Rapport Final : Système d'analyse de traces sportives

Guillaume Vacherias et Eric Herqué
INFO5

28 Janvier 2022 - 18 Mars 2022

Contents

1 Introduction

1.1 Sujet

Ce projet de recherche porte comme objectif la conception d'un système d'analyse de trace sportives. Le projet consiste à récolter et rendre évaluable les différentes données provenant de différents équipements connectés portés par un utilisateur en activité sportive.

Le principal défis de ce projet est de pouvoir rendre exploitable les données collectées durant un instant précis. En effet, les données de l'utilisateur peuvent arriver sous différents formats de fichier et à des instants différents. Il est alors important de pouvoir aligner les différentes données pour pouvoir par la suite les traiter. Notre projet a donc pour but de produire une solution permettant de réaliser cet alignement des données grâce à une interface intuitive fournissant à l'utilisateur un affichage.

1.2 Cahier des charges

Lors de la phase de conception du projet, nous avons définit les fonctionalités nécessaires à notre solutions:

Dans un premier temps, nous avons définit les fonctionalités primaires qui sont:

- un affichage graphique des données
- un curseur permettant de déplacer les graphes
- un moyen d'importer des fichiers
- un moyen d'exporter le résultat

Puis dans un second temps nous avons rajouté les éléments secondaires, tel que:

- un élément permettant de rogner les tracés

2 Développement de la solution

2.1 Technologies utilisées

Pour réaliser notre solution, nous nous sommes orienté vers une approche client-serveur car c'est ce qui nous a paru être le choix le plus logique. En effet, pour l'instant cette solution est déployée localement, mais dans des perspectives futures, elle pourrait être déployée à plus grande échelle.

Pour cela nous avons implémenté notre solution en Python, en utilisant la librairie Bokeh. Suite à une étape de veille technologique, notre choix technologique s'est porté sur cette librairie car elle nous permet de répondre au cahier des charges.

Bokeh est une librairie permettant à la fois de créer une solution de type client-serveur et de visualiser des données. La librairie permet l'utilisation de différents widgets prêts à l'emploi et la gestions de ces widgets grâce à des callbacks Python.

2.2 Architecture technique

2.2.1 Application Bokeh

Bokeh permet une visualisation interactive des données avec Bokeh server en créant une application du code fourni. Bokeh serveur utilise cette application pour créer une session et des documents reliant les pages web connectées.

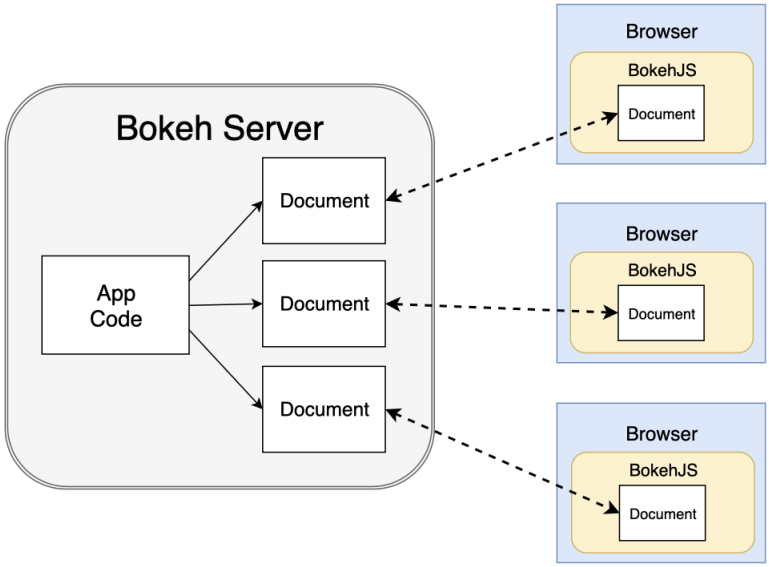

Figure 1: Bokeh Server

Le schéma illustre une application hosté par Bokeh server créant ainsi des documents côté serveur pour chaque connexion établis.

2.2.2 Design de l'application

Nous avons suivis le design pattern Model View Controller. Concernant la partie Control, nous définissons tous les widgets et le graphe au moment du lancement de l'application. Pour la partie Model, les callbacks associé aux widgets seront en charge de l'action attendu. Une fois que le callback a fini son exécution, la partie View qui dans notre application correspond à la classe Plot va initier les updates sur l'interface utilisateur.

