises.

CSS

C'est un langage qui permet de gérer la présentation d'un document textuel à balisage (c'est donc le cas pour un document HTML, mais aussi d'autres comme les documents XML). Il commence à être développé au milieu des années 90 mais n'est couramment utilisé qu'à partir des années 2000.

JSDOC

C'est aussi un langage de balisage mais lui est utilisé directement dans un code source JavaScript. Des balises sont ajoutées au code, permettant de générer automatiquement par la suite

une documentation de celui-ci. Ce langage, comme le langage qu'il permet de commenter, se rapprochent d'autres langages de balisages pour documenter des codes sources tel que Javadoc.

2.1.2 Librairies

Three.js

C'est une librairie JavaScript sous licence MIT, permettant une libre utilisation et commercialisation, créée par Mr.doob (pseudo GitHub). Elle offre la possibilité de représenter des objets 3Ds dans une page web, grâce à une balise `<canvas>` de HTML.

3D Force Graph

La librairie 3D Force Graph s'appuie sur la librairie Three.js en proposant des fonctionnalités permettant la visualisation de graphes. Son auteur Vasco Asturiano, l'a développé sous licence MIT.

FontAwesome

Cette librairie collaborative sous licence libre de toute utilisation et commercialisation, est implémentée en JavaScript dans lequel a été ajouté du style en cascade. Elle permet notamment de pouvoir ajouter des icônes communes à une page web grâce à un balisage HTML.

2.2 Architecture

2.2.1 Entité à développer

Une architecture composée de deux niveaux, constituée d'un côté serveur et client, sera utilisée pour la conception de l'outil. Le client demande des ressources que le serveur fournit sans intermédiaire. Seule la partie client sera développée lors de ce projet. La partie serveur sera mise en place dans un second temps, par l'entreprise Gencovery. Concernant l'architecture logique de l'outil développé, cette dernière sera l'**architecture MVC** (*Model-View-Controller*[1]) (Figure 2.1). Elle est très souvent utilisée pour les interfaces graphiques et les applications web, car elle permet de diviser le programme en trois sous parties inter-connectées. Les trois entités qui la composent sont :

- le Modèle (*Model*), qui gère les données de l'application ;
- la Vue (*View*), qui s'occupe de l'affichage des données de l'application ;
- le Contrôleur (*Controler*), qui permet de gérer l'interaction utilisateur et de lier les deux entités précédemment présentées, les gardant ainsi séparées.

Durant ce projet, le côté client sera développé grâce à l'utilisation d'un serveur local, en s'inspirant de l'architecture MVC, séparant bien les données biologiques de la représentation graphique.

1. Modèle-Vue-Contrôleur

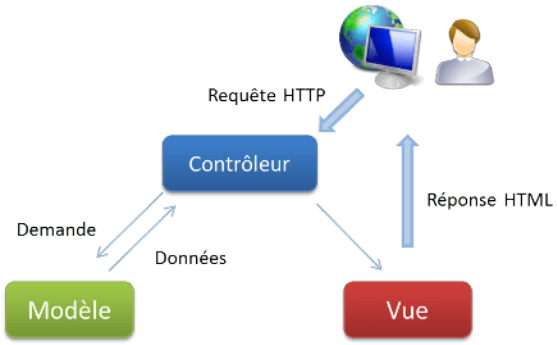


FIGURE 2.1 – Architecture MVC

Source : http ://www.webcky.fr/blog/2016/09/16/developpement-site-web-avec-larchitecture-mvc-en-isere/

2.2.2 Choix du type de graphe

L'utilisation de la méthode des coordonnées parallèles n'est pas envisageable car elle ne correspond pas aux standards actuels, ni aux habitudes des biologistes. Le choix de graphe classique est donc adopté.
L'utilisation de graphes simples dans cette application entraînerait un problème qui pourrait être appelé "multiplicité de liens". En effet, lors d'une réaction métabolique, plusieurs métabolites entrent en jeu, il n'est donc pas pertinent de les représenter sous forme d'arêtes.
Il a donc été décidé d'utiliser le modèle de **graphe biparti**, en représentant à la fois les réactions et les métabolites comme des noeuds. Ceci permet de faciliter la visualisation et d'éviter la multiplicité de liens, ainsi qu'un accès simplifié aux différentes données des réactions et des métabolites via un clic sur le noeud.

2.2.3 Conception des classes

L'application sera constituée de neuf types de classes principalement (Figure 2.2) :
- Le type carte : représenté par la classe `Map`, stockant un ou plusieurs graphes
- Le type graphe : représenté par la classe `Graph`, permettant de stocker tous les éléments composant le graphe, et de les mettre en relation
- Le type élément : représenté par la classe `Element`, dont héritent deux autres classes : `Relation` et `Entity`
- Le type réaction : représenté par la classe `Reaction`, héritant de la classe `Relation`
- Le type composé : représenté par la classe `Compound` héritant de la classe `Entity`, et représentant les métabolites
- Le type gène : représenté par la classe `Gene` héritant lui aussi de la classe `Entity` et représentant les gènes
- Le type réseau : représenté par la classe `Pathway`, contenant les informations propres au réseau étudié

