

PCD TELECOM Nancy 2A 2013-2014

Projet *Open Data*

1. Contexte

Ce projet de deuxième année permet d'approfondir par la pratique les méthodes et techniques acquises en programmation objet, à travers la conception et le développement d'un produit logiciel en Java. A partir d'un sujet à la fois précis et ouvert, vous serez amenés à assurer les différentes phases inhérentes à la conduite d'un projet informatique en suivant une méthode de type agile.

2. Objectifs

Les objectifs principaux du module sont de :

- conduire en équipe un projet informatique de bout en bout, de la phase d'analyse des besoins jusqu'à la livraison d'un produit fini, selon un processus itératif,
- concevoir et développer en Java un produit logiciel ayant une architecture claire et de qualité, en particulier en faisant appel à des patrons de conception (design patterns),
- maîtriser les fonctionnalités des ateliers de développement, et se familiariser avec les outils de gestion de version et de suivi des bogues.

3. Sujet

Le projet consiste à concevoir, développer et tester une plateforme de planification de trajets multi-modaux s'appuyant sur des données ouvertes (de type *open data*).

Les deux sources de données considérées pour le projet seront celles fournies par la communauté urbaine du Grand Nancy (<http://opendata.grand-nancy.org/>) et celle fournie par la SNCF (<http://www.data.sncf.com/>).

Cette plateforme permettra aux utilisateurs de consulter les données relatives au transport et de planifier automatiquement des trajets d'un arrêt à un autre en utilisant un ou plusieurs modes de transport (trains, bus, tramways, vélos, autopartage autopi).

Elle sera composée d'une **application cliente** java (qui pourra être déployée sur plusieurs machines) permettant d'accéder aux différentes fonctionnalités de la plateforme, d'une **application serveur** permettant d'assurer son administration (minimaliste) par le web, et d'une **base de données**. L'application cliente ne pourra accéder directement à la base de données mais devra passer par l'application serveur.

Une part importante du travail consistera à analyser les différentes données disponibles en se focalisant dans un premier temps sur deux modes de transport (réseaux ferroviaires et inter-urbains) et à déterminer leurs limites pour définir les fonctionnalités envisageables pour la planification.

Les données nécessaires à la plateforme devront ensuite être transférées dans la base de données. Le transfert devra être autant que possible automatisé pour faciliter les mises à jour.

Chaque utilisateur disposera d'un compte (login et mot de passe) pour accéder à son dossier depuis l'application cliente. Ce dernier contiendra les coordonnées de l'utilisateur ainsi que ses préférences éventuelles quant aux modes de transport.

La gestion des comptes (création, modification, suppression) se fera par le web au travers de l'application serveur.

Une fois connectés à l'application cliente, les utilisateurs pourront à travers les fonctionnalités de base :

- consulter les données relatives au transport (par exemple, informations sur une ligne de bus, un arrêt, un train ou une gare),
- planifier des trajets en indiquant leurs contraintes (points de départ et d'arrivée, horaires, modes de transport possibles).

La planification pourra tout d'abord se limiter à des trajets directs, puis être étendue à des trajets indirects (avec changements). Si aucune réponse n'est trouvée, l'application devra faire des suggestions à l'utilisateur pour améliorer sa requête.

Le logiciel doit impérativement être modulaire et facilement extensible pour permettre l'ajout de nouveaux modes de transport.

Optionnellement (en fonction de l'avancement de votre projet), vous pourrez étendre la plateforme pour permettre les fonctionnalités avancées suivantes :

- mise à jour automatique des données ouvertes vers la base de données,
- extension aux autres modes de transport (pistes cyclables et autopartage),
- gestion de l'historique des requêtes utilisateurs.

Les fonctionnalités implantées dépendront de votre analyse des données ouvertes. Elles pourront donc varier d'un groupe à un autre, et ce en fonction des contraintes fixées par votre client (en l'occurrence, votre encadrant).

4. Déroulement du projet

4.1 Les groupes

Les groupes seront impérativement constitués de 4 étudiants.

Les étudiants d'un groupe peuvent appartenir à des groupes de TD différents.

Les groupes rencontreront leur encadrant à chaque itération et fourniront systématiquement un produit fini. La fréquence des réunions peut être augmentée en fonction des difficultés rencontrées par le groupe. Nous entendons par produit fini :

- une version compilée et *packagée* de l'application associée à une documentation contenant les instructions de lancement,
- l'ensemble du code source utilisé dans cette version de l'application, identifié dans le gestionnaire de version (utilisation de la fonction de *tagging* du gestionnaire de versions). Le code source doit être documenté (*javadoc*) et respecter un certain nombre de conventions de nommage (cf. section dédiée). Un document contenant les instructions de compilation de ce code source doit également être présent.

