

Android



Índice

I. Qué es Android.

- Introducción.
- Instalación de Eclipse y su SDK.
- Traducción.
- Imágenes.

II. Programación en Android.

- Java.
- XML.
- Entorno de desarrollo.
- Funcionalidad soportada por Android.
- Componentes de una aplicación.
- Organización de un proyecto.

III. Kora.

I. Qué es Android

I. Qué es Android.

- Sistema operativo para dispositivos móviles basado en el kernel Linux, desarrollado por Google, entre otros.
- La versión más extendida es la 1.6, aunque la rama 2.0.X se extiende cada vez más. Ya no se venden dispositivos con 1.5.
- El lenguaje de programación principal es Java, aunque se permiten programas en C/C++ nativos con la NDK.
- Las aplicaciones se ejecutan en una “sandbox”. Cada aplicación es un proceso independiente con su máquina virtual asociada.
- Uso extensivo de XML: layouts, traducción, información de la aplicación, recursos, etcétera.
- El entorno de desarrollo elegido oficialmente es Eclipse, para el que hay plugins oficiales.

Instalación de Eclipse + plugins.

- Que rule el pen drive.
- Es necesaria una conexión a Internet para descargar los plugins.

Traducción y creación de frases

- La traducción se realiza sobre ficheros XML en la carpeta de recursos.
- Pueden traducirse también otros elementos como iconos, imágenes o cualquier otro tipo de recurso.

```
values-<idioma>/strings.xml
```

```
values-<idioma>/icon_#.png
```

- Para no-informáticos, tratar directamente con ficheros XML puede ser extraño o incómodo. También lo es para nosotros aunque estamos acostumbrados.
- Para traducción no es necesario el Eclipse, aunque conviene un editor con resaltado de sintaxis. O mejor, un programa de traducción: aiLocalizer.
 - Solo para Windows (aunque de fuentes abiertas).
 - Requiere .NET 2.0.
 - Ambos paquetes están también en el pen drive.

Imágenes del proyecto

- Iconos Oxygen para acciones predeterminadas.
- Es conveniente usar imágenes escalables y proveer distintos tamaños de icono.
- Proveer distintas imágenes para la misma acción: icono, icono simplificado, icono en alto contraste, animación en lengua de signos, ¿fotografía?
- La aplicación necesita un icono.
- Se almacenan en la carpeta res/drawable-<resolución>.
- Debido a la mayor flexibilidad requerida, conviene meterlas en un directorio especial.

II. Programación para Android.

Funcionalidad que soporta Android

- Varios formatos de imagen, pero no soporta SVG.
- Renderizado de texto para múltiples idiomas.
- Aceleración gráfica mediante OpenGL ES.
- Geolocalización, ya sea mediante GPS o mediante el router wifi al que estemos conectado.
- Bluetooth.
- Tratamiento de ficheros XML (modelos DOM, SAX y Pull).
- Bases de datos relacionales mediante SQLite.
- Seguridad para aplicaciones basada en permisos.
- Separación modelo-vista-controlador.
- Compartición de datos y funcionalidad entre aplicaciones:
 - Intents.
 - Content providers.
- Facilidad para crear una interfaz para las opciones de una aplicación.
- Servicios de telefonía.

Java

- Las aplicaciones típicas (incluida esta) se escriben en Java. Y el hecho de escribirse en Java conlleva sus ventajas e inconvenientes. Desde mi punto de vista son:
- Ventajas:
 - Si no te sales de lo típico, las cosas funcionan bien.
 - Recolección de basura automática.
 - Documentación bastante buena.
- Inconvenientes:
 - Si te sales de lo normal tienes que liarla más. La documentación ya no es tan buena.
 - Nombres de clases y métodos que pueden llegar a ser muyyyy largos.
 - Pese a que la JVM de Android (Dalvik) es muy rápida, hay que tener cuidado con el rendimiento de la aplicación, aunque en este caso no es un problema.

XML

- Android utiliza bastantes ficheros en XML:
 - Para layouts.
 - Para la configuración de la aplicación (AndroidManifest.xml)
 - Para los ficheros de traducción.
- También provee métodos DOM, SAX y Pull para tratar ficheros XML desde código escribiendo parsers personalizados.

Eclipse. Entorno de desarrollo en general.

- Eclipse no me gusta.
 - Es lento.
 - La interfaz está sobrecargada.
 - En Linux provoca fugas de memoria en el servidor gráfico.
- El emulador de Android es bastante bueno aunque también es lento.
- El editor de layouts está bien aunque uno acaba siempre editando el XML a pelo.
- La generación automática del fichero de identificadores de recursos es muy útil.

Componentes de una aplicación

- Como se comentó al principio, cada aplicación se ejecuta separadamente de las otras.
- Las aplicaciones no tienen `main()`. Cada aplicación se divide en varias `Activities` que pueden ser expuestas para su uso desde otras aplicaciones.
- Para reutilizar funcionalidad entre aplicaciones, simplemente se inicia la `Activity` correspondiente que se requiera.

Componentes de una aplicación (II).

Activities

- Una `Activity` se define como “una interfaz de usuario enfocada a una actividad que el usuario quiera realizar”.
- Cuando una `Activity` inicia otra, esta suele mostrarse a pantalla completa, pero las `Activities` pueden embeberse en zonas de la pantalla o en diálogos.
- Para crear una `Activity` simplemente se crea una clase que herede de ella. Android provee varias `Activities` con funcionalidad implementada: `ListActivity`, `TabActivity`, `PreferenceActivity`...

Componentes de una aplicación (III). Services.

- Se ejecutan en segundo plano y no tienen interfaz gráfica.
- Es posible conectarse a un servicio que se esté ejecutando, y si es necesario, se comienza previamente.