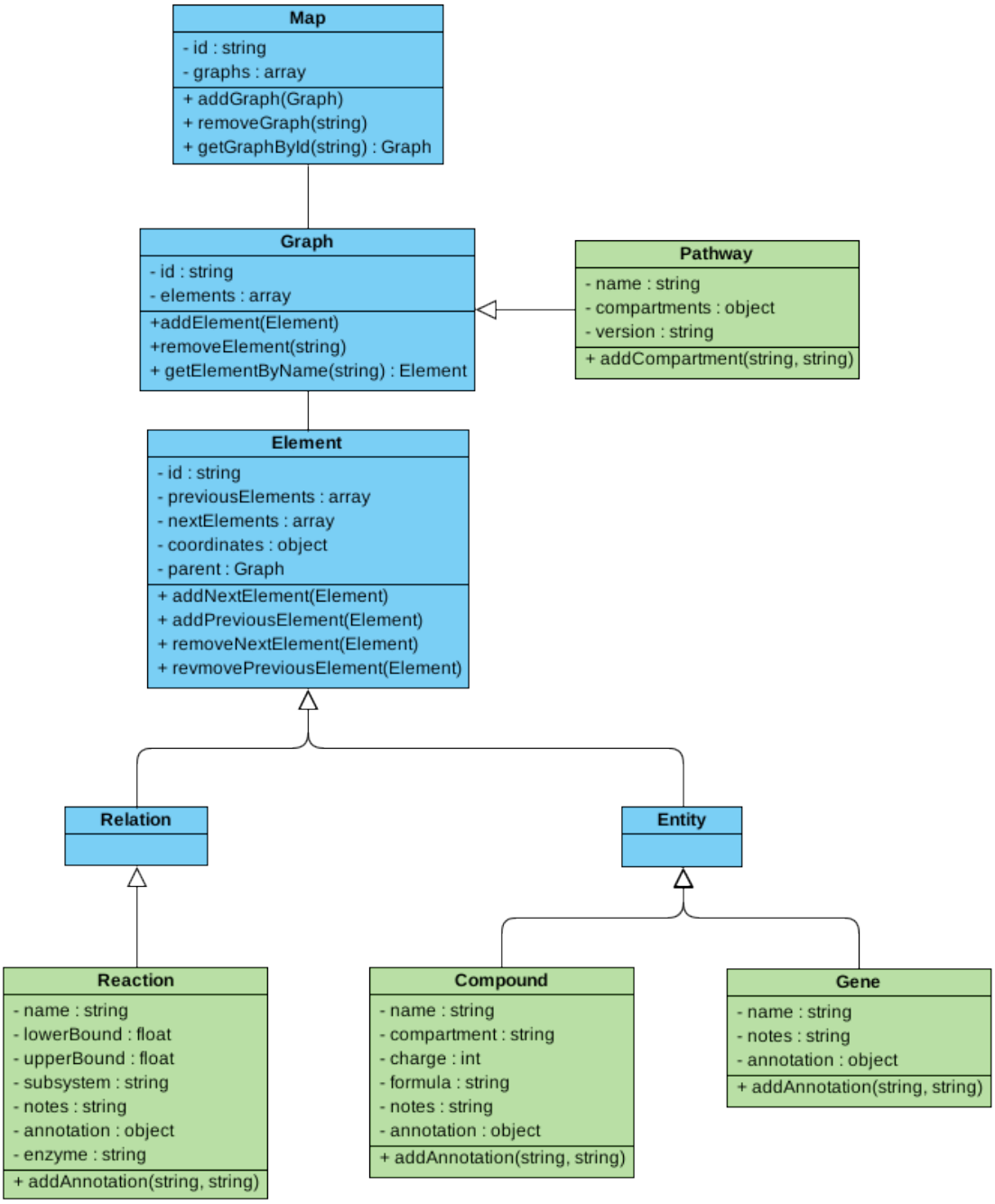


FIGURE 2.2 – Diagramme UML de l'application

Les classes en bleu, représentent la partie graphe alors que les classes en vert représentent l'aspect biologie. Sur cette figure, les accesseurs (getters et setters) ne sont pas représentés.

Une représentation en FIGURE 2.3 donne une idée des objets des classes `Compound` et `Reaction` attendus lors de l'instanciation des classes à partir du fichier JSON contenant les informations

relatives à la glycolyse chez la bactérie *Escherichia coli*. Ces objets représenteront ensuite les noeuds lors de la création du graphe.

Compound

```
id : "adp_c"
nextElements : […]
previousElements : […]
coordinates : {"x" : …,"y": …,"z": …}
parent : e_coli_core

name : "ADP C10H12N5010P2"
compartment : "c"
charge : -3
formula : "C10H12N5010P2"
notes : "adp_c"
annotation : {"bigg.metabolite": ["adp"],
       "biocyc": ["META:ADP"], … }
```

(a) Attendu de l'instanciation d'un objet `Compound`, à partir des données du premier métabolite du fichier.

Reaction

```
id : "PFK"
nextElements : […]
previousElements : […]
coordinates : {"x" : …,"y": …,"z": …}
parent : e_coli_core

name : "Phosphofructokinase"
lowerBound : 0.0
upperBound : 1000.0
subsystem : "Glycolisis/Gluconeogenesis"
notes : "PFK"
annotation : {"bigg.reaction": ["PFK"], "ec-
          code": ["2.7.1.11"], … }
enzyme : " ... "
```

(b) Attendu de l'instanciation d'un objet `Reaction`, à partir des données de la première réaction du fichier.

FIGURE 2.3 – Représentation des objets attendus pour le premier métabolite et la première réaction du fichier JSON

2.3 Lecture des fichiers

Il sera tout d'abord nécessaire de réaliser une lecture de fichiers de type JSON, les fichiers de type SBML seront traités si le temps le permet.
Une fonction permettra lorsqu'elle est appelée de lire un fichier et de renvoyer un objet, de type `Map`.

2.4 Proposition d'un visuel pour l'interface web

Une maquette de l'application a été réalisée afin d'avoir une représentation simple et visuelle du résultat attendu à la fin du projet (FIGURE 2.4). Cette dernière a été validée par le client.

