INTRODUCCION

El conocimiento de programación de aplicaciones para móviles ha pasado de manera muy rápida de ser un conocimiento más, a ser una necesitad debido a la rápida implantación y evolución de las plataformas móviles.

Esta rápida evolución crea incertidumbre sobre que tecnologías son las más adecuadas para la programación de móviles. Una de las arquitecturas más implantada es la proporcionada por el sistema Android.

Las aplicaciones móviles también llamadas Apps, son programas informáticos diseñados para ofrecer distintos servicios en dispositivos móviles. Esto permite a los usuarios realizar determinadas funciones de una forma muy rápida (con menos clics) y acceder a contenidos de forma optimizada para su lectura en pantallas de reducidas dimensiones.

El dispositivo móvil Se puede definir como un aparato de pequeño tamaño, con algunas capacidades de procesamiento, con conexión permanente o intermitente a una red, con memoria limitada, que ha sido diseñado específicamente para una función, pero que puede llevar a cabo otras funciones más generales. De acuerdo con esta definición existen multitud de dispositivos móviles, desde los reproductores de audio portátiles hasta los navegadores GPS, pasando por los teléfonos móviles, los PDA’s o los Tablet PC’s.

Modelado de objetos de acceso a datos en dispositivos móviles Modelado de objetos en dispositivos móviles.

1. Un Objeto de Acceso a Datos o Data Access Object (DAO) son una serie de objetos que le permiten tener acceso y manipular datos mediante programación en bases de datos locales o remotos. Puede utilizar DAO para administrar bases de datos, así como sus objetos y su estructura.

Los Objetos de Acceso a Datos pueden usarse en Java para aislar a una aplicación de la tecnología de persistencia Java subyacente(API de Persistencia Java), la cual podría ser JDBC, JDO, EJB,CMP(Persistencia controlada por el Contenedor), TopLink, Hibernate, iBATIS, o cualquier otra tecnología de persistencia. Usando Objetos de Acceso de Datos significa que la tecnología subyacente puede ser actualizada o cambiada sin cambiar otras partes de la aplicación.

2. Usando Objetos de Acceso de Datos significa que la tecnología subyacente puede ser actualizada o cambiada sin cambiar otras partes de la aplicación.

JDBC: Java Database Connectivity, más conocida, es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

Otra ventaja de usar objetos de acceso a datos es que cualquier objeto de negocio (aquel que contiene detalles específicos de operación o aplicación) no requiere conocimiento directo del destino final de la información que manipula.

3. Proceso de programación de objetos de acceso a datos en dispositivos móviles.

ESTRUCTURA DE XML:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE Edit_Mensaje SYSTEM "Edit_Mensaje.dtd">

<Edit_Mensaje>
    <Mensaje>
        <Remitente>
            <Nombre>Nombre del remitente</Nombre>
            <Mail> Correo del remitente </Mail>
        </Remitente>
        <Destinatario>
            <Nombre>Nombre del destinatario</Nombre>
            <Mail>Correo del destinatario</Mail>
        </Destinatario>
        <Texto>
            <Asunto>
```

```
            Este es mi documento con una estructura muy sencilla
            no contiene atributos ni entidades...
        </Asunto>
        <Parrafo>
            Este es mi documento con una estructura muy sencilla
            no contiene atributos ni entidades...
        </Parrafo>
    </Texto>
  </Mensaje>
</Edit_Mensaje>
```

DIFERENCIA ENTRE XML Y JSON:

A continuación se muestra un ejemplo simple de definición de barra de menús usando JSON y XML.

JSON:

```
 {"menu": {
   "id": "file",
   "value": "File",
   "popup": {
```

```
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
 }
}
```

XML:

