OCTO Technology
There is a better way

Architecture Android
Les spécificités de l'OS

Philippe Prados

50, avenue des Champs-Elysées
75008 Paris - FRANCE

Tél : +33 (0)1 58 56 10 00
Fax : +33 (0)1 58 56 10 01
www.octo.com

1er Cabinet d'Architecture

MÉTHODES DE DÉVELOPPEMENT AGILE

Great place to work®

L'ENGAGEMENT

LE PRAGMATISME **NOS VALEURS** FORFAITAIRE

DE BEAUX SUCCÈS DEPUIS 13 ANS MULTI TECHNO

L'EXCELLENCE ET LE PLAISIR

Partage de nos savoirs

Offre d'OCTO

- OCTO dispose d'une offre originale pour bâtir des SI et des applications innovantes en phase avec la complexité des enjeux métiers de ses clients.
- Elle se décline autour de 4 thématiques.

Architecture & Technologie

- Définition d'architecture
- Audits techniques et applicatifs
- Aide au choix de solutions / Technologies
- Expertises : IHM, Cloud, NoSQL, BI, sécurité
- Task force

Accompagnement au changement

- Accompagnement / coaching projets en mode agile (MOA+ MOE + prod)
- Mise en place de pratiques Lean IT
- Mise en place de démarches de tests et d'industrialisation

Pilotage du SI

- Réalisation de schémas directeurs IT
- Réalisation d'audits de SI
- Application Portfolio Management
- Gouvernance et urbanisme

Réalisation de projets stratégiques & innovants

- Brainstorm et conception de produits
- Spécifications et développements agiles itératifs
- Mise en production et suivi du produit

- Philippe PRADOS (pprados@octo.com)

- Consultant
- Gplus.to/pprados
- @pprados
- LinkedIn

- Quels sont les principes ayant guidés à la conception d'Android ?
- Pourquoi le framework est-il conçu comme cela ?
- Quels sont les avantages des choix initiaux ?

- Exécution dans un environnement extrêmement contraint :
  - CPU peu puissante
  - Peu de mémoire
  - Réseau chaotique
  - Batterie à économiser au maximum
  - Traitements prioritaires
    (la réception d'un appel)
- Il faut économiser la moindre ressource !
- Sécurité in-board (protection entre et contre les applications)
- Cela apparaît dans tous les choix d'architecture du framework

IOS à les mêmes contraintes

- Activités
- Fournisseurs de contenus
- Broadcasts
- Services
- IDL

- L'architecture a résisté à cinq ans de concurrence forte
  - Sans remise en cause des fondamentaux
- Évolution à la marge
  - Interface utilisateur
  - Notion de « plusieurs utilisateurs »
  - Renforcement de la sécurité

IOS utilise une approche totalement différente

- Basées sur le modèle des pages Web.
- Identifiées par une « Intention », sorte de requête HTTP enrichie (URL, action, type mime, extra)
- Cycle de vie fragile (comme les pages Web).
- Une activité meurt lors de l'affichage de la suivante ou du basculement de l'écran.
- Mais, si Android est sympa, il peut la garder en cache avant de la reprendre.
- Les données d'une activité ne sont pas pérennes
- Android mémorise l'historique de navigation entre les activités (d'où le bouton « retour »)
- Ainsi,
  - une activité peut déclencher l'affichage d'une activité d'une autre application
  - Et il est possible de revenir à l'activité précédente (même d'une autre application).

- Permettent d'exposer des données aux autres applications via une API de type REST.
- Toutes les données sont identifiées par une URI (content://contacts/people/1)
- Doivent offrir les quatre services de manipulations classiques (création, modification, consultation, effacement)
- La consultation doit répondre à une requête et retourner un Curseur.
- L'implémentation est libre (SQLite interne, fichiers plats, requêtes Web, etc.)
- Ne permet pas les jointures entres fournisseurs de contenus.

