Cahier des charges

Projet “Formations”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Date | Rédigée par | Description | Statut |
| 1.0 | 31/05/2021 | Nicolas Rousset |  | Approuvé |
|  |  |  |  |  |
|  |  |  |  |  |

# Introduction

Félicitations, vous avez été sélectionnées sur appel d’offre pour développer le projet “formations”. Vous avez été réparties en 4 équipes :

* Equipe 1 : responsable de la tâche « remboursement »
* Equipe 2 : responsable de la tâche « annulation »
* Equipe 3 : responsable de la tâche « replacement formateur »
* Equipe 4 : responsable de la tâche « fusion formations »

Ce cahier des charges s’applique à toutes les équipes, le différences étant détaillées dans les sous-parties (de toute façon, tout le monde doit faire des tests unitaires).

Pour ce TP, vous serez sur la surveillance d’un lead developper. Vous ne pouvez pas pusher directement sur la branche « main », il vous faut passer par un PR (pull request). Les PR se lancent depuis l’interface du projet sous github, c’est une demande pour merger une branche dans la branche main. Cela correspond aux bonnes pratiques réelles, ce qui peut changer c’est qui est habilité à effectuer la code review.

# Step 1 : mise en place du projet

Le repository du projet est :

<https://github.com/Aenori/TP_real_project>

Les comptes github qui m’ont été indiqués ont reçus une invitation, pour l’accepter, il faut aller ici :

https://github.com/Aenori/TP\_real\_project/invitations

La première étape est donc de cloner localement le repository :

git clone [git@github.com](mailto:git@github.com):Aenori/TP\_real\_project.git

### Notes sur les commandes git

Les principales commandes dont vous allez avoir besoin sont :

* git commit
* git push
* git pull
* git merge main
* git add (sert également à résoudre les conflits => dans certains IDE elle est également appelée par “marquer comme résolu”)

Note : il est recommandé de pratiquer directement git en command line. Il est effectivement toujours possible de ne pas pouvoir utiliser son IDE préféré (ssh cela vous dit quelque chose ?), et les IDE ne font qu’appeler git.

# Step 2 : Le projet !

## Task n°1 – ajouter le nom de votre équipe au fichier « doc/equipe.csv »

Il y a un fichier csv (comma separated value) dans le sous répertoire doc. Ajoutez une ligne contenant votre numéro d’équipe, éventuellement un nom d’équipe et les noms de l’équipe.

Par exemple :

1;Les Avengers;IronMan;Hulk ;Captain America

Ajoutez votre nom d’équipe à la fin du fichier.

Cette task ne nécessite pas de tests unitaires.

## Task n°2 – ajouter une méthode *is$task* à l’objet Formation

Ajouter une méthode is$task à l’objet org.formation.model.Formation, qui renvoit True, si cette fonction a été concernée par votre fonctionnalité. Décliné par fonctionnalité, cela donne :

|  |  |
| --- | --- |
| Equipe 1 : Remboursement | Créer une méthode publique isRembourse qui renvoit True si la formation est remboursée, i.e si son statut est égal à "Remboursee" |
| Equipe 2 :  Annulation | Créer une méthode publique isAnnulee qui renvoit True si la formation est remboursée, i.e si son statut est égal à "Annulee" |
| Equipe 3 :  Remplacement formateur | Créer une méthode publique hasFormateurRemplace qui renvoit True si le formateur a été remplacé, i.e si son statutFormateur est égal à "Annulee" |
| Equipe 4 :  Fusion formation | Créer une méthode publique isFusionnee qui renvoit True si la formation a été fusionnée, i.e si son statut est égal à "Fusionnee" |

A noter que dans un vrai projet, il est préférable d’écrire les noms de fonction 100 % en anglais. Notamment car les frameworks utilisent des convention de nommage en anglais (getter / setter / etc ...)

Ajouter un test unitaire qui vérifie les 2 cas (un cas où la valeur renvoyée est True, un cas où elle vaut False)

## Task n°3 – ajouter un exemple dans les fixtures

Ajouter dans le fichier Fixture.java un exemple de formation qui répond True à la fonction développée à la task 2.

Ajouter un test unitaire qui vérifie que l’appel à Formation.getAll() renvoit bien exactement une liste contenant exactement un objet Formation renvoyant True.

NB : le terme fixture est officiel, et consiste exatement à cela, fournir un jeu de test pour le développement et/ou les tests unitaires, mais l’implémentation choisie ici n’est pas très orthodoxe.

## Task n°4 – création d’un bean dédié à la task et d’un fonction *selectIs$task*

Créez un bean (qui dans le cas présent sera surtout un namespace) dans le package org.formation.bean correspondant à votre tâche (ie RemboursementBean / AnnulationBean / RemplacementBean / FusionBean) qui contient une fonction select$task (ie selectRemboursees / selectAnnulation / selectRemplacementsFormateur / selectFormationsFusionnees) qui renvoit la liste de toutes les formations répondant au critère implementés à task numéro 3.

Vous accédez aux formations par la méthode Formation.addAll() (qui est en quelque sorte un mock pour hibernate)

Ajoutez le test unitaire correspondant, en créant un fichier du même nom que votre Bean.

## Task n°4 bis – création d’un **test unitaire avec Mock**

Ajoutez au fichier de test unitaire précédemment créé un test unitaire qui « mock » la méthode Formation.addAll() pour renvoyer une liste contenant 2 éléments.

## Task n°5 – visualisation HTML de la sélection

Modifier la page /$task (Remboursement /Annulation / Remplacement / Fusion) pour qu’elle affiche la liste générées à la page précédente. Vous avez besoin de changer 2 fichiers :

$task.java (par exemple RemboursementServlet.java)

$task.jsp (par exemple Remboursement.jsp)

Pour le coup on ne demande pas de tests unitaires.

## Task n°6 – ajouter la fonction do$task en java

## Task n°7 – appeler la fonction do$task depuis un formulaire HTML

## Task n°8 – appeler la fonction do$task **en AJAX**

## Task n°9 – mise à jour de l’affichage HTML en javascript

## Task n°10 – ajouter l’évènement à l’historique