```
 <menu id="file" value="File">
   <popup>
     <menuitem value="New" onclick="CreateNewDoc()" />
     <menuitem value="Open" onclick="OpenDoc()" />
     <menuitem value="Close" onclick="CloseDoc()" />
   </popup>
 </menu>
```

Manipulación de datos en dispositivos móviles

- Las bases de datos contienen datos empresariales que graban los flujos de mensaje desplegados y a los que se accede a través de estos. Debe crear conexiones desde el nodo de integración a la base de datos utilizando ODBC o JDBC.

Una conexión a base de datos es un archivo de configuración donde se especifica los detalles físicos de una base de datos como por ejemplo el tipo de base de datos y la versión, y los parámetros que permiten una conexión JDBC desde el Integration Toolkit a la base de datos.

Las conexiones ODBC a bases de datos las gestiona internamente el nodo de integración y, por consiguiente, las opciones de agrupación de conexiones configurables disponibles en el controlador ODBC no se deben utilizar.

- Bases de datos estáticas

Son bases de datos de sólo lectura, cuya información de datos históricos solo sirve para estudiar la evolución de alguna entidad durante el tiempo o tomar determinadas decisiones por parte del usuario que consume la información.

Bases de datos dinámicas

Estas son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización, borrado y adición de datos, además de las operaciones fundamentales de consulta. Un ejemplo de esto puede ser la base de datos utilizada en un sistema de información de un supermercado, una farmacia, un videoclub o

una empresa o una red social.

Las bases de datos en dependencia de su ubicación se clasifican en locales y remotas.

Base de datos local

Reside por lo general en el mismo dispositivo o terminal desde donde se consulta la información. Su acceso es muy rápido y por lo general contiene información que no es compartida con otros usuarios.

Normalmente se componen de un programa o motor para realizar las consultas y de un archivo con la información.

Es el caso típico de las bases de datos de escritorio, como Microsoft Access o las bases de datos de dispositivos móviles como SQLite.

Servicios Web

Un servicio Web (en inglés, Web Service) es una tecnología que utiliza un conjunto de protocolos y estándares abiertos que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios Web para intercambiar datos en redes de ordenadores como Internet.

Los servicios Web suelen exponer determinados métodos que nos permiten de manera segura y en un entorno distribuido, acceder a la información contenida en las bases de datos remotas.

Aplicación web estática

Una aplicación web estática es lo que normalmente entendemos como sitio web y se caracteriza por lo siguiente:

- Se suelen desarrollar en HTML y CSS y puede utilizarse algo de JavaScript. Pueden presentar contenido digital con movimiento, como vídeos, audio, *banners*, o GIF animados.
- No dispone de muchas funcionalidades y el usuario no puede modificarla por sí mismo. Ofrecen poca o ninguna interactividad.
- Su actualización es compleja, es un proceso lento, tedioso y manual.
- Cambiar los contenidos también es complicado, se necesita modificar el HTML (recodificar la página) y actualizarlo en el servidor.
- Se suelen emplear para ofrecer información concisa y permanente.

Algunos ejemplos de aplicaciones web estáticas son portfolios, currículums digitales, páginas de presentación de empresas, WebQuest, etc.

Aplicación web dinámica

Las aplicaciones web dinámicas presentan los siguientes rasgos:

- Mayor complejidad técnica.
- Utilizan bases de datos para cargar la información y los contenidos se actualizan cada vez que el usuario accede a la aplicación.

- La actualización de los contenidos es sencilla, la mayoría se administra mediante un CMS. No se requiere acudir al servidor.
- Para su desarrollo existen numerosos lenguajes, como PHP o ASP.
- Permiten implementar numerosas funcionalidades, como foros o bases de datos.
- Admite muchas posibilidades de diseño y presentación.
- Hay interacción en ellas. El usuario puede realizar cambios.

Algunos ejemplos de aplicaciones web dinámicas son los blogs personales y corporativos, las páginas de noticias y actualidad y las revistas y periódicos digitales.

Servicios SMBD Móviles

Un SMBD móvil debe ofrecer los servicios de un SMBD tradicional, además de funcionalidad adicional requerida por los SMBD móviles, que incluye la capacidad de:

Comunicarse con el servidor centralizado de la base de datos utilizando técnicas como la comunicación inalámbrica o el acceso a Internet.

Replicar los datos en el servidor de base de datos centralizado y en el dispositivo móvil.

Sincronizar los datos del servidor de base de datos centralizado y en el dispositivo móvil.

Capturar datos de varias fuentes, por ejemplo, de Internet.

Gestionar datos en el dispositivo móvil.

Analizar los datos almacenados en el dispositivo móvil.

Crear aplicaciones móviles personalizadas.

Ventajas e Inconvenientes

Las principales ventajas que hemos detectado en las bases de datos móviles son las siguientes:

