

*

0cm7ex plus 1ex minus .2ex1.5ex plus .2ex *

0cm4.5ex plus 1ex minus .2ex1.5ex plus .2ex *

*

*

Compte Rendu

Sonny Klotz - Jean-Didier Pailleux - Malek Zemni

*Interface de chargement, de contrôle
et d'analyse statistique des données
pour la constitution d'un graphe de flux*

28 mai 2017



Module *Projet*

Table des matières

1	Architecture de l'application	1
1.1	Organigramme et données échangées	1
1.2	Fonctionnalités des modules	3
2	Outils et langages de programmation	4
2.1	Contraintes	4
2.2	Choix des outils et des langages	5
3	Points délicats	5
3.1	Le paradigme client serveur	5
3.2	Les applets	6
3.3	Ajax	6
4	Fonctionnement de l'application	6
4.1	Format du fichier CSV	6
4.2	Interface web	6
4.3	API de chargement des données et d'analyses descriptives	7
5	Bilan technique du projet	7
5.1	Problèmes rencontrés	7
6	Organisation interne du groupe	8
7	Coûts	9

Introduction

Ce document est le compte-rendu final de notre travail qui s'inscrit dans le cadre du module *Projet* de la licence informatique de l' **UVSQ**. Le sujet de ce projet a été proposé par l'entreprise **DCbrain**.

Ce projet s'inscrit dans un thème d'actualité, le **Big Data**. La notion Big Data fait référence aux masses de données collectées et l'enjeu consiste à extraire de ces données une valeur ajoutée.

Les techniques utilisées pour répondre aux problèmes sont empruntées aux statistiques et l'on va parler d'**analyse descriptive de données**¹ : une branche des statistiques qui étudie les données par le biais d'informations synthétiques, sans supposer qu'elles suivent un comportement particulier.

Notre client, **DCbrain**, emploie une appore basée sur une représentation des **réseaux physiques** de leur client en **graphes de flux**. Les graphes de flux permettent de représenter les données liées au flux du réseau, repérer les anomalies sur le réseau et simuler les évolutions de celui-ci.

Les graphes de flux peuvent être utilisés pour tout réseau physique de fluide, tels les réseaux électriques et c'est pourquoi les clients de DCbrain sont des groupes comme ERDF ou la SNCF. Par exemple, on pourra représenter les canaux qui acheminent l'électricité par des arcs et les connections par des noeuds.

Fournir aux utilisateurs une application web est l'objectif du projet. Le produit représente un outil complémentaire au travail de DCbrain et ses fonctionnalités permettent aux utilisateurs de charger des données, les visualiser et les analyser.

Deux documents supplémentaires sont livrés avec le code source, un manuel utilisateur décrivant comment faire fonctionner notre application, et une documentation technique des modules de notre produit.

1 Architecture de l'application

1.1 Organigramme et données échangées

Cet organigramme a été préalablement établi dans le cahier des charges du produit. Il représente la décomposition en modules de l'application, ainsi que les informations qui circulent entre ces modules.

On notera les arrangements faits dans la représentation des informations transmises entre les modules de l'organigramme. Ces arrangements sont principalement effectués pour des fins de compatibilité avec les outils utilisés et seront explicitement justifiés.

