



# Conexión a servidor

ANDROID



Por Diego Ruiz Fernandez

# ¿Qué necesitamos?



Aplicación Android



Aplicación Web



Base de datos

## Flujo general

Aplicación Android



Servidor  
mas scripts php



Base de datos MySQL



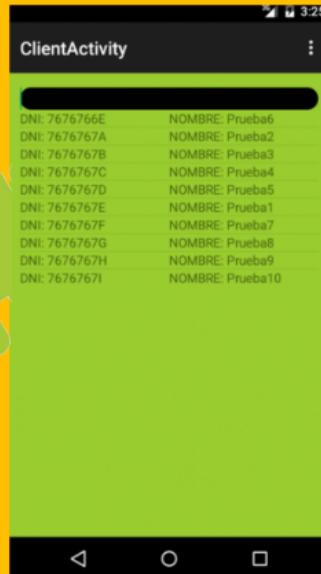
Recepción de datos  
formato json

# Aplicación Android

En nuestra aplicación deberá poder iniciar sesión y si el resultado es satisfactorio pues poder ver la lista de empleados o de clientes.



## Funciones a realizar



# ¿Que necesitamos?

Para nuestra aplicación Android necesitaremos la ayuda de una clase auxiliar que será la que nos ayude a conectar con el servidor, se podría hacer sin una clase auxiliar simplemente poniendo el código en nuestra clase Asyntax pero seria mucho mas extenso y complejo además de poco recomendable.

Pero vamos por partes primero crearemos la clase auxiliar en nuestro caso la clase “Httppostaux” la puedes llamar como quieras aconsejable ponerle un nombre que tenga relacion con la hacion que va ha realizar.

## Clase Auxiliar

Lo primero es tener 2 atributos en nuestra clase un objeto tipo `InputStream` que será el encargado de recoger el flujo devuelto de la conexión y un atributo tipo `String` que veremos mas adelante para que nos sirve.

Después de esto vamos a definir un método que será:  
`private void httppostconnect(ArrayList<NameValuePair> parametros, String urlwebserver)`

Este metodo nos servira para realizar la conexión a el servidor web y poder recibir el flujo de datos de el servidor.

```
private void httppostconnect(ArrayList<NameValuePair>  
parametros,String urlwebserver)
```

Veamos los parametros ArrayList<NameValuePair> parametros y String urlwebserver:

-ArrayList<NameValuePair> son los parametros que se le pasaran a la peticion del servidor. Mediante la encapsulacion de clave, valor.  
NameValuePair es una interface que implementa este tipo de informacion.



Ejemplo de el uso de NameValuePair, como es una interface no podremos añadir directamente un objeto de esa clase en el ArrayList asi que tendremos que buscar una clase que implemente dicha interfaz que seria BasicNameValuePair veamos un ejemplo de su uso:

```
String email = "admi@gmail.com"  
String passwrod = "admin"  
ArrayList<NameValuePair> postparameterssend = new  
ArrayList<NameValuePair>();
```



```
postparameters.add(new BasicNameValuePair("usuario", email));  
postparameters.add(new BasicNameValuePair("password", password));  
La clave y el valor siempre tendran que ser de tipo String y serviran para luego  
con esa clave interactuar el scrip php con la base de datos. Mas info:  
http://developer.android.com/reference/org/apache/http/NameValuePair.html  
http://developer.android.com/reference/org/apache/http/message/  
BasicNameValuePair.html
```



-String urlwebserver : este parámetro es la dirección ip del equipo o dominio del servidor mas la ruta donde se encuentra nuestros scrip php. Ej:

```
public static final String IP_SERVER = "10.0.2.2:8080"; //ip local del sdk de android y  
private final String URL_CONECT = "http://" + IP_SERVER+ "/ChiefManagement/acces.php";
```

Si en vez de ser un ip pues tendríamos que poner el nombre del dominio como seria:

"http://www.google.com/ChiefManagement/acces.php"



