**PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES**

**UF1:**

1. **Qué es una actividad y cuál es su evento básico**

Una actividad es un componente de la aplicación Android que proporciona interacción con el usuario. Se puede interpretar como una ventana tradicional en la que se desarrolla una acción simple.

onCreate(). Sirve para inicializar los componentes gráficos.

1. **Ciclo de vida de una actividad y sus eventos**

Las actividades tienen 3 estados:

Resumed: está ejecutándose.

Paused: parada pero aún visible.

Stopped: parada y no visible.

Cuando la aplicación empieza llama al método onCreate() donde se establecen sus

componentes, después se llama al método onStart() para que sea visible. El siguiente método es onResume(), y empieza a correr la actividad. Si se lanza la actividad se pausa ésta, que después volverá al método onResume() para continuar con su ejecución. Aunque también puede darse el caso de que se pause por otras aplicaciones, que entonces necesitaremos volver a cargar los componentes. En vez de pausar, se puede detener llamando al método onStop(), la forma de volver a utilizar esta actividad es llamando al método onRestart(), a no ser que se haya detenido por otras aplicaciones, que entonces, al igual que en el caso anterior, necesitaremos volver a cargar los componentes. Por último, el método onDestroy() libera los recursos de la actividad y se termina.

1. **4 preguntas de verdadero o falso y razonar sobre toast y notificaciones en la barra de estado.(las 4 cosas básicas de cada uno)**

Un **toast** es un mensaje que se muestra en pantalla durante unos segundos al usuario para luego volver a desaparecer automáticamente sin requerir ningún tipo de actuación por su parte. Aunque se le puede indicar que se muestre en una ubicación en concreto, por defecto aparecen en la parte inferior de la pantalla, sobre un rectángulo gris ligeramente translúcido.

La barra de estado de Android se encuentra situada en la parte superior de la pantalla. La parte izquierda de esta barra está reservada para visualizar notificaciones. Cuando se crea una nueva notificación, aparece un texto desplazándose en la barra, y a continuación, un pequeño icono permanecerá en la barra para recordar al usuario la notificación.

El usuario puede arrastrar la barra de notificaciones hacia abajo, para mostrar el listado de las notificaciones por leer.

Una notificación puede ser creada por un servicio o por una actividad. Aunque dado que la actividad dispone de su propio interfaz de usuario, parece que las notificaciones son el mecanismo de interacción más interesante del que disponen los servicios. Las notificaciones pueden crearse desde un segundo plano, sin interferir con la actividad que en ese momento esté utilizando el usuario.

1. **Qué es un fragmento**

Un fragmento es na parte constituyente de una actividad. Formaremos las actividades a partir de uno o varios fragmentos.

1. **Pregunta de relacionar(actividad, servicio, broadcast, proveedor de contenido e intent): saber que es cada cosa.**

Proveedor de contenido

Un mecanismo para exponer nuestros datos internos al resto de aplicaciones, no son

necesarios si solo se prevé hacer un uso internos de los datos de la aplicación.

Broadcast Receiver

Es un componente que puede recoger mensajes desde cualquier aplicación.

Intent:

Como mecanismo para invocar componentes, los intents son bastante fáciles de comprender. Básicamente nos permiten llamar a aplicaciones externas a la nuestra, lanzar eventos a los que otras aplicaciones puedan responder, lanzar alarmas etc.

Servicio:

Un servicio es un componente de aplicación que puede realizar operaciones de larga duración en segundo plano y no proporciona una interfaz de usuario. Otro componente de aplicación puede iniciar un servicio y continúa ejecutándose en segundo plano incluso si el usuario cambia a otra aplicación. Además, un componente puede enlazar a un servicio para interactuar con él e incluso realizar comunicación entre procesos (IPC).

1. **Que layout usa orientation y un ejemplo de gravity.**

LinearLayout, y un ejemplo de gravity es Debe declararse dentro del TextView al que se le quiere dar alineación. La forma de declararlo es: android:gravity= y entre comillas se escribe cualquiera de los valores center, left, bottom o right.

**¿Cómo y cuándo se termina una operación de arrastre?**

Si bien para comenzarla se llama a la función startDrag(), se termina de forma automática por parte del SSOO al dejar caer el elemento en alguna vista.

**¿Qué es la sombra y por qué es obligatorio crear una?**

La sombra es el mecanismo que tiene el usuario para determinar por dónde está el elemento que se está arrastrando.

**¿Puede cualquier vista recibir un objeto arrastrado?**

Verdadero, pero se tiene que registrar para ello y al recibir el primer evento comunicar al SSOO al dejar caer el elemento en alguna vista.