- Fournisseurs de contenus pour les contacts, les images, les vidéos, les son, etc.
- Permet aux éditeurs de proposer des personnalisations au niveau design, sans casser la compatibilité.

- Composants à l'écoute d'événements génériques
  - Activation / désactivation Wifi ou Bluetooth
  - Installation d'une application
  - Push
  - Etc.
- Permettent de réveiller l'application lors d'événements majeurs.
- Une application n'a pas besoin d'être vivante pour recevoir un événement !

- Deux usages :
  - Offrir des traitements en tâche de fond
  - Permettre la communication entres objets de différents processus
- Éviter de perdre l'application pendant un traitement
- Reprise possible après un crash

- AIDL (Android Interface Definition Language)
  - Invocation de méthode entres applications
  - très utilisés par le framework Android (même si les développeurs l'ignorent)
  - Utilise un module noyau linux spécifique. Il injecte les éléments de sécurités.

IOS permet sous certaines contraintes, les traitements en tâche de fond.

- Dans le fichier `AndroidManifest.xml` de chaque application.
- Permet d'identifier les applications à déclencher, sans avoir à les exécuter (enregistrement passif)
- Ainsi, les applications peuvent réagir aux événements sans consommation de ressources
  - Ouverture d'une activité dont l'Intent correspond à un filtre
  - Réception d'un évènement broadcast
  - Démarrage d'un service

IOS: les applications doivent être activent pour réagir.

- Android gère le cycle de vie des applications :
  - Évite autant que possible de tuer une application
  - Mais n'hésite pas à le faire s'il a besoin de ressources complémentaires
- À minima, comme sur d'autres plates-formes mobiles, il n'y a qu'une seule application active à la fois, celle présentée à l'utilisateur.
- Lorsque l'utilisateur retourne à une application (bouton retour, sélection de l'application, etc.) il peut être nécessaire de la réinitialiser si elle n'était plus active (d'où un délai parfois long).
- Si un service est en fonctionnement, l'application n'est pas tuée, sauf nécessité absolue

IOS: même approche.

- Android sépare la notion de processus et la notion d'applications
- Une application peut être portée par plusieurs processus
- Plusieurs applications peuvent partager le même processus
- Il est même possible de lancer un OS Android complet dans un unique processus
- Les processus s'exécutent avec des utilisateurs différents
- C'est un élément de sécurité important pour isoler les applications.
- Android peut tuer un processus après l'avoir prévenu. Il en contrôle la consommation des ressources.

IOS: ne permet pas d'avoir plusieurs processus

- Le processus `system_app` porte le framework Android.
- Contient les informations de tous les fichiers `AndroidManifest.xml` des applications
- Permet de gérer le cycle de vie des autres processus
- S'il meurt, le téléphone reboot.

- Machine virtuelle utilisant un byte code orienté registre
- Op-code aligné sur la mémoire et optimisé par la résolution des offsets, lors de la première exécution
- Peux être mappé directement en mémoire
- GC asynchrone
- Compilation JIT basé sur le flux de traitement et non sur la structure du langage (détection des boucles dans le flux)

IOS utilise une compilation native, pas de GC

- De très nombreuses astuces sont utilisées pour optimiser le code
  - L'utilisation de constantes numériques à la place de chaîne de caractères
  - Une communication optimisée entre les processus
  - Exécution de requêtes SQL en mode batch
  - Etc.
- Pour accélérer le démarrage d'un processus, un machine virtuelle java (Dalvik) est initialisé avec une liste de classes, puis attend une commande pour se dédoubler, associer un utilisateur Unix et démarrer l'application.
- Ainsi, une partie du framework est partagée entre les processus

- Publication des applications sur le Play Store avec signature numérique
- Algorithmes de chiffrements disponibles (flux, fichier)
- Pas de conteneur chiffré disponible avant la version 4
- Isolation des applications (user Linux différent)
- Privilèges pour accéder aux services sensibles.
- Possibilité d'ajouter de nouveaux privilèges
- Présentation des privilèges AVANT l'installation de l'application