Ahora veremos el interior del código del método `private void httppostconnect(ArrayList<NameValuePair> parametros, String urlwebserver)`

```
/*
 * Metodo para realizar las peticiones http
 * @param parameters son los paramentros de la peticion si los hay sirve
 * @param urlwebserver urlwebserver es la direccion donde tendra que conectarse
 */
private void httppostconnect(ArrayList<NameValuePair> parametros,
    String urlwebserver) {

    //
    try {
        HttpClient httpclient = new DefaultHttpClient();
        HttpPost httppost = new HttpPost(urlwebserver);
        httppost.setEntity(new UrlEncodedFormEntity(parametros));
        // ejecuto peticion enviando datos por POST
        HttpResponse response = httpclient.execute(httppost);
        HttpEntity entity = response.getEntity();
        is = entity.getContent();

    } catch (Exception e) {
        Log.e("log_tag", "Error in http connection " + e.toString());
    }
}
```



Vamos a explicar línea a línea lo que hace:

-**HttpClient**: Clase encargada de enviar la información almacenada en **HttpPost** a nuestro **WebService**. Mas info:

<http://developer.android.com/reference/org/apache/http/client/HttpClient.html>

-**HttpPost**: almacena los datos que serán enviados, por medio de **HttpClient**, a nuestro **WebService**. Mas info:

<http://developer.android.com/reference/org/apache/http/client/methods/HttpPost.html>



-`httpPost.setEntity(new UrlEncodedFormEntity(parametros));`

Lo que estamos haciendo es a la variable `HttpPost` fijarle los datos para la conexión y los parámetros.

-`HttpResponse`: Clase que nos permite guardar la respuesta del servidor a la petición hecha por el cliente mediante el método `httpClient.execute(httpPost);`

Mas info:

<http://developer.android.com/reference/org/apache/http/HttpResponse.html>



-**HttpEntity**: nos permite interactuar con los datos de la respuesta del servidor.  
Como por ejemplo coger el flujo de entrada para poder leer los datos del servidor.

Ej:

```
HttpEntity entity = response.getEntity();
```

```
InputStream is = entity.getContent();
```

Mas info:

<http://developer.android.com/reference/org/apache/http/HttpEntity.html>



Siguiente método de nuestra clase auxiliar ahora nuestro atributo String creado anteriormente nos será de utilidad puesto que será donde guardemos los datos del InputStream recogido anteriormente. No explicare nada porque ya deberíamos saber entenderlo

```
/*
 * Metodo para convertir la respuesta a la peticion http en String
 */
public void getpostresponse() {

    try {
        BufferedReader reader = new BufferedReader(new InputStreamReader(
                is, "iso-8859-1"), 8);
        StringBuilder sb = new StringBuilder();
        String line = null;
        while ((line = reader.readLine()) != null) {
            sb.append(line + "\n");
        }
        is.close();

        result = sb.toString();
        Log.e("getpostresponse", " result= " + sb.toString());
    } catch (Exception e) {
        Log.e("log_tag", "Error converting result " + e.toString());
    }
}
```

Luego crearemos los métodos para convertir los datos del atributo String a objetos JSONObject o JSONArray según convenga. Mas info: Juanvi.

```
/**  
 * Metodo para pasar los datos obtenidos de un String de la peticion http a un objeto JSONObject  
 * @return un objeto JSONObject o null sino se a podido parsear  
 */  
public JSONObject getjsonObject() {  
    try {  
        JSONObject jobject = new JSONObject(result);  
  
        return jobject;  
    } catch (JSONException e) {  
        Log.e("log_tag", "Error parsing data " + e.toString());  
        return null;  
    }  
}  
  
/**  
 * Metodo para pasar los datos obtenidos de un String de la peticion http a un objeto JSONArray  
 * @return un objeto JSONArray o null sino se a podido parsear  
 */  
public JSONArray getJSONArray(){  
    try {  
        JSONArray jArray = new JSONArray(result);  
  
        return jArray;  
    } catch (JSONException e) {  
        Log.e("log_tag", "Error parsing data " + e.toString());  
        return null;  
    }  
}
```