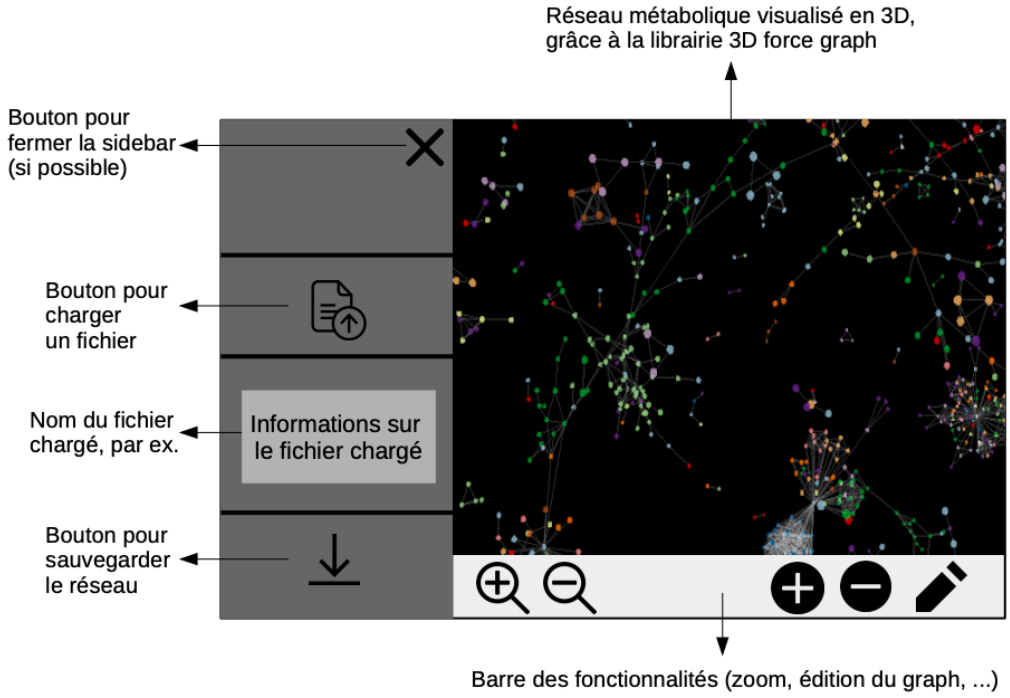


FIGURE 2.4 – Maquette de l'application

2.5 Partage des tâches

Le concept général de l'application ainsi que l'implémentation des classes et des fonctions seront réfléchis par tout le groupe lors de discussions quasi-quotidiennes grâce à divers outils permettant un contact constant avec chaque membre (Annexes A, B). Cela, grâce à un serveur Discord et ses salons textuels ou vocaux portant sur les tâches à accomplir. Il permettra de garder une trace écrite de l'ensemble du travail effectué, ainsi que les difficultés rencontrées. Les fichiers, quant à eux, seront déposés sur **GitLab** (via Git) afin que l'ensemble du groupe y ait accès.
Aussi, pour avancer rapidement sur l'application et se familiariser avec le travail en équipe, il a été décidé de travailler en binômes, donc une répartition du travail a été imaginée. L'un des groupes s'occupera de la création des classes, de la lecture et écriture des fichiers, puis du chargement et de la gestion des données. Tandis que l'autre, se chargera de la réalisation de l'interface web, ce qui englobe les fonctionnalités utilisateur rattachées à cette dernière et l'affichage permettant la visualisation du graphe.
Étant donné qu'une liaison entre la librairie JavaScript avec les données et la visualisation est requise, le groupe entier participera à la création des objets 3D Force Graph.

Chapitre 3

Réalisation

3.1 Création des classes

Les classes ont été implémentées selon le diagramme présenté précédemment (Figure 2.2). Pour chacune des classes, un constructeur a été défini permettant ainsi la création des objets. Les accesseurs (*getters* et *setters*) de chacun des attributs ont également été implémentés. Certaines méthodes ont été implémentées, cependant elles ne sont pas actuellement utilisées par l'application. Elles sont présentes dans le but de l'évolution de l'application et notamment la mise en place de nouvelles fonctionnalités. A noter que des valeurs par défaut sont présentes dans les constructeurs. Chacune des classes ont été implémentée dans un fichier indépendant. De plus, ces classes sont exportées, ce qui permet leur utilisation dans n'importe quel fichier JavaScript, via un import.

3.2 Lecture du fichier JSON

Lorsque l'utilisateur charge le fichier JSON qu'il souhaite visualiser, ce dernier est tout d'abord lu en tant que chaîne de caractères. Cette chaîne est ensuite stockée dans un objet JavaScript, ayant pour clé le nom du fichier (sans l'extension `.json`) et comme valeur la chaîne de caractères contenant le contenu du fichier chargé. Dans un second temps, le contenu du fichier, est transformé en objet JSON. Enfin, les objets (`Map`, `Pathway`, *etc.*) sont créés, d'après cet objet JSON. Il faut savoir que l'objet JSON est stocké dans un objet JavaScript, dans le but de simplifier l'enregistrement du graphe par la suite (Section 3.4). Cet objet est de type : `{ NomDuFichier : {objet JSON}}`.

3.3 Lien avec la librairie de visualisation

La librairie de visualisation 3D Force Graph se base sur des objets JavaScript pour créer les modèles. Ces objets sont de type : `{ nodes:[], links:[] }`.
Il a donc été nécessaire de créer une méthode permettant la transformation de l'objet `Pathway` en objet lisible par la librairie.
Pour ce faire, la méthode crée l'objet vide, et ajoute dans ce dernier, un nouvel élément noeud (*node*), pour chaque élément présent dans l'objet `Pathway`. À chaque élément, la classe est vérifiée. Si l'élément est une réaction (instance de la classe `Reaction`), des liens (*links*) sont également créés, entre la réaction et les éléments précédents, ainsi que les éléments suivants. Le

fait de créer les liens uniquement pour les réactions permet d'éviter les doublons.

Cas concret : La glycolyse chez *Escherichia coli*

Le script a premièrement été testé avec le fichier JSON réduit, correspondant à la glycolyse d'*Escherichia coli*, précédemment mentionné (Section 1.4). Un aperçu du résultat est visible en FIGURE 3.1.

