Université Claude Bernard Lyon 1

Projet en bioinformatique 3

Cahier des Charges

Conception d'une interface web pour la suite d'outils Bio++

Hadrien Guichard
Konstantinn Bonnet
Johanna Galvis-Lascroux

October 7, 2020

Sous la supervision de:
Laurent Gueguen, Maître de conférences Universitaire Lyon 1
Guillaume Launay, Maître de conférences Universitaire Lyon 1

25 septembre - 17 décembre 2020

Laboratoire de Biométrie et Biologie Évolutive UMR CNRS 5558
Université Claude Bernard Lyon 1
43 bd du 11 novembre 1918
Villeurbanne 69622, FRANCE

Table des matières

1 Introduction

1.1 Contexte

Dans tous les domaines de la biologie de nouveaux programmes sont développés au quotidien. Ces programmes sont en constante évolution et de plus en plus complexes. Avec cette complexité grandissante, ces outils peuvent se retrouver en pratique peu utilisés par de nombreux utilisateurs potentiels. En effet, l'installation ou l'appel de programmes via la ligne de commande peut vite être rédhibitoire. La montée du cloud computing et des solutions no code accessibles par des interfaces web illustre ce mouvement de démocratisation des outils en les rendant plus facile d'utilisation, sans installation locale. L'objectif de ce projet est de développer une interface web permettant l'utilisation de bppsuite (CITATION) dont le paramétrage de l'appel peut devenir très lourd.

1.2 Existant

Bio++ (CITATION) est une suite de librairies en C++ dédiées à la phylogénie et à l'analyse d'évolution de séquences biologiques. Au sein de cette suite, bppsuite est une solution stand alone qui permet l'utilisation de libraries de Bio++ (bpp-core, bpp-seq, bpp-phyl, bpp-popgen) dont le but est de simuler l'évolution de séquences, des analyses phylogéniques, l'estimation du meilleur modèle d'évolution... Cette suite manipule des objets variées tels que des abres, des séquences, des modèles, des process dont les possibilités combinatoires sont infinies. Voici la liste des modules de Bio++ sur lesquels nous allons travailler:

bpp-seq

Cette librairie est composée de classes qui stockent les séquences biologiques. Ces séquences peuvent être de l'ADN, des codons, de l'ARN ou des acides aminés. Cette librairie supporte aussi les séquences alignées.

bpp-phyl

Cette librarie contient les classes et les méthodes pour étudier l'évolution phylogénique et moléculaire. C'est elle qui définit la classe Tree (plus complexe dans la branche newlik en développement) qui est une liste de noeuds. Un noeud contient les methodes permettant d'accéder aux noeuds père et fils ainsi que d'autre information comme sont nom, la longuer de la branche vers le noeud père... Elle contient aussi les outils permettant de reconstruire un arbre basé sur l'évolution des séquences. Cette reconstruction peut se faire avec différentes méthodes : maximum de parsimony, par distance, par vraisemblance... De nombreux modèles evolutifs sont aussi fournis avec cette librarie. Tous les modèles classiques de substitution pour les nucléotides ou les protéines sont inclus, de plus l'implémentation de modèles est possible.

bpp-popgen

Cette librairie a été conçue pour l'étude de génétique de population. Elle se base sur bpp-seq et permet de manipuler des jeux de séquences ainsi que de calculer différentes statistiques comme le polymorphisme, l'hétérozygosie...

bpp-core

Cette librarie définie tous les objets de base (string, numbers, vectors) utilisés par les librairies qui composent Bio++.

bppsuite

Toutes les libraries citées précédement sont indépendantes les unes des autres. Le rôle de bppsuite est d'offir des utilitaires prêts à l'emploi qui permetent de combiner tous ces modules afin de mener des analyses complexes avec une sytaxe commune. En combinant des jeux de données grands, avec une multitude d'abres et de modèles différents les possibiltés d'appel de bppsuite sont nombreuses et complexes. Le paramétrage de bppsuite peut se faire via la ligne de commande mais en pratique elle est plutot faite via un fichier texte de configuration. Ce fichier définie les objets à manipuler et les paramètres qui leur sont associés.

