

Micro Servicio

Klisman Mendoza y Daibiris Rodríguez

az3301990@gmail.com, daibiris@hotmail.com

Sistemas Distribuidos - Universidad Nacional Experimental de Guayana

Resumen

La arquitectura de Microservicios es un enfoque de desarrollo de una aplicación como un conjunto de servicios más pequeños, cada uno se ejecuta en su propio proceso y se exponen, normalmente, a través de protocolos HTTP mediante XMLHttpRequest. Estos servicios están contruidos alrededor de funcionalidades de independencia de despliegue a través de un sistema de automatización. Además pueden estar escritos en diferentes lenguajes de programación y utilizar diferentes tecnologías de almacenamiento de base de datos.

Introducción

El avance tecnológico han obligado a la sociedad a no solo desarrollar sino a incrementar la calidad de diferentes formas de producción, la potencia y las posibilidades de los instrumentos de micro servicio las cuales han cambiado profundamente tanto en concepción como en su uso, como sucede en la mayoría de los campos técnicos la tecnología de información puede referir a los medios colectivos para reunir almacenar y luego procesar dicha información.

De esta manera, surge la necesidad de los microservicios para combatir los aspectos de deficiencia y obtener una mejor tolerancia a fallas, alto rendimiento, transparencia, disponibilidad de la información, y en conjunto los sistemas web mejorar la manera en como prestar microservicios o recursos a través de internet y las redes.

Teniendo en cuenta estos aspectos se decidió crear un microservicio de central de citas que se desarrollara de manera de microservicios, con la finalidad de que este fuese mediante una base de datos para que preste el mencionado microservicios. Esta visión es conceptualizada debido a que los microservicios es un enfoque para desarrollar una aplicación software como una serie de pequeños servicios, cada uno ejecutándose de forma autónoma y comunicándose entre sí, a través de peticiones HTTP,xml y request.

Método Experimental

La metodología aplicada para la implementación y ejecución del proyecto es la metodología en cascada.

La metodología en cascada para Sáez (2009), el ciclo de vida clásico es: “Uno de los modelos más utilizado. En él este modelo, cada etapa deja el camino preparado para la siguiente, de forma que la última no debe comenzar hasta que no ha acabado la anterior. De esta forma, se reduce mucho la complejidad de la gestión, y no se puede dar por terminada una etapa hasta que haya cumplido totalmente con sus objetivos.” (p.297).

Fase 1. Definición:

El proceso de reunir los requisitos para el análisis del sistema fueron los requerimientos obtenidos mediante las pautas establecidas por el profesor de la asignatura Sistemas Distribuidos.

Fase 2. Diseño:

El diseño es realmente un proceso que se centra en tres atributos distintos de programa: estructura de datos, arquitectura de software y representaciones de interfaz.

El proceso del diseño traduce requisitos en una representación del software donde se pueda evaluar su calidad antes de que comience la codificación.

Fase 3. Codificación

Se implementó el código necesario para probar los servicios, correspondientes al proyecto.

Fase 4. Pruebas

Una vez se ha generado el código, comienzan las pruebas de ejecución. El proceso de pruebas se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales; es decir, realizar las pruebas para la detección de errores

y asegurar que la entrada definida produce resultados reales de acuerdo con los resultados requeridos.

Instalación del entorno de trabajo

Los pasos que aquí se detallan servirán para llegar a configurar un entorno válido y así, posteriormente, obtener una instalación satisfactoria de AppServ y PostgreSQL en el equipo. Estos pasos están pensados para todos aquellos usuarios que utilicen la plataforma para desarrollar módulos.

Obtención de un entorno de trabajo

Será necesario que en nuestra máquina tengamos instalado un servidor web. Además, trabaja con bases de datos.

Por todo esto, se aconseja utilizar cualquiera de los paquetes existentes en la red que contienen estos elementos básicos y además se encuentran de forma gratuita.

Instalación de AppServ en una máquina con SO Windows XP

1. El paquete se puede descargar de la siguiente página: <http://www.descargar21.com/appserv/> y una vez aquí ir a la opción [Descargar Gratis \(20.8 MB\)](#).
2. Abrimos el instalador appserv-win32-2.6.0.exe que hemos descargado en el paso 1, veremos el asistente de instalación, donde tendremos que seguir los pasos.
3. Esperamos a que termine la instalación, y por último pulsamos en **Finish**.
4. Para poder hacer funcionar nuestro programa, necesitamos configurar dos archivos el **httpd.conf** de apache y el **php.ini** de php.

