Aplicación de verificación mediante webservice con arquitectura REST

Alberto Lara Alberola

Lenguaje: Java

Aplicación para Android

Desarrollado sobre SO Linux 20.04.

Entorno desarrollo: Android Studio Bumblebee — 2021.1.1

6 de abril de 2022

Webservice

Para la creación del servicio web, que dará soporte a la aplicación, se ha utilizado el conjunto Lamp. Esta formado por Apache Web Server, MySQL, el lenguaje PHP y es usado en entornos Linux. Pero en este caso hemos utilizado la base de datos PostgreSQL al ya disponer de ella, el uso de una u otra base de datos no influirá en la aplicación siempre y cuando adaptemos el archivo PHP y la consulta en SQL a la base de datos especifica que queramos utilizar. También podemos basar esta elección en si necesitamos una BD relacional, OR o XML. En este caso la intención es que el webservice nos entrege un objeto Json.

Tabla en la BD

Se ha creado una tabla llamada *registros* dentro de la base de datos *aplicacion*. Esta compuesta por las columnas *usuario*, *contraseña* y *nombre* que son strings y *premium* que es un booleano.

Código PHP

El código es bastante simple ya que vamos a buscar una coincidencia entre un nombre de usuario y una contraseña, así que admitirá el paso de dos parámetros. En caso de no haber coincidencia nos devolverá *false* que usaremos en la lógica de la aplicación para determinar acciones.

```php
<?php

$usuario=$_REQUEST['user'];
$password=$_REQUEST['pas'];
$conexion= pg_connect("host=localhost dbname=aplicacion user=
    odoo password=admin options='--client_encoding=UTF8'");
$query= "SELECT * FROM registros WHERE usuario='$usuario' AND
    contraseña='$password'";
$consulta = pg_query($conexion, $query);

/*Devuelve las coincidencias como un objeto Json*/
$datos=pg_fetch_object($consulta);

/*Codificamos datos en JSON y especificamos también que queremos
     la salida en caracteres unicode
para no tener problemas con acentos o caracteres especiales*/
echo json_encode($datos, JSON_UNESCAPED_UNICODE);

?>
```

Nos devolverá un objeto JSON de esta manera:

```
{"usuario":"albertus","contraseña":"al3","nombre":"Alberto Lara"
    ,"premium":"f"}
```

Proyecto Android

Interfaz. Lenguaje XML

La interfaz utilizada para esta primera versión es bastante simple. Como layout se ha utilizado *constraintlayout* que facilita poder organizar la interfaz de manera libre usando *constraints* en los elementos. Primero vemos una imagen con temática de la aplicación, debajo de esta encontramos un *textview*, que nos informa que debemos identificarnos, y dos *edittext* donde pondremos el usuario y la contraseña que serán usados como parámetros para la consulta en el *webservice*. Finalmente observamos un botón que se ha personalizado mediante un recurso *drawable*. Todo el código esta comentado para más detalle.

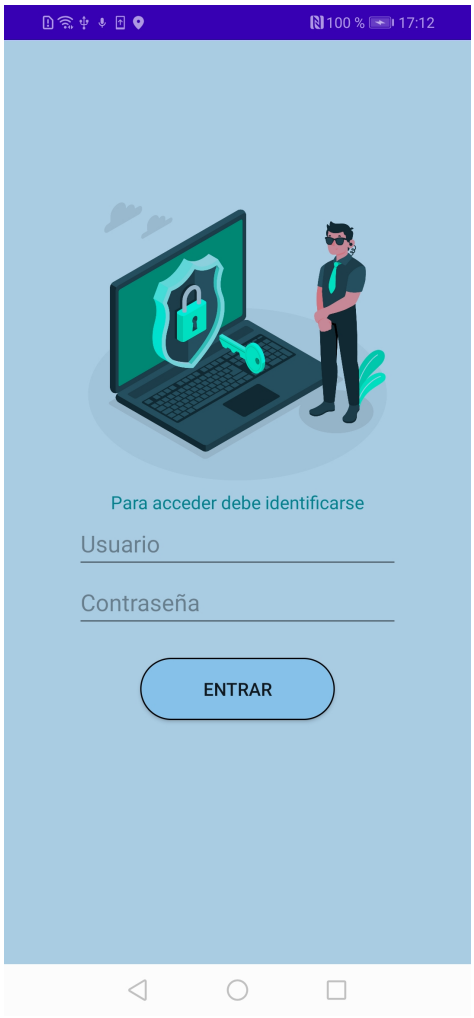

Imagen capturada en dispositivo real.

Lógica. Lenguaje Java

La aplicación se ha creado usando como SDK mínimo **API 26 Android 8.0 Oreo**. Para crear una aplicación como esta, es necesario la utilización de hilos asíncronos para operar con el *webservice* ya que *Android* no permite ejecutar operaciones de red en el hilo principal, esto genera un error. Esto nos obligara a tener en cuenta que el hilo secundario debe entregar los datos al hilo principal o bien implementar que el hilo secundario haga todas las operaciones. Si elegimos que el hilo secundario haga todas las operaciones vamos a tener que ingeniárnoslas bastante para notificar mensajes, como un toast, en el contexto principal, indicando cualquier cosa al usuario. Esto es debido a que el hilo secundario no puede mostrar mensajes en el contexto principal y si quieres poder habilitarlo vas a tener que implementar mucho código, lo que dará como resultado **mayor coste de producción de la aplicación así como de su mantenimiento y documentación**.

La aplicación se ha desarrollado en base a que el hilo secundario entregue los datos al principal. En un primer momento se implemento la aplicación utilizando la clase *AsyncTask* pero no tiene definido un método para hacer que el hilo principal espere a que el secundario coloque los datos en las variables. Se puede pensar que utilizando *yield()* o *sleep()* sobre el hilo principal se podría lograr algo, pero debemos pensar que también cabe la posibilidad de que el webservice tarde en responder por cualquier motivo, además que debe permitir que sea cíclico, es decir yo puedo equivocarme 10 veces seguidas al introducir los parámetros y el método que invoca el botón entrar debe de poder repetirse infinitas veces sin mostrar ningún indicio de fallo. Utilizando estos métodos sobre el hilo principal la aplicación tendera a ser inestable. Si a esto lo sumamos que la clase *AsyncTask* se considera obsoleta, definitivamente no es el camino.

Para Implementar la aplicación se ha utilizado la clase *ExecutorService*. Dispone de los métodos que necesitamos para pasar información al hilo principal y nos permite crear el código casi completamente implementando métodos y clases, con lo cual **todo nuestro código será reutilizable**.