1. On utilisera l'acronyme ADD par la suite pour parler d'analyse descriptive de données.

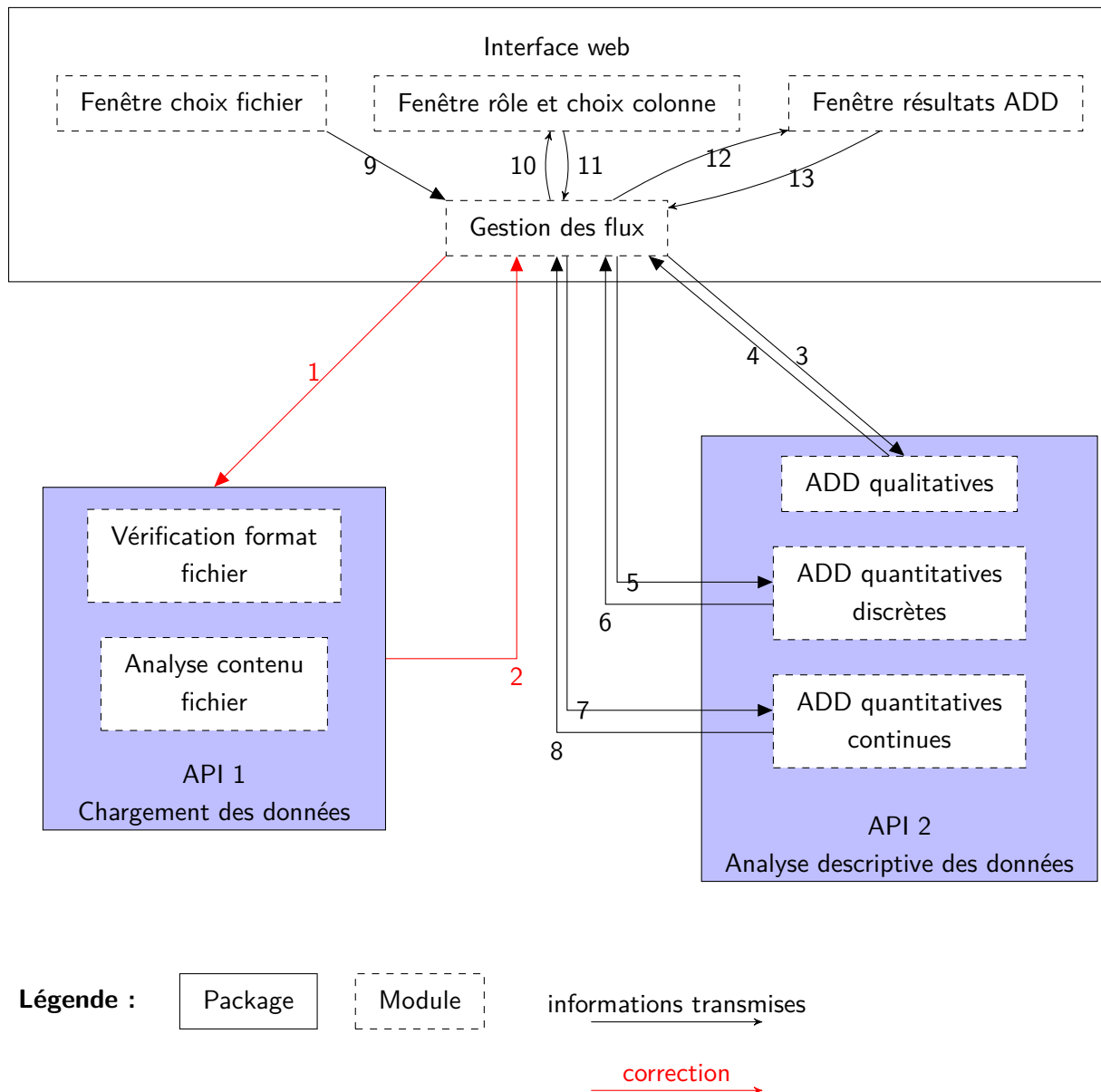


FIGURE 1 – Organigramme des différents modules du logiciel

Notes :

- (1) Chemin du fichier CSV importé : lancement des fonctionnalités du module de chargement de manière indépendante : la vérification de son format et l'analyse de son contenu
- (2) Le résultat du chargement du fichier CSV importé :
- soit un message d'erreur signalant un problème lié au fichier
 - soit deux structures, l'une contenant les données du fichier, et l'autre contenant une description du nom, du type, et des erreurs de chaque colonne de données du fichier

Justification de la modification : les outils utilisés pour faire communiquer les modules ne permettait

pas la transmission d'un certain type d'objets, dont les objets représentant les fichiers. On a donc décidé de récupérer directement les résultats de l'analyse du fichier et d'y inclure la gestion des potentielles erreurs.

