

# TALLER 9. Aplicaciones web

## Inicio

El objetivo de este taller es entender los conceptos básicos relacionados con desarrollo de aplicaciones web, como mecanismo para la visualización de los datos.

Se trabajará sobre el proyecto de fin de curso. Supondremos que se requiere una aplicación web para interactuar con los usuarios finales. Se desarrollará una aplicación web que acceda al servidor IoT desarrollado, para ejecutar las operaciones GET.

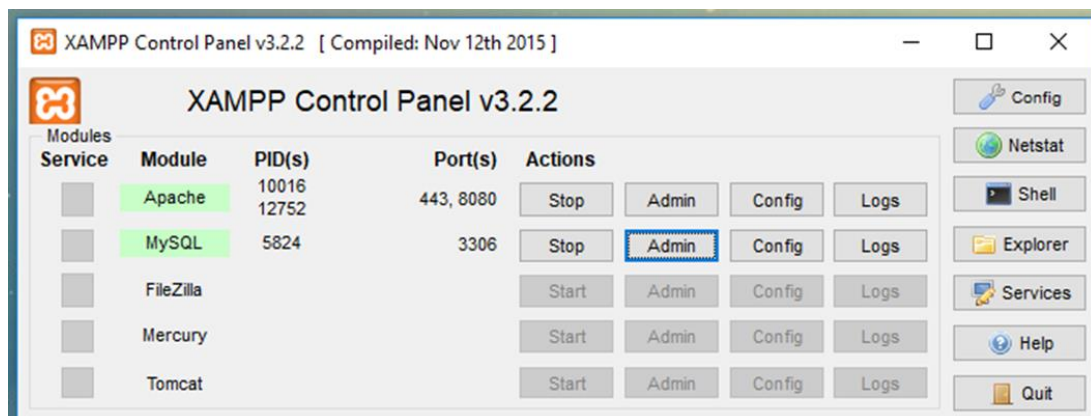
El desarrollo del taller se realizará en 2 fases:

1. Configuración y desarrollo de la aplicación web para acceder al servidor IoT remoto (aws) desde un servidor web local.
2. Configuración de un servidor web remoto (máquina virtual aws) donde se monte el servidor web y se suban los archivos de la aplicación web desarrollada. Este servidor deberá estar montado en una máquina diferente a la máquina del servidor Rest.

## Procedimiento – Servidor web local

1. Instalación de Apache

La instalación de Apache se puede hacer a través de XAMPP, el cual ya lo tenemos instalado.

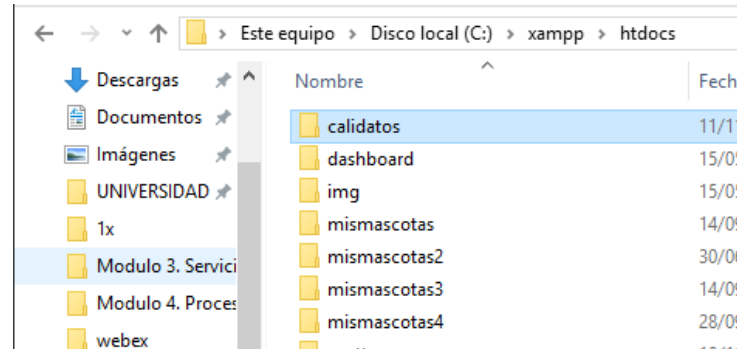


2. Definición de las funcionalidades que prestará la aplicación web a desarrollar. Partiremos de un ejemplo simple con requerimientos básicos.
  - Restringir el ingreso a la aplicación a algunos usuarios únicamente.
  - Mostrar un listado de las variables que se pueden ver a través de la aplicación.
  - Permitir que el usuario seleccione la variable que quiere ver.

- Mostrar el histórico de datos asociados a la variable seleccionada por el usuario.