Después codificaremos los métodos para la obtención de los datos desde nuestra clase Android. Para obtener datos JSONArray:

```
/**  
 * Metodo para obtener un JSONArray mediante una peticion webservice  
 * @param parameters son los paramentros de la peticion si los hay sino solo le pasamos un ArrayList  
 * @param urlwebserver es la direccion donde tendra que conectarse para obtener los datos de la base  
 * @return un objeto JSONAarry si hay datos o null si no hay datos  
 */  
public JSONArray getserverdataArray(ArrayList<NameValuePair> parameters,  
        String urlwebserver) {  
  
    // conecta via http y envia un post.  
    httppostconnect(parameters, urlwebserver);  
  
    if (is != null) {// si obtuvo una respuesta  
  
        getpostresponse();  
  
        return getjsonArray();  
    } else {  
  
        return null;  
    }  
}
```



# Y para la obtención de un objeto JSONObject

```
/**  
 * Metodo para obtener un objeto JSONObject mediante una peticion a un webservice  
 * @param parameters son los paramentros de la peticion si los hay sino solo le pasamos un ArrayList  
 * @param urlwebserver urlwebserver es la direccion donde tendra que conectarse para obtener los datos  
 * @return un objeto JSONObject si hay datos o null sino hay datos  
 */  
public JSONObject getserverdata(ArrayList<NameValuePair> parameters,  
        String urlwebserver) {  
  
    // conecta via http y envia un post.  
    httppostconnect(parameters, urlwebserver);  
  
    if (is != null) {// si obtuvo una respuesta  
  
        getpostresponse();  
  
        return getjsonObject();  
    } else {  
  
        return null;  
    }  
}
```



## Clase Android

Ahora vamos con las clases Android. Para nuestro ejemplo nos fijaremos en la clase LoginActivity que será la encargada de recibir los datos del usuario para iniciar sesión y con la ayuda de la clase auxiliar y una clase AsyncTask realizará la petición al servidor. Todo esto se hará cuando pulse el botón Iniciar Sesión

No explicaré el código Android y me centraré en el código interesante para nuestro webservice, es decir las comprobaciones de campos vacíos et...

Crearemos un método boolean loginstatus(String email, String password) que nos devolverá true si la comprobación del servidor con el email es correcto o false sino es correcto.

Tendremos que tener iniciada nuestra clase auxiliar e inicializada.

```
private Htppostaux post;
private ProgressDialog pDialog;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);
    // Inicializo los atributos
    edEmail = (EditText) findViewById(R.id.ed_email);
    edPass = (EditText) findViewById(R.id.ed_password);
    btStart = (Button) findViewById(R.id.bt_iniciar);
    post = new Htppostaux();
}
```



Nuestro código del método por partes, Primera parte: Como podéis comprobar aquí pasamos los parámetros para la petición y la URL\_CONNECT que es la ip y la ruta anteriormente descrita  
Y obtenemos los datos del servidor.

```
/*
 * Valida el estado del logueo solamente necesita como parametros el usuario
 * y password
 */
public boolean loginstatus(String email, String password) {
    int logstatus = -1;

    /Creamos un ArrayList del tipo nombre valor para agregar los datos
    /recibidos por los parametros anteriores y enviarlo mediante POST a
    /nuestro sistema para realizar la validacion

    ArrayList<NameValuePair> postparameterssend = new ArrayList<NameValuePair>();

    postparameterssend.add(new BasicNameValuePair("usuario", email));
    postparameterssend.add(new BasicNameValuePair("password", password));

    // realizamos una peticion y como respuesta obtenes un array JSON
    JSONArray jdata = post.getserverdataArray(postparameterssend, URL_CONECT);
```