4.2 Le planning

Semaine 38 : constitution des groupes.

Semaine 40 : affectation des encadrants, première réunion avec l'encadrant.

Semaine 41 : première version des use cases, classement et planification des itérations.

La suite de la progression suivra une **démarche itérative** selon une méthode de type **agile** (*extreme programming*).

Semaine 43 : première itération (démonstration, mise à jour du planning).

Semaine 46 : deuxième itération (démonstration, mise à jour du planning).

Semaine 49 : troisième (dernière) itération (démonstration).

Semaine 51 : soutenance finale (démonstration et rendu du projet).

4.3 Les itérations

Les étudiants sont responsables de la gestion de leur projet. Ils doivent se répartir les rôles et les tâches pour mener à bien le travail de conception et de développement. Il appartient aux groupes de contacter par eux-mêmes leur encadrant pour fixer les différents rendez-vous.

La première étape consistera à analyser les besoins et à rédiger le cahier des charges, l'encadrant jouant le rôle du client. A partir du cahier des charges, les étudiants établiront un planning prévisionnel. Ce dernier sera réalisé en classant les différents cas d'utilisation par ordre de priorité. Une évaluation du temps d'implantation de chaque cas sera effectuée. Pour chaque itération, un sous ensemble des cas sera sélectionné pour être implanté.

Au cours d'une itération, il sera réalisé une analyse et la conception de ce qu'il faut faire. Les cas sélectionnés seront ensuite implantés. Des tests pourront être mis en place pour valider la réalisation des cas. **Chaque itération doit aboutir à une nouvelle version du logiciel final.**

L'analyse et la conception devront mettre en évidence les éléments principaux des cas à réaliser. Ils devront être mis à jour à chaque itération.

A la fin de chaque itération, il sera fait un bilan de ce qui a été fait par rapport aux prévisions. Un document d'une page ou deux sera fourni à chaque itération pour en faire le bilan (cas implantés, choix de conception). Les **principaux diagrammes de conception** seront également fournis et mis à jour (diagrammes de classes, diagrammes de séquences et éventuellement diagrammes d'états). La planification pourra être remise en cause à ce moment là.

Des **feuilles de temps** seront utilisées pour mesurer le temps passé par chacun sur le projet. Elles devront en outre mesurer le temps passé à l'implantation de chaque *use case*. Chaque feuille est individuelle. Si vous faites de la programmation par paire, chacun comptera le temps passé intégralement.

Vous ne serez pas notés sur le temps passé. Le compte de temps servira surtout à obtenir une évaluation objective du temps que vous avez consacré au projet.

4.4. Soutenance finale (démonstration)

Les soutenances finales se dérouleront la semaine 51.

Elles consisteront en une démonstration du produit final sur une machine, qui devra mettre en évidence les fonctionnalités et l'ergonomie de la solution proposée.

Les étudiants n'ont pas à préparer de transparents. En revanche, ils devront remettre lors de cette démonstration un **rapport synthétique de 10 pages** environ. Ce rapport devra décrire les fonctionnalités du produit (3 pages), l'architecture du produit (6 pages) et un guide d'utilisation (1 page). Il devra également mentionner un lien (avec les accès) à partir duquel les enseignants pourront accéder au site du projet.

5. Evaluation

Les enseignants se réunissent périodiquement pour évaluer l'état d'avancement des différents projets. Le travail demandé peut varier d'un groupe à l'autre en fonction des opportunités ou des difficultés rencontrées lors du projet, mais ces différences sont bien évidemment prises en compte lors de l'évaluation finale.

5.1 Note de suivi

La **note de suivi** est donnée par votre encadrant de projet. Les critères principaux pour cette note seront les suivants :

- **qualité de l'implantation** : le logiciel fonctionne conformément au cahier des charges et aux évolutions décidées au cours du projet,
- **qualité de la conception** : le logiciel est construit de façon modulaire. Il suit une architecture Modèle-Vue-Contrôleur (MVC). La conception est justifiée,
- **qualité du code** : le code est écrit en respectant des règles communes et homogènes (nommage, indentation ...).

Cette note tiendra également compte de l'évolution du projet et du respect du principe des itérations.

5.2 Note de soutenance

La **note de soutenance** est donnée par un autre enseignant (autre que votre encadrant de projet). Elle prend principalement en considération la **qualité de votre démonstration**, les **fonctionnalités** et l'**ergonomie du produit final** ainsi que la **qualité du rapport** fourni.

5.3 Note finale

La **note finale** sera la moyenne de la **note de suivi** et de la **note de soutenance**.

La note finale sera la même pour tous les membres d'un même groupe. Tous les membres d'un groupe sont censés participer aux réunions avec leur encadrant. En cas de défaillance d'un ou plusieurs membres d'un groupe, les notes pourront être individualisées en tenant compte de cette participation et des feuilles de temps individuelles.

