Génie logiciel avancé - 2016
TD n°3
Système de planification des vols

La présentation rendue les semaines précédentes doit être complétée par toutes les informations nécessaires à la bonne compréhension du projet par un intervenant externe.

L'objectif de ce TD est de compléter ce livrable et de mettre en place votre environnement de développement pour le vendredi 12 février au plus tard.

Question 1 – Mise à niveau

Si nécessaire, vous devez intégrer les remarques qui ont été formulées suite à votre rendu de la semaine précédente.

Question 2 – Gestion du code source

Chaque groupe doit créer un dépôt Git afin de déposer l'ensemble des codes sources relatifs au projet. L'utilisation d'une interface graphique comme SourceTree[1] est très fortement recommandée.

Le dépôt doit être créé sur un des plateformes suivantes : GitHub, le serveur Git de l'université, ou encore BitBucket (en utilisant un compte académique).

Un accès au dépôt doit être inclus dans la présentation du projet afin que vos enseignants puissent accéder à l'ensemble des codes sources.

L'ensemble des éléments des questions suivantes doit être déposé sur le dépôt.

[1] https://www.sourcetreeapp.com/

Question 3 – Couche de persistance

A partir des objets métiers choisis durant les TP précédents ainsi qu'à partir de la liste des web-services que vous avez définis, vous devez spécifier l'interface de la couche de persistance.

Un résumé de ces interfaces doit être inclus dans la présentation globale du projet.

Cette réalisation doit être faite en Java en implémentant :

- les classes correspondantes aux objets métiers ;
- les interfaces qui définiront les méthodes de vos DAO.

Pour cela, vous pouvez vous appuyer sur l'exemple suivant qui présente un objet métier ainsi qu'une interface :

```
public class Action {

	public String title;
	public String content;
	public String username;
	public DateTime deadLine;

}


public interface ActionDao {

	/**
	 * @return this list of actions
	 */
	List<Action> getActions();

	/**
	 * @param username
	 * @return the list of actions assigned to a specific user.
	 */
	List<Action> getActions(String username);

	/**
	 * @return this list of actions with an overdue deadline
	 */
	List<Action> getOverdueActions();

}
```

Question 4 – Bouchons

Un bouchon est une implémentation d'un composant dans le seul but de rendre accessible une interface.

Afin de développer les IHM sans attendre la vraie implémentation des web-services, vous devez réaliser un bouchon pour ce composant. Il s'agit de réaliser une implémentation extrêmement simple qui permet de démarrer un serveur web et de rendre disponible l'ensemble des web-services que vous avez spécifiées durant le TD précédent.

Vous pouvez récupérer via GitHub un projet Java qui met en place Jetty et Jersey via l'url suivante : https://github.com/ktekkal/jetty-jersey

Le plus simple est d'importer ce projet dans Eclipse (une version intégrant un support Maven) comme par exemple la version « Eclipse IDE for Java Developers »[2].

Deux web-services sont disponibles une fois le serveur lancé :

- méthode GET : http://localhost:8080/example/aircraft
- méthode POST : http://localhost:8080/example/aircraft

[2] http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/mars1