## Segunda parte de nuestro código del método loginstatus.

Acedemos a los datos y en función de lo que devuelva pues será true o false.

```
// si lo que obtuvimos no es null
if (jdata != null && jdata.length() > 0) {

    JSONObject json_data; // creamos un objeto JSON
    try {
        json_data = jdata.getJSONObject(0); // leemos el primer segmento
                                            // en nuestro caso el único
        logstatus = json_data.getInt("logstatus"); // accedemos al valor
        Log.e("loginstatus", "logstatus= " + logstatus); // muestro por
                                                       // log que
                                                       // obtuvimos
    } catch (JSONException e) {
        e.printStackTrace();
    }

    // validamos el valor obtenido
    if (logstatus == 0) { // [{"logstatus": "0"}]
        Log.e("loginstatus ", "invalido");
        return false;
    } else { // [{"logstatus": "1"}]
        Log.e("loginstatus ", "valido");
        return true;
    }

} else { // json obtenido invalido verificar parte WEB.
    Log.e("JSON ", "ERROR");
    return false;
}
```



Después crearemos una clase Asyntack para que se ejecute nuestra petición al servidor en segundo plano. Mas info: apuntes de Julita. Llamamos a loginstatus en doInBacground si devuelve true en onPostExecute nos lanzara a otra activity.

```
/**  
 * Acción principal  
 */  
protected String doInBackground(String... params) {  
    // obtenemos email y pass  
    email = params[0];  
    pass = params[1];  
  
    // enviamos y recibimos y analizamos los datos en segundo plano.  
    if (loginstatus(email, pass) == true) {  
        return "ok"; // login valido  
    } else {  
        return "err"; // login invalido  
    }  
}
```



## Lo que realizaría en onPostExecute:

```
/**  
 * Método que se ejecuta después de la acción principal  
 */  
protected void onPostExecute(String result) {  
  
    pDialog.dismiss(); //ocultamos progres dialog.  
    Log.e("onPostExecute=",""+result);  
  
    if (result.equals("ok")){  
        //Lanzamos otra activity donde tendremos las opciones de elegir clientes y empleados  
        Intent intent=new Intent(LoginActivity.this,OptionActivity.class);  
        startActivity(intent);  
  
    }else{  
        //Mostramos mensaje de error  
        Toast.makeText(LoginActivity.this, getResources().getString(R.string.tx_toast_error),  
    }  
}  
}
```



Para obtener las listas de clientes o empleados aparte de tener nuestro adapter y clases envolventes para los datos y clases que gestionen los datos el código relacionado con el webService seria aparte de pasarle los parámetros y la url como en el anterior todo esto dentro de un método y luego llamado desde nuestra clase AsynTack.

Básicamente recoge y recorre todos los datos obtenidos luego habría que fijarlos al adapter.

```
JSONObject jobject = post.getserverdata(postparametersend, URL_CONECT_CLIE);
//Para pasar el JSONObject a un array con los datos de la base de datos
JSONArray jdata = jobject.optJSONArray("clientes");
// si lo que obtuvimos no es null
if (jdata != null && jdata.length() > 0) {
    try {
        for (int i = 0; i < jdata.length(); i++) {
            Cliente cliente = new Cliente();
            //Cojemos el elemento del array de JSON
            JSONObject jsonArrayChild = jdata.getJSONObject(i);
            //Cojemos los datos de objeto JSONObject mediante el get y el nombre de la columna de la
            cliente.setDni(jsonArrayChild.getString("dni"));
            cliente.setNombre(jsonArrayChild.getString("nombre"));
            cliente.setApellido1(jsonArrayChild.getString("primerApellido"));
            cliente.setApellido2(jsonArrayChild.getString("segundoApellido"));
            cliente.setTelefono(Integer.parseInt(jsonArrayChild.getString("telefono")));
            cliente.setDireccion(jsonArrayChild.getString("direccion"));
            //Añadimos a nuestro gestor un cliente con los datos de la base de datos
            gestorClie.addCliente(cliente);
        }
        return true;
    } catch (JSONException e) {
        e.printStackTrace();
        return false;
    }
}
```



## Aplicación Web

Básicamente es la encargada de la comunicación entre dispositivo y base de datos.

