UNIVERSITÉ DE RENNES 1

DUDCL

Projet–

Mathieu Acher et Noël Plouzeau
Contact : mathieu.acher@irisa.fr

2015/2016

istic Informatique Électronique

Projet

L'objectif pédagogique du projet est d'une part de pratiquer les différentes techniques de développement vues dans les différents modules d'autre part de se mettre en situation en travaillant par petit groupe.

Vous allez :

- Utiliser le langage Java et quelques APIs pour traiter des données
- Transformer des données et donc s'appuyer sur différents formats/schemas (CSV, XML, JSON, etc.)
- Utiliser des technologies Web (HTML, CSS, JavaScript, etc.)
- Assurer une certaine qualité de code (notamment avec des tests unitaires)

En termes de méthode il faudra être capable de :

- Comprendre des exigences et les modéliser (par exemple avec UML)
- Travailler collectivement sur un code et projet complexe (par exemple en utilisant des outils de « versioning » comme git ou des outils de suivi de projet intégrés dans github)
- Respecter les délais impartis
- Présenter son travail au fur et à mesure pour valider les choix de conception et technologiques

Suivi et déroulement des séances

Nous (Noël Plouzeau et Mathieu Acher) encadrerons les séances mais nous ne serons pas systématiquement présents. Vous allez travailler en autonomie et vous nous solliciterez pour raffiner les exigences/valider les choix technologiques (plutôt Mathieu) ou valider les choix de conception (plutôt Noël).

Nous jouerons de manière artificielle le rôle du « client » et nous vous aiderons sur les détails techniques et le suivi de projet. Un point quotidien sera réalisé.

Evaluation

Au cours du projet, il sera demandé d'effectuer une tâche précise, non triviale dans un laps de temps prédéfini. Il est attendu :

- une implémentation
- des cas de test associés pour valider l'implémentation
- une documentation (README.md) du projet en anglais décrivant l'objectif, le résultat, la licence, les technologies utilisées, ainsi que l'architecture du projet
- une démonstration du résultat final : une vidéo (« screencast ») de 2' minimum (3' maximum)
- des instructions pour déployer le résultat (et notamment ré-exécuter la démonstration) seront également à inclure dans le README.md

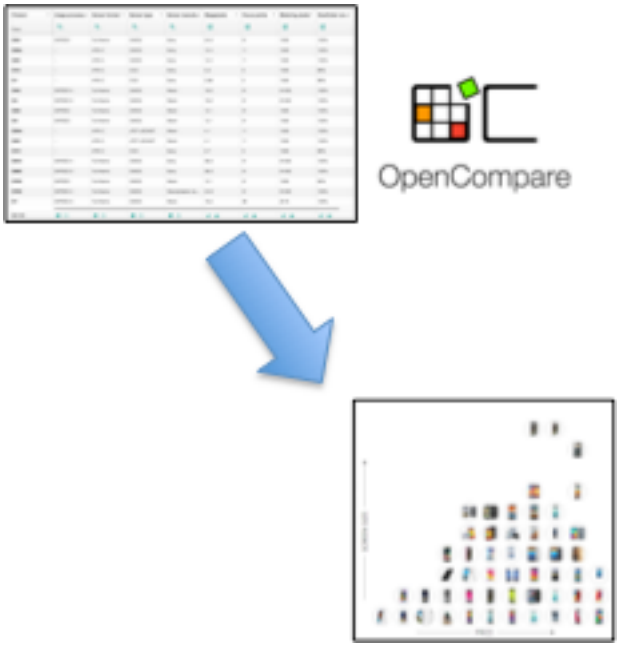
Le code sera nécessairement hébergé sur github, dans un repository privé ou publique. La date limite de rendu est le vendredi 20 mai 15h30.

Quelques remarques

- Le cahier des charges ci-dessous est volontairement imprécis et incomplet. C'est à vous d'identifier ces manques et de les éclaircir en proposant des solutions et en validant les choix avec le client.
- La qualité du code (tests compris) sera évaluée.
- Il est fortement conseillé d'avoir un « produit » qui fonctionne le plus rapidement possible pour ensuite itérer dessus et l'améliorer.
- Il n'est pas interdit de discuter, au cours du projet et avec le client, de la possibilité de ne pas implémenter certaines fonctionnalités

« OpenCompare et product charts »

opencompare.org a pour but d'aider une communauté d'utilisateurs à importer, éditer, visualiser, et exploiter des *matrices de comparaison*, typiquement pour choisir des produits dans un domaine donné.



Le but de ce projet est de générer, à partir d'une matrice de comparaison, une visualisation interactive appelée « product chart ».

