TD Intégration continue

Introduction

Le but de ce TD est de mettre en place une série d'étapes d'automatisation à partir d'un projet Android existant.

Le projet que vous utiliserez :

- se compile et s'exécute
- possède des tests unitaires

Partie 1 : Automatisation

Objectifs

Le but est de vous passer d'Android Studio pour compiler un fichier apk à partir du code source. Il s'agit de la première étape d'automatisation.

1.1 Mise en place

1. Télécharger et extraire l'archive contenant le projet
2. Importer le projet dans Android Studio (`Import Project (Gradle ...)`)
3. Compilez et exécutez sur un simulateur

Si à l'étape 3, vous avez des erreurs sur des dépendances nécessaires, installez les depuis Android Studio.

1.2 Compiler sans Android Studio

La compilation et packaging d'un fichier `apk` à partir du code source s'effectue avec le moteur de production Gradle.

1. Faites un clean du projet depuis Android Studio.
2. Fermez Android Studio.
3. Lancez un terminal
4. Placez vous dans le répertoire du projet
5. Trouvez la ligne de commande permettant de compiler et packager un apk.
6. Vérifiez que votre apk peut s'installer sur un emulateur (drag & drop sur un émulateur ouvert)

1.3 Script d'automatisation

Le but est de réaliser un script shell `.sh` qui permette de démarrer les tests, lancer la compilation, renommer le fichier `apk` et le déplacer dans un répertoire personnalisé.

1. Créer un fichier `build-apk.sh` à la racine du projet
2. Lui donner les permissions d'exécution
3. Faire en sorte que le script exécute dans l'ordre : `clean`, `tests`, `compilation`.
4. Faire en sorte qu'après la compilation, le script déplace le fichier `apk` produit au chemin `build/apk` depuis le répertoire racine

5. Faire en sorte que le script renomme le fichier apk au format `sample-dmYHMS.apk` (jour/mois/année/Heure/Minute/Seconde) avant de le placer dans le bon répertoire

Partie 2 : Travis-CI

Objectifs

Le but est versioner le projet et de le pousser sur un repository GitHub et qu'à chaque commit un build se lance sur Travis-CI qui va exécuter les tests et démarrer la compilation.

2.1 Créer le projet sur Github

1. Se connecter sur votre compte personnel GitHub
2. Créer un nouveau repository
3. Pousser les sources du projet sur le repository nouvellement créée

2.2 Lier le projet à travis-ci

1. Créer un compte Travis-CI à partir de votre compte GitHub
2. Effectuer un build manuel à partir de la branche master qui va normalement échouer

2.3 Faire fonctionner un build avec travis-ci

1. Créer une nouvelle branche
2. Créer le fichier `.travis.yml` à la racine du projet
3. Configurer le `yml` pour qu'il puisse dans l'ordre effectuer les opérations suivantes à chaque commit : `clean`, `tests`, `compilation`. Aidez vous de la documentation de Travis pour les projets Android.
4. Vérifier le bon fonctionnement en faisant des commits sur votre branche et en vérifiant sur le portail Travis-CI le résultat de vos builds.

Partie 3 : Fastlane

Objectifs

Le but est d'utiliser l'outil d'automatisation en ligne de commande Fastlane pour se passer d'un script shell et simplifier la mise en place de chaines d'automatisation qui seront ensuite appellées par votre outil d'intégration continue.

3.1 Mise en route fastlane

1. Installer ruby via `homebrew` sur votre machine : `brew update && brew install ruby`
2. Installer fastlane : `sudo gem install fastlane -NV --no-ri --no-rdoc`
3. Tester la lane fastlane dans le projet en exécutant depuis un terminal à la racine du projet `fastlane test_lane`

3.2 Lane basique

L'objectif sera d'effectuer le même script shell (1.3) mais en utilisant Fastlane. La plupart des actions faites en shell peuvent être écrites plus simplement en ruby.

1. Créer une lane

2. Exécuter une tâche gradle simple avec la lane (clean par exemple)
3. Exécuter la séquence des tâches : `clean`, `compilation` dans la lane
4. Récupérer le dernier apk produit et le déplacer dans `build/apk` en le renommant `[project-name]-[date].apk`.