- Quelques vulnérabilités découvertes sur les applications `root` (de moins en moins) ou les surcouches constructeurs
- Maintenant, `root` n'a plus tous les droits

Apple est très stricte pour la publication d'applications

- Authentification
  - L'utilisateur du téléphone est considéré comme « autorisé »
  - Valide si mécanisme de blocage du terminal (pin)
  - Pour les traitements sensibles, demander confirmation d'un autre PIN
  - La dernière version d'Android propose le multi-compte
- Habilitation
  - Les applications utilisent des users linux différents
  - De nouveaux privilèges peuvent être déclarés par les applications
  - Habilitation pour tous, limitée aux mêmes auteurs des applications ou limitée au système.
  - Permet de partager des secrets entre applications du même auteur

IOS a une approche d'habilitation au premier usage.

- Répertoire de travail par application
- Droit limité à l'utilisateur associé à l'application (ou aux autres applications de même signature)
- Carte SD considérée comme publique (sinon il faut chiffrer les données)
  - Dernièrement, ajout d'un privilège pour avoir droit de lire la carte SD
- Chiffrement « gratuit » si l'application est installée sur le carte SD
  - Chiffrement associé au terminal
- Partage de fichier/flux
  - Possibilité de modifier les droits pour permettre un accès aux autres utilisateurs =>Risque d'exposer des fichiers sensibles
  - Passage de handle fichier d'une application à une autre (permet de ne pas exposer le fichier aux autres applications. Juste l'accès)
  - Depuis v4, possibilité d'ouvrir un pipe entre les applications (évite de créer un fichier temporaire pour partager des données)
- Toutes les « ressources » (fichiers xml, images, styles, etc) sont accessibles à toutes les applications

IOS ne permet pas le partage de fichier.

- Framework centralisé et protégé compatible OAuth2 (settings/account)
- A UTILISER systématiquement
- Ne pas demander les user/password dans chaque application
- Permet de proposer un token aux autres applications sans exposer les ids
- Plus complexe à coder, mais plus d'ouverture et de sécurité
- Reset automatique de tous les passwords lors d'un changement de carte SIM
- Secrets en caches dans le frameworks

IOS ne propose pas de compte centralisé.