Componentes de una aplicación (IV).

Broadcast receivers.

- Son componentes que esperan que se produzca un mensaje de multidifusión para reaccionar ante los mismos.
- Una aplicación puede definir varios para reaccionar ante los eventos que considere importantes.
- Tampoco tienen interfaz gráfica pero como reacción a un evento producido pueden invocar una Activity o llamar al gestor de notificaciones.

Componentes de una aplicación (V).

Content providers.

- Sirven para exponer al resto de aplicaciones datos propios de la aplicación.
- Los datos pueden estar contenidos en ficheros de datos o texto, o en bases de datos.
- Se invocan mediante un `ContentResolver`, que se encarga de localizar el proveedor de datos, pasarle los parámetros indicados y devolver los datos obtenidos.

Componentes de una aplicación (VI).

Intents.

- Son un mecanismo de mensajes para activar Activities, Services y Broadcast Receivers.
- Contienen la actividad o el servicio a invocar y el resultado que se espera que devuelvan (si se desea un resultado).
- Para los broadcast receivers incluyen información sobre el evento que se ha producido.
- Se inician actividades nuevas mediante los métodos `startActivity()` de la clase `Context` y `startActivityForResult()`.
 - Al iniciar una actividad mediante la segunda, al acabar esta llama al método `onActivityResult()`, que contiene el resultado de la ejecución y los datos devueltos.
- Para iniciar un servicio se utiliza el método `startService()` o el método `bindService()` de la clase `Context` para conectarse a uno que ya esté funcionando. Para multidifusión, métodos `sendBroadcast()` y sus variaciones.

Componentes de una aplicación (VII). Otros.

- Diálogos.
- Gestión de la pila de actividades para cada aplicación (tareas).
- Estados de las tareas. Métodos onCreate(), onStart(), onRestart(), onResume(), onPause(), onStop(), onDestroy().

Cómo está organizado un proyecto Android. Comenzar un proyecto.

- Carpeta `src`: código fuente en Java.
- Carpeta `gen`: donde se sitúa el fichero `R.java`
- Carpeta `res`: donde se sitúan los recursos del programa.
 - `res/drawable`: imágenes e iconos utilizados.
 - `res/layout`: ficheros XML para layouts de actividades.
 - `res/values`: ficheros para traducción.
 - `res/xml`: ficheros XML genéricos.
 - `AndroidManifest.xml`: información sobre la aplicación.
- Para comenzar un proyecto, en Eclipse:
 - Fichero → Nuevo → Proyecto de Android.

Comencemos un
proyecto de ejemplo.

III. Kora.

Diseño de la aplicación

- Varios usuarios por dispositivo, cada uno con necesidades y permisos de uso distintos.
- Se distinguen dos tipos básicos de interacción: encendido/apagado, selección en un intervalo discreto.
- Cada dispositivo puede soportar varias interacciones distintas.
- Puede convenir un tipo de vista distinto para cada interacción, y puede ser conveniente presentarlo de una u otra forma según la necesidad del usuario.

III. Organización de actividades

- Pantalla inicial.
- Actividad de control de dispositivos.
 - Selección de dispositivo.
 - Interacción con dispositivo elegido.
- Actividad de administración (TabActivity).
 - Usuarios.
 - Perfiles de usuario.
 - Perfiles de dispositivos.

Gestión de ajustes y perfiles

- Las clases que representan los modelos de ajustes son `User`, `UserProfile` y `DeviceProfile`.
- Todas se gestionan de forma centralizada en el `SettingsManager`.
- Al instanciarse el `SettingsManager` se cargan los ajustes desde la base de datos.
- Perfiles por defecto y personalizados.
- Cada vez que se modifica un perfil se guarda automáticamente en la base de datos.
- El problema de las `SharedPreferences` y la `PreferenceActivity`.

Control de dispositivos. Selección.

- La actividad de selección de dispositivo es la `DeviceSelectionActivity`.
- Muestra una rejilla con los dispositivos disponibles. La interacción con la misma depende del perfil de usuario.
- ¿Uso de widgets propios (extender clase `View`) o reutilizar componentes de Android?
- Es necesario controlar los eventos del usuario, los que se produzcan en los dispositivos y los automáticos.
- Métodos de paginación.
- Resaltado de elementos seleccionados y elegidos. Confirmación.

Control de dispositivos. Manejo.

- La actividad de manejo de dispositivos es la DeviceActivity.
- Básicamente es una lista de controles para un dispositivo, que puede tener uno o varios controles.
- El control gráfico para cada interacción puede tener representaciones distintas, según su funcionalidad y el perfil del usuario.
- Cuestiones:
 - ¿Dispositivos con un solo control deberían integrarse en la pantalla de selección?
 - ¿Posibles métodos de interacción? Es importante tener en cuenta el barrido y el texto-a-voz.

Recursos.

- Iconos típicos.
 - Tomados del proyecto Oxygen (<http://oxygen-icons.org>).
 - Escalables, generados para múltiples resoluciones.
- Son necesarios iconos para dispositivos.
- Aplicación desarrollada en inglés.
- La aplicación necesita un icono fácilmente reconocible, amigable y colorido.
- La ventana de inicio podría ser rediseñada e incluso adaptable a la necesidad del último usuario que inició la aplicación.

Resumen

- Implementación retrasada. Quedan por implementar toda la parte de control de dispositivos y parte de las preferencias.
- Análisis y diseño decididos, pero quedan pequeñas cuestiones pendientes.
- Es muy importante el uso de recursos gráficos y sonoros, que deben adaptarse al usuario.
- Al ser una aplicación libre y con un fuerte componente educativo y social, es muy conveniente que esté disponible en varios idiomas.
- Al estar enfocada a un uso no técnico, conviene disponer de un manual de usuario detallado.