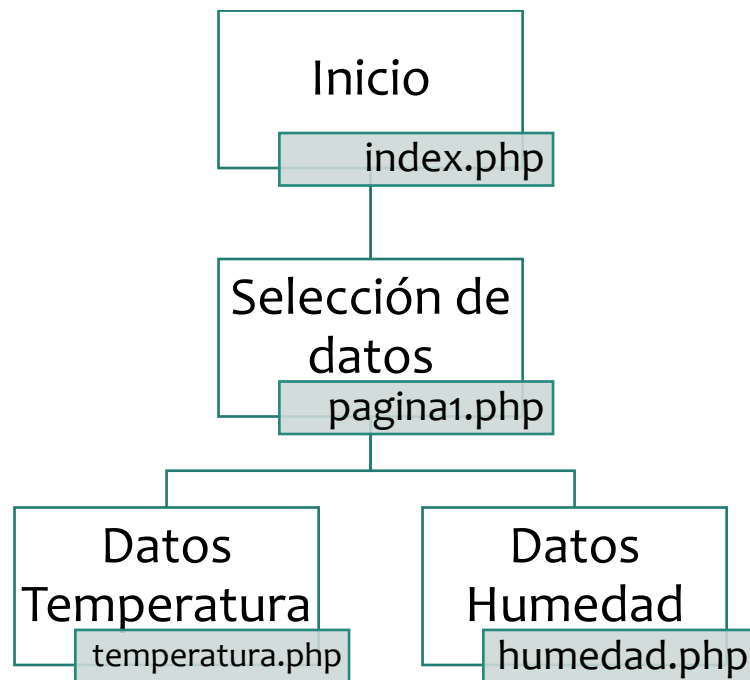
### 3. Creación de la carpeta que contendrá la aplicación

Dentro de la carpeta **htdocs** que se encuentra en **Xampp**, creamos una carpeta a la cual le daremos un nombre relacionado con el proyecto, en mi caso la llamaré **calidatos**.



### 4. Creación de las páginas html de la aplicación web.

Lo primero es definir la estructura de la aplicación web.



La página **index.php** contendrá un formulario de ingreso, en donde se pedirá un login y una contraseña. Si los datos son correctos se accederá a **pagina1.php** en caso contrario se continuará en la misma página.

El código básico de esa página es el siguiente:

```

<!DOCTYPE html>
<html>
  <head>
    <title>CaliDatos</title>
  </head>
  <body>

```

```

<?php
    if(isset($_POST['enviar'])){
        $login = $_POST['login'];
        $password = $_POST['pass'];
        if ($login=="admin" && $password=="1234"){
            session_start();
            $_SESSION['usuario']=$login;
            header("Location: pagina1.php");
        }
        else{
            session_start();
            $_SESSION['usuario']="";
            header("Location: index.php");
        }
    }
    else{
?>
    <br>
    <h1>CALI DATOS</h1>
    <h2>INGRESO</h2>
    <form action="index.php" method="POST">
        LOGIN: <br>
        <input type="text" name="login" width="50"><br><br>
        PASSWORD: <br>
        <input type="password" name="pass" width="50"><br><br>
        <input type="submit" name="enviar" value="ENVIAR">
    </form>
    <?php } ?>
</body>
</html>

```

En **pagina1.php** lo que se hace es validar que se inició la sesión y poner hipervínculos a las páginas de **temperatura.php** y **humedad.php**. El código básico de **pagina1.php** es:

```

<!DOCTYPE html>
<html>
    <head>
        <title>calidatos/pagina1.php</title>
    </head>
    <body>
        <?php
            session_start();
            $us=$_SESSION['usuario'];
            if($us=="")
            {
                header("Location: index.php");
            }
        ?>
        <h1>CALI DATOS</h1>
        <h2>Seleccione una variable:</h2>
        <h3><a href="temperatura.php">TEMPERATURA</a></h3>
        <h3><a href="humedad.php">HUMEDAD</a></h3>
        <h4><a href="logout.php">CERRAR SESION</a></h4>
    </body>
</html>

```

Las páginas de **temperatura.php** y **humedad.php** verifican inicialmente que la sesión haya sido iniciada y después llaman al servidor IoT a través de una operación GET y muestran los datos en una tabla. El código de estas páginas se muestra a continuación:

### Temperatura.php

```
<html>
<head>
  <title>calidatos/pagina1.php</title>
</head>
<body>
  <?php
    session_start();
    $us=$_SESSION['usuario'];
    if($us=="")
    {
      header("Location: index.php");
    }
  ?>
  <h1>CALI DATOS</h1>
  <h2>Datos de Temperatura</h2>
  <table border="2px">
    <tr><th>ID NODO</th><th>TEMPERATURA</th><th>FECHA</th></tr>
    <?php
      $url_rest = "http://ec2-54-226-231-84.compute-1.amazonaws.com:3000/datos";
      $curl = curl_init($url_rest);
      curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
      $respuesta = curl_exec($curl);

      if($respuesta===false){
        curl_close($curl);
        die("Error...");
      }

      curl_close($curl);
      //echo $respuesta;
      $resp = json_decode($respuesta);
      $tam = count($resp);

      for ($i=0; $i<$tam; $i++){
        $j = $resp[$i];
        $idnodo = $j-> idnodo;
        $temp = $j-> temperatura;
        $fecha = $j-> fecha;
        echo "<tr><td>$idnodo</td><td>$temp</td><td>$fecha</td></tr>";
      }
    ?>
  </table>
  <a href="pagina1.php">Volver</a><br>
  <a href="logout.php">Cerrar sesión</a>
</body>
</html>
```

### humedad.php

```
<html>
<head>
  <title>calidatos/pagina1.php</title>
</head>
```

```

<body>
<?php
    session_start();
    $us=$_SESSION['usuario'];
    if($us=="")
    {
        header("Location: index.php");
    }
?>
<h1>CALI DATOS</h1>
<h2>Datos de Humedad</h2>
<table border="2px">
    <tr><th>ID NODO</th><th>HUMEDAD</th><th>FECHA</th></tr>
    <?php
        $url_rest = "http://ec2-54-226-231-84.compute-1.amazonaws.com:3000/datos";
        $curl = curl_init($url_rest);
        curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
        $respuesta = curl_exec($curl);

        if($respuesta===false){
            curl_close($curl);
            die ("Error...");
        }

        curl_close($curl);
        //echo $respuesta;
        $resp = json_decode($respuesta);
        $tam = count($resp);

        for ($i=0; $i<$tam; $i++){
            $j = $resp[$i];
            $idnodo = $j->idnodo;
            $hum = $j->humedad;
            $fecha = $j->fecha;
            echo "<tr><td>$idnodo</td><td>$hum</td><td>$fecha</td></tr>";
        }
    ?>
</table>
<a href="pagina1.php">Volver</a><br>
<a href="logout.php">Cerrar sesión</a>
</body>
</html>

```

Por último creamos una página denominada **logout.php** para cerrar la sesión. El código es el siguiente:

```

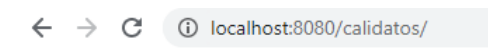
<?php
    session_start();
    session_destroy();
    header("Location: index.php");
?>

```

## 5. Prueba del servicio

Como la aplicación web es local, nos ubicamos en el navegador y colocamos la url <http://localhost:8080/calidatos/>, en mi caso se usa el puerto 8080 porque mi apache se está ejecutando en ese puerto. Si el puerto usado es 80 no hay necesidad de colocarlo en la url.

El resultado es el siguiente:



## CALI DATOS

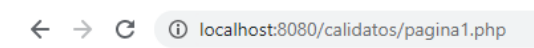
### INGRESO

LOGIN:

PASSWORD:

ENVIAR

Cuando el usuario ingresa las credenciales adecuadas (admin y 1234) pasa a la siguiente página:



## CALI DATOS

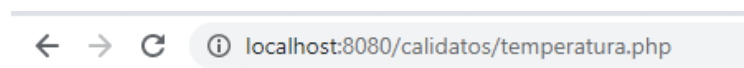
**Seleccione una variable:**

[TEMPERATURA](#)

[HUMEDAD](#)

[CERRAR SESION](#)

Cuando selecciona una de las variables se despliegan los datos:



## CALI DATOS

### Datos de Temperatura

ID NODO	TEMPERATURA	FECHA
1	25	10/11/2020 5:13 p.m.
2	25	10/11/2020 5:13 p.m.

