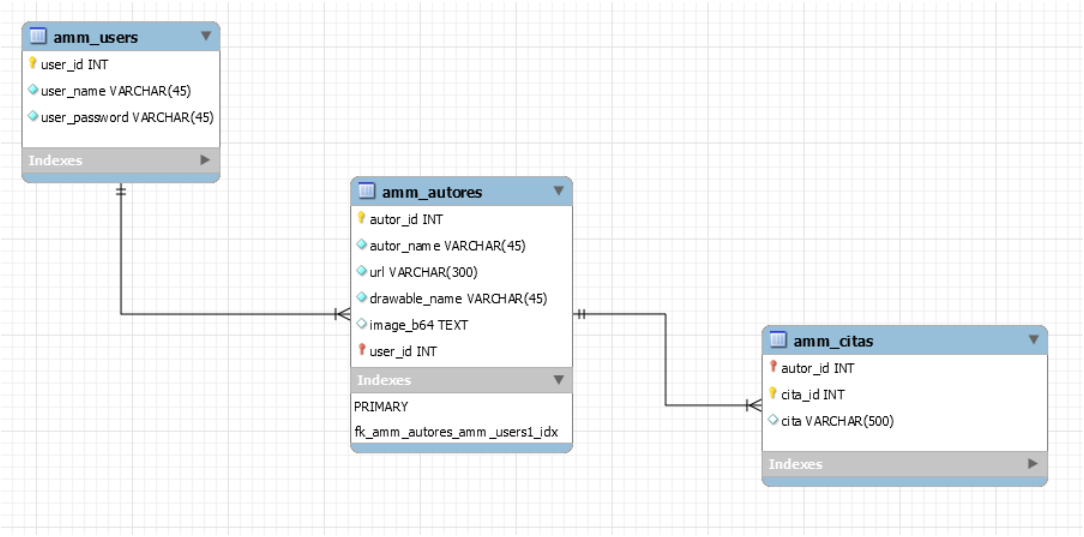
E0606 – CitasCelebres

Basándose en el ejemplo de aplicación E0606_JSON en este mismo tema, y utilizando las funciones del API/REST en http://umhandroid.momrach.es/services modificar la aplicación de CitasCélebres para que recoja del servicio el conjunto de autores y citas que haya en la base de datos para un usuario (alumno) determinado.

Cada alumno usará su nombre de alumno (ammNN) como user_name para no interferir con los dados de otros alumnos.

1. Modelo de datos

El modelo de datos soportado en el servidor para la aplicación citas célebres es el siguiente:



Aunque el conocimiento del modelo de datos es transparente desde para App, puesto que llamamos al servicio y éste nos devuelve un JSON con estos datos, no está de más conocer el modelo de datos.

Cada usuario tiene un conjunto de autores, y estos a su vez tienen un conjunto de citas.

Los usuarios de la tabla amm_users son ammNN donde NN es vuestro número de alumno.

El password es ammNN (donde NN es el número de alumno). Aunque para estel ejercicio este campo no se usará.

El servicio

El único servicio que vamos a utilizar es get_citas_celebres.php cuyos JSON de entrada y salida se listan más abajo en el apartado REST/API.

Simplemente llamando a esta URL en un AsyncTask pasándole por HTTP POST el nombre del usuario (amm01 por ejemplo) nos devolverá un JSON completo con toda la información de autores y citas que haya en la base de datos relativos a dicho usuario.

La App

La APP Citas Celebres cargará inicialmente todos los autores y citas, generando a su vez un conjunto dinámico de botones que ubicará en un scroll view en la Main Activity.

Habrá que adaptar la versión anterior de la App CitasCelebres lo suficiente como para que todo se cargue dinámicamente por ejemplo los botones y las imágenes de los autores, que se reciben codificados en base64 desde el servidor.

