Programmation Mobile

Android est une plateforme logicielle pour les équipements embarqués. Souvent il faut une interface simple et tactile.

Androïde : Un robot qui ressemble à un humain.

Plateforme logicielle achetée par google en juillet 2005.

En octobre 2008 Android est devenu une plateforme open source

Il y a 2.7 million d’applications disponibles sur le playstore.

Définir la version minimum d’Android à partir de laquelle on veut utiliser l’application.

Faire attention à la résolution à la résolution visée.

Android :

- Développement complet : OS, Middleware, Applications

Middleware : Couche entre le système d’exploitation et l’application

Avant version 5 : Java 🡪 bytecode 🡪 Dalvik bytecode 🡪 Linux OS

APK fichier archive de l’application

Dalvik Runtime : machine virtuelle dalvik, optimiser pour les téléphones

Application framework : Activity manager (gère l’application : appli arrêtée, mise ne veille, en arrière-plan)

Concept :

**Activités :** écran de l’application change à chaque fois que tu changes d’écran.

**Vues :** permet d’interagir avec l’utilisateur

**Content provider :** permet de fournir du contenu aux autres applications (n’a pas d’interface)

**Services :** Tâches de fond (pendant que d’autres applications sont en premier plan)

**Intents :** Envoie d’un message entre deux activités ou entre deux applications (quand on démarre une appli, le système envoie un intent à l’appli pour lui dire « démarre »)

**Broadcast receivers** : appli qui ne fait rien tant qu’elle ne reçoit pas de message. Dès qu’elle en reçoit, elle le traite plus se désactive.

**Permission :** L’application demande avant d’utiliser tels ou tels choses. En tant que développeur il faut déclarer toutes les permissions que l’on souhaite utiliser

Ce qui permet d’avoir accès aux autres applications ou aux différents capteurs.

Application lifecycle :

Le framework (plateforme, système d’exploitation d’Androïd) de l’Androïd qui décide quoi faire à n’importe quel moment. C’est plus l’application qui gère tout ça.

Chaque application possède un processus.

Thread : Fil d’exécution à l’intérieur d’un processus. C’est le programmeur qui découpe les thread.

Un processus peut à tout moment être tué, mis en arrière-plan ou en pause pour récupérer des ressources (ex : mémoire, batterie).

Activating components : intents

Démarrer une activité : Context.startActivity()

Démarrer un service : si pas ouvert : Context.startService()

Sinon : Context.bindService()

Démarrer un broadcast : Context.sendBroadcast()

Finir une activité : finishActivity()

Service : stopself() , Context.stopService()

Activity lifecycle :



Interface Graphique :

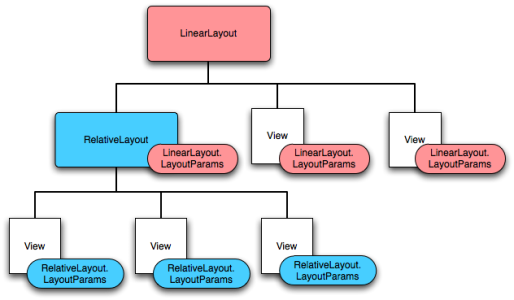
Une interface est égale à une activité (en java, c’est une classe)

Tous les éléments de graphiques que l’on va pouvoir insérer dans une fenêtre vont hériter de la classe View.

Pour dire à Android que c’est cette vue-là qu’on veut afficher : setContentView(View v)

Layout ou ViewGroup :

Un layout, c’est quelque chose qui vient regrouper un ensemble de vue et c’est le layout qui donne la façon dont sont arrangées les différents view



Handling UI Event :

**Event Listener:** Réagit aux évènements: onClick(), onLongClick(), onFocusChange(), onKey(), onTouch(), onCreateContextMenu().

**Register listener for a specific view :** Dès qu’on appuie sur un bouton, ça appellee la méthode onClik avec l’objet myListener : button.setOnClickListener(myListener)

Notification utilisateur :

**Toast notifications :** sorte de petite fenêtre popup

**Status Bar Notification :** ajouter un icone dans la barre de notification

Fichier WML : Fichier déclaratif

Charger un fichier XML : steContentView(R.layout.myUI)

🡪 cherche dans R(res) puis dans layout, le fichier myUI.