Permiten la movilidad de los usuarios, por lo que no es necesario estar físicamente en la organización para acceder a sus datos. Éstos pueden ser accedidos remotamente.

El mercado potencial de este tipo de bases de datos es bastante amplio, ya que multitud de empresas de todo tipo poseen trabajadores que necesitan acceder a los datos de la compañía mientras se encuentran en localizaciones remotas.

Estas bases de datos poseen un gran ámbito de aplicación ya que en principio cualquier base de datos relacional puede ampliarse para ofrecer los servicios de las bases de datos móviles.

Los principales inconvenientes que hemos observado en las bases de datos móviles son los siguientes:

Los enlaces de comunicaciones juegan un papel importante es el desarrollo de estos sistemas, por lo que su dependencia puede suponer un freno para ellos.

Los datos pueden estar replicados, por lo que consistencia y coherencia de los mismos son fundamentales y puede generar conflictos importantes.

El tratamiento de fallos es un aspecto delicado ya que al tratarse de un entorno distribuido, los fallos de transmisión de datos deben de solucionarse y detectarse de forma eficiente para que no produzcan errores en la información tratada.

Persistencia de datos en los dispositivos móviles

- **Concepto de persistencia en dispositivos móviles**

La persistencia en el ámbito de una aplicación indiferentemente si es una aplicación Android o de cualquier otro tipo consiste en que los datos manipulados por la aplicación "sobrevivan" a la ejecución de la misma en el tiempo; en otras palabras; consiste en almacenar los datos en un medio secundario, no volátil para posterior reconstrucción y utilización; por lo tanto son independientes en el tiempo del proceso que los creó.

- **Retos de la persistencia en los dispositivos móviles**

Al manipular los datos se deben encontrar maneras de resolver problemas secundarios que impidan se pierdan los datos que el usuario esta manejando.

- **Formas de persistencia en los sistemas operativos de los dispositivos móviles:**

Preferencias: Almacena datos primitivos y privados en pares clave-valor. ***Almacenamiento de archivos:*** Almacena archivos que tu app pretenda compartir con otras apps, incluidos archivos multimedia, documentos y otros.

Datos estructurados: se utilizan normalmente en bases de datos relaciones, como por ejemplo en bases MySQL. Este tipo de datos se suelen almacenar y mostrar en forma de filas y columnas, y son fácilmente ordenables y procesados por diversas herramientas.

- **Tipos de persistencia:**

Local: Almacenamiento en el propio móvil, espacio limitado, acceso rápido, alto coste de consultas complejas.

Remota: El móvil solo muestra datos, que lee de un servidor, cada acceso a datos requiere una consulta, alta latencia, muy sensible a desconexiones.

Cacheo/Hoarding: En dispositivos móviles: Copia local de alguna información del servidor, comunicación con el servidor para sincronizar, funcional (más o menos) desconectado. Latencia según acierto.

Proceso de programación de persistencia en dispositivos móviles

Con Shared Preferences podemos almacenar y recuperar en el formato clave-valor información como texto, booleanos y números; lo que lo convierte en potencial para almacenar configuraciones del usuario como: estilos, preferencias, etc.

Los modos de acceso posibles son:

MODE_PRIVATE: Sólo nuestra aplicación tiene acceso a estas preferencias.

MODE_WORLD_READABLE: Todas las aplicaciones pueden leer estas preferencias, pero sólo la nuestra puede modificarlas.

MODE_WORLD_WRITEABLE: Todas las aplicaciones pueden leer y modificar estas preferencias.

Su uso es sencillo; si queremos recuperar una "preferencia" que hemos llamado "MiPreferencia" basta con emplear el siguiente código:

SharedPreferences preferencia =

getSharedPreferences("MiPreferencia",Context.MODE_PRIVATE);

Para obtener la información de la preferencia:

SharedPreferences preferencia =

boolean = preferencia.getBoolean("habilitar_imagenes", true)

Los parámetros consisten en:

- El nombre de la preferencia que queremos recuperar.
- Valor por defecto que será utilizado si la preferencia no existe en el sistema.

Para guardar o modificar información de la preferencia:

SharedPreferences preferencia =

boolean = preferencia.putBoolean("habilitar_imagenes", false);

Persistencia en Android: Almacenar archivos en memoria

Este tipo de persistencia es uno de los más conocidos debido a que son soportados por la mayoría de los lenguajes de programación aparte de Java; consiste en guardar y recuperar la información en archivos; Android permite escribir y leer archivos que se encuentren ubicados en la propia Memoria Interna del dispositivo; al igual que con Shared Preferences.

- **Elementos para tomar en cuenta en el desarrollo de aplicaciones orientadas a móviles:**

Interfaz simple, Visibilidad en IOS y Android, Seguridad, Funcionamiento offline de la APP, Actualizaciones periódicas de la APP, Comentarios y medios de contacto, Opciones de personalización, El sistema de búsqueda, Analítica, Interoperabilidad.

- **Lentitud en las consultas**