1.3 Objectifs

La complexité grandissante du fichier de paramétrage, associé à une grammaire propre peut être rédhibitoire à l'utilisation de bppsuite. L'objectif de ce projet est de démocratiser l'utilisation de cet outil en développant une interface web intuitive qui permettera à l'utilisateur de réaliser la modélisation qu'il désire et retournera le fichier de configuration. Cette interface devra pouvoir jongler entre le paramétrage de bppsuite et une repésentation visuelle de la modélisation. Cette modélisation sera ensuite éxécutées par bppsuite et les fichiers de sorties devront aussi pouvoir être accessible par l'interface. Au long terme, le but est de faire tourner bppsuite sur un serveur situé sur le campus, offrant aux utilisateurs curieux une version sans code et installation afin de pouvoir facilement utiliser ces outils.

1.4 Acceptabilité

Les critères principaux sont une interface graphique constituée d'icônes permettant de déclarer et paramétrer les différents objets de façon intiutive avec une représentation visuelle de la modélisation en cours d'élaboration. Cette interface web doit être flexible pour pouvoir facilement implémenter les nouvelles fonctionnalités de Bio++ qui est en constante évolution ainsi qu'ajouter des modèle mis au point par l'utilisateur. L'interface doit être facuilement implémentable sur un serveur pour le développement à long terme du projet.

2 Matériel et méthodes

2.1 Besoins fonctionnels principaux

L'objectif principal est de produire une interface web fonctionnelle pour les outils *Bio++*, et de la deployer sur un serveur du laboratoire d'accueil accessible depuis l'extérieur. L'interface doit impérativement être simple et accessible pour un publique non-expert, ciblé essentiellement sur des biologistes travaillant sur des projets en génétique évolutive. En outre, ce projet permet à un utilisateur potentiel de s'abstraire de l'installation des différents outils, qui nécessite une connaissance des environnements Unix-like leur administration, et la compilation de logiciels C++. En d'autres termes, l'idée de ce projet est de permettre de rendre la suite accessible à un publique beaucoup plus large en baissant significativement l'effort nécessaire pour se l'approprier.

Pour ce faire, l'interface doit pouvoir faire l'intermédiaire entre un utilisateur et la suite *Bio++*, afin de pouvoir spécifier un modèle phylogénétique et rentrer des données, de façon transparente par rapport à l'implementation des outils.

En effet, la suite d'outils est imposante en premier abord, nécessitant l'apprentissage d'un langage déclaratif spécifique à *Bio++*, l'utilisation et le paramètrage des différents logiciels de la suite, en passant par l'entrée des données et la récupération des sorties, ainsi que leur interprétation.

2.2 Considérations additionnelles

L'interface doit rendre la suite plus attractive à de nouveaux utilisateurs. Ainsi, sa bonne performance est très importante. Sa réactivité, efficacité et la rapidité du rendu doit donc être bien prise en compte dans son design et implémentation.

L'introduction à la suite *Bio++* par le biais de la nouvelle interface sera facilité par l'accès à des templates de modèles évolutifs, et un tutoriel rapide, tous transposés depuis des documents et exemples déjà disponibles dans les dépôts de *Bio++*. Leur présence sur l'interface permettra de comprendre par l'exemple les possibilités de la suite. Un accès au reste de la documentation déjà générée permettra de plus à des utilisateurs expérimentés d'aller plus loin dans leurs investigations, en passant par des modèles de plus en plus complexes.

2.3 Design

Pour répondre à ces besoins, le design considéré est celui d'une architecture serveur-client classique.

La partie client sera divisée en deux composantes, l'une gérant l'entrée et la validation des données et paramètres, et l'autre le rendu des résultats et leur visualisation.

La partie serveur elle communiquera avec le client pour générer les lignes de commandes et fichiers d'entrée pour lancer les outils *Bio++*, puis renvoyer les résultats du calcul à l'utilisateur.

Le déployement de l'outil devra être facile et rapide pour prendre en compte des changements de serveur ou d'organisation du système de fichiers différente dans l'avenir. Sa prise en charge par une autre équipe après le terme du projet doit être sans effort. Les dépendances du projet doivent donc en outre être faciles à gérer. L'implémentation ne doit pas utiliser des chemins en dur sur le système de fichiers et utiliser des fichiers de configuration pour s'abstraire de sa localisation.

En termes de performance, le temps de réponse du serveur doit être minimale, sans temps de latence supplémentaire par rapport au temps de calcul de la suite. De l'autre côté, la partie client doit être peu lourde pour conserver une rapidité même sur des ordinateurs peu puissants, ce qui est le cas assez fréquemment.