- (3) Ensemble de données de type qualitatif
- (4) Erreur ou effectifs, effectifs cumulés, fréquences, fréquences cumulées
- (5) Ensemble de données de type quantitatif discret
- (6) Erreur ou indicateurs de tendance central, de dispersion et de forme, les anomalies, la distribution des données, un diagramme à moustaches
- (7) Ensemble de données de type quantitatif continu
- (8) Même données que (6)
- (9) Chemin du fichier CSV importé
- (10) Informations du (2) et ensemble de données contenu dans le fichier CSV
- (11) Signal de validation du choix de la colonne, et noms des colonnes
- (12) Envoi des résultats d'analyses de (4), (6) et (8)
- (13) Signal de contrôle : demande d'exportation des résultats de l'ADD, analyse d'une autre colonne, ou importation d'un autre fichier

1.2 Fonctionnalités des modules

Package Chargement des données

1. Module Vérification format fichier :
Vérification de l'existence du fichier, de l'extension, de la lisibilité du contenu (texte brut ou formaté) et l'accessibilité en lecture.
2. Module Analyse contenu fichier :
Ce module comprend deux fonctionnalités principales :
 - [leftmargin=*]Lecture du contenu du fichier CSV : on détermine le délimiteur de données du fichier, puis on lit ses données ligne par ligne. Ces données sont stockées dans une **première structure**. Description des données de chaque colonne du fichier CSV : on stocke les noms des colonnes fournis dans le fichier, ensuite on détermine les types de données attendus à partir de ces noms, et finalement, on parcourt la structure contenant les données du fichier en comparant le type de ces données au type attendu. Si ces types ne correspondent pas, on indique une description de l'erreur.
 - Ces trois informations (nom, type et erreurs) sont stockées dans une **deuxième structure**.
Ce module va donc fournir les deux structures décrites ci-dessus.

Package Analyse descriptive de données

- ± Module ADD qualitatives :
 - [leftmargin=*]
 - Calcul des effectifs, effectifs cumulés, fréquences et fréquences cumulées des données

— Préparation des représentations graphiques : diagramme en secteur et histogramme

2. Module ADD quantitatives discrètes :

[leftmargin=*]Statistiques : moyenne, quantiles, variance, écart-type, coefficients de symétrie et d'aplatissement de Fisher Anomalies de Tukey : toute donnée hors de l'intervalle $[Q1 - 1.5 \cdot IQ ; Q3 + 1.5 \cdot IQ]$, où $Q1$ et $Q3$ sont les quartiles, et IQ l'écart inter-quartiles. Préparation des représentations graphiques : fonction de distribution, fonction de distribution cumulative, boîte à moustaches

3. Module ADD quantitatives continues :

[leftmargin=*]Discretisation de l'étendue (découpage en classe d'intervalles) Calcul des quantiles pour le cas de données continues Préparation des représentations graphiques : fonction de distribution cumulative

Package Interface web

1. Module Gestion des flux :

[leftmargin=*]

- Gestion des branchements : exécution normale ou arrêts pour cause d'erreur.
- Interface entre les différentes fonctionnalités : communique les données nécessaires entre les modules.

2. Module Fenêtre choix fichier :

[leftmargin=*]Récupération un fichier CSV en renseignant son chemin en parcourant l'arborescence de fichiers, ou de la manière d'un Drag & Drop.

3. Module Fenêtre rôle et choix colonne :

[leftmargin=*]Affichage des données du fichier CSV : noms de colonnes, leurs valeurs et du nombre de lignes du fichier. Affichage des données erronées (description de l'erreur). Affichage d'un échantillon de données du fichier à l'aide de filtres. Sélection et envoi d'une colonne de mesures pour lancer l'analyse sur celle-ci.

4. Module Fenêtre résultats ADD :

[leftmargin=*]Affichage des résultats d'analyse descriptive : informations statistiques de l'API 2 et représentations graphiques pour visualiser les données dans leur ensemble. Une fonctionnalité de retour en arrière permet de sélectionner une nouvelle colonne sans relancer l'applet.

2 Outils et langages de programmation

2.1 Contraintes

Lors de la réalisation de l'application, des contraintes ont dû être respectées. Ces contraintes ont été imposées par notre client **DCbrain** lors de la remise du sujet du projet. On peut en énumérer trois :

1. Le produit doit fournir une application web : notre produit sera une application fonctionnant sur un navigateur web, appelée **applet**.
2. Le produit doit être développé avec un langage de programmation compatible avec l'analyse de données : le langage de programmation choisi doit inclure des bibliothèques qui permettent d'analyser les données (tâches de data mining et de machine Learning).
3. Le produit doit fournir des API (ensemble de packages) pour le chargement et l'analyse de données : le code source de l'application doit intégrer des API d'analyse descriptive de données qui pourront être réutilisés par le client.