Le traitement se fera en Java en réutilisant l'API de manipulation des matrices.

Un « product chart » consiste en :

- Un placement des produits sur un axe en 2 dimensions avec des abscisses X et des ordonnées Y. X et Y correspondant à des caractéristiques de produits (prix, taille, poids, performance, sécurité, emprunte écologique, etc.) ;
- Un panneau de configuration/filtrage qui dynamiquement est à même de mettre à jour le placement des produits (et d'éliminer les produits qui ne correspondent plus aux souhaits de l'utilisateur)

Des implémentations d'un tel système existent [1][3], on pourra s'en inspirer.

Si les produits ont une image associée, alors elle sera utilisée pour visualiser le produit (on utilisera un rond pour encadrer l'image); sinon un simple « point » sera affiché. Au survol de la représentation du produit, on peut lire son nom et ses caractéristiques.

La procédure de génération de « product chart » est configurable, par exemple, on peut choisir les caractéristiques en jeu pour les X et Y. Il est également possible d'associer une 3ème caractéristique en jouant sur la grosseur de la représentation. Enfin une 4ème dimension en jouant sur la couleur [1] est envisageable.

Au niveau de la technologie Web, on utilisera du HTML, du CSS et du JavaScript. La solution doit fonctionner sur n'importe quels appareils (téléphones portables, tablettes, ordinateurs, etc.). La procédure prendra en entrée une matrice de comparaison, et génèrera statiquement un ensemble de fichiers HTML, CSS, JavaScript. Le résultat doit être exploitable sur n'importe quel navigateur.

Deux librairies seront utilisées :

- NVD3 http://nvd3.org/
- Plot.ly https://plot.ly/javascript/

Votre solution devra fonctionner avec les deux librairies, i.e., votre procédure Java sera donc paramétrisable pour afficher le « product chart » soit avec NVD3, soit avec Plot.ly

A terme il est espéré que la procédure soit intégrée à opencompare.org. Ce travail d'intégration pourra être éventuellement abordé si le projet avance (très) bien.

Comment commencer ?

- Bien lire et comprendre le cahier des charges
- Pour charger une matrice de comparaison (PCM), se référer à [4] et mettre en place les outils (utilisation de git, Maven, et d'un IDE) pour que chaque membre du groupe puisse contribuer
- Explorer NVD3 et Plot.ly
- Mettre en place un « repository » github
- Assigner les tâches, planifier l'effort

Synthèse

Le « produit » final sera une procédure Java, qu'on peut lancer via java ou Maven, capable de prendre n'importe quel PCM en entrée et de produire un « product chart » (une page Web lisible dans n'importe quel navigateur). La procédure sera paramétrisable pour (1) spécifier le choix des caractéristiques (X, Y, etc.) (2) spécifier la librairie JavaScript visée (nvd3 ou plot.ly)

Il est attendu que cette procédure fonctionne au minimum sur un jeu de données de 1400 PCMs (http://tinyurl.com/DUDCL-projet1516) ainsi que sur deux cas d'étude : « Pokemon » et « Star Wars »

On attend quatre démonstrations dans le « screencast » :

- Une démonstration sur les 1400 PCMs : sélectionner parmi les 1400 un jeu de données dans lequel votre application est particulièrement intéressante… Montrer également les limites et les expliquer
- Une démonstration sur le cas d'étude Pokemon : vous utiliserez les CSVs disponibles ici https://github.com/phalt/pokeapi/tree/master/data/v2/csv
- Une démonstration sur le cas d'étude Starwars : vous utiliser les fichiers JSON disponibles ici https://github.com/phalt/swapi/tree/master/resources/fixtures
- Une démonstration où vous utiliserez une source de données de votre choix (eg données issues de l'open data)

La date limite de rendu est le vendredi 20 mai 15h30.

Une démonstration sera effectuée quelques minutes après et nous commenterons ensemble les résultats.

[1] Murashkin, A., M. Antkiewicz, D. Rayside, and K. Czarnecki, "Visualization and Exploration of Optimal Variants in Product Line Engineering", Software Product Line Conference, Tokyo, Japan, 2013. http://gsd.uwaterloo.ca/node/528

[2] opencompare.org

[3] productcharts.com

[4] https://github.com/OpenCompare/getting-started

[5] Génération d'un panneau de configuration pour pouvoir « configurer » dynamiquement (sélectionner des caractéristiques intéressantes avec des checkbox et des sliders) et ainsi avoir un impact sur le « product chart »

https://github.com/OpenCompare/editor/tree/master/app/configurator