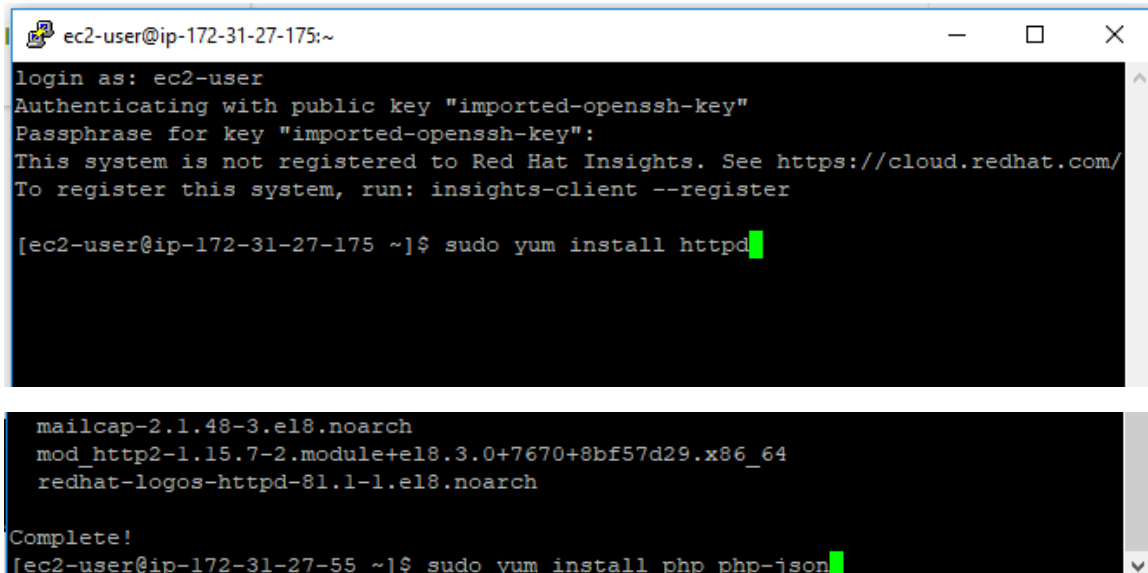
[Volver](#)

[Cerrar sesión](#)

# Procedimiento – Servidor web remoto

## 1. Instalación de apache

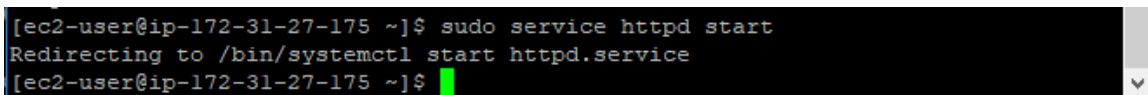
En la instancia de aws creada, instalar apache, php y el soporte de json para php



```
ec2-user@ip-172-31-27-175:~  
login as: ec2-user  
Authenticating with public key "imported-openssh-key"  
Passphrase for key "imported-openssh-key":  
This system is not registered to Red Hat Insights. See https://cloud.redhat.com/  
To register this system, run: insights-client --register  
  
[ec2-user@ip-172-31-27-175 ~]$ sudo yum install httpd  
  
mailcap-2.1.48-3.el8.noarch  
mod_http2-1.15.7-2.module+el8.3.0+7670+8bf57d29.x86_64  
redhat-logos-httpd-81.1-1.el8.noarch  
  
Complete!  
[ec2-user@ip-172-31-27-55 ~]$ sudo yum install php php-json
```

