TP N° 6
Construction d'une API Java avec Gradle

Enoncé du TP

Il s'agit de construire une API de calcul matriciel et de la déployer.

1. Partie 1

1.1 Création du projet

1. Créez un projet Java / Gradle sous *IntelliJ IDEA* en spécifiant *com.example* comme nom du groupe et *1.0* comme version (ces informations peuvent être modifiées dans le fichier *build.gradle*).
2. Intégrez le code source et les tests unitaires (le dossier src), les dépendances du fichier *build.gradle* et le dossier *Features* qui contient le *feature file*.

1.2 Tests unitaires

1. Lancez l'exécution des tests unitaires avec Gradle.
2. Visualisez le rapport de tests généré dans le dossier **build/reports/tests**.

1.3 Rapport de tests Cucumber

1. Intégrez et configurez le plugin de reporting de Cucumber (La génération du rapport doit se faire dans le dossier **build/reports/cucumber**
2. Relancez l'exécution des tests unitaires et vérifiez la génération du rapport dans le dossier spécifié.

1.4 Code review

1. Ajoutez le Plugin Gradle de Jacoco et de SonarQube.
2. Ajoutez une dépendance entre les tâches *test* et *sonarqube* (le lancement des tests implique le lancement de l'analyse du code après).
3. Démarrez *SonarQube* et lancer le build du projet.
4. Visualisez les résultats d'analyse sur l'interface web de SonarQube.

1.5 Documentation

1. Lancez la génération de la documentation de l'API.
2. Visualisez le résultat.

1.6 Génération du projet

Créez une tâche *generateMatrixAPI* qui copie dans un dossier *MatrixRelease/Matrix_v_1.0* (ce dossier ne doit pas être créé dans le dossier build) :

1. Le dossier *reports* du dossier build.
2. Le dossier *docs* du dossier build.
3. Le dossier *libs* du dossier build.
4. Ajoutez la contrainte de dépendance entre *generateMatrixAPI* et *build et javadoc* (pour chaque lancement de *generateMatrixAPI* les tâches *build* et *javadoc* doivent être lancées avant).

1.7 Déploiement du Jar

1. Créez un compte sur *https://mymavenrepo.com/*.
2. Activez l'authentification HTTP en écriture
3. Créez un utilisateur en écriture.
4. Ajoutez le plugin de déploiement Maven.
5. Configurez *publishing* et *publications* et lancer la tâche *publish* pour le déploiement du Jar.
6. Ajoutez la contrainte de dépendance entre la tâche *publish* et *generateMatrixAPI* (pour chaque lancement de *generateMatrixAPI* la tâche *publish* doit être lancée après).
7. Lancez la tâche *generateMatrixAPI* et vérifiez le déploiement du fichier Jar.

1.8 Utilisation de l'outil de collaboration Slack

Il s'agit de notifier l'équipe de développement qui utilise l'outil *Slack* pour collaborer. Pour cela :

1. Créez un compte dans https://slack.com/
2. Créez un *Workspace* et une *Channel*.
3. Activez dans slack Incoming Webhooks.
4. Ajoutez le plugin *Slack.*
5. Configurer le plugin *Slack.*
6. Testez l'envoi d'un message en lançant la tâche *publishToSlack*.
7. Créez une dépendance entre la tâche *publishToSlack* et *publish* (l'envoi du message sur Slack se fait après le déploiement).

2. Partie 2

Suivez les étapes suivante pour tester l'intégration de la librairie déployée:

1. Créez un nouveau projet Gradle sous *IntelliJ IDEA*
2. Ajoutez un utilisateur en lecture dans *mymavenrepo.*
3. Ajoutez le repository *Maven* dans *repositories*
4. Ajoutez dans *dependencies* la librairie déployée en suivant l'une des deux syntaxe:
    a. implementation group: '*_group*', name: '*_name*', version: ***'_version'***
    b. implementation "*_group*:*_name*:*_version'*

_group: le nom du groupe, ***_name***: le nom de l'API, ***_version***: la version

3. Intégration des Plugins

Plugin	**Liens web**	**Id du Plugin**
JaCoCO	https://docs.gradle.org/current/userguide/jacoco_plugin.html	jacoco
Reporting Cucumber	https://github.com/SpacialCircumstances/gradle-cucumber-reporting	
Sonarqube	https://plugins.gradle.org/	org.sonarqube
Maven	https://mymavenrepo.com/docs/gradle.auth.html	maven-publish
Slack	https://plugins.gradle.org/	net.madeng.slack

Configuration Slack WebHook

https://api.slack.com/incoming-webhooks