¿Qué necesita para funcionar?

En este caso un servidor web como puede ser apache, scripts php que permiten la conexión e interaccionar con la base de datos.

Para el servidor vamos a utilizar el programa XAMPP

# XAMPP

Programa para el servidor  
apache, instalación y  
configuraron

Instalación de XAMPP es una instalación normal como cualquier programa. Después de instalado deberemos ir a el directorio donde se ha instalado y configurar unos datos para el servidor apache que trae el programa.

Pulsamos en la carpeta apache.

anonymous	18/01/2015 19:41	Carpeta de archivos
apache	18/01/2015 19:41	Carpeta de archivos
cgi-bin	18/01/2015 19:46	Carpeta de archivos
contrib	18/01/2015 19:41	Carpeta de archivos
FileZillaFTP	18/01/2015 19:46	Carpeta de archivos
htdocs	18/01/2015 19:55	Carpeta de archivos
img	18/01/2015 19:41	Carpeta de archivos
install	18/01/2015 19:46	Carpeta de archivos
licenses	18/01/2015 19:41	Carpeta de archivos



## Luego pulsar en conf

bin	18/01/2015 19:41	Carpeta de archivos
conf	18/01/2015 19:46	Carpeta de archivos
error	18/01/2015 19:41	Carpeta de archivos
icons	18/01/2015 19:41	Carpeta de archivos
include	18/01/2015 19:41	Carpeta de archivos

Y despues dentro de conf editar el archivo httpd.conf y cambiar  
ServerName localhost:80 por ServerName localhost:8080.  
Listen 80 por Listen 8080.

extra	18/01/2015 19:46	Carpeta de archivos
original	18/01/2015 19:41	Carpeta de archivos
ssl.crt	18/01/2015 19:41	Carpeta de archivos
ssl.csr	18/01/2015 19:41	Carpeta de archivos
ssl.key	18/01/2015 19:41	Carpeta de archivos
charset.conv	17/07/2014 13:50	Archivo CONV 2 KB
httpd.conf	18/01/2015 20:52	Archivo CONF 21 KB

Después en la misma carpeta de conf entrar en la carpeta extra.

 extra	18/01/2015 19:46	Carpeta de archivos
 original	18/01/2015 19:41	Carpeta de archivos
 ssl.crt	18/01/2015 19:41	Carpeta de archivos
 ssl.csr	18/01/2015 19:41	Carpeta de archivos
 ssl.key	18/01/2015 19:41	Carpeta de archivos
 charset.conv	17/07/2014 13:50	Archivo CONV 2 KB

Dentro de extra editar el archivo httpd-ssl.conf y cambiar: <VirtualHost \_default\_:443> por <VirtualHost \_default\_:4430>.

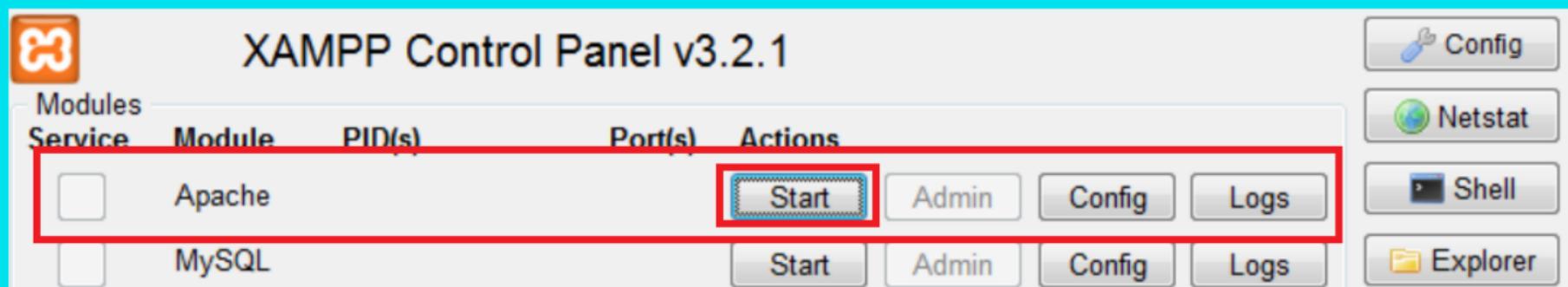
ServerName localhost:443 por ServerName localhost:4430.