La suite *Bio++* peut prendre en charge des types de données multiples, un ou plusieurs sous-modèles, avec potentiellement des paramètres différents, et une multitude de combinaisons qui rend le nombre de possibilités rapidement ingérable. Le programme initial sera donc très simple mais modulaire et extensible, pour implémenter les fonctionnalités basiques, qui vont soutenir le reste de l'interface, très rapidement. Au début, l'interface ne gérera que des combinaisons d'entrées et de modèles très simples, puis de plus en plus de possibilités seront ajoutées de façon itérative. Ceci permettra de tester facilement au fur et à mesure les nouvelles fonctionnalités, de permettre de limiter l'implémentation dans le cadre de ce projet aux fonctionnalités principales, de donner la capabilité d'ajouter le reste par la suite, et de s'adapter à l'évolution future de la suite.

2.4 Implementation

L'interface sera basée sur une installation de Linux, adaptée au serveur disponible hébergé par le laboratoire d'accueil.

Les données d'entrée sont une ou plusieurs séquences d'alphabet différent (protéique, nucléotidique, etc.), un ou plusieurs arbres phylogénétiques élaborés précedemment, **et autres**. Les séquences sont assumées propres et alignées au préalable.

cf notes: i/o

Le paramétrage se fera en commençant par un modèle bare-bones, ou un template de modèle existant, par le biais de boutons et de menus déroulants pour rajouter de plus en plus d'éléments au modèle en fonction des besoins. Le modèle courant sera visualisé sur un graphe acyclique dirigé, sur lequel des éléments pourront directement être ajoutés, copiés, masqués, etc. **interface principal = DAG?**

Les sorties seront une phylogénie optimisée et des visualisations.

Les logiciels ne dépendent pas de données volumineuses ou de connection à une base de données, donc la gestion de l'espace de disque dur ne sera pas conséquente. Par contre, les outils *Bio++* peuvent, pour des modèles relativement complexes, prendre beaucoup de temps de calcul. Les résultats devront ainsi potentiellement être rendus en différé, par exemple par notification par e-mail, et en fournissant un lien unique vers eux.

Le serveur peut être implémenté sous Python avec Flask.

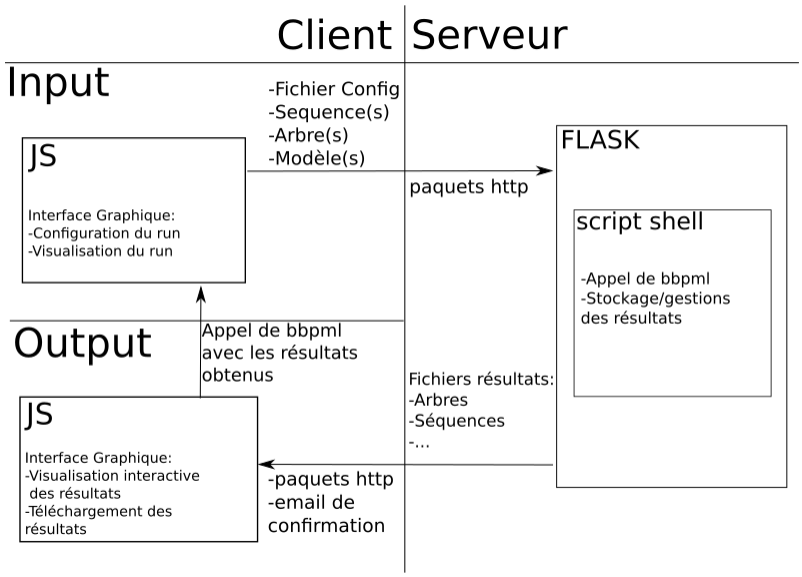
Le client utilisera Typescript transpilé en javascript. Les sorties sont attendues à être assez simples, pour ne nécessiter que des visualisations relativement simples sur graphiques SVG générées par D3.js. Une implémentation single-page est souhaitée.

L'aspect visuel sera géré par des fichiers CSS.

Les pages exposées à l'utilisateur seront clairement documentées, soit sur une page dédiée, soit directement et de façon non-intrusive sur l'interface. De plus, des tool-tips sur les différentes parties de l'interface la rendront plus facilement utilisable. Une page indiquera les auteurs, des contacts, des liens vers les dépôt et leur site officiel, et les articles de recherche sur la suite.

cf notes: détails d'implementation

Le développement se fera en local sur machine personnelles avant de le déployer pour qu'il soit accessible publiquement, pour avoir directement un site fonctionnel dès publication.