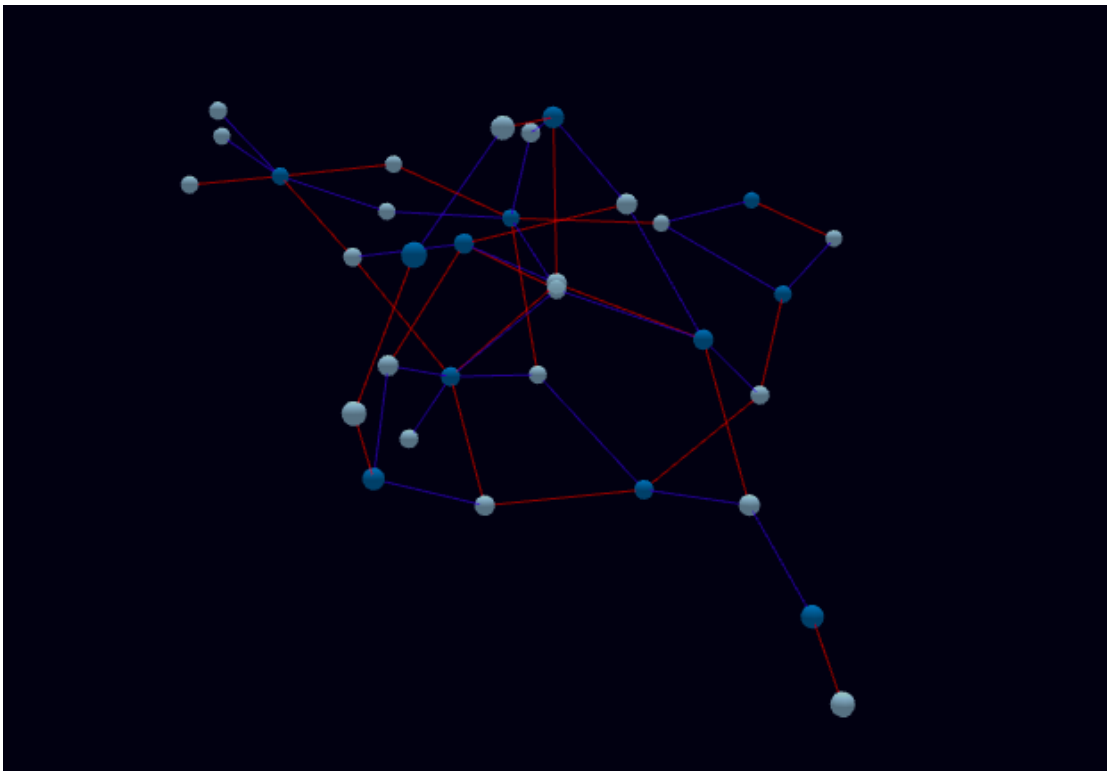


FIGURE 3.1 – Résultat du script permettant la visualisation du graphe avec la librairie 3D Force Graph.

Ce modèle tridimensionnel correspond à la glycolyse chez Escherichia coli. Les sphères claires correspondent aux métabolites, et les sphères foncées aux réactions. Quant aux liens, les rouges relient un réactif à une réaction, et les bleus relient une réaction à un produit.

Une fois le graphe affiché, il est possible de changer son orientation, et de déplacer les noeuds. Ceci permet une **édition du graphe** si l'utilisateur le souhaite.

3.4 Inclusion du graphe dans l'interface

L'interface web élaborée de manière indépendante, est visible en FIGURE 3.2. Elle permet l'inclusion du graphe précédemment élaboré sur la droite, et dispose également d'une barre latérale permettant l'import du fichier par l'utilisateur. Cette dernière est rétractable (en cliquant

sur la croix en haut à droite de la barre) afin d'augmenter l'espace accordé au graphe si besoin. D'autres fonctions sont également disponibles, et seront présentées plus tard.
Enfin, des informations concernant la manipulation sont visibles en bas de page. Elles donnent des instructions concernant le zoom et la rotation du graph.

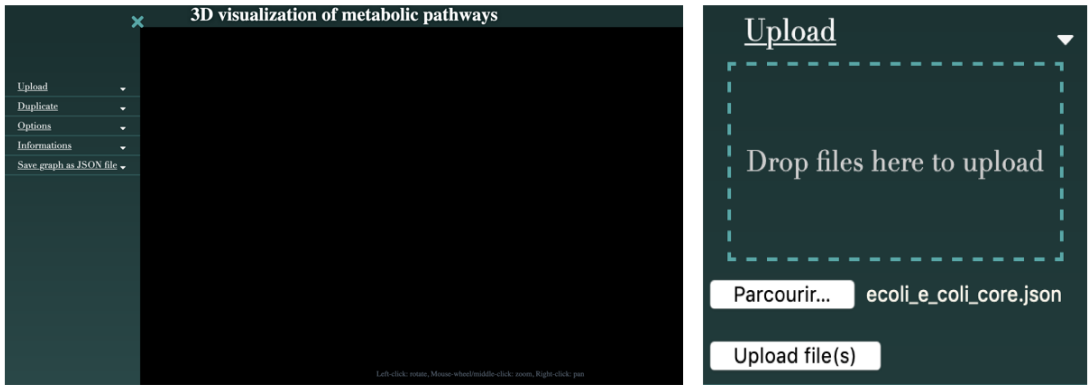


FIGURE 3.2 – Interface de l'application web et onglet "Upload"

L'interface finale avant le chargement d'un fichier JSON. On retrouve plusieurs onglets qui serviront, pour l'utilisateur, à afficher le réseau métabolique qu'il désire. Il pourra choisir le fichier qu'il veut lire grâce à l'onglet "Upload". Il devra parcourir ses fichiers avant de cliquer sur "Upload file(s)" qui enclenchera la lecture de ce dernier, et l'affichage du graphe correspondant directement au fichier importé.

Résultat final de l'affichage du réseau métabolique de l'organisme entier de la bactérie *Escherichia coli*

La FIGURE 3.3 donne un aperçu de l'affichage du réseau métabolique de *Escherichia coli* dont les informations étaient stockées dans le fichier `e_coli_core.json`, dans l'interface web. Ce fichier est disponible dans le dossier `test_data/`.

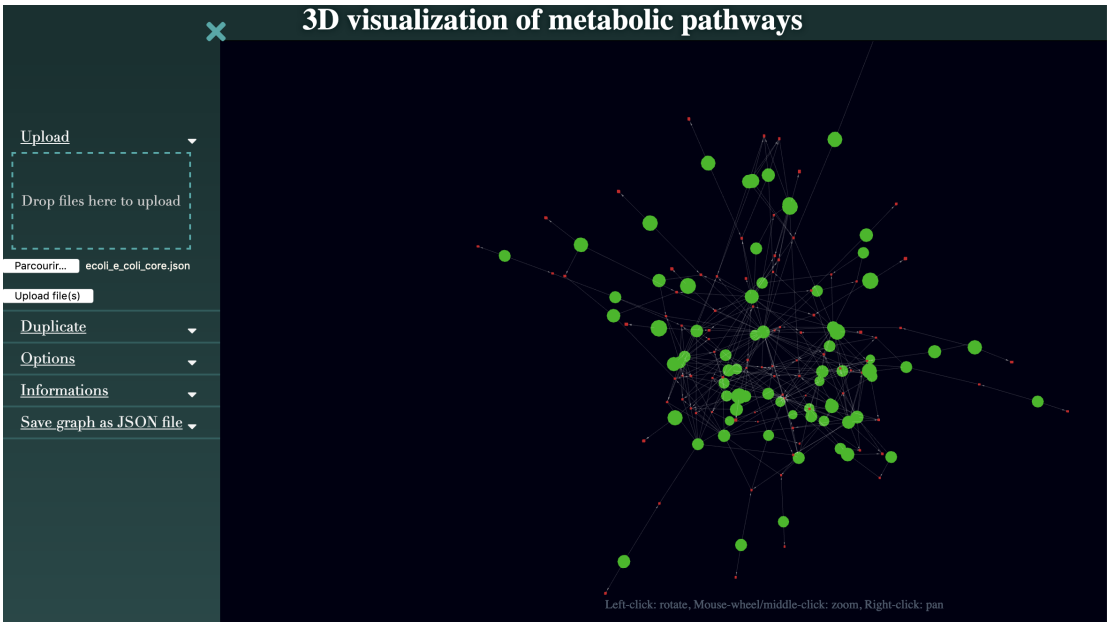


FIGURE 3.3 – Réseau métabolique de l'organisme Escherichia coli dans l'application web

L'interface finale après le chargement du fichier JSON contenant les informations de tout l'organisme Escherichia coli donne ceci. Un graphe dont l'affichage des réactions et des métabolites a été personnalisés, est visible.

