JJD311: Desarrollo Avanzado de Apps para Android 5 Lollipop – skill test

Nombre completo:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Nota:\_\_\_\_\_\_\_\_

**Q1 – Señale qué afirmaciones son ciertas con respecto a los sistemas de almacenamiento en Android**

a) Algunas de las opciones que ofrece la plataforma son *Shared Preferences*, *Internal storage* y *SQLite databases*

b) En el *Internal storage*, los ficheros son guardadas en la ruta “*/data/data/nombre.del.paquete/files/* ”

c) Cuando una aplicación es desinstalada, cualquier base de datos que se hubiese implementado es eliminada

d) Todos los dispositivos Android permiten el “almacenamiento externo compartido”, para así guardar ficheros que puedan ser fácilmente accedidos por cualquier aplicación

e) Por defecto, los ficheros guardados en la memoria interna son privados a la aplicación

**Q2 – En Android, la *Google Maps API Key* se necesita para:**

a) acceder al servicio de mapas de Google, *Google Maps*

b) firmar una aplicación que integre *Google Maps* para su posterior subida a Google Play Store

c) permitir que una aplicación pueda ser indexada por el buscador de *Google Maps*

d) Ninguna de las anteriores

**Q3 – La gestión de bases de datos en Android se realiza mediante:**

a) Mongo DB

b) SQL

c) SQLite

d) MySql

e) Android SQL

**Q4 – En referencia a las operaciones de *networking* en Android, señale las opciones correctas:**

a) En general, se utiliza un objeto de la clase HttpURLConnection

b) Es obligatorio incluir en el *Manifest* el permiso android.permission.INTERNET

c) En el caso de descarga de datos, HttpURLConnection entrega un objeto de tipo InputStream

d) Es obligatorio incluir en el *Manifest* el permiso android.permission.ACCESS\_NETWORK\_STATE

e) b y d

**Q5 – El gran inconveniente de usar entidades de tipo AsyncTask en Android es:**

a) que inicialmente debe crearse un Thread, para que la AsyncTask no se ejecute en el *UI Thread*

b) que el prototipo de la clase AsyncTask obliga a declarar los datos genéricos a utilizar, reduciendo así la reusabilidad del código

c) que incluso utilizando el método executeOnExecutor(), un conjunto de varias AsyncTask solo pueden ejecutarse secuencialmente

d) que los AsyncTask no manejan automáticamente los cambios de configuración del sistema, pudiendo causar *memory leaks*

**Q6 – En el prototipo de una AsyncTask del tipo AsyncTask<String, Integer, Void>, los argumentos indicados representan (marque todas las que sean correctas):**

a) que el constructor del AsyncTask no requerirá del Context de la Activity como argumento de entrada

b) que se utilizará un array de elementos de tipo Integer para informar al usuario del progreso del procesamiento

c) que en el método onPostExecute() no se utilizará ningún resultado devuelto por doInBackground(), ya que el tipo es Void

d) que el prototipo del AsyncTask está incompleto, pues por defecto se requiere de 4 argumentos

**Q7 – El método runOnUiThread() sirve para:**

a) actualizar la UI desde un thread distinto al principal

b) hacer referencia a un *worker thread* desde el hilo principal

c) pedir al sistema que desde esa línea de código en adelante, el resto de la aplicación se ejecute exclusivamente en el hilo principal

d) anidar un *worker thread* dentro de otro

**Q8 – En un ContentProvider, el método query(…):**

a) solo requiere de un argumento de entrada; concretamente un String con la sentencia *SQL* a ejecutar

b) debe devolver un objeto de tipo Cursor, que será null en caso de error

c) es independientemente de si encuentra o no algún registro (fila) que coincida con la petición, ya que su tipo de retorno es void

d) No requiere un URI, sino una URL

**Q9 – A la hora de crear una base de datos en Android (señale todas las opciones que sean correctas):**

a) El método que realmente crea la base de datos en el sistema es SQLiteDatabase.execSQL()