Voici un schéma illustrant notre design :

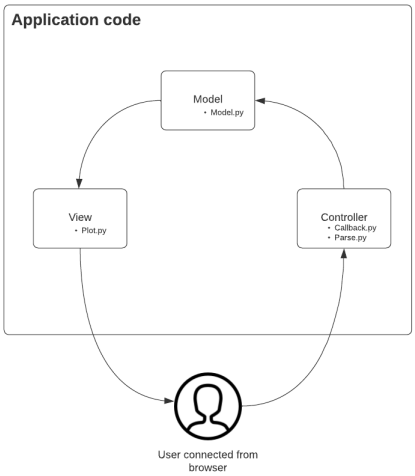

Figure 2: Model View Controller architecture

2.3 Réalisation techniques

Voici une capture d'écran de la solution côté front end :

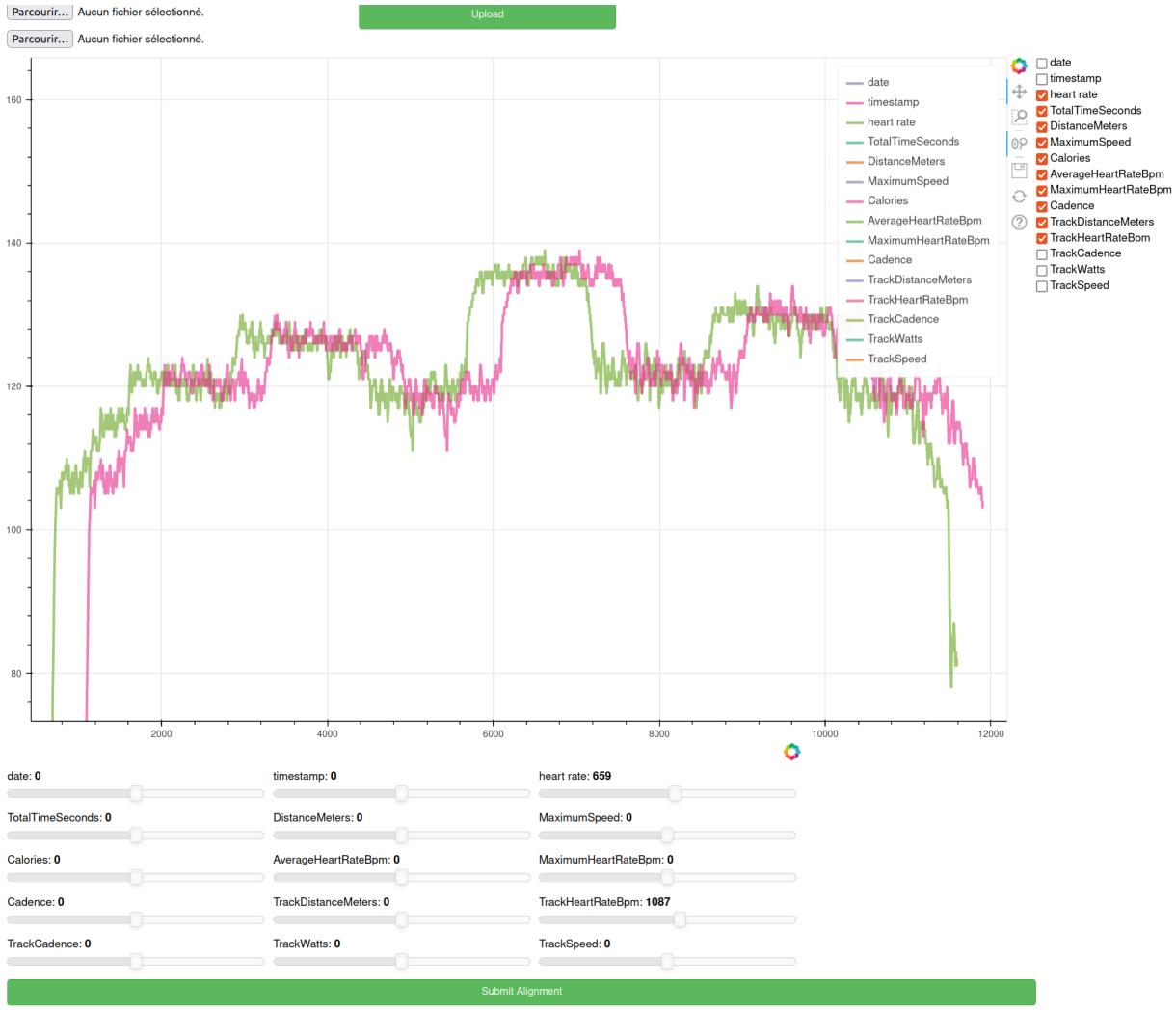

Figure 3: Résultat de l'interface

Tout d'abord, en haut de la page, nous retrouvons le widgets permettant d'importer les fichiers de l'utilisateur. Nous avons des file inputs pour parcourir les fichiers dans l'arborescence côté utilisateur, et un bouton d'upload permettant de les envoyer dans le back-end et donc de produire les graphes.