3.5 Enregistrement des modifications du graphe (format JSON)

Une fois que l'utilisateur souhaite enregistrer le graphe, qu'il a éventuellement édité (positions des noeuds), il suffit de cliquer sur le bouton "Save graph" (FIGURE 3.4). Cela provoque la récupération de toutes les coordonnées de chacun des noeuds. Ces dernières seront ensuite modifiées dans l'objet JSON qui contient toutes les informations. À partir de cet objet, l'objet est converti en chaîne de caractères, puis est introduit dans un fichier. Le fichier est ensuite stocké dans le dossier "Téléchargements" de l'ordinateur de l'utilisateur. Le fichier a pour nom le même nom que le fichier initialement chargé dans l'application, avec une extension `"_GENCOVERY"` ajoutée. Cette extension est modifiable simplement en changeant une variable constante.

FIGURE 3.4 – Onglet "Save graph as JSON file" de l'interface de l'application web

Cet onglet permet d'enregistrer les nouvelles modifications du graphe grâce au bouton dédié à cela. L'utilisateur peut alors enregistrer le nouveau graphe sous forme d'un fichier JSON.

3.6 Réalisations supplémentaires

3.6.1 Affichage de graphes multiples

L'application a la possibilité d'afficher plusieurs fichiers JSON afin de comparer le métabolisme de plusieurs organismes simultanément, sur la même page (Figure 3.5). L'utilisateur charge les fichiers qu'il souhaite visualiser dans la zone prévue à cet effet.

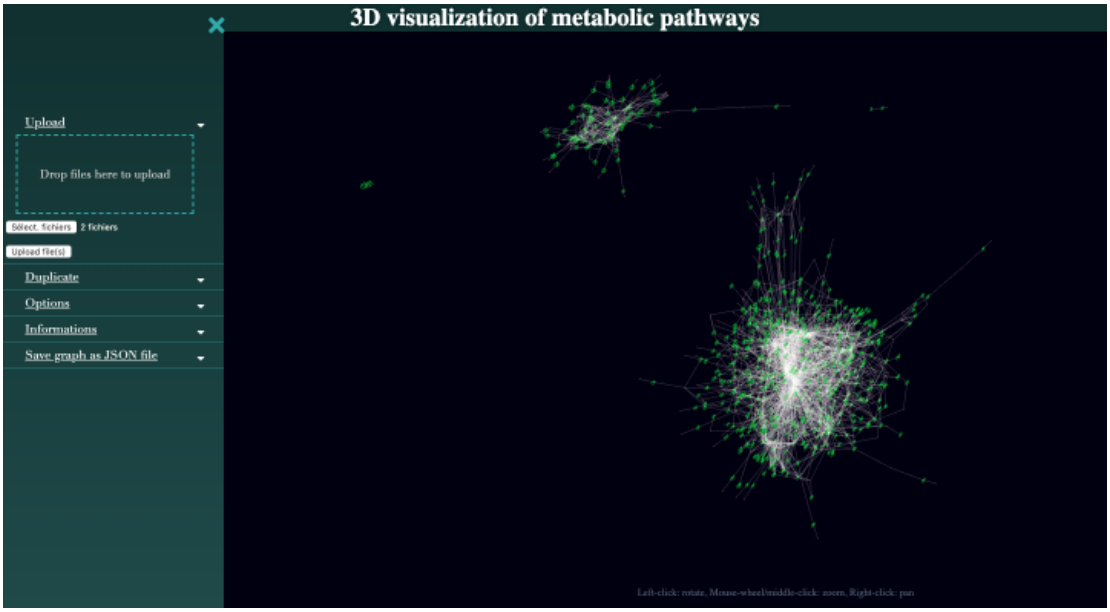


Figure 3.5 – Capture d'écran d'une visualisation d'un multi-graphe

Deux fichiers JSON ont été chargés : le fichier JSON d'Escherichia coli et le fichier JSON de Thermotoga maritima.

Chargement

L'application a été organisée de façon à ce que plusieurs graphes puissent être affichés en même temps. En effet, l'objet `Map`, contient une liste de graphes (instances de la classe `Graph`). Le chargement des fichiers fonctionne de la même manière que pour un seul fichier. Le code a été implémenté de façon à gérer une liste de fichiers. De ce fait, plusieurs objets JSON sont stockés dans l'objet JavaScript. À partir de ces objets, les graphes sont créés.

Création de l'objet 3D Force Graph pour plusieurs graphes

La fonction servant de lien entre les graphes et la librairie 3D Force Graph est adaptée au chargement de plusieurs fichiers. En effet, lors de l'ajout des noeuds à la liste, l'id des objets est modifié. en y ajoutant comme extension le nom du graphe auquel il appartient. De ce fait, il est possible de différencier les noeuds et, *a fortiori* les graphes.

Enregistrement

De la même manière qu'avec l'affichage mono-graphe, la sauvegarde des graphes récupère tout d'abord l'ensemble des coordonnées des noeuds, puis les modifie dans les objets JSON. Les

fichiers sont ensuite téléchargés et stockés dans le dossier "Téléchargements", tout comme le chargement d'un seul fichier.

3.6.2 Duplication des noeuds

La visualisation implémentée, met en évidence un problème de lisibilité du graphe car plusieurs liens pointent parfois sur le même noeud. C'est surtout problématique pour certains cofacteurs impliqués dans un grand nombre de réaction tel que H^+, CO_2, H_2O, ADP, ATP.
Pour résoudre ce problème de lisibilité de la visualisation graphique, il est proposé à l'utilisateur de pouvoir les **dupliquer**. Avec cette fonctionnalité les cofacteurs sélectionnés sont dupliqués dans la construction du graphe jusqu'à avoir un noeud pour chaque lien le concernant.