2.5 Extensions possibles

L'interface sera implémentée en français durant le projet, mais à terme, elle devrait être en anglais d'abord, avec une option pour la traduire en français.

La fonctionnalité de l'interface est primordiale, mais un travail sur son aspect visuel est souhaitable également.

3 Contraintes et difficultés

3.1 Coûts

Aucun coût financier n'est associé au projet. Le serveur web sur lequel le site sera déployé est déjà fourni par le laboratoire fourni. Les dépendances de la suite *Bio++* et du site sont gratuites et open source, et disponibles sur *github.com*. Les exemples de modèles et de données permettront facilement de tester de développer le site sur machine personnelle.

3.2 Délais

Le délai pour rendre le délivrable final est relativement court, d'environ 10 semaines. Uniquement 2 jours par semaine sont entièrement dédiées au projet, le reste comportant demi-journées ou journées entières de cours.

Le suivi de l'enseignement et la préparation aux examens dès fin novembre nécessite une organisation efficace un travail régulier, et une bonne communication entre les membres du projet et les responsables pédagogique et de stage.

3.3 Autres contraintes

Un problème potentiel auxquels des solutions doivent être prévues à l'avance est un reconfinement possible dû au COVID-19, qui pourrait entraver la communication entre les membres et surtout avec les responsables du projet.

Des dispositifs de l'université, dont des salles de travail disponibles aux étudiants du master bioinformatique, et des systèmes de visioconférence, pourraient aider à pallier à cette contrainte.

L'inexperience des membres du projet en phylogénomique et en l'implémentation de sites web, nécessitera un apprentissage rapide et efficace des technologies et pratiques courantes.

- developpement iteratif

- respect de programming standards rigoureux

- tests a chaque etape

- coordination et review entre collaborateurs, nott pour bugs

L'évolution constante de *Bio++* laisse la possibilité de survenue de bugs cassant aussi la fonctionnalité du site.

4 Déroulement du projet

4.1 Planification

Le développement de l'interface web pour l'utilisation du logiciel dédié au calcul de Maximum de vraisemblance *bppml* suivra les étapes suivantes:

- Installation et test de fonctionnement de pbbsuite: la suite logicielle est installée en local sur nos machines. Parmi les douze logiciels qui en font partie, spécifiquement *bppml* a été choisi par le demandeur de la prestation comme cible du développement web du présent travail. Des exemples de fichiers de configuration (contenant *a minima* les éléments: alignement, arbre(s), modèle(s)) ont été mis a notre disposition afin de tester plusieurs scenarios possibles, du plus simple, en passant par les cas les plus fréquents "standard", jusqu'aux modèles plus exigeantes comptant d'avantage de paramètres. Pendant cette étape nous aurons l'opportunité d'explorer la modélisation des processus évolutifs en phylogénie.

- Développement d'une application en locale: le framework python FLASK et des librairies npm (le gestionnaire de paquets officiel de Node.js) ont été choisis en raison du bon rapport entre flexibilité et stabilité, ayant une ample gamme de librairies disponibles pour un développement rapide et au même temps efficace, ce qui est convenable en raison des contraintes de temps du projet.

  On produira un premier prototype *bpp-web*, acceptant en entrée un modèle ayant les paramètres minimaux requis. Cette application intermédiaire sera rapidement enrichie pour offrir à l'utilisateur de *bppml* la totalité des possibles affectations des variables (plusieurs arbres, plusieurs modèles). Ceci sera réalisé en trois phases parallèles:

  1. Développement de l'interface entrée qui permettra un usage intuitif de la part du client. Un script .js capture les variables insérées par l'utilisateur.
  2. Scripting pour la transmission des variables depuis celles capturées par .js en passant par le langage python (Flask) jusqu'un fichier .txt qui est donné en entrée au logiciel *bppml*.
  3. Scripting pour la capture soit des résultats, soit des messages d'erreur (stderr) et transmission à travers le framework Flask vers le client, et visualisation de l'arbre obtenu *in situ*.

  La possibilité de boucler des résultats vers l'entrée client sera étudiée.