b) Es obligatorio crear un constructor en la clase que hereda de SQLiteOpenHelper

c) b

d) Para escribir en la base de datos, el desarrollador debe llamar al método getReadableDatabase()

e) a y b

**Q10 – Señala las opciones verdaderas en referencia a la clase ConnectivityManager:**

a) La llamada getSystemService(Context.CONNECTIVITY\_SERVICE) devuelve un objeto del tipo ConnectivityManager

b) ConnectivityManager.getActiveNetworkInfo() permite al desarrollador conocer el estado de la red

c) Para obtener una instancia de ConnectivityManager es necesario instanciar la clase, es decir, new ConnectivityManager()

d) La clase ConnectivityManager gestiona también la conexión de la aplicación con el *Sensor Framework*

e) Ninguna de las anteriores

**Q11 – Un Fragment (señale las opciones verdaderas):**

a) es una entidad definida en la plataforma Android que permite compilar el código de una aplicación por paquetes o fragmentos

b) representa un comportamiento o una porción de la UI de una Activity

c) tiene un ciclo de vida que incluye el *callback* onActivityAssigned()

d) puede entenderse como una sección modular de una Activity que tiene su propio ciclo de vida

**Q12 – Para comunicar un Fragment con la Activity que lo contiene:**

a) es recomendable definir una interfaz dentro del Fragment, y obligar a la Activity a implementarla

b) deben usarse objetos de tipo Intent y/o Bundle

c) debe utilizarse forzosamente un Service y un BroadcastReceiver

d) ha de inicializarse el Fragment mediante el método startFragmentForResult()

**Q13 – El ciclo de vida de un Fragment hasta que toma el foco (*foreground*) es:**

a) onCreate() -> onCreateView() -> onActivityCreated() -> onAttach() -> onStart() -> onResume()

b) onFragmentCreate() -> onFragmentAttached() -> onFragmentStart() -> onFragmentResume()

c) onCreate() -> onStart() -> onResume()

d) onAttach() -> onCreate() -> onCreateView() -> onActivityCreated() -> onStart() -> onResume()

**Q14 – Un Loader en Android sirve para:**

a) referenciar un ContentProvider y relacionarlo con un ListView

b) indicar en los ficheros de compilación (*Gradle*) que la aplicación va a ser ejecutada por lotes

c) hacer que la compilación de una aplicación sea progresiva, y por tanto compatible con ART

d) proceder a la carga de datos asíncrona en una Activity o Fragment

**Q15 – Indique las opciones correctas a la hora de hacer un *parsing* a un fichero XML:**

a) XML son las siglas de *eXtreme Matching League*

b) El método nextTag() permite saltar a la siguiente etiqueta XML (ya sea de inicio o de cierre)

c) La clase a emplear es XmlPullParser

d) El desarrollador debe decidir inicialmente qué campos del *feed XML* le interesan, para lo que tendrá que inspeccionar el fichero correspondiente previamente

e) La constante que indica El final del fichero XML es XmlPullParser.END\_OF\_CURRENT\_DOCUMENT

**Q16 – La librería Picasso permite:**

a) utilizar gráficos *OpenGL* para crear animaciones

b) descargar una imagen e introducirla en un View de forma rápida, elegante, eficiente y asíncrona

c) crear un pool de hilos de ejecución (*thread pool*) para ejecutar tareas en paralelo

d) aligerar la carga de procesamiento del sistema, ya que se encarga de gestionar la UI

**Q17 – En referencia a la inclusión de Google Maps en una aplicación Android, se puede definir un Marker como:**

a) una marca de agua, configurable por el desarrollador, que aparece en el extremo superior derecho del mapa

b) una marca de tiempo que permite definir un movimiento de cámara sobre un mapa

c) un punto de debug (*breakpoint*) para la depuración de aplicaciones que incluyan *Google Maps*

d) un marcador que puede ser incluido en ciertas localizaciones del mapa implementado