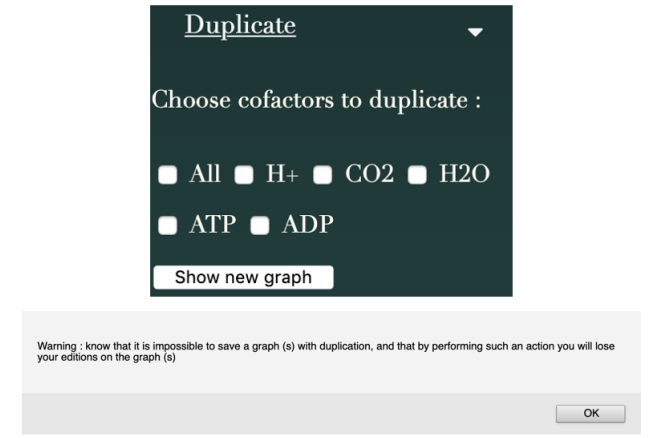


FIGURE 3.6 – Duplication des noeuds grâce à l'onglet "Duplicate" de l'application web

L'onglet "Duplicate" permet de choisir si l'on souhaite dupliquer certains noeuds, et donc des cofacteurs (H^+, CO_2, H_2O, ADP, ATP, ou tous). En effet, lorsque l'on charge un graphe, toutes les réactions ou métabolites comprenant un cofacteur qui apparaît souvent seront reliés à ce dernier rendant la lisibilité du réseau assez problématique. L'apparition d'un message est déclenché à chaque clic sur cet onglet, afin de prévenir l'utilisateur que la duplication d'un cofacteur ne pourra pas permettre l'enregistrement du graphe.

Cette fonctionnalité est pilotée directement par l'utilisateur (FIGURE 3.6). Il doit choisir s'il veut dupliquer des cofacteurs (tous ou une sélection). Une fois ce choix fait via l'onglet dédié dans l'interface, une liste de cofacteurs est construite. Un objet `Cofactor` contient l'id du métabolite correspondant, son nom et le nombre de fois où le métabolite est présent dans les réactions de la voie visualisée.

Lorsqu'un ou plusieurs cofacteurs sont dupliqués, le fichier JSON n'est pas affecté. Cela a seulement un impact sur la création de l'objet 3d force, soit le graphe visible, lors de la création des noeuds.
Pour une problématique de justesse biologique, il n'est pas possible de sauvegarder le graphe avec la duplication des cofacteurs. Il ne s'agit que d'une option de visualisation graphique des données chargées.

3.6.3 Modification de l'apparence de la visualisation graphique

Cette fonctionnalité a été ajoutée dans le but d'offrir plus d'interaction utilisateur, pour qu'il puisse choisir lui-même l'apparence de la visualisation, mais aussi visualiser des modifications sur le graphe courant (FIGURE 3.7). Cependant, contrairement aux informations d'édition sur le graphe (déplacement des noeuds), les modifications de l'apparence ne sont pas enregistrées lors de la sauvegarde du fichier. En effet, il n'y a aucun interêt biologique à conserver la personnalisation de l'affichage.

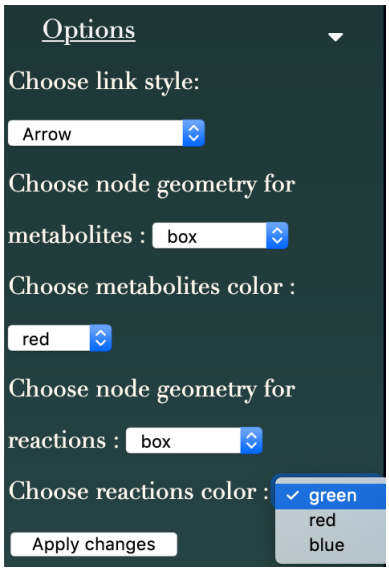


FIGURE 3.7 – Personnalisation de l'apparence du réseau métabolique

L'onglet "Options" permet de personnaliser la forme des noeuds et des liens du graphe.

Les noeuds

Le type de graphe choisi étant le graphe biparti, il est nécessaire de différencier les deux types de nœuds afin de faciliter la visualisation. Mais la possibilité de laisser le choix de l'apparence à l'utilisateur est aussi un point intéressant. Il a donc été décidé de proposer une combinaison déterminée par l'utilisateur parmi un certain nombre d'icônes tridimensionnels présents dans la librairie Three.js (FIGURE 3.8), mais aussi de couleurs, tout en différenciant toujours les représentations des réactions et des métabolites par une taille différente. Ainsi, les deux composantes du graphe se distinguent de part la taille : les réactions étant toujours plus petites que les métabolites quelque soit la combinaison choisie par l'utilisateur.

FIGURE 3.8 – Choix de formes possible pour les noeuds du réseau métabolique

Voici les trois formes que peut choisir l'utilisateur pour les noeuds correspondant aux métabolites ou aux réactions : "sphere", "box", "torusknot". Il peut également choisir une couleur pour les noeuds entre le rouge, le vert et le bleu.

Les liens

Lors de la visualisation d'un réseau, il est nécessaire d'afficher le sens des réactions, autrement dit, les réactifs et produits de chaque réaction. Pour cela, deux fonctions ont été mises en place :

- un affichage des liens sous forme de flèches : les flèches pointant sur les réactions représentent des liens "réactif - réaction" et les flèches pointant sur des métabolites représentent des liens "réaction - produit".
- un affichage des liens sur lesquels se déplacent des microparticules, représentant les flux. Lorsque sur un lien les particules se déplacent vers une réaction, c'est un lien "réactif - réaction", et lorsqu'elles se déplacent vers un métabolite, c'est un lien "réaction - produit".

Un aperçu de ces deux affichages est disponible en FIGURE 3.9

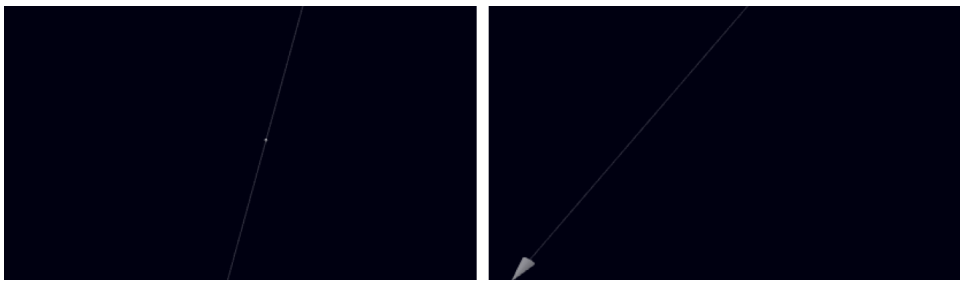


FIGURE 3.9 – Choix de formes possible pour les liens du réseau métabolique

Il existe deux formes que l'utilisateur peut choisir pour les liens qui relient les métabolites aux réactions. Il peut y avoir une flèche qui indique le sens de la réaction, ou des particules qui se dirigent le long d'un axe également dans le sens de la réaction.