Au milieu, nous avons la partie visuelle représentée par un graphe, affichant toutes les données à visualiser. Sur la droite, on retrouve une liste de checkboxs, avec les différents champs affichables sur le graphe. L'utilisateur se retrouve avec la possibilité d'afficher ou de cacher certains champs pour pouvoir produire son fichier final.

Enfin, en bas, nous avons tous les sliders permettant de réaliser les décalages des différents graphes. Chaque slider correspond à un graphe. Pour conclure, en dessous de ces sliders on retrouve le bouton "Submit Alignment" permettant à l'utilisateur de créer son fichier final d'alignement des données.

Chaque élément présent dans l'interface utilisateur est un widget Bokeh. Ils sont gérés par des callbacks permettant de réaliser les différentes fonctionnalités.

3 Gestion de projet

3.1 Déroulemet du projet

Il est mené à Polytech Grenoble et sont impliqués dans le projet notre équipe de conception, composée de deux étudiants INFO5, ainsi que notre enseignant encadrant.

L'équipe est ainsi composée de:

- Vivien QUEMA, Technical lead et enseignant encadrant
- Guillaume VACHERIAS, Chef de projet et développeur
- Eric HERQUE, SCRUM Master et développeur

3.2 Méthode agile

Au niveau de l'organisation, nous suivons une organisation agile et hybride. En effet, lors de projets informatique, il est de plus en plus commun de travailler selon la méthode agile, souvent à l'aide d'un SCRUM Master.

Nous avons aussi jugé bon d'inclure des phases de peer programming, lorsque l'on ne pouvait pas répartir des tâches entre les membres du groupe.

3.3 Organisation

Le projet est divisé en deux principaux sprints, guidés par des daily meeting et des réunions organisés de manière fréquente. Nous avions 2 daily meeting par semaine dû au fait qu'il y avait un autre projet en parallèle (ECOM). Puis grâce à notre flexibilité nous pouvions communiquer à tout moment par messages, et placer des réunions lorsque cela était possible.

3.4 Conception et définition des tâches

Au début du projet, nous somme passés par une phase de conception pour cerner les objectifs du projet. Pour cela, nous sommes passés par une phase de brainstorming, mettant en évidence les principaux axes de développement ainsi que leurs liens.

S'en est suivi la définitions des différentes tâches à réaliser qui a ensuite été explicité sur notre backlog, outil de gestion de projet agile. Nous nous sommes ainsi basé sur le backlog pour savoir quoi faire, connaître l'avancement du projet et l'ordre de difficulté/priorité des tâches et aussi pour rajouter des tâches.