Ces contraintes se sont avérées déterminantes pour les choix techniques que l'on a fait, et en l'occurrence, le choix des langages de programmation.

2.2 Choix des outils et des langages

Les contraintes imposées par le client ainsi que les exigences définies nous ont permis de fixer nos choix sur plusieurs langages de programmation qui vont interagir ensemble :

[leftmargin=*]**Python** : d'une part, ce langage est adapté pour l'ADD, puisqu'il est doté de nombreux modules permettant d'effectuer du calcul scientifique. D'autre part, **Python** est compatible avec le développement d'applications web avec l'aide de frameworks dédiés. Nous avons choisi **Flask**, une bibliothèque qui s'efforce d'être la plus simple possible pour une prise en main, à l'opposé d'autres bibliothèques. C'est donc plus abordable pour une première expérience sur le développement d'applications web. **HTML et CSS** : l'application web nécessite le recours à des langages de balisage comme **HTML** pour la présentation et **CSS** pour la mise en forme des pages web. **JavaScript** : c'est le langage qui nous permet de dynamiser nos pages web rendues côté client, c'est pour cela qu'on parle d'application web. **jQuery** : librairie JavaScript pour la manipulation simplifiée de concepts avancés, dont la gestion des événements et **Ajax** (concept détaillé dans la prochaine partie). **c3js** : module de représentations graphiques JavaScript réutilisables. Il est basé sur **d3js**, une librairie JavaScript pour une manipulation de documents dirigée par les données. **Sphinx** : framework Python pour la génération de documentation. Sa puissance vient de la génération automatique directement à partir du code source.

3 Points délicats

Nous dédions cette partie pour expliquer les concepts informatiques qui nous ont posé le plus de problèmes. Ces concepts n'étaient pas maîtrisés voire inconnus avant d'aborder le projet, ils ont donc grandement faussé notre estimation quant à la charge de travail que représente le projet car ils ont mal été pris en compte, ou pas du tout.

3.1 Le paradigme client serveur

client : navigateur web serveur : là où est hébergé notre code source et nos templates. client fait requête au serveur, le serveur répond en envoyant la page web demandant (html).

Pb : Comment dynamiser la page ?

3.2 Les applets

balise script, et javascript, qui contiennent du code source exécuté côté client (par le navigateur web).

Pb : Comment utiliser nos outils et nos données stockées sur serveur avec une applet dynamique.

3.3 Ajax

Asynchronous JavaScript And XML. Requêtes et envoi uniquement des données nécessaire (pas toute une page web), donc rafraichissement partiel de la page. Code plus léger et moins de chargement.

4 Fonctionnement de l'application

4.1 Format du fichier CSV

Le format du fichier a été établi par le client **DCbrain**. Son contenu est assez structuré et est décrit par des colonnes aux types prédéfinies :

timestamp	parent	enfant	mesure 1	mesure 2
January 1st 2017, 15 :00 :00.000	102	95	26644.235	176.253

Le graphe de flux utilisé par **DCbrain** pour analyser les réseaux de ses clients est orienté, d'où l'utilisation des nœuds *parent* - *enfant* numérotés. Ceux-ci permettent d'identifier une connexion précise. Les colonnes *mesure* représentent des données mesurées sur une connexion à un temps donné. Le nombre de ces colonne n'est pas limité et en pratique il ne dépassera pas la dizaine.

4.2 Interface web

Notre applet est destiné à des utilisateurs n'ayant pas de connaissances avancées pré-requises pour sa manipulation. En effet, uniquement des capacités à interpréter des informations statistiques basiques et à lire de représentations graphiques sont demandées. Son utilité va être déterminée par des graphiques et une présentation des valeurs intuitifs qui vont guider l'utilisateur dans l'interprétation des résultats.

Le scénario standard d'utilisation est simple. Une première fenêtre laisse le choix à l'utilisateur de sélectionner un fichier au format défini à la sous-partie précédente.

Si le fichier n'est pas au bon format, il ne sera pas traité et il faudra sélectionner un autre fichier. Sinon, la deuxième fenêtre peut être affichée. Sur cette fenêtre sont affichées dans un tableau les données du fichier qui peuvent être soumis à différents traitements : renommage de l'en-tête des

colonnes et filtrage des valeurs. Ensuite, on peut valider l'analyse sur les valeurs filtrées pour passer à la dernière fenêtre. Cette fenêtre dispose d'un système de navigation entre les différents graphiques disponibles, et un compte-rendu synthétique des analyses descriptives.