L'utilisateur peut choisir l'affichage désiré grâce à un menu déroulant.

3.6.4 Convertisseur SBML vers JSON

Il a été décidé de réaliser un convertisseur indépendant, qui sera ajouté plus tard comme module de l'application web.
Celui-ci est codé en langage Python 3 et dispose pour le moment d'une interface graphique simple (visible en FIGURE 3.10).

FIGURE 3.10 – Aperçu du convertisseur SBML vers JSON

Il permet à l'utilisateur d'importer un fichier au format SBML (*level*[1] 3 uniquement). Une vérification du type de fichier permet l'affichage d'une erreur dans le cas ou le *level* est inférieur à 3, ou dans le cas où le fichier n'est pas au format SBML. Une fois le fichier importé, ce dernier est traité de la manière suivante :

- Récupération des différents éléments dans des listes de dictionnaires (Réactions, Métabolites, Gènes, Compartiments).
- Rassemblement de ces éléments dans un seul dictionnaire.
- Conversion du dictionnaire obtenu en fichier JSON (grâce à la librairie json).

Il faut cependant noter que certaines informations présentes dans les fichiers JSON ne sont pas présentes dans les fichiers SBML, comme par exemple, les données *lowerbound* et *upperbound* (bornes des flux métaboliques des réactions), qui seront par défaut aux valeurs -1000 et 1000 ou encore, les sous systèmes auxquels appartiennent les réactions. Ces informations ne pourront donc pas être affichées lors d'un clic sur un noeud dans l'application.

3.6.5 Drag and drop

Pour plus de facilité pour le choix des fichiers et afin d'éviter à l'utilisateur de parcourir l'arbre des répertoires sur son ordinateur à la recherche des fichiers qu'il souhaite visualiser, l'idée de faire un espace glisser-déposer avait été émise. De manière indépendante, cet espace réagit bien lorsqu'il est seul sur une page HTML. Cependant, lors de son intégration à l'application il ne réagit plus à la présence de fichiers sur l'espace de dépôt qui leur est dédié. Le glisser-déposer est assez complexe à intégrer à une application web car il met en jeu le côté serveur. Il peut fonctionner sur certains ce qui n'est pas le cas sur d'autres. Son intégration devra être faite plus tard, en accord avec le serveur qui sera utilisé. Une fois cela fait, il devra être relié à l'application afin de récupérer les fichiers pour qu'ils soient ensuite traités. Cet outil fonctionnera de la même manière et en parallèle de l'outil actuellement utilisé pour importer les fichiers.

1. Niveau

Conclusion

Le but de ce projet était le développement d'une interface web permettant de visualiser des cartes de réseaux métaboliques en trois dimensions pour répondre aux besoins de la biologie et étudier le métabolisme différemment qu'en deux dimensions pour une meilleure lisibilité.
L'outil développé est fonctionnel, et permet la visualisation tridimensionnelle des réactions métaboliques à partir de l'importation de fichiers JSON, contenant les informations qui permettent la construction du graphe désiré. Le chargement des données est basé sur des fichiers de type `.json`, mais pourra aussi se faire à partir de fichiers `.xml` (SBML *level* 3) grâce à un convertisseur qui transformera ce dernier en JSON. Il est possible de visualiser plusieurs graphes en même temps comme l'application permet un chargement multiple de fichiers. Que ce soit donc un ou plusieurs graphes, ils sont directement modélisés sur l'interface web. La rapidité de l'affichage dépend des formes des noeuds choisies par défaut. Actuellement, la fluidité n'est pas optimale mais elle peut être améliorée en choisissant des formes plus simples telles que des sphères, ou en utilisant des images plutôt que des formes en trois dimensions. Comme ce sont des graphes bipartis, les nœuds représentent donc des réactions et des métabolites et les liens de simples liaisons entre ces derniers. Il est possible de les personnaliser : couleur et forme pour les nœuds, ainsi que flèches ou "flux de particules" pour les liens.
La dimension 3D offre la possibilité de naviguer dans et autour du graphe, en zoomant plus ou moins sur les endroits désirés rendant son étude plus dynamique. Ce développement permet donc une visualisation des cartes de réseaux métaboliques, et permet des perspectives d'évolution. En effet, plusieurs points peuvent être améliorés comme ajouter directement le convertisseur (SBML à JSON) dans l'interface de façon à ce que le chargement d'un fichier SBML permettent l'affichage d'un graphe directement. D'ailleurs, lors d'un chargement de plusieurs fichiers, attribuer un halo lumineux d'une couleur spécifique à un graphe au niveau des noeuds permettrait de mieux visualiser les différents graphes.
À ce jour, les informations sur les gènes présents dans les fichiers JSON ne sont pas utilisées. Seule la création des objets `Gene` est faite. Il est prévu par la suite, une poursuite du développement afin d'afficher ces informations dans l'application.
L'affichage de représentations en 3D des molécules serait un plus. Ces représentations seraient visibles en cliquant sur un nœud, ce qui déclencherait l'ouverture d'une fenêtre dans laquelle ces dernières pourraient être manipulées et permettrait donc étudier directement les molécules du métabolisme en question. Mettre en place des méthodes pour enlever, afficher et ajouter des nœuds dans le respect de la cohérence en biologie serait intéressant. Mais aussi, afficher la réaction à laquelle appartient un nœud du type "A + B $\longrightarrow$ C" lors d'un clic sur ce dernier, ou encore mettre en place une zone glisser-déposer fonctionnelle avec le serveur qui hébergera l'application pour faciliter le chargement de fichiers côté utilisateur.
Les possibilités d'évolution sont nombreuses, cependant la prochaine étape majeure serait de mettre l'application sur un serveur, permettant ainsi une mise en service fonctionnelle avant de choisir et développer des fonctionnalités nouvelles.

Références

[1] Base de données BiGG. https://http://bigg.ucsd.edu/. [En ligne ; accessible le 04-03-2020].

[2] Base de données BioModels. https://www.ebi.ac.uk/biomodels-main/. [En ligne ; accessible le 04-03-2020].