- La Version définitive de *bpp-web* reposera sur le même framework développé à l'étape précédente, ce qui sera réalisé en deux pas simultanés:

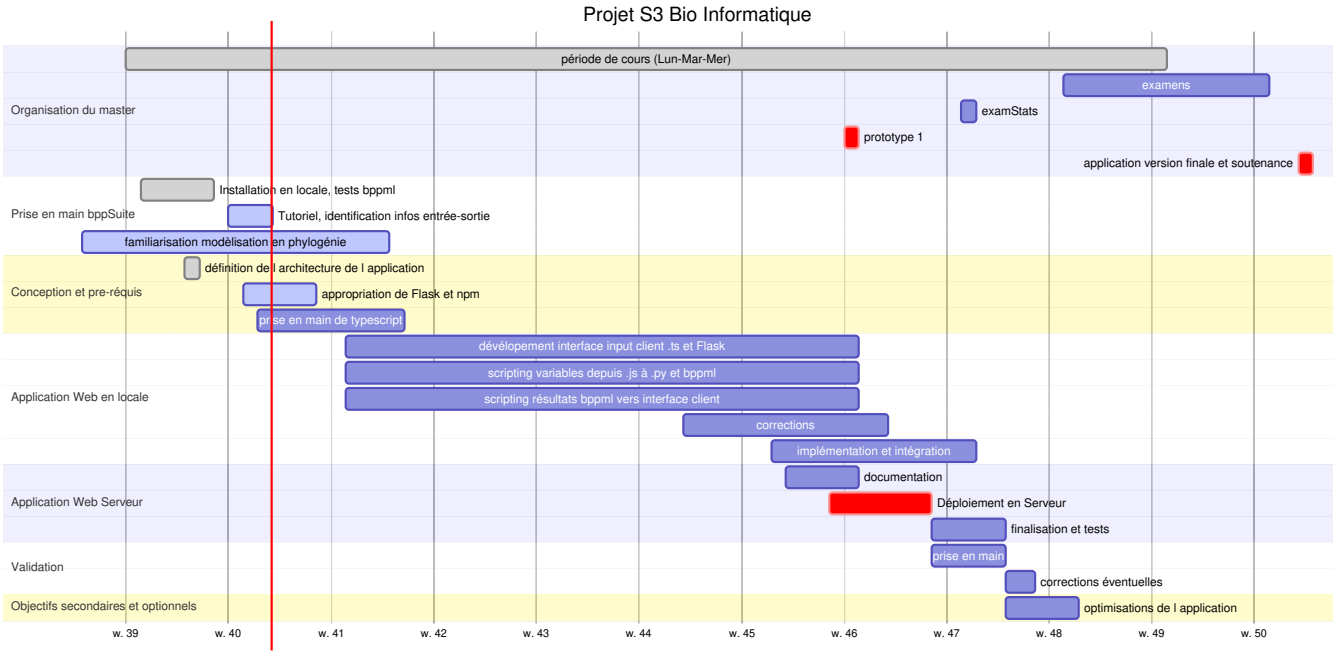
  – la mise en place et déploiement de l'interface web sur un Serveur Linux.

  – l'exécution des derniers tests et correction d'éventuels erreurs.

  Les délais pour chacune des étapes du projet sont visualisés dans la figure 1.

  La prestation sera orientée vers la possibilité de faire évoluer le code (par exemple, par d'autres équipes invitées) afin de couvrir progressivement d'avantage de logiciels faisant partie de *bppsuite* et *Bio++*.

Figure 1: Chronologie de l'exécution du projet



4.2 Plan d'assurance qualité

Pour contrôler la qualité du logiciel, à chaque test effectué la sortie sera évaluée sous critères de cohérence, complétude (aucun fichier de sortie omis ou délaissé) et structure. Sur des modèles plus complexes, l'aval des sorties par l'expert du logiciel *bppml* sera nécessaire.

4.3 Documentation

La documentation sera apportée en totalité dans un dépôt GitHub.

4.4 Responsabilités

4.4.1 Maîtrise d'ouvrage

L'équipe Bio++ development team, représentée par M. Laurent Guéguen est le demandeur du présent travail. Les délais données correspondent a ceux de la matière Projet du M2 bio-info, ayant pour dates clés les suivantes:

- 25/09 : description des besoins par le maître d'ouvrage.
- 30/09 : apport des exemples et Tutoriel par le maître d'ouvrage.
- 09/10 : rendu du cahier des charges
- 17/12 : délivrable et soutenance

Il n'y a pas de budget officiel institutionnel consacré à ce projet.

4.4.2 Maîtrise d'oeuvre

Les trois branches simultanées de développement sont attribuées comme suit:

- Hadrien : interface entrée client, capture des variables.
- Konstantinn : conduction des variables depuis.js vers .py et *bppml*.
- Johanna : interface sortie client, visualisation des résultats.

Le tuteur pédagogique est M. Guillaume Launay.