GL MIAGE - PROJET

Le but de ce projet est de créer une application de calcul d'itinéraire pour terminal mobile. Les objectifs du projet sont de mettre en pratique ce que nous avons vu en cours:

- *travailler à plusieurs (groupes de 4) sur le développement d'une application,*
- *réussir à livrer un produit dans les temps,*
- *avoir le code le plus fiable et le plus maintenable possible.*

Spécifications

On veut offrir une application pour terminal mobile qui permette de calculer un itinéraire dans le métro parisien. On supposera que le terminal mobile supporte la géolocalisation. Vous ne devez pas coder la partie *front-end* (l'application mobile), mais vous devez coder l'application *back-end*.

À partir de la position du terminal, on souhaite que l'utilisateur indique l'endroit où il désire se rendre, et l'application calculera alors le meilleur itinéraire en fonction de critères tels que:

- l'itinéraire le plus rapide en temps (l'utilisateur a rendez-vous à une heure précise),
- l'itinéraire avec le moins de changements de lignes (l'utilisateur doit porter des objets encombrants),
- l'itinéraire passe par certains points donnés par l'utilisateur (l'utilisateur veut passer par le Trocadéro).

On prendra également en compte les possibles incidents:

- problème à une station: il faut éviter toutes les lignes qui passent par cette station,
- problème sur une voie (entre deux stations): on peut éventuellement trouver un chemin qui passe par la station de départ et/ou par la station d'arrivée.

Pour réaliser ce projet, vous devrez:

- modéliser les lignes de métro, et les stations (un sous-ensemble pourra convenir),
- à chaque fragment de ligne, un temps de parcours est associé, et un indicateur permettant de savoir s'il y a ou non un incident,
- à chaque station, un temps d'arrêt est associé, et un indicateur permettant de savoir s'il y a ou non un incident,
- on doit être capable à partir d'une position du terminal mobile de trouver la (ou les) station(s) la(es) plus proche(s), et de proposer un itinéraire en fonction de la station d'arrivée, des préférences de l'utilisateur et des incidents potentiels sur la ligne ou aux stations,
- on supposera qu'il y a dix rames par heure qui partent de la station de départ.

Objectifs et évaluation

Les objectifs principaux du projet sont les suivants:

- travailler à plusieurs (groupes de 4) sur le développement d'une application,
- réussir à livrer un produit dans les temps,
- avoir le code le plus fiable et le plus maintenable possible.

Seule une version en ligne de commande devra être fournie. Éventuellement, si vous avez le temps, une version graphique pourra être développée (bonus).

De même, pour pouvoir utiliser JUnit sans avoir à utiliser de Mock, vous stockerez en dur le maillage du réseau de métro (et non dans une base de données).

Voici les différents points qui serviront à l'évaluation de votre travail:

1/ réponse aux besoins exprimés

2/ conception

- cohésion et couplage des composants,
- adaptabilité des composants (ajout d'une station, d'une ligne, d'un incident...)

3/ réalisation

- tests, résultats des tests,
- cohésion des composants,
- couplage des composants,
- lisibilité/simplicité du code,
- structuration du projet.

4/ utilisation optimisée d'un SCM (Git) et de la méthode Scrum.

Vous disposez de 6 semaines pour réaliser ce projet.

Vous devez penser ces 6 semaines comme 6 sprints. Pour chacun de ces sprints, vous devrez donc déterminer le *backlog*, créer la liste des tâches, vous engager sur une ou plusieurs tâches, faire un *burndown chart* à l'issue de chaque séance, et repartir sur un nouveau sprint.

Pour chaque tâche, vous devrez réaliser les tests unitaires vous permettant de valider les composants, et de refactoriser si besoin en toute « sécurité ».

Travail à rendre

On vous demande de rendre votre projet en incluant les composants suivants:

- les sources (l'utilisation d'un dépôt Git est exigé),
- les tests unitaires,
- les *backlogs* précis de chaque sprint,
- les tâches, l'estimation, et leurs développeurs (2 maxi sur une tâche),
- les *burndown charts* à l'issue de chaque sprint,
- la version fonctionnelle obtenue à l'issue de chaque sprint.

Vous aurez donc une archive dont la structure ressemblera à ceci:

- Sources
- Tests
- Sprint 1: backlog, tâches, estimations, burndown chart, version livrée à l'issue du sprint
- Sprint 2: …