**Q18 – En referencia a los mecanismos de seguridad empleados en el desarrollo en Android, podemos afirmar que (mara aquella que NO es verdadera):**

a) las aplicaciones han de firmarse con un certificado

b) está basado en el *kernel* de Linux (user/group ID)

c) el modelo de seguridad declarativa, definida por permisos

d) Ninguna de las anteriores

**Q19 – Un InputStream, como el devuelto tras establecer una comunicación HTTP, cumple que (señale todas las que sean correctas):**

a) no puede emplearse para la lectura de ficheros de texto, ya que para eso se emplea la clase BufferReader

b) se referencia mediante el método getInputStream()

c) puede servir tanto para cargar una imagen como para leer un fichero de texto

d) debe ser cerrado como cualquier *stream*, mediante el método close()

**Q20 – Señale las opciones correctas en referencia a los posibles valores de latencia de un sensor.**

a) SENSOR\_DELAY\_FULL

b) SENSOR\_DELAY\_DEFAULT

c) SENSOR\_DELAY\_GAME

d) SENSOR\_DELAY\_UI

e) SENSOR\_DELAY\_NORMAL

**Q21 – El elemento <uses-feature> del Manifest permite:**

a) invocar diferentes *callbacks* para el correcto empaquetado de la aplicación en el fichero de extensión ‘.apk’

b) que una aplicación sea depurable (*debuggable*) usando un emulador y/o un terminal físico

c) declarar un tipo especial de permiso que hace que una aplicación pueda implementar servicios alojados en el *Application Framework* del *Android Stack*

d) filtrar las aplicaciones basándose en la presencia o no de ciertas características, como por ejemplo sensores

**Q22 – La razón principal por la que se insta al desarrollador a usar un RecyclerView en vez de un ListView es (señale la opción que más se ajuste):**

a) Realmente, la documentación oficial muestra la clase RecyclerView como obsoleta (*deprecated*)

b) Gracias a las librerías de compatibilidad (*support libraries*), los RecyclerView están disponibles para cualquier versión de Android (es decir, desde la API 1), por lo que no tiene sentido no usarlos

c) *Material Design* introdujo la clase RecyclerView, y es obligatorio usarla cada vez que se pueda. De no hacerse así, la aplicación puede ser rechazada por los administradores del *Google Play Store*

d) Un RecyclerView hace un uso más eficiente de los recursos del sistema, instanciando solo aquellas vistas que están a punto de ser representadas, es decir, que son o van a ser visibles por el usuario

**Q23 – Entre las best practices a considerar cuando se utilizan sensores en una aplicación Android, cabe destacar (señale todas las que sean ciertas):**

a) No deben realizarse tareas “pesadas” en el método onSensorChanged()

b) Deben anularse las suscripciones de los *listeners* a sensores cuando ya no sea necesario usarlos, es decir en la llamada a onDestroy() del ciclo de vida de la Activity en cuestión

c) Los tiempos de latencia (*sensor delay*) deben ser elegidos de forma conservadora, para no malgastar recursos valiosos como la batería del terminal

d) En general, no debe usarse ningún sensor, ya que con la llegada de *Material Design* la gestión de los sensores se ha delegado al sistema operativo

**Q24 – Las dos clases principales que permiten la gestión y reproducción de contenido multimedia en Android son:**

a) MediaPlayer y MediaRecorder

b) MediaRecorder y AudioManager

c) MediaRecorder y AppCompat

d) AudioManager y MediaPlayer

**Q25 – Un ListActivity permite al desarrollador (señale la opción verdadera):**

a) utilizar una View, declarada en el layout, con el identificador @android:id/empty, y que puede ser empleada para mostrar cierta información en el caso de que el ListView correspondiente no albergue ningún registro

b) desentenderse de la creación del *layout* de la Activity, ya que este viene definido implícitamente y por tanto no se crea ningún fichero XML

c) hacer un mejor uso del *Manifest*, ya que una ListActivity no tiene que declararse en él

d) Ninguna de las anteriores