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 obligatoire

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.
2. Intégrez le code source et les tests unitaires.

1.2 Tests unitaires

1. Lancez l'exécution des tests unitaires avec Gradle.
2. Visualisez le rapport des tests.

1.3 Qualité des tests unitaires

1. Intégrez le plugin *JaCoCo* (pour la couverture des tests).
2. Lancez le calcul de la couverture du code en générant un rapport HTML.
3. Visualisez le rapport généré.

4. Spécifiez la couverture minimale des tests à 80%.
5. Lancez le test de vérification de la couverture minimale des tests.
6. Intégrez le plugin *Pitest* pour les tests de mutation.
7. Configurez dans la tâche *pitest* les classes du code et celles du test (*targetClasses* et *targetTests*).
8. Lancez les tests de mutation.

1.4 Code review

1. Ajoutez le Plugin Gradle de Sonarqube.
2. Démarrez *SonarQube* et lancer la commande Gradle pour analyser la qualité du code.
3. Visualisez les résultats.

1.5 Mise à jour des tâches de vérification

Ajoutez les dépendances entre les tâches *pitest*, *jacocoTestReport*, *jacocoTestCoverageVerification, sonarqube* et *check* (les tâches *pitest*, *jacocoTestReport*, *jacocoTestCoverageVerification* et *sonarqube* doivent être lancées après le lancement de la tâche *check*).

1.6 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 même dossier du projet) :

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 de 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).

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 optionnelle

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