**Describe los tipos de servicios.**

Servicios que corren durante todo el tiempo (Ejecutados) y servicios que se lanzan cuando se solicitan (Enlazados). El primer caso los lanza el SSOO o la aplicación y se deben parar ellos mismos o una actividad, mientras que los segundos se terminan en el momento de finalizar el trabajo.

**Mecanismos de comunicación entre un servicio y el usuario.**

Toast y mensajes en la barra.

**¿Qué es un proveedor de contenido?**

Un mecanismo para exponer nuestros datos internos al resto de aplicaciones, no son necesarios si solo se prevé hacer un uso internos de los datos de la aplicación.

**¿Qué es una clase contractual?**

Es una clase abstracta que se distribuirá con nuestra aplicación con los metadatos necesarios para acceder al proveedor de contenido por parte de otras aplicaciones. No es obligatoria pero facilitará la labor de programación a los demás mucho.

**¿Qué es un broadcast receiver?**

Es un componente que puede recoger mensajes desde cualquier aplicación.

**Tipos de mensajes que puede recoger un broadcast receiver.**

Síncronos y asíncronos.

**Verdadero o falso: Los sensores siempre estarán presentes si existen en el dispositivo.**

Falso, el usuario tiene la posibilidad de activar o no un sensor, por lo que es obligatorio comprobar su disponibilidad.

**Tipos de sistemas de posicionamiento.**

Basados en satélites, basados en redes, bien wi-fi o bien de telefonía.

**¿Qué es el Google Maps API?**

Una interface de programación de Google que proporciona localización.

**Tipos de alertas de proximidad.**

De entrada en un radio y posición, o de salida.

**¿Cuál es el evento obligatorio de todas las actividades? ¿Cuál es su funcionalidad?**

onCreate(). Sirve para inicializar los componentes gráficos.

**¿Cuáles son los pasos que debemos seguir para añadir una nueva actividad a nuestra aplicación?**

En primer lugar extender de la clase Activity, e implementar al menos el evento onCreate().

Definir la actividad en el fichero de manifiesto y por último lanzar la actividad, bien por parte del Sistema operativo, o por nosotros mediante la llamada a los métodos startActivity() o startActvivityForResult().

**¿Qué evento utilizarías para grabar el estado de una aplicación?**

onPause y onSaveInstanceState ya que los eventos onStop y onDestroy no se garantiza que se llamen.

**¿Qué es un fragmento?**

Un fragmento es na parte constituyente de una actividad. Formaremos las actividades a partir de uno o varios fragmentos.

**¿Cómo añadimos un fragmento a una actividad?**

Mediante las transacciones, que se encargan de la gestión para añadir, remplazar o borrar un fragmento de una actividad.

FragmentManager fragmentManager = \_getFragmentManager();

FragmentTransaction fragmentTransaction = fragmenManager.beginTransaction();

Fragment newFragment = new ExampleFragment();

FragmentTransaction transaction = getFragmentManager().beginTransaction();

transaction.replace(R.id.fragment\_container, newFragment);

transaction.addToBackStack(null);

transaction.commit();

**¿En qué recurso añadiremos las cadenas de texto? ¿De qué forma declararías una cadena?**

Values/strings.xml

**¿En qué recurso añadiremos los iconos de la aplicación?**

Drawable

**¿Cómo hacemos en Android para tener otras configuraciones de idiomas?**

Para cada traducción crearemos un directorio con el calificador del idioma correspondiente (values-calificador).

**UF2:**

1. **Tareas asíncronas.**

Es un objeto instanciado de una clase derivada de AsyncTask. Esta clase presenta unas características especiales de gestión muy apropiadas para el uso en nuestras aplicaciones, permite modificar el UI, a la vez que puede ejecutar código en una hebra separada evitando toda gestión de hebras creación y mantenimiento.

1. **Handlers**

* Permiten poner en cola una acción que se realiza en un subproceso distinto al suyo.
* Cada instancia de la clase Handler está asociada a un sólo hilo y a la cola de mensajes de este.
* Comunicación del hilo secundario con el hilo principal o UIThread a través de un controlador.
* Proporciona varios métodos para poner en cola un mensaje en la parte delantera o enviar al final de la cola después de un determinado tiempo ( sendMessageAtFrontOfQueue(Message msg) y sendMessageAtTime(Message msg, long uptimeMillis) ).

1. **Que hebras pueden acceder y modificar la UI**

La hebra principal.

1. **Permisos para escribir en una base de datos externa**

android.permission.WRITE\_EXTERNAL\_STORAGE en el manifest.xml.