```
private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {
  // Do the long-running work in here
  protected Long doInBackground(URL... urls) {
     int count = urls.length;
     long totalSize = 0;
     for (int i = 0; i < count; i++) {
        totalSize += Downloader.downloadFile(urls[i]);
        publishProgress((int) ((i / (float) count) * 100));
        // Escape early if cancel() is called
        if (isCancelled()) break;
     }
     return totalSize;
  }

  // This is called each time you call publishProgress()
  protected void onProgressUpdate(Integer... progress) {
     setProgressPercent(progress[0]);
  }

  // This is called when doInBackground() is finished
  protected void onPostExecute(Long result) {
     showNotification("Downloaded " + result + " bytes");
  }
}
```

- **Alto consumo de batería**

La duración de batería es el aspecto más importante de la experiencia del usuario en dispositivos móviles. Un dispositivo sin batería no ofrece ninguna funcionalidad. Por esta razón, es sumamente importante que las apps respeten este aspecto tanto como puedan.

Hay tres conceptos importantes que debes tener en cuenta para que tu app consuma poca batería:

Crea apps de inicialización retrasada.

Aprovecha las funciones de la plataforma, ya que que pueden ayudarte a administrar el consumo de batería de tu app.

Usa herramientas que te permitan identificar las causas del consumo de batería.

- **Dificultad de sincronización**

Cuando intentes sincronizar tu proyecto, es posible que recibas el siguiente mensaje de error: "Connection to the Internet denied. ('Permission denied: connect')". Puedes eliminar este mensaje de error agregando la propiedad de sistema -Djava.net.preferIPv4Stack=true a tu archivo gradle.properties en Android Studio de la siguiente manera:

org.gradle.jvmargs=-Xmx2048m -XX:MaxPermSize=512m -Djava.net.preferIPv4Stack=true

- **Modo off-line**

Para indicarle al sistema de compilación de Android que use los componentes sin conexión que descargaste y descomprimiste, debes crear una secuencia de comandos, como se describe a continuación.

def reposDir = new File(System.properties['user.home'], ".android/manual-offline-m2")

def repos = new ArrayList()

```
reposDir.eachDir {repos.add(it) }
repos.sort()

allprojects {
  buildscript {
    repositories {
      for (repo in repos) {
        maven {
          name = "injected_offline_${repo.name}"
          url = repo.toURI().toURL()
        }
      }
    }
  repositories {
    for (repo in repos) {
      maven {
        name = "injected_offline_${repo.name}"
        url = repo.toURI().toURL()
      }
    }
  }
}
```

- **Recuperación de conexión**

Antes de que una app cliente intente trabajar con un archivo para el que tiene un URI de contenido, puede solicitar información sobre el archivo desde la app de servidor, incluido el

tipo de datos y el tamaño del archivo. El tipo de datos ayuda a la app cliente a determinar si puede procesar el archivo, y el tamaño del archivo la ayuda a configurar el almacenamiento en búfer y en caché del archivo.

```
...

val mimeType: String? = returnIntent.data?.let { returnUri ->
    contentResolver.getType(returnUri)
}
...
```

Proceso de selección de los mecanismos de tolerancia a fallos en el desarrollo de aplicaciones de dispositivos móviles

Una app para Android falla cada vez que se produce una salida inesperada a causa de una señal o excepción no controlada. Una app escrita con Java falla si muestra una excepción no controlada, representada por la clase Throwable. Una app escrita con lenguajes de código nativo falla si se produce durante su ejecución una señal no controlada, como SIGSEGV.

COMPONENTES. Sistemas de hardware respaldados por sistemas idénticos o equivalentes, sistemas de software respaldados por otras instancias de software, fuentes de energía que se hacen tolerantes a fallas utilizando fuentes alternativas.

REQUISITOS. Ningún punto único de falla: si el sistema falla, debe continuar funcionando durante la reparación sin interrupciones, aislamiento de fallas de los componentes defectuosos, contención de fallas para evitar la propagación de la falla, disponibilidad de modos de reversión.

REFERENCIAS

INSTITUTO TECNOLOGICO DE TUXTEPEC. (2017, mayo). *ADMINISTRACION DE APLICACIONES PARA DISPOSITIVOSMOVILES* (N.º 6). ANAYA MANZANO JUAN. https://es.slideshare.net/juan_anaya/unidad-4-administracin-de-datos-en-dispositivos-mviles?from_action=save

AYALA WILSON, J. O. S. E. M. A. N. U. E. L. (2013, 1 octubre). *Bases de datos en dispositivos móviles - Introducción*. Jose Manuel Ayala Wilson. Recuperado 30 de julio de 2022, de https://jmaw.blogspot.com/2012/07/bases-de-datos-en-dispositivos-moviles_23.html

C. (2021, 16 diciembre). *Tolerancia a fallos, qué es y técnicas*. Ciberseguridad. https://ciberseguridad.com/guias/prevencion-proteccion/tolerancia-fallos/

Android Developers. (2022, 22 mayo). *Desarrolladores de Android* |. Recuperado 31 de julio de 2022, de https://developer.android.com/?hl=es-419