Listen 443 por Listen 4430.

 extra	18/01/2015 19:46	Carpeta de archivos
 original	18/01/2015 19:41	Carpeta de archivos
 ssl.crt	18/01/2015 19:41	Carpeta de archivos
 ssl.csr	18/01/2015 19:41	Carpeta de archivos
 ssl.key	18/01/2015 19:41	Carpeta de archivos
 charset.conv	17/07/2014 13:50	Archivo CONV 2 KB



Para arrancar el apache vamos a inicio y en XAMPP pulsamos XAMPP control panel y después aparecerá una ventana y donde pone apache le damos a start.



Ahora empezaremos a colocar nuestros scripts php que deberán situarse en una carpeta que creemos por ejemplo el nombre del proyecto y esto irá dentro de la carpeta de instalación de XAMPP dentro de la carpeta htdocs

anonymous	18/01/2015 19:41	Carpeta de archivos
apache	18/01/2015 19:41	Carpeta de archivos
cgi-bin	18/01/2015 19:46	Carpeta de archivos
contrib	18/01/2015 19:41	Carpeta de archivos
FileZillaFTP	18/01/2015 19:46	Carpeta de archivos
htdocs	18/01/2015 19:55	Carpeta de archivos
img	18/01/2015 19:41	Carpeta de archivos
ChiefManagement	24/01/2015 20:26	Carpeta de archivos
dashboard	18/01/2015 19:41	Carpeta de archivos
forbidden	18/01/2015 19:41	Carpeta de archivos
img	18/01/2015 19:41	Carpeta de archivos
restricted	18/01/2015 19:41	Carpeta de archivos

ChiefManagement es la carpeta donde tengo los scripts php de mi proyecto



# SCRIPTS PHP

-php: es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.

Nuestra aplicación contara con los siguiente archivos

php:

acces.php

config.php

conectbd.php

funciones\_bd.php

showcustomers.php

showemployers.php

-acces.php: Encargado de comprobar si el email y el password son correctos.

```
/*LOGIN*/  
//variables de la peticion recibida  
$usuario = $_POST['usuario'];  
$passw = $_POST['password'];  
  
//inicializa funciones_bd.php  
require_once 'funciones_bd.php';  
$db = new funciones_BD();  
    //llama a la funcion login que se encuentra en funciones_bd  
if($db->login($usuario,$passw)) {  
    ...  
    $resultado []=array ("logstatus"=>"0");  
} else {  
    $resultado []=array ("logstatus"=>"1");  
}  
  
echo json_encode ($resultado);
```



-config.php: encargado de definir los datos de la base de datos MySQL

```
/**  
 * Database config variables  
 */  
define ("DB_HOST", "localhost");//ip de la base de datos  
define ("DB_USER", "root");//cambiar por el nombre de usuario definido en la configuracion de la BD.  
define ("DB_PASSWORD", "root");//Modificar por el password elegido  
define ("DB_DATABASE", "chiefmanagement");//Nombre de la base de datos reemplazar si se utilizo otro diferente
```