Finalement, on a le choix de revenir à la fenêtre de filtrage des valeurs, ou bien de télécharger des résultats complets d'ADD sur notre ordinateur.

4.3 API de chargement des données et d'analyses descriptives

Une partie de notre travail consiste à fournir des paquets réutilisables pour le chargement des données et l'ADD. Pour mettre en avant nos résultats, nous avons décidé d'écrire un script en réutilisant les fonctionnalités de nos API.

Son principe est d'effectuer des analyses descriptives sur chaque colonne pour chaque arc distinct d'un fichier toujours au format présenté plus tôt. Ce script écrit dans un nouveau fichier `.csv` est sortie dans un format similaire pour permettre une réutilisation et une interprétation facile.

Les résultats satisfaisants de ce script nous ont incité à le réutiliser pour la fonctionnalité de téléchargement des analyses de notre applet.

5 Bilan technique du projet

Notre produit final, c'est à dire l'application, se comporte comme prévu : l'application est fonctionnelle, la liaison entre ses différents modules réussit bien et les différentes fonctionnalités fournissent le résultat attendu.

Cependant, nous aurions souhaité enrichir

Tests unitaires : Les modules de notre application ont été testés et le code des tests est disponible avec notre code source.

5.1 Problèmes rencontrés

Problèmes résolus :

Lors de la réalisation de l'application, on a été confrontés à plusieurs problèmes, principalement des verrous techniques, qui ont perturbé le bon déroulement de notre travail :

[leftmargin=*]Les premiers verrous rencontrés sont liés aux outils et aux langages de programmation utilisés. En effet, la majorité de ces outils sont nouveaux pour nous, ce qui a nécessité un temps d'apprentissage et d'adaptation. Cependant, on a finalement réussi à assez maîtriser ses outils pour assurer la réalisation de notre application. Lors de la manipulation de ces outils, on a rencontré des problèmes de compatibilité lors de l'interaction entre les différents langages de programmation utilisés. Un des problèmes principaux était le type de données transférées

entre les différents modules.

Par exemple, la bibliothèque **Flask** ne permet pas la transmission d'objets qu'elle considérerait comme non sérialisable (en particulier les fichiers) et ceci a demandé des aménagements dans le schéma de transfert d'informations. Les fonctions manipulant des fichiers ont dû être modifiées pour utiliser d'autres structures compatibles (divergence avec les spécifications). Les tests ont pu mettre en lumière un défaut de précision des nombres flottants. Les fréquences cumulées étaient calculées en additionnant plusieurs fois les fréquences d'apparition, ce qui propage les imprécisions de calcul. Ce problème a été résolu en passant par les effectifs cumulés pour n'avoir qu'un calcul flottant à effectuer. Un dernier problème concerne le graphe des séries temporelles réalisé avec c3js.

Nous ne pouvons pas avoir de représentation logique si plusieurs valeurs sont mesurées au même instant.

Cela dit, sur un même arc la série temporelle est cohérente, et nous avons pu résoudre le problème en mettant à disposition une série temporelle par connexion différente.

... mettez vos trucs ...

Problèmes non résolus :

Certains problèmes rencontrés n'ont pas été entièrement résolus. Ces problèmes ne sont pas déterminants pour l'acceptabilité de notre produit et sont surtout liés aux performances :

[leftmargin=*]Lors du chargement d'un fichier de grand volume, un temps est nécessaire pour que le système d'affichage des données et de filtrage soit opérationnel. L'augmentation du temps d'attente n'est pas linéaire en la taille du jeu de données mais moins performant. Nous aurions aimé atteindre des bonnes performances sur des gros volumes pour avoir une application compétitive. Une représentation graphique, aurait été un réel atout pour notre application, les boîtes à moustaches de Tukey.

Alors même qu'il est annoncé dans les spécifications, il n'est cependant pas disponible dans la librairie c3js, et donc n'a pu être réalisé dans les délais.

Nous avons dans notre travail préparé toutes les données nécessaires à cette représentation, y compris les requêtes Ajax pour envoyer ces données au navigateur.

