

# Présentation Cahier des Charges

Sonny Klotz - Jean-Didier Pailleux - Malek Zemni

UVSQ

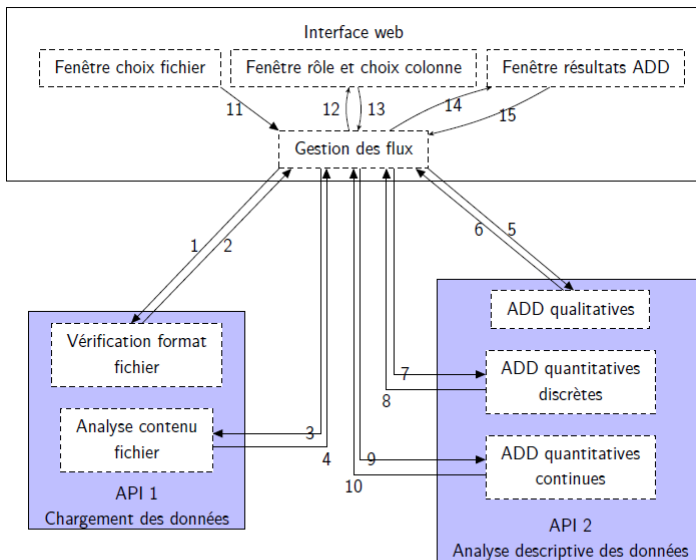
01/06/2017

Projet de L3 informatique UVSQ, remis par DCbrain.

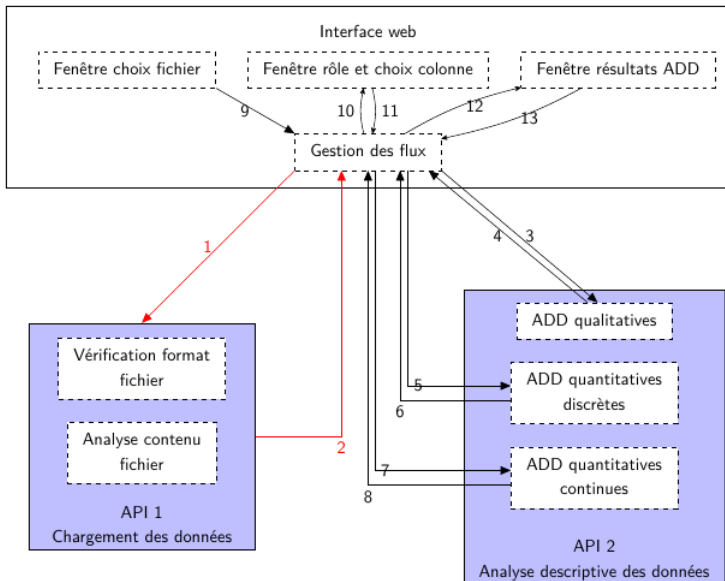
- Projet découlant d'un thème : le **Big Data**.
- Analyse descriptives de données pour répondre au problème du **Big Data**.
- Utilisation de graphe de flux par DCbrain pour visualiser le réseau (détections d'erreurs, optimisation).
- **Objectif** : Fournir application web, outils permettent de charger des données, de les visualiser et les analyser.

- 1 Architecture
- 2 Outils, langages de programmation
- 3 Fonctionnement de l'application
- 4 Bilan technique
- 5 Organisation interne du groupe
- 6 Coûts
- 7 Conclusion

# Architecture



# Architecture



## **API 1** : Chargement des données

*Vérification format fichier :*

- Format csv
- Ouverture en lecture
- Texte brut ou formaté

## API 1 : Chargement des données

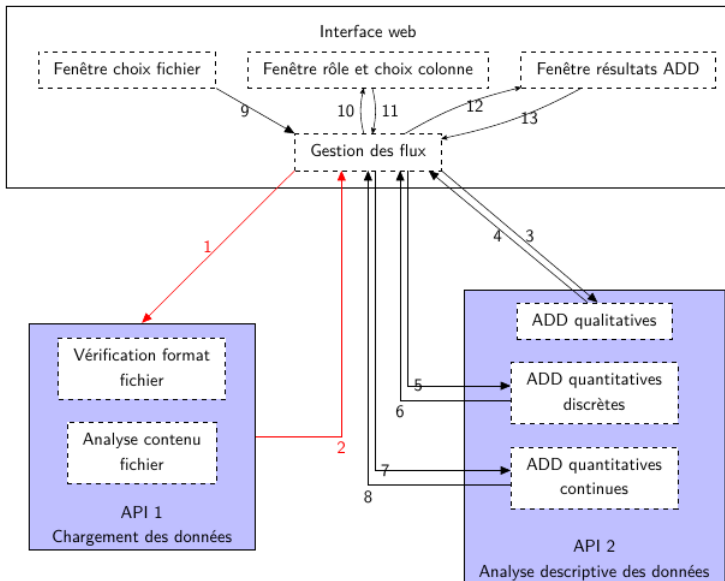
### *Vérification format fichier :*

- Format csv
- Ouverture en lecture
- Texte brut ou formaté

### *Analyse contenu fichier :*

- Lecture des données du fichier ligne par ligne + stockage de ces données dans une **structure 1**
- Description des données de chaque colonne : type, nom et données erronées + stockage dans une **structure 2**.