-conectbd.php: Encargada de la conexión a la base de datos.

```
class DB_Connect {

    // constructor
    function __construct() {
    }

    // destructor
    function __destruct() {
        // $this->close();
    }

    // Connecting to database
    public function connect() {
        require_once 'config.php';
        // connecting to mysql
        $con = mysql_connect(DB_HOST, DB_USER, DB_PASSWORD);
        // selecting database
        mysql_select_db(DB_DATABASE);

        // return database handler
        return $con;
    }

    // Closing database connection
    public function close() {
        mysql_close();
    }
}
```



-funciones\_bd.php: Encargada de guardar las peticiones a las bases de datos y demás funciones.

```
class funciones_BD {  
  
    private $db;  
  
    // constructor  
  
    function __construct() {  
        require_once 'connectbd.php';  
        // connecting to database  
  
        $this->db = new DB_Connect();  
        $this->db->connect();  
  
    }  
  
    // destructor  
    function __destruct() {  
    }  
}
```



```
public function login($user,$passw){  
  
$result=mysql_query("SELECT COUNT(*) FROM administradores WHERE email='$user' AND password='$passw' ");  
$count = mysql_fetch_row($result);  
  
/*como el usuario debe ser unico cuenta el numero de ocurrencias con esos datos*/  
/*  
 *  
 */  
  
if ($count[0]==0){  
    return true;  
  
}else{  
    return false;  
}  
}
```



```
public function clientes() {
    $query_search = "select * from clientes order by dni";
    $query_exec = mysql_query($query_search) or die(mysql_error());
    $json = array();
    if(mysql_num_rows($query_exec)) {
        while($row=mysql_fetch_assoc($query_exec)) {
            $json['clientes'][]=$row;
        }
    }
    echo json_encode($json);
}

public function empleados() {
    $query_search = "select * from empleados order by dni";
    $query_exec = mysql_query($query_search) or die(mysql_error());
    $json = array();
    if(mysql_num_rows($query_exec)) {
        while($row=mysql_fetch_assoc($query_exec)) {
            $json['empleados'][]=$row;
        }
    }
    echo json_encode($json);
}
```



-showcustomers.php: Encargada de obtener los datos de los clientes.

```
require_once 'funciones_bd.php';
$db = new funciones_BD();
$db->clientes();
```

-showemployers.php: Encargada de obtener los datos de los empleados.

```
require_once 'funciones_bd.php';
$db = new funciones_BD();
$db->empleados();
```



## Base de datos

Para nuestro ejemplo necesitaremos tener instalado MySQL workbench pero el propio XAMPP nos podría permitir la conexión a mysql y phpadmin para realizar la creación de tablas e inserción de datos.

Lo único que necesitamos es la creación de una base de datos con los campos que queremos y que coincidan con los campos de los scripts php para poder interactuar con ellos si fuera necesario.

# Referencias

En este apartado destacaremos las referencias tanto de fuentes consultadas como las referencias de software utilizado.

## Fuentes Utilizadas

En este apartado mostraremos los enlaces a las fuentes donde hemos encontrado información para la realización del proyecto

# Uso de web service en android con php y MySQL

- [http://www.tutorialspoint.com/android/android\\_php\\_mysql.htm](http://www.tutorialspoint.com/android/android_php_mysql.htm)
- <http://androideity.com/2012/07/05/login-en-android-usando-php-y-mysql/>
- <http://www.androidhive.info/2012/05/how-to-connect-android-with-php-mysql/>
- <http://picarcodigo.blogspot.com.es/2014/05/webservice-conexiones-base-de-datos.html>



# Corregir errores de XAMPP

-<http://tutorialesyopiniones.blogspot.com.es/2013/01/corregir-error-al-iniciar-xampp-apache.html>

# Clases y código Android

-<http://developer.android.com/index.html>

# Software

En este apartado dejaremos enlaces para poder descargar el software necesario para el funcionamiento de nuestra aplicación

# XAMPP

-<https://www.apachefriends.org/es/download.html>



## Android y eclipse

-<http://www.javaya.com.ar/androidya/detalleconcepto.php?codigo=132&inicio=0>

# MySQL

-<http://www.mysql.com/downloads/>