1. **Paquete y clases java para internet.**

El paquete recoge todas las capacidades de red básicas, proporciona clases para crear flujos de datos y sockets, gestión básica de HTTP y recursos. Cualquier documentación de java tradicional es aplicable a este paquete.

Android.net.http.

Android.net.rtp.

Android.net.sip.

Android.net.wifi.

1. **Ordenar de mayor a menor importancia:**
   1. **Procesos vacios**
   2. **Procesos en 2º plano**
   3. **Servicios**
   4. **Procesos visibles**
   5. **Procesos en primer plano.**

**E D, C, B, A**

**UF3:**

1. **Qué es el viewport**

Área del plano cercano que se representará en el dispositivo. Todo lo que quede fuera de viewport no se visualizará. La esquina superior izquierda se corresponde exactamente con la superior izquierda del dispositivo en formato “norma”.

1. **Por qué es bueno diseñar el juego previamente en papel**

Nos va a permitir ordenar todas las ideas, dar un sentido y una coherencia al juego que de otra manera sería imposible, así como detectar las necesidades de elementos gráficos y de sonido de manera temprana. Va a permitir dividir el desarrollo en fases y si fuera necesario cada fase podría ser implementada por equipos de programación diferentes.

1. **Qué es una animación**

Una animación es la modificación de una o varias propiedades de los objetos a través del tiempo, con el camino de aspecto de dicho objeto. El sistema de animaciones de Android proporciona tres mecanismos diferentes para animar objetos en pantalla.

El primero y obsoleto se denomina animación de vistas. Podemos animar propiedades simples del contenido de una View, no de cualquier objeto.

El segundo mecanismo son las animaciones de Drawables, que cargamos un conjunto de imágenes y hacemos visualizarlo uno detrás de otro, simulando el movimiento, similar a una presentación de fotografías rápidas. Para cada frame podemos establecer propiedades individuales, así como controlar la secuencia de reproducción.

Por último el tercer mecanismo de animación nos permite realizar tareas más elaboradas, es un framework potente y robusto que facilita el cambio de cualquier propiedad, de cualquier objeto a lo largo del tiempo. Para crear una animación de este tipo, tendremos que determinar la duración de la animación, la repetición de la misma, cómo se cambian las propiedades de los objetos a lo largo del tiempo, si existen conjuntos de objetos que se animarán conjuntamente, y por último, la tasa de refresco de la animación.

1. **Por qué no se deben poner .png en /res**

Porque Android modifica los ficheros gráficos en formato png que ahí se sitúen para mejorar el rendimiento, usaremos mejor el directorio /assets.

1. **Qué permisos necesitaremos para un juego que se conecte a internet, reproduzca sonido y escriba en la tarjeta sd.**

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

<uses-permission android:name="android.permission.AUDIO" />

<uses-permission android:name="android.permission.INTERNET " />

**Enumera los diferentes métodos a la hora de realizar juegos para Android.**

Directamente en Java, usando OpenGL, otras técnicas avanzadas como NDK.

**¿Por qué es importante crear nuestro juego en papel antes de lanzarnos a programar?**

Nos va a permitir ordenar todas las ideas, dar un sentido y una coherencia al juego que de otra manera sería imposible, así como detectar las necesidades de elementos gráficos y de sonido de manera temprana. Va a permitir dividir el desarrollo en fases y si fuera necesario cada fase podría ser implementada por equipos de programación diferentes.

**Se ha introducido en el desarrollo el uso de un tiempo delta, ¿por qué?**

La capacidad gráfica de cada dispositivo es diferente, haciendo el desarrollo del juego también distinto, de esta manera controlamos que la experiencia visual sea la misma en todos los dispositivos.

**¿Por qué es tan importante que no usemos el directorio /res de nuestro proyecto para dejar ficheros en formato png?**

Porque Android modifica los ficheros gráficos en formato png que ahí se sitúen para mejorar el rendimiento, usaremos mejor el directorio /assets.

**Diferencia entre una proyección ortogonal y una en perspectiva.**

En la primera no se deforman los objetos con la profundidad y no hay sensación de ella.

**¿Qué es el estado en OpenGL?**

Es el mecanismo por el cual OpenGL establece parámetros clave en el renderizado de un mundo. Estos parámetros permanecen fijos hasta que los cambiamos.

**¿Qué diferencias existen entre el sistema de coordenadas de las texturas y el del mundo OpenGL?**

Las texturas sitúan el origen en la esquina superior izquierda y el valor máximo es uno en ambos sentidos, independientemente del tamaño de la misma. Mientras que el origen en OpenGL de coordenadas se localiza en la esquina inferior izquierda.