# Architecture

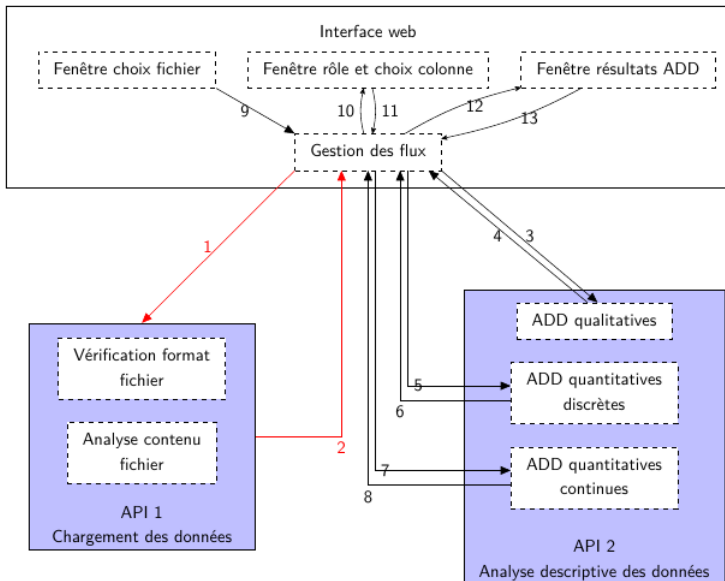




## API 2 : Analyse descriptives des données

- **Données à analyser** : Données d'une colonne (Avec un possible filtrage).
- **Retours de l'analyse** : Informations statistiques et représentations graphiques.
- **ADD quantitatives continues** : Discrétisation des valeurs.

# Architecture



## Interface web

### *Gestion des flux :*

- **Flux d'exécution** : Gestion des branchements et arrêts de l'application en cas d'erreur(s).
- **Flux de données** : Rôle d'interface pour communiquer les données entre les différents modules

## Interface web

### *Gestion des flux :*

- **Flux d'exécution** : Gestion des branchements et arrêts de l'application en cas d'erreur(s).
- **Flux de données** : Rôle d'interface pour communiquer les données entre les différents modules

### *Fenêtre choix fichier :*

- **Choix du fichier** : Parcours de l'arborescence de fichiers - Drag&Drop.

## *Fenêtre rôle et choix colonne :*

- Affiche sous forme d'un tableau : nom des colonnes - nombre de lignes et de colonnes - un échantillon grâce à une navigation.
- Affichage des données erronées + description.
- Sélection et envoi d'une colonne de mesures pour analyse.

## *Fenêtre rôle et choix colonne :*

- Affiche sous forme d'un tableau : nom des colonnes - nombre de lignes et de colonnes - un échantillon grâce à une navigation.
- Affichage des données erronées + description.
- Sélection et envoi d'une colonne de mesures pour analyse.

## *Fenêtre résultats ADD :*

- Affichage des résultats d'analyse descriptive : informations statistiques + représentations graphiques.
- Fonctionnalité de retour en arrière pour analyser une nouvelle colonne.
- Fonctionnalité de téléchargement des résultats au format .csv

- 1 Architecture
- 2 Outils, langages de programmation
- 3 Fonctionnement de l'application
- 4 Bilan technique
- 5 Organisation interne du groupe
- 6 Coûts
- 7 Conclusion

## Contraintes

- 1 Le produit doit fournir une application web.
- 2 Le produit doit être développé avec un langage de programmation compatible avec l'analyse de données.
- 3 Le produit doit fournir des API pour le chargement et l'analyse de données.

## Outils et langages de programmation

- **Python** : adapté pour l'ADD et le développement d'applications web
- **Flask** : framework web Python
- **HTML et CSS** : présentation et mise en forme des pages web
- **JavaScript** : dynamiser les pages web
- **jQuery** : gestion des événements et **Ajax**
- **c3js** : module de représentations graphiques
- **Sphinx** : framework Python, génération de documentation



- 1 Architecture
- 2 Outils, langages de programmation
- 3 Fonctionnement de l'application**
- 4 Bilan technique
- 5 Organisation interne du groupe
- 6 Coûts
- 7 Conclusion

Malek (3 :30)

- 1 Architecture
- 2 Outils, langages de programmation
- 3 Fonctionnement de l'application
- 4 Bilan technique**
- 5 Organisation interne du groupe
- 6 Coûts
- 7 Conclusion

## Paradigme client / serveur



- **Client** : Navigateur Web -> Requête URL au Serveur.
- **Serveur** : Réception de la requête -> retour de la page demandée.
- **Flask** : framework web choisi.

