université de BORDEAUX

College Science and Technology

Cahier des charges

Développement d'un outil client pour la visualisation de l'annotation fonctionnelle de données protéomiques

Auteurs :

Julie Blasquiz

Romain Luck

Marc Mongy

Michèle Raffaelli

Encadrants : Patricia Thebault, Karine Dementhon

Bordeaux

26-02-2018

LaTeX

Table des matières

1 Contexte et état de l'art

Dans le cadre d'une étude sur les *Candida* 3 espèces vont être étudiées. Elles ont été sélectionnées en fonction de leur différence en terme de stratégies de pathogénicité chez l'Homme. L'objectif de cette étude est de comparer par une analyse protéomique, les mécanismes de survie de ces espèces phagocytées dans les macrophages.
Les expériences reposent sur une infection *in vitro* de macrophages avec les pathogènes d'intérêt. En contrôle, les levures seules en absence des macrophages sont utilisées. Les extraits protéiques de levures internalisées ou seules sont préparés. Les protéines sont identifiées et quantifiées par chromatographie liquide couplée à de la spectrométrie de masse en tandem (LC-MS/MS), afin de sélectionner les protéines statistiquement dérégulées en condition de phagocytose macrophagique. Les résultats préléminaires sont obtenus par spectomètrie de masse en tandem afin de sélectionner des protéines statistiquement dérégulées. Entre 400 et 1000 protéines, selon l'espèce Candida, sont sélectionnées.
L'analyse de ces protéines ce fait manuellement par BLAST afin d'attribuer une fonction putative à chaque protéine dérégulée.
Le projet de stage de l'étudiante en Master 2 Microbiologie Immunologie est centré autour du lien intéractions moléculaires et comportements communs ou atypiques

chez les levures étudiées. Devant de telles quantités de données à traiter il semble évident qu'un traitement manuel est trop fastidieux. C'est pourquoi une solution faisant appel à des outils bioinformatiques va être proposée. Ces outils existent déjà mais il faut les chercher séparèment sur les différentes plateformes correspondantes. L'intérêt de ce projet, résumé en figure 1, est donc de proposer, sous forme d'une page Internet, un accès aux différentes informations nécessaires à l'étude menée (sur la base d'une Gene Ontology rentrée par le biologiste sur la page Internet).

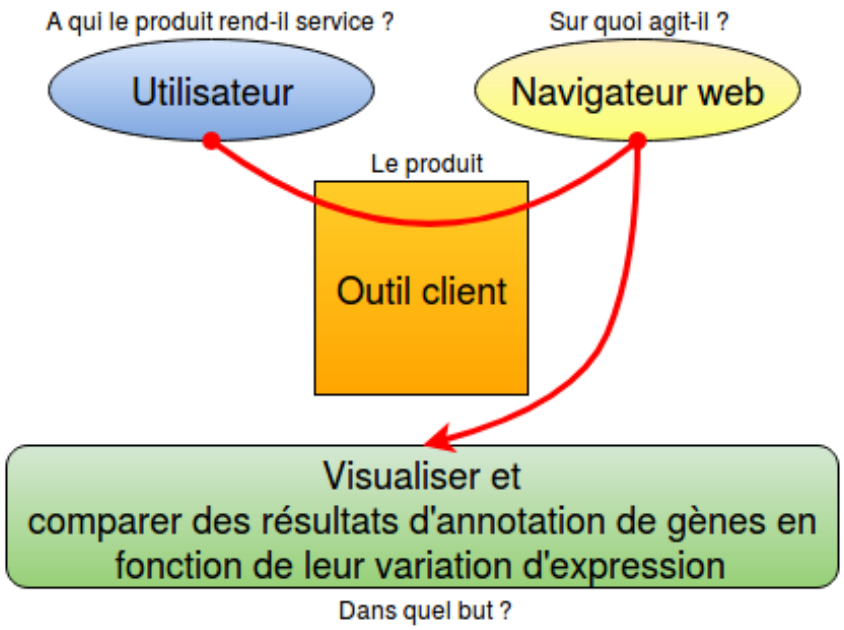

FIGURE 1 – Boîte à cornes représentant la demande du client

Ainsi il peut avoir un accès simple, clair et rapide à toutes les informations qu'il désire en un clic.

En alliant ainsi biologie et informatique il est ainsi possible de faire gagner un temps précieux au biologiste et de fournir un outil de recherche qui aura pour vocation d'être réutilisable dans un autre contexte également (nécessité de créer un outil qui ne soit pas ultra spécifique mais qui permette toutefois de répondre précisément à une question posée). De ce fait il est important de travailler en étroite collaboration avec les biologistes.

2 Besoins fonctionnels

L'outil client développé ici dans le cadre du projet de programmation doit répondre à certains besoins. L'ensemble de ces besoins est résumé en figure 2. On distingue les besoins fonctionnels (en vert sur la figure 2) des besoins non fonctionnels (en violet sur la figure 2).