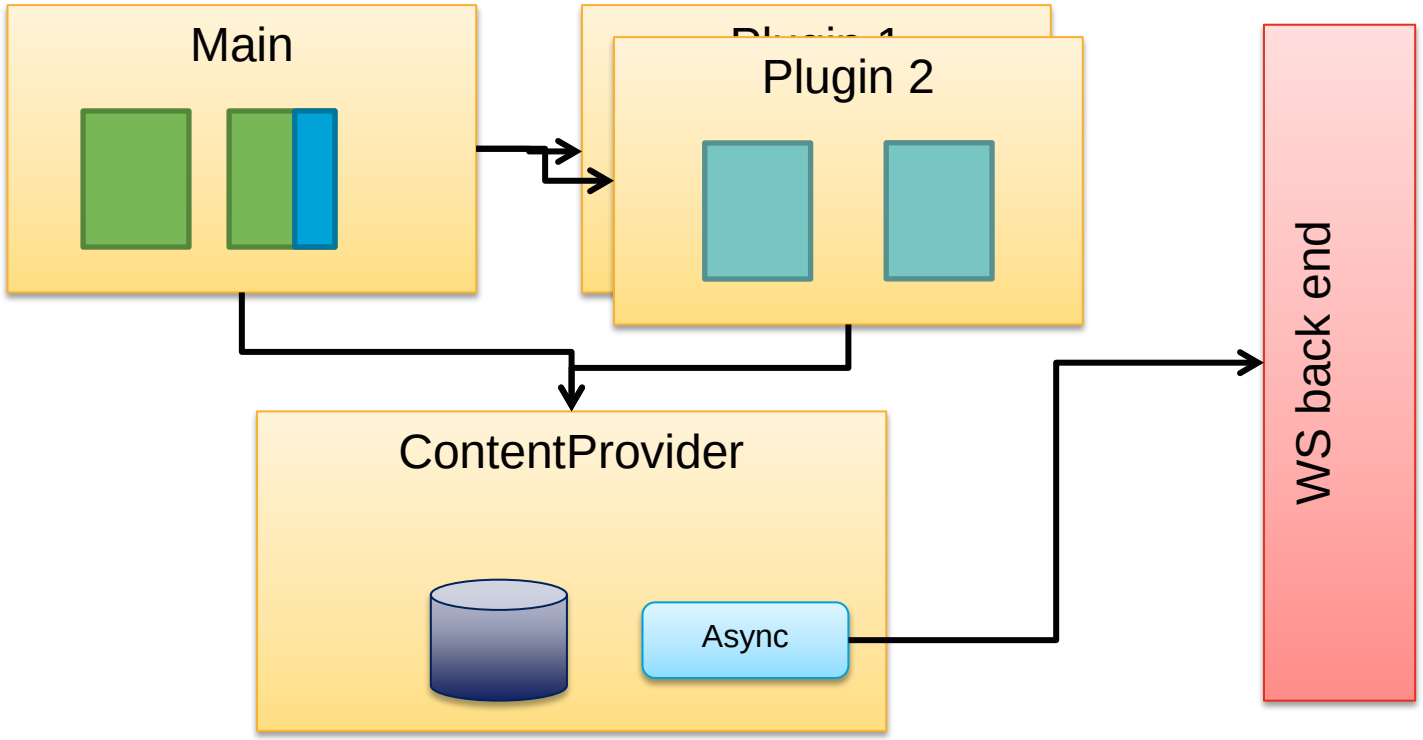
Comment sont vérifiées les permissions ?

- Aucun service ou périphérique critique n'est directement accessible aux applications (/dev n’est pas accessible)
- Les applications doivent communiquer avec le processus `system_app`
- Ce dernier vérifie les privilèges du processus appelant
- Car le mécanisme Binder (AIDL) injecte l'UID et le PID de l'appelant
- Les permissions sont déclarées par les applications dans `AndroidManifest.xml`

- Très différents d'un développement Java classique
- Il faut oublier les bonnes pratiques J2EE pour privilégier les performances
- Le code est complexe, difficile à déverminer, car il existe de nombreux scénarios d'usages (basculement de l'écran, perte de l'application en cours de route, etc.).
- Utilisation systématique du multitâche
- Mais, le résultat peut être portable sur téléphone, télévision, tablette, voiture, etc.

- En acceptant de suivre les bonnes pratiques
  - + Très forte possibilité d'intégration entre les applications
  - + Invocation d'écrans/services des autres applications
  - + Possibilité de publier des services/écrans/données pour les autres applications
  - + En cas d’ambiguïté, l'utilisateur sélectionne l'application
- Le framework est très ouverts
- Pas d'adhérence avec un fournisseur de contact particulier, un partage de contenu, etc.
- La notion d'Intent est proposé comme spécification pour le Web (Web Intent)

Exemple d'architecture modulaire



APK

Activité

- Mise à jour individuelle de chaque module
- Extension de l'application suivant les plugins présent
- Synchronisation au bon moment

Par rapport à IOS

- Langage compilé
- Très peu de partage entre les applications
- Pas de navigation entres applications
- Isolation très forte entre les applications
- Sécurité non extensible

- Multitâche de moins en moins limité
- Acceptation de plusieurs hauteurs des écrans (pas encore de plusieurs largeurs)

- Deux environnements
  - Eclipse
  - Android Studio (Beta)
- Les sources
  - http://source.android.com/
- Émulateur PC très rapide
  - Genymotion (http://www.genymotion.com/)
- Pour tester
  - Téléphone Android 2.3
  - Téléphone Android 4.3 (Famille Nexus pour suivre rapidement les versions)
  - Tablette 7"
  - Tablette 10"

Des questions ?