Extensión de la clase CitasCelebres

Para realizar esta tarea la App deberá incluir un nuevo método void LoadFromServer(String userName) como método de la clase CitasCelebres que ya tenemos construida. Este método montará un JSON con la entrada necesaria al servicio, instanciará un objeto de la clase HTTPAsyncTask (ver a continuación) y se lo pasará cuando llame a su método .execute()

También añadiremos un método void UseServerJsonData(JSONObject jsonObject) que recibirá un objeto JSON con el retorno del servicio. El método se encargará de construir la lista de botones para colocar en el Scroll View de la

MainActivity, y de añadir a la lista de autores cada uno de los objetos autor que irá construyendo conforme lea el objeto JSON recibido.

Creación de la clase HTTPAsyncTask

Para procesar asíncronamente la solicitud al servidor del servicio deberá implementar una nueva clase class HTTPAsyncTask extends AsyncTask <String, void, String> donde implementará los métodos siguientes miembros y métodos:

private CitasCelebres citasCelebres - Variable miembro que almacenará una instancia del objeto CitasCelebres de la MainActivity, para que el método onPostExecute tenga acceso a dicha clase para llamar a un método de la misma para procesar el JSON que se recibe del servidor.

private ProcessDialog dialog – Un cuadro de diálogo que se mostrará mientras el método doInBackground esté procesando la solicitud al servidor.

HTTPAsyncTask(MainActivity mainActivity) – El constructor, que recibirá el contexto de la MainActivity para poder tener acceso al UIThread en el resto de sus métodos. Inicializará una variable miembro de la clase con dicho contexto, es decir this.citasCelebres = mainActivity.citasCelebres; . Además creará un nuevo cuadro de dialogo que asignará a la variable miembro dialog, es decir: dialog= new ProcessDialog(mainActivity); que recibirá el contexto donde debe ejecutarse, el de la Main Activity, es decir en el UIThread.

void onPreExecute() – Que lanzará el cuadro de diálogo mediante su método .show() habiendo establecido previamente un mensaje que indique que se están cargando los datos, espere…

String doInBackground(String… datos) – que realizará el trabajo. Recibe un array de dos Strings donde el primero será la URL del servicio y el segundo un string JSON que será el que reciba el servicio. Este JSON lo montaremos en la clase CitasCelebres, en el método LoadFromServer donde llamarán al método execute pasándole estos dos Strings. Este método realizará todo el trabajo llamando a un nuevo método String doPost(String urlServicio, String strJson) que es donde colocaremos el código. Se puede colocar todo el código directamente en doInBackground, pero así queda más legible. doInBackground devolverá el string json que doPost le devuelva y que será la entrada a onPostExecute.

doPost(String urlServicio, String strJson) - Este método es el que instancia la conexión con el servidor y realiza la solicitud y captura del JSON de retorno. Es prácticamente igual al que tenemos en la aplicación de ejemplo T0606_JSON.

void onPostExecute(String result) – Este método recibe el string JSON que se ha recibido desde el servidor, lo convertirá en un objeto JSON y lo pasará al método UseServerJsonData de la clase CitasCelebres para ello deberá usar el miembro privado citasCelebres que se instanció en el constructor.

Creación dinámica de los botones

Los botones de los autores se crearán dinámicamente, para ello podéis revisar en el blog de la asignatura un Post donde se explica cómo se pueden crear botones dinámicamente. Se explica no sólo cómo se crean y añaden a un Linear Layout sino que también explica cómo hacer que cada botón tenga su onClick vía una clase propia ButtonsOnClickListener que implementa un View.OnClickListener.

Creación dinámica de los Drawables

Se recibe para cada autor su drawable codificado en base64, con lo que habrá que instanciar un objeto drawable para cada uno dinámicamente en el momento que se procesan los datos en el método UseServerJsonData.

En el blog de la asignatura está explicado cómo se puede leer una imagen base64 en Android y cómo guardarla en el directorio de almacenamiento interno del dispositivo, que es privado para la aplicación y se elimina cuando ésta se desinstala. La entrada tiene por nombre Leer imagen JSON base64 y almacenarla en la memoria interna

2. Funciones del API/REST CitasCelebres

ATENCION: Para este ejercicio únicamente con el servicio get_citas_celebres.php es suficiente, pero si el alumno quiere ampliar el ejercicio dotando a la App de las capacidades de añadir autores y citas, necesitará los demás servicios.