Les meilleurs projets pourront être présentés dans le cadre des journées portes ouvertes de Telecom Nancy.

6. Environnement de travail

Il n'y a pas de contraintes strictes sur l'environnement de développement intégré choisi. Nous vous conseillons l'un des deux suivants :

- Netbeans (<http://www.netbeans.org>)
- Eclipse (<http://www.eclipse.org>)

Par ailleurs, nous vous recommandons fortement d'aller regarder des outils tels que Ant (<http://ant.apache.org>), voire Maven (<http://maven.apache.org>) qui peuvent vous aider et simplifier la gestion de votre projet.

Vous serez amenés à utiliser le langage Java pour l'application cliente et le langage PHP (serveur web Apache et base de données MySQL) pour l'application serveur.

6.1 Convention de codage

Il vous est demandé de commenter votre code en respectant la syntaxe *javadoc* et de respecter autant que possible les conventions de nommage :

- <http://java.sun.com/docs/codeconv/CodeConventions.pdf>
- <http://gee.cs.oswego.edu/dl/html/javaCodingStd.html>

Les environnements de développement intégré peuvent vous aider à respecter ces conventions. La plupart proposent des guides (*templates*) pour suivre et vérifier ces conventions. Il existe des outils intégrés ou non tels que Checkstyle.

6.2 Hébergement du projet et gestion de versions

Le projet devra être hébergé sur la forge de l'école (<http://redmine.esial.uhp-nancy.fr>). Vous devrez donc créer un projet, y ajouter les différents membres de votre projet ainsi que votre encadrant universitaire.

A chaque fin d'itération, vous définirez une étiquette *tag* que vous associerez à la version correspondante dans le gestionnaire de versions. Cette étiquette respectera la nomenclature suivante : **Rnuméro-itération** où **numéro-itération** sera remplacé par le numéro de l'itération correspondante.

Ainsi à la fin du projet, il est évident que votre gestionnaire de versions doit pouvoir vous fournir cinq versions de votre application étiquetées (**R1**, **R2** et **R3**). Chacune de ces versions devant forcément compiler avec succès.

6.3 Documentation

La documentation devra être mise à jour régulièrement au cours du projet et sera accessible en ligne.

Vous pouvez soit la rédiger sous forme de pages *wiki* en utilisant le module intégré à la forge, soit la rédiger sous un autre format électronique et la mettre à disposition dans le gestionnaire de versions.

Dans le cas où vous rédigeriez votre document dans un format non conventionnel, merci de bien vouloir également fournir une version pdf.

6.4 Outils d'analyses de code utilisables

Vous pourrez être amenés à utiliser les outils d'analyses et de tests suivants :

- CheckStyle (pour vérifier les standards de codage)
<http://checkstyle.sourceforge.net/>
- JUnit (pour les tests unitaires)
<http://junit.org/>
- PMD (pour l'analyse de code)

<http://pmd.sourceforge.net/>

- JDepend (pour la qualité de la conception)
<http://www.clarkware.com/software/JDepend.html>

7. Travail personnel et honnêteté

Ne trichez pas! Ne copiez pas! Si vous le faites, vous serez lourdement sanctionnés. Nous ne ferons pas de distinction entre copieur et copié. Vous n'avez pas de (bonne) raison de copier. En cas de problème, nous sommes prêt à vous aider. Encore une fois: en cas de doute, envoyez un courriel à vos enseignants, ça ne les dérange pas.

Par tricher, nous entendons notamment :

- Rendre le travail d'un collègue avec votre nom dessus,
- Obtenir une réponse par Google ou autre et mettre votre nom dessus,
- Récupérer du code et ne changer que les noms de variables et fonctions ou leur ordre avant de mettre votre nom dessus (« *Moving chunks of code around is like moving food around your plate to disguise the fact that you haven't eaten all your brussel sprouts.* »),
- Permettre à un collègue de *s'inspirer* de votre travail. Assurez vous que votre répertoire de travail n'est lisible que par vous-même.

Il est plus que très probable que nous détectons les tricheries. Chacun a son propre style de programmation, et personne ne code la même chose de la même manière. De plus, il existe des programmes très efficaces pour détecter les similarités douteuses entre copies (MOSS, <http://theory.stanford.edu/~aiken/moss/>).

En revanche, il est possible (voire conseillé) de discuter du projet et d'échanger des idées avec vos collègues. Mais vous ne pouvez rendre que du code écrit par vous-même. Vous indiquerez dans votre rapport toutes vos sources d'inspiration (comme les sites Internet de vulgarisation de l'informatique que vous auriez consultés).

Bon courage pour votre projet.

L'équipe pédagogique