## 2. Inicio del servidor apache y creación de la aplicación

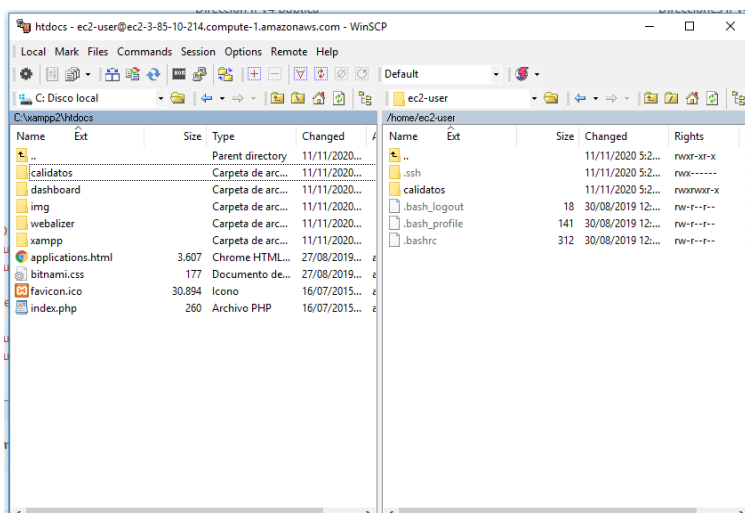
Con el siguiente comando iniciamos el servidor web:



```
[ec2-user@ip-172-31-27-175 ~]$ sudo service httpd start  
Redirecting to /bin/systemctl start httpd.service  
[ec2-user@ip-172-31-27-175 ~]$
```

La carpeta donde se deben guardar las páginas web es [/var/www/html](#) allí debemos guardar la carpeta con la aplicación. Los archivos se pueden subir con [winscp](#)

Inicialmente se suben a la carpeta [/home/ec2-user](#) que es la carpeta sobre la que se tienen todos los permisos



En el servidor copiamos los archivos a la carpeta correspondiente:

```
[ec2-user@ip-172-31-27-55 ~]$ sudo cp -R calidatos /var/www/html
[ec2-user@ip-172-31-27-55 ~]$
```

Desde la consola de aws se debe habilitar el acceso al puerto 80 del servidor web.

Reglas de entrada

Tipo	Protocolo	Intervalo de puertos	Origen	Descripción: opcional
SSH	TCP	22	Personalizada	
TCP personalizado	TCP	80	Personalizada	

Agregar regla

NOTA: al modificar las reglas existentes, se eliminará la regla editada y se creará otra nueva con los nuevos detalles. Esto provocará que se interrumpa el tráfico que depende de dicha regla durante un breve período de tiempo, hasta que se pueda crear la regla nueva.

Cancelar Previsualizar los cambios Guardar reglas

Para evitar problemas de bloqueos por seguridad se debe deshabilitar el **selinux** en la máquina virtual, para ello editamos el archivo **config** que se encuentra en **/etc/selinux/** y cambiamos el siguiente parámetro a **disabled**:

```
[ec2-user@ip-172-31-27-55 ~]$ sudo vi /etc/selinux/config
```

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#   enforcing - SELinux security policy is enforced.
#   permissive - SELinux prints warnings instead of enforcing.
#   disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these three values:
#   targeted - Targeted processes are protected,
#   minimum - Modification of targeted policy. Only selected processes are protected.
#   mls - Multi Level Security protection.
SELINUXTYPE=targeted

-- INSERT --
```

Para poder editar el archivo lo primero que debe hacer es presionar la tecla **i**, para salvar presiona la **esc** luego la tecla **dos puntos (:)** y la tecla **x**, le da **enter** y listo.

Salvamos el archivo, y reiniciamos la máquina con el comando **sudo reboot**, en ese momento la instancia se desconecta y debemos volvernos a conectar.

Debemos volver a levantar apache con el comando **sudo service httpd start**



### 3. Prueba del servicio

Nos ubicamos en el navegador y colocamos la ruta de nuestro servicio, el cual será el dns de la instancia, más la ruta del proyecto que en mi caso es **calidatos**



## CALI DATOS

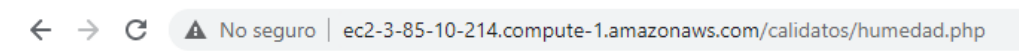
### INGRESO

LOGIN:

PASSWORD:

ENVIAR

Ingresamos con las credenciales y seleccionamos cualquiera de las variables, y se desplegarán los datos:



## CALI DATOS

### Datos de Humedad

ID NODO	HUMEDAD	FECHA
1	50	10/11/2020 5:13 p.m.
2	50	10/11/2020 5:13 p.m.

[Volver](#)

[Cerrar sesión](#)