[3] Identification de motifs dans les réseaux métaboliques. https://tel.archives-ouvertes.fr/tel-00195401/PDF/these.pdf/. [En ligne ; accessible le 05-04-2020].

[4] Gaëlle Richer. Passage à l'échelle pour la visualisation interactive exploratoire de données : approches par abstraction et par déformation spatiale. https://tel.archives-ouvertes.fr/tel-02453395/document. [En ligne ; accessible le 17.04.2020].

[5] Base de données ncbi. https://www.ncbi.nlm.nih.gov/. [En ligne ; accessible le 18-05-2020].

[6] Base de données pubmed, appartenant à ncbi. https://www.ncbi.nlm.nih.gov/pubmed/. [En ligne ; accessible le 18-05-2020].

[7] Base de données uniprot. https://www.uniprot.org/. [En ligne ; accessible le 18-05-2020].

[8] Consortium gene ontology. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3037419/?tool=pubmed. [En ligne ; accessible le 18-05-2020].

[9] Règles de cohérence pour l'annotation génomique : développement et mise en oeuvre in silico et in vivo. https://tel.archives-ouvertes.fr/tel-00350902/document. [En ligne ; accessible le 12-04-2020].

[10] The goa database : Gene ontology annotation updates for 2015 = "https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4383930/", note = "[en ligne ; accessible le 12-04-2020]".

[11] Gene ontology annotation (goa) database. https://www.ebi.ac.uk/GOA/index. [En ligne ; accessible le 12-04-2020].

[12] Moteur de recherche quickgo. https://www.ebi.ac.uk/QuickGO/. [En ligne ; accessible le 05-04-2020].

[13] Systems biology ontology = "http://www.ebi.ac.uk/sbo/main/static;jsessionid=A468FD4C6208129F816E6F3922B24A9D?page=FAQ", note = "[en ligne ; accessible le 12-04-2020]".

[14] Methodes qualitatives pour la construction et l'analyse des reseaux moleculaires sbgn = "https://tel.archives-ouvertes.fr/tel-01391465/document", note = "[en ligne ; accessible le 12-04-2020]".

[15] Regles de coherence pour l'annotation genomique = "https://tel.archives-ouvertes.fr/tel-00350902/document", note = "[en ligne ; accessible le 12-04-2020]".

[16] Conversion of kegg metabolic pathways to sbgn maps including automatic layout = "https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-14-250", note = "[en ligne ; accessible le 12-04-2020]".

[17] Sangdun Choi. Systems biology for signaling networks. https://books.google.fr/books?id=-cnVcd5X4oEC&printsec=frontcover&dq=Systems+Biology+for+Signaling+Networks&hl=fr&sa=X&ved=0ahUKEwit2u2C0MjpAhVBzRoKHRWZCkYQ6AEIKDAA#v=onepage&q=Systems%20Biology%20for%20Signaling%20Networks&f=false. [Publié en 2010 - p172].

[18] Le langage sbgn. https://sbgn.github.io/. [En ligne ; accessible le 17.04.2020].

[19] Demir E. *et al.* The BioPAX community standard for pathway data sharing. *Nature Biotechnology*, 28(9) :935–942, September 2010. Number : 9 Publisher : Nature Publishing Group.

[20] Beurton-Aimar M. SBML, langage de modélisation des réseaux biochimiques. In *Biologie Systémique - Standards et modèles pour la biologie*, Collection Écrin, page 288. 1 edition, October 2007.

[21] Reactome. Outil de visualisation Reactome. https://reactome.org/. [En ligne ; accessible le 02-04-2020, dernière MAJ ; version 72 12/03/2020].

[22] Karp P. *et al.* The BioCyc collection of microbial genomes and metabolic pathways. *Briefings in Bioinformatics*, 20(4) :1085–1093, July 2019. Publisher : Oxford Academic.

[23] Kutmon M. *et al.* PathVisio 3 : An Extendable Pathway Analysis Toolbox. *PLOS Computational Biology*, 11(2) :e1004085, February 2015. Publisher : Public Library of Science.

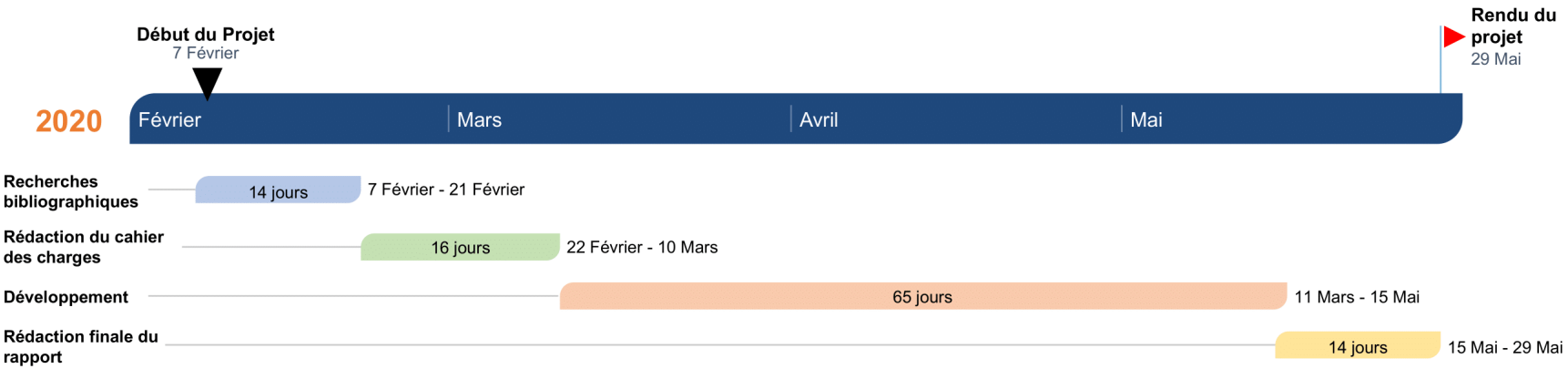
[24] Cytoscape Consortium. Logiciel de visualisation Cytoscape. https://cytoscape.org/. [En ligne ; accessible le 02-04-2020, et téléchargeable version 3.7.2].

[25] Akira Funahashi, Mineo Morohashi, Hiroaki Kitano, and Naoki Tanimura. CellDesigner : a process diagram editor for gene-regulatory and biochemical networks. *BIOSILICO*, 1(5) :159–162, November 2003.

Annexe A

Échéancier du projet

Échéancier du projet



Annexe B

Planning prévisionnel du développement

Déroulement du développement (Prévisionnel)