## Templates HTML

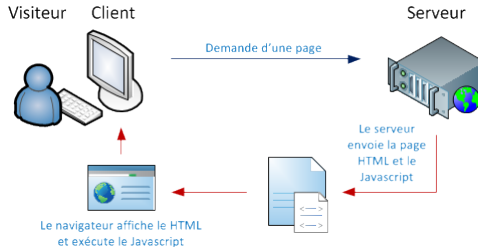


- **Template** : Page html formatée générique, avec des balises.
- **Flask** :  
Création de la page html finale.  
Balises peuvent contenir du code Python.

## Enjeux

Pages dynamiques - Circulation des données.

## Les applets



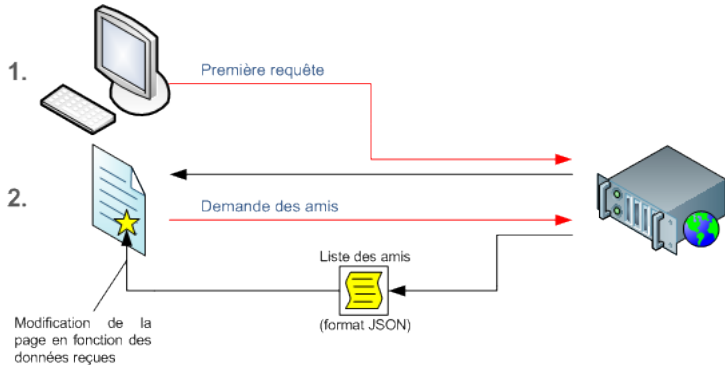
- 1 Balises `<script>` pour HTML.
- 2 Code exécuté côté client : **JavaScript**

**Dynamisme** : Animations, programmation événementielle, Html modifié en direct.

## Enjeu

Circulation des données nécessaire : Côté client  $\Leftrightarrow$  Côté serveur.

## Ajax (Asynchronous JavaScript and Xml)



- Rafraichissement **partiel** de la page.
- JSON : syntaxe des objets JavaScript, légère.
- Mécanisme asynchrone : fonctions **callback**

### ■ Divergence avec les spécifications

Problème : Utilisation des fichiers.

Caractéristique : Indépendance ressources Client / Serveur.

Solution : envoi Ajax.

### ■ Séries temporelles

Problème : Affichage incohérent.

Caractéristique : Plusieurs mesures à un temps donné.

Solution : Une série temporelle par arc distinct.

### ■ Performances (non résolu)

Problème : Affichage du jeu de données.

Caractéristique : Complexité non-linéaire.

Solution (éventuelle) : Module JavaScript personnel.



- 1 Architecture
- 2 Outils, langages de programmation
- 3 Fonctionnement de l'application
- 4 Bilan technique
- 5 Organisation interne du groupe**
- 6 Coûts
- 7 Conclusion

Module	Malek	Sonny	Jean-Didier
Gestion des flux			x
Fenêtre choix fichier			x
Fenêtre rôle et choix colonne	x		
Fenêtre résultats ADD		x	
ADD qualitatives			x
ADD quantitatives discrètes		x	
ADD quantitative continues		x	
Vérification format fichier	x		
Analyse contenu fichier	x		

- Groupe de trois personnes.
- Planning respecté.
- Travail en groupe.

- 1 Architecture
- 2 Outils, langages de programmation
- 3 Fonctionnement de l'application
- 4 Bilan technique
- 5 Organisation interne du groupe
- 6 Coûts**
- 7 Conclusion

Module	Estimation	Coût final
Gestion des flux	15	<b>98</b>
Fenêtre choix fichier	30	<b>72</b>
Fenêtre rôle et choix colonne	65	<b>180</b>
Fenêtre résultats ADD	100	<b>200</b>
ADD qualitatives	60	53
ADD quantitatives discrètes	100	89
ADD quantitatives continues	85	93
Vérification format fichier	30	35
Analyse contenu fichier	60	70
<b>Coût Total</b>	<b>545</b>	<b>850</b>

## Justifications

**Ajax** : Charge supplémentaire code.

**Séries temporelles** : Deux fonctionnalités supplémentaires.

**Filtrage des valeurs** : Demandées après l'écriture du cahier des charges.

## Plan

- 1 Architecture
- 2 Outils, langages de programmation
- 3 Fonctionnement de l'application
- 4 Bilan technique
- 5 Organisation interne du groupe
- 6 Coûts
- 7 Conclusion**

## Support :

Masse de données collectées sur un ***réseau physique*** assimilé à un ***graphe de flux***.

## Applet :

Traitement des données à l'aide d'analyses descriptives.

Résultats : statistiques et visuels.

## Améliorations possibles :

Interface-web : performances de l'affichage.

API extensibles :

- ADD : analyses multidimensionnelles et nouvelles représentations graphiques.