GrrGrrCar

1 - Objectif

Le but de ce projet est de concevoir un système de co-voiturage, GrrGrrCar, qui permet de joindre l'utile et l'agréable en profitant d'un trajet en voiture pour se plaindre de tout. Votre application doit permettre à un ensemble d'utilisateurs de proposer ou de réserver ces trajets en voiture d'une ville à une autre.

2 - Fonctionnalités demandées

L'architecture de votre système doit reposer sur un serveur central qui exposera des services avec une API REST.
Cette API devra proposer au minimum :
- d'enregistrer un nouveau compte (login/password+infos) et de s'authentifier avec ce compte,
- d'ajouter un trajet d'une ville à une autre, avec une adresse, une date et une heure de départ, un nombre de places, et un tarif,
- de rechercher des trajets par ville de départ, ville d'arrivée et de date. Si aucune correspondance exacte n'est trouvée, de proposer les trajets les plus « proches » à cette date,
- de demander à réserver une place sur un trajet disponible,
- pour un conducteur, d'accepter ou de refuser une demande de place sur un trajet qu'il a proposé,
- de valider la fin du trajet pour les passagers,
- de mettre une note (de 0 à 5 Grr) et un éventuel commentaire pour chaque participant au trajet (le conducteur peut noter les passagers et les passagers peuvent noter le conducteur,
- d'afficher les informations sur tout utilisateur (inscrit depuis, notes reçues, commentaires, …)

Vous devrez proposer au moins un client pour votre API.

3 - Travail à réaliser

Vous devez constituer (librement) des groupes de 4 personnes et vous inscrire sur https://orleans.miage.fr/confluence/display/prjm1sem2/Projet+2018
Chaque groupe devra développer le(s) application(s) décrite(s), en utilisant Maven et le dépôt Git mis à sa disposition.

Pour (dernier délai) le samedi 14 avril 20h, chaque groupe devra me rendre par mail un document de proposition d'architecture technique (2 pages max) pour l'application, avec le mode de fonctionnement global et la liste des frameworks et services externes envisagés. Ces choix techniques devront être validés (avec d'éventuelles modifications) avant de pouvoir être utilisés dans la réalisation du projet.

Pour le code, l'accent devra être mis sur une architecture modulaire et extensible à l'aide d'interfaces, la répartition équitable des implémentations à réaliser au sein de l'équipe, la documentation du code (javadoc) et surtout des APIs REST (swagger), la réalisation de tests (unitaires, fonctionnels,…) et la qualité générale du code.

Pour le rapport d'étude final, vous devrez expliciter les services externes (google, open street map, …) utilisés dans votre solution.

Vous serez évalués sur **l'ensemble de ces critères**.

A la fin du projet, vous devez fournir :
- le code source de toutes les applications (serveur et client(s)) sur le dépôt Git,
- un (ou plusieurs) pom.xml permettant de construire la ou les applications,
- un shell script permettant d'initialiser des données dans le système (commandes curl)
- un jeu de test en SoapUI pour votre API,
- une documentation technique (PAS une doc utilisateur !) décrivant l'architecture générale et le fonctionnement global de votre application, l'API d'accès ainsi que l'ensemble des classes et les liens entre elles (Environ 10 pages, à sauver en .doc/.odt dans le répertoire "documentations" du dépôt de votre projet)
- une README.md ou README.ad expliquant comment déployer le serveur et lancer le(s) client(s).

Ce projet donnera lieu à une soutenance orale de démonstration de 20 minutes le 24 mai.