Cela dit, le code JavaScript nécessite la maîtrise d'une librairie telle d3js pour faire du graphisme. Malgré la puissance de ce type de librairie, nous n'avons pas pu être assez performant avec pour obtenir des graphismes présentables et interactifs originaux. D'où le retrait de la fonctionnalité.

6 Organisation interne du groupe

Ce tableau établi dans le cahier des charge, illustre l'assignation des modules pour chaque membre du groupe :

Module	Malek	Sonny	Jean-Didier	Total
--------	-------	-------	-------------	-------

Gestion des flux			x	1
Fenêtre choix fichier			x	1
Fenêtre rôle et choix colonne	x			1
Fenêtre résultats ADD		x		1
ADD qualitatives			x	1
ADD quantitatives discrètes		x		1
ADD quantitative continues		x		1
Vérification format fichier	x			1
Analyse contenu fichier	x			1

Cette répartition a été parfaitement respectée. Elle nous a permis de travailler efficacement et assez indépendamment, ce qui prouve que l'assignation des modules a été judicieusement faite.

Nous sommes également restés en contact pendant toute la phase de développement pour s'entraider pour la prise en main des nouveaux outils.

7 Coûts

Ce tableau indique les coûts estimés lors de l'établissement du cahier des charges et les coûts finaux, en nombre de lignes de code et pour chaque module :

Module	Nombre de lignes	Justification	Coût final
Gestion des flux	15	Mise en forme du main, appel de l'application, gestion des routes pour flask et des échanges de données avec ajax	98
Fenêtre choix fichier	30	Fonctions pour : Drag& Drop + Système de fichiers	72
Fenêtre rôle et choix colonne	65	Communication avec le module application + Affichage de l'interface + lecteurs des valeurs + affichage des valeurs	180
Fenêtre résultats ADD	100	Envoi d'informations au module application + Construction des graphes d'ADD	200
ADD qualitatives	60	Fonctionnalités pur occurrences et fréquences + données des graphes	53
ADD quantitatives discrètes	100	Formules attachées à l'ADD quantitatives discrètes + données des graphes	89
ADD quantitatives continues	85	Formules attachées à l'ADD quantitatives continues + données des graphes	93
Vérification format fichier	30	Ouverture fichier + vérification si ouverture en lecture + présence de texte formaté ou non	35
Analyse contenu fichier	60	Recopie et vérification + initialisation de la structure + Parcours du fichiers avec condition + Fonction pour donner nom et type de colonne	70
Coût Total	545	Estimation totale du coût	850

Les coûts estimés pour les modules effectuant de simples calculs sont respectés. Par contre, les modules de l'interface web ont plus de lignes que prévu. Ceci est dû à l'implémentation de fonctionnalités

d'affichage dynamique dans nos fenêtres et les échanges de données Ajax qui ont allourdi le code source sans que nous ayons réussi à le prédire.

Conclusion

Ce document résume tout ce qui a été établi avant, pendant et après la réalisation de notre œuvre. Cette œuvre est une application web de visualisation et d'analyse de données. Son développement a été réparti en plusieurs modules indépendants, ce qui facilite sa maintenance et permet ainsi au client d'étendre ses différentes fonctionnalités.

Dans le point où l'on est actuellement, on peut revoir, avec du recul, la manière avec laquelle notre produit a été réalisé. En effet, les verrous majoritaires concernent les échanges Ajax et les représentations graphiques sur un site web. Privilégier le langage JavaScript au détriment de Python pour le traitement des données limiterait les échanges avec le serveur (ce qui augmente aussi les performances). Néanmoins, et dans le cadre pédagogique de ce projet, les délais imposés ne laissent pas suffisamment de marge pour une maîtrise de ces vastes outils dont la prise en main n'est pas rapide.

Ce travail a été réalisé par un groupe de 3 personnes. Le sujet a été très bien appréhendé par les membres du groupe et l'entente a été excellente. Grâce aux informations préalablement établies dans le cahier des charges et dans les spécifications, le travail au sein du groupe a pu être mené de façon assez indépendante, sans générer de conflits ou de problèmes d'organisation.

Cette épreuve constitue donc une expérience enrichissante pour nous, non seulement sur le plan compétence, mais aussi sur le plan de travail collectif, qui nous inspirera probablement pour de futurs projets.