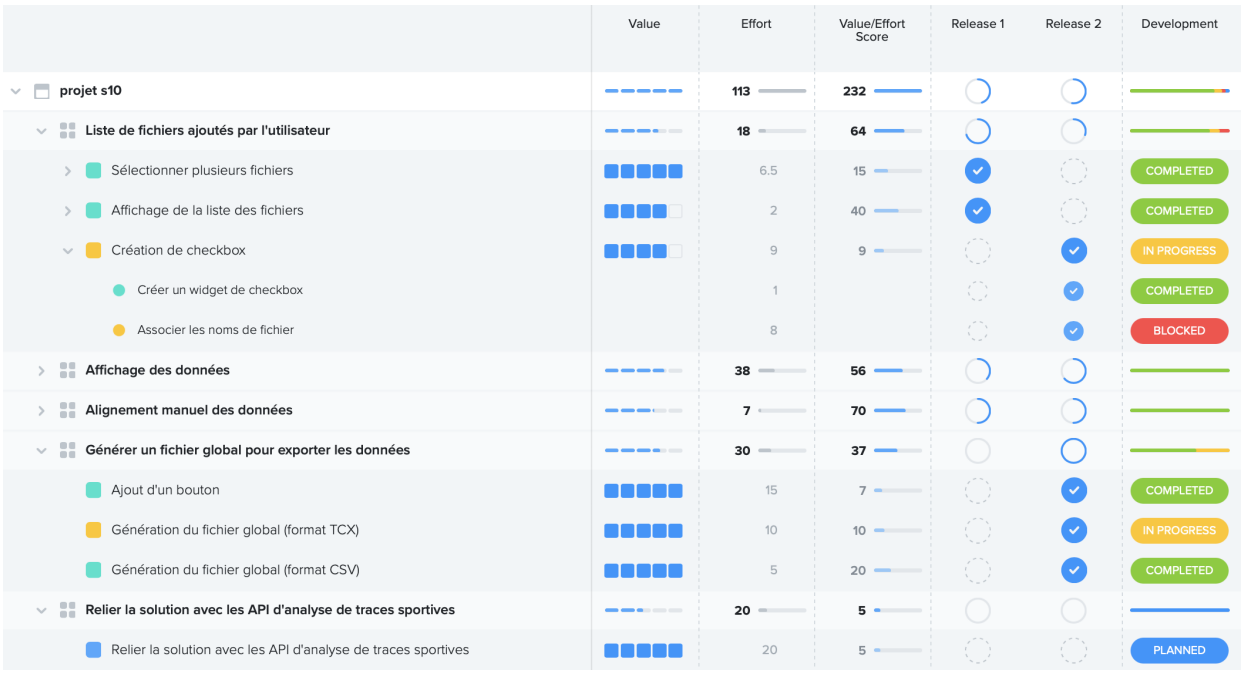

Figure 4: Backlog

L'avantage de cette organisation agile était de pouvoir définir quelles tâches réaliser dans le sprint, de jauger la difficulté de chaque tâches (en équipe) et de pouvoir rajouter des tâches à tout moment.

3.5 Sprints

Le premier sprint a eu pour but de développer la base de notre application client-serveur. Il a débuté suite à la phase de conception qui a eu lieu au début du projet. Le but de ce sprint était de:

- Obtenir une interface côté client
- Obtenir un back-end exploitable
- Trouver un moyen de faire des graphes pouvant être déplacés par l'utilisateur

Ce sprint a pris le plus de temps car nous avons procédé à une mise à niveau et à une veille technologique conséquente. En effet, il a commencé le 28 janvier et a pris fin le 28 février. Au début du projet, nous étions contraint de produire une solution en NodeJS, technologie que nous ne maîtrisions pas. Nous avons donc dû nous documenter et trouver un moyen de répondre au cahier des charges.

Une fois la prise en main effectuée, nous avons dû trouver le moyen de réaliser les graphes, dans le but de procéder à l'alignement manuel des données.

Les librairies que nous avons utilisées pour tracer les graphes sont **Plotly.js**, **Nodeplotlib.js**, **Chart.js**, **Billboard.js**. Certaines de ces librairies ne fournissent pas assez de documentation sur le traçage dynamique des valeurs, ce qui est le cas pour les librairies **Nodeplotlib.js**, **Billboard.js**. En effet, l'interface doit fournir un traçage dynamique sur le graphe pour permettre à l'utilisateur de réaligner (translater) les courbes.

En ce qui concerne la librairie Chart.js, elle ne permet pas d'accéder aux valeurs des courbes translatées, donc la production du fichier global contenant les données réalignées est impossible.

De l'autre côté **Plotly.js**, permet le réalignement des courbes grâce aux sliders et permet l'accès aux nouvelles valeurs réalignées. Cependant, il n'existe aucun moyen d'attribuer un slider à une courbe particulière du graphe. Plus précisemment, la fonctionnalité du slider ne permet pas manipuler toutes les courbes du graphes une par une.

Ne trouvant aucune solution pour pallier à notre problème, le sprint s'est conclu sur un changement de technologie. Effectivement, suite à nos recherches, nous avons trouvé l'outil parfait pour réaliser notre solution. Nous somme donc passés de NodeJS à Python. La librairie **Bokeh** sur Python nous permet de visualiser des données, de construire une architecture client-serveur et nous offre une certaine liberté grâce à l'implémentation de callbacks.