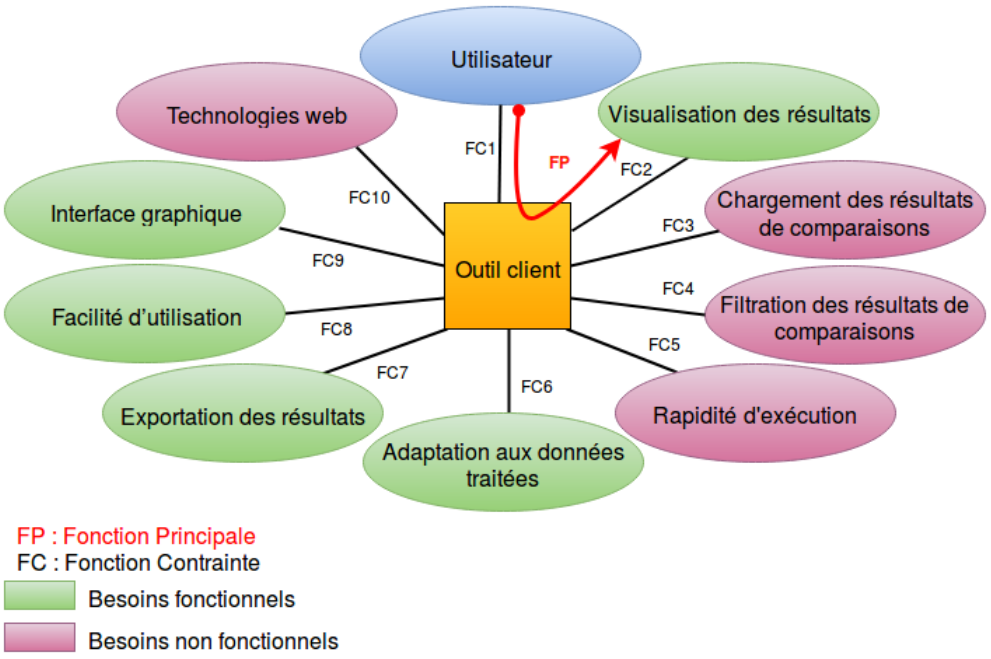

FIGURE 2 – Diagramme pieuvre représentant les besoins de l'outil client

Les besoins fonctionnels sont définis comme étant des fonctionnalités demandées spécifiquement par le client et dont l'action peut être visualisée par l'utilisateur du produit.

Ici, le besoin fonctionnel majeur de l'outil est la visualisation des résultats (FC2 sur la figure 2) : des graphes de

résultats de comparaisons doivent s'afficher après analyse des données de l'utilisateur.

Un autre besoin fonctionnel est celui de l'adaptation de l'outil aux données traitées (FC6 sur la figure 2) : l'utilisateur doit avoir la possibilité de rentrer, en temps que critère, sur quel organisme portent les données et la fonction biologique du sujet d'étude.

Il a également la possibilité de choisir le type de graphe qu'il souhaite obtenir en résultat (histogramme ou diagramme de Venn).

L'exportation des résultats de visualisation (FC7 sur la figure 2) répond au besoin fonctionnel suivant : l'utilisateur doit avoir la possibilité d'exporter les résultats de visualisation qu'il a obtenu.

La facilité d'utilisation de l'outil (FC8 sur la figure 2) va être également considérée comme un besoin fonctionnel : l'outil doit être intuitif et simple d'utilisation.

Enfin, le dernier besoin fonctionnel n'est autre que celui de l'interface graphique (FC9) : l'interface de l'outil web développée doit être agréable au visuel.

3 Besoins non fonctionnels

Les besoins non fonctionels sont les éléments qui participent au bon fonctionnement du programme mais qui ne sont pas directement visibles par l'utilisateur.

Technologies web (FC10 figure 2) : le script doit pouvoir être accessible depuis un site internet. Pour ce faire les technologies HTML et CSS seront utilisées qui permettent de créer des pages web, ainsi que le language de programmation javascript associé à jvquerry et nodejs qui permettent entre autres de réaliser des scripts, rendre les pages web dynamiques, permettre des interactions entre la page et le serveur.

Chargement des résultats de comparaison (FC3 figure 2) : Une fois les résultats calculés ils doivent être sufisamment légers pour pouvoir être chargés sans problèmes sans nécéssiter trop de processeur ou de débit internet.

Filtration des résultats de comparaison (FC4 figure 2) : Une fois les résultats calculés et chargés l'utilisateur doit être capable de choisir ceux qui l'intéressent pour les afficher.

Rapidité d'exécution (FC5 figure 2) : Dans l'éventualité ou le client doit utiliser le script de manière intensive il est conseillé que le script soit optimisé pour réduire les temps de calcul et utiliser moins de processeur.

4 Les choix

Pour la conception du serveur, au moins 2 alternatives existent, PHP et NodeJS. Chercher à comparer les avantages et inconvénients de l'un et de l'autre reviendrait à déclencher une guerre de trolls.

NodeJS nous a été imposé, afin d'offrir une continuité avec Javascript, déjà utilisé en front-end, probablement en raison de sa conception récente. Le fait de pouvoir utiliser les packages NPM de AmiGO (`https://www.npmjs.com/package/amigo2`) a sans doute aussi orienté le choix de façon non négligeable.

Puisque AmiGO est une application web de référence en matière d'ontologie génétique, utiliser son API JavaScript (`http://api.berkeleybop.org/amigo2/docs/index/General.html`) semble pertinent.

Pour la présentation des résultats, sous forme d'histogrammes ou de diagrammes de Venn, l'API D3 (Data-Driven Documents) constituée d'une collection de scripts à modifier selon nos besoins (`https://github.com/d3/d3/wiki/Gallery`) nous a été vivement suggérée.

Afin de visualiser d'éventuels réseaux mététaboliques issus de la comparaison des jeux de données, KEGG (Kyoto Encyclopedia of Genes and Genomes) (`http://www.genome.jp/kegg/pathway.html`) nous a été vivement recommandé, probablement en raison de son API (`http://www.kegg.jp/kegg/rest/keggapi.html`) ainsi que de

son wrapper utilisable sous NodeJS (`https://github.com/knicos/kegg`).

Références

Lesbats J. (2018). Analyse fonctionnelle de gènes impliqués dans la résistance des levures Candida à la phagocytose macrophagique. Université de Bordeaux, UFR Sciences de la Santé.