El API está compuesta por una serie de ficheros .php que son llamados mediante HTTP – POST pasándole en cada caso el string JSON correspondiente. Los servicios devuelven un string JSON con el resultado. Algún servicio no precisa de la recepción de datos.

A continuación enumeramos los servicios disponibles, su entrada, salida y su funcionalidad.

2.1. get_citas_celebre.php

```
/*
Recibe:
Un Json con el nombre del usuario
JSON:
{
  "user_name": "amm01"
}

Devuelve:
Un Json con código y texto de resultado
Un array de autores, donde para cada autor se reciben sus datos y un array de sus citas
JSON:
{
  "code": 0,
  "result": "OK",
  "autores": [ {"Nombre": "Miguel de Cervantes",
                        "BiografiaUrl": "http://cervantes.es",
                        "Drawable": "cervantes",
                        "DrawableB64" : "Base64 encoded image",
                        "Citas": [ "En un lugar de la mancha\nde cuyo nombre no quiero
acordarme",
                                    "No ha mucho que vivía un hidalgo ",
                                      "De los de lanza en astillero",
                                      ...
                                     ]
                        },
                        ...
                   ]
}
*/
```

2.2. add_autor.php

```
/*
Recibe:
Un Json con los datos del autor.
Si el autor ya existe para el user_id pasado, actualiza sus datos con los que se pasan.
Si el autor no existe para el user_id pasado, lo crea para ese usuario con los datos pasados.
JSON:
{
  "user_id": id_usuario
  "Nombre": "Miguel de Cervantes",
  "BiografiaUrl": "http://cervantes.es",
  "Drawable": "cervantes"
}

Devuelve:
Un Json con código y texto de resultado.
Además devuelve el author id que se ha dado al nuevo autor.
JSON:
{
  "code": 0,
  "result": "OK",
  "autor_id"
}
*/
```

2.3. set_autor_image.php

```
/*
Recibe:
Un Json con el nombre y el id del autor y la imagen del autor en formato base64
El alto y/o el ancho de la imagen no deben superar los 1200 pixels
JSON:
{
  "autor_id": autor_id
  "Nombre": "Miguel de Cervantes",
  "Imagen": "base64 image data"
}
Devuelve:
Un Json con código y texto de resultado.
JSON:
{
  "code": 0,
  "result": "OK"
}
```

```
*/
```

2.4. add_cita.php

```
/*Recibe:
Un Json con los datos del autor y la cita a añadir
Si el autor no existe devuelve error.
JSON:
{
  "autor_id": autor_id,
  "Nombre": "Miguel de Cervantes",
  "Cita": "En un lugar de la mancha\nde cuyo nombre no quiero acordarme"
}
Devuelve:
Un Json con código y texto de resultado.
También devuelve el cita_id creado
JSON:
{
  "code": 0,
  "result": "OK",
  "cita_id"
}*/
```

2.5. get_autor.php

```
/*Recibe:
Un Json con el nombre del usuario y el nombre del autor
JSON:
{
  "user_name": "amm01",
  "Nombre": "Miguel de Cervantes",
}
Devuelve:
Un Json con código y texto de resultado
Junto con los datos del autor y un array de citas del autor
JSON:
{
  "code": 0,
  "result": "OK",
  "Nombre": "Miguel de Cervantes",
  "BiografiaUrl": "http://cervantes.es",
  "Drawable": "cervantes",
  "citas": [ "En un lugar de la mancha\nde cuyo nombre no quiero acordarme",
             "No ha mucho que vivía un hidalgo ",
             "De los de lanza en astillero"
               ]
}*/
```

2.6. get_cita.php

```
/*
Recibe:
Un Json con el nombre del autor y el número de cita
JSON:
{
  "autor_id": autor_id,
  "cita_id": cita_id
}

Devuelve:
Un Json con código y texto de resultado y la cita.
JSON:
{
  "code": 0,
  "result": "OK",
  "cita": "En un lugar de la mancha\nde cuyo nombre no quiero acordarme"
}
*/
```