Le second sprint s'est déroulé du 28 février au 16 mars. Durant ce sprint nous avons implémenté la globalité des fonctionnalités décrites dans le backlog, à l'aide de Bokeh. Nous avons tout d'abord reproduit ce que nous avions fait durant le sprint 1, puis nous avons développé les fonctionalités de notre produit:

- Import des fichiers à visualiser
- Affichage dynamique
- Alignement manuel des graphes (sliders)
- Export de l'alignement

4 Outils

Dans le cadre du projet, voici la liste des outils que nous avons utilisé :

- Gitlab
- Product Board : Permet de créer un backlog et donc de voir la visualisation des tâches et sprints complétés
- Discord, Téléphone : Réunion avec Monsieur Quéma pour lui informer de notre avancement ainsi que d'éventuelles questions sur le projet.

5 Métriques logicielles

5.1 Ligne de code

Pour notre application, nous avons produit 304 lignes de code en Python ainsi que 52 lignes de code en HTML.

5.2 Performance

Nous avons 2 formats de fichiers, csv et xml sur lesquels nous pouvons réaliser 3 types d'évaluation de performance. Nous évaluons donc le temps que met le fichier pour être envoyé côté serveur, le temps du parsing du fichier pour extraire les données à tracer ainsi que le temps du tracage.

Format	Word	Temps upload (ms)	Temps ploting (ms)	Temps parsing (ms)
xml	151994	0.05084848403930664	3.343161106109619	2.633176803588867
csv	54664	0.04486560821533203	0.9594240188598633	0.04404592514038086
csv	111574	0.07366824150085449	0.907494068145752	0.06849980354309082

Table 1: Temps de performance

On retiendra aussi que le fichier xml a beaucoup plus de données à extraire que dans les fichiers csv. De plus, la manipulation des données sur un fichier csv est plus facile que sur un fichier xml.

Une autre mesure de performance à prendre en compte est la génération du fichier global. En prenant les données des 3 fichiers utilisés pour le test précédent, ce test a été effectué en faisant l'alignement sur les champs Heart rate du fichier csv et tcx :

Temps d'alignement (ms)
0.17492961883544922

Table 2: Temps d'alignement

5.3 Contribution

Cette contribution ne prend que en compte la version final réalisé avec la librairie **Bokeh**. Elle ne prend pas en compte les versions précédente utilisant d'autre librairie. De plus, ces chiffres ne prennent pas en compte le peer programming :

Membre du groupe	Nombre de commit	Nombre de lignes de code
Guillaume Vacherias	25	208
Eric Herqué	4	96

Table 3: Contribution

- Guillaume : réalisation de l'architecture du code implémenté
- Eric : réalisation des sprints et des tâches

6 Perspectives futures

Nous avons réalisé la génération du fichier contenant les données alignées. Le fichier généré est en format csv. Nous pouvons aussi générer le fichier en format xml, qui est une fonctionnalité que nous n'avons pas eu le temps d'implémenter.

De plus, une piste d'amélioration serait de relier notre application avec les logiciel d'analyse de trace tel que :

- Nolio
- Golden Cheetah
- Training Peaks

En effet, récupérer les API de ces sites permettrait de fournir directement sur notre interface une analyse des données après l'alignement, rendant l'analyse de traces plus facile.

Enfin, une autre piste serait de déployer cette solution, en vue de la rendre accessible à plus grande échelle. En l'état actuel, le serveur de notre produit se lance uniquement en local. Etant donné que son architecture est de type client-serveur, nous pourrions imaginer un déploiement de celle-ci.

7 Conclusion

En définitive, le rendu final du projet pose les principales bases de la solution, à savoir l'import de fichiers TCX/CSV, la visualisation et translation de données et l'export du résultat final. L'utilisateur peut ainsi réaliser l'alignement souhaité pour par la suite réaliser des analyses.

Bibliographie

[1] Plotly.js,
`https://plotly.com/javascript/`

[2] Chart.js,
`https://www.chartjs.org/`

[3] Billboard.js,
`https://naver.github.io/billboard.js/`

[4] nodeplotlib,
`https://www.npmjs.com/package/nodeplotlib`

[5] Bokeh,
`https://bokeh.org/`

[6] XML Minidom,
`https://docs.python.org/3/library/xml.dom.minidom.html`

8 Glossaire

- *Callback* : Fonction de rappel ou fonction de post-traitement est une fonction qui est passée en argument à une autre fonction
- *Widget* : Composant d'interface graphique
- *GUI* : Graphic User Interface
- *MVC* : Model View Controller
- *CSV* : Comma-Separated Values
- *XML* : Extensible Markup Language