Para configurar el httpd.conf de apache hacemos lo siguiente:

- En nuestro caso el archivo se encuentra en "C:\AppServ\Apache2.2\conf\httpd.conf".
- Una vez abierto el archivo, se le indica que cargue el archivo libpq.dll de postgresql, y está ubicado en "C:/Archivos de programa/PostgreSQL/9.3/bin/libpq.dll".

```

#LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
#LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
#LoadModule proxy_connect_module modules/mod_proxy_connect.so
#LoadModule proxy_http_module modules/mod_proxy_http.so
#LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
LoadModule negotiation_module modules/mod_negotiation.so
LoadModule rewrite_module modules/mod_rewrite.so
LoadModule setenvif_module modules/mod_setenvif.so
#LoadModule speling_module modules/mod_speling.so
#LoadModule status_module modules/mod_status.so
#LoadModule unique_id_module modules/mod_unique_id.so
LoadModule userdir_module modules/mod_userdir.so
#LoadModule usertrack_module modules/mod_usertrack.so
#LoadModule vhost_alias_module modules/mod_vhost_alias.so
LoadModule ssl_module modules/mod_ssl.so
LoadModule php6_module C:/AppServ/php6/php6apache2.2.d11

#Configuración para conectar php con postgresql
LoadFile "C:/Archivos de programa/PostgreSQL/9.3/bin/libpq.dll"

#
# 'Main' server configuration
#
# The directives in this section set up the values used by the 'main'
# server, which responds to any requests that aren't handled by a
# <virtualHost> definition. These values also provide defaults for
# any <virtualHost> containers you may define later in the file.
#
# All of these directives may appear inside <virtualHost> containers,
# in which case these default settings will be overridden for the
# virtual host being defined.

```

Figura 1: Archivo httpd.conf.

Una vez modificado el archivo httpd.conf, paramos a modificar el archivo php.ini. Una vez abierto el archivo, activamos la extension=php_pgsql.dll quitandole el ";" :

```

;extension=php_mysqli.dll
;extension=php_mysql.dll
;extension=php_mysqli.dll
;extension=php_oci8.dll
;extension=php_openssl.dll
;extension=php_pdo.dll
;extension=php_pdo_firebird.dll
;extension=php_pdo_mysql.dll
;extension=php_pdo_oci.dll
;extension=php_pdo_oci8.dll
;extension=php_pdo_odbc.dll
;extension=php_pdo_pgsql.dll
;extension=php_pdo_sqlite.dll
;extension=php_pgsql.dll
;extension=php_pspell.dll
;extension=php_shmop.dll
;extension=php_soap.dll
;extension=php_sockets.dll
;extension=php_sqlite.dll
;extension=php_sybase_ct.dll
;extension=php_tidy.dll
;extension=php_xmllrpc.dll
;extension=php_xsl.dll
;extension=php_zip.dll

; Module Settings :

```

Figura 2: Archivo php.ini.

Guardamos y reiniciamos el Appserv

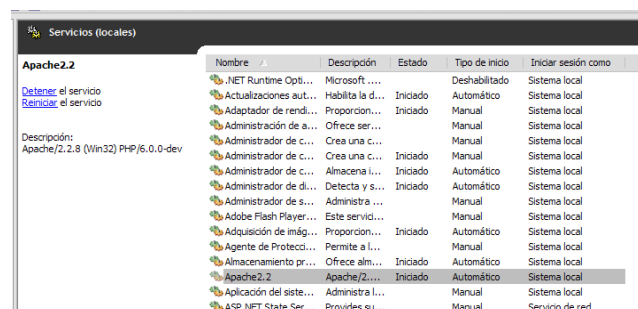


Figura 3: Servicios.

Instalación de PostgreSQL en una máquina con SO Windows XP

Descarga el instalador del programa de la página en <https://www.filehorse.com/es/descargar-postgresql-64/14414/> Se descarga el

instalador postgresql-9.3.0-1-windows-x64.exe, veremos el asistente de instalación, donde tendremos que seguir los pasos.

Abrir pgAdmin III

Para administrar la Base de Datos de PostgreSQL Inicio>Programas > PostgreSQL> Ejecuta pgAdmin III

Una vez abierto el pgAdmin III, tienes que hacer clic dos veces en PostgreSQL 9.3 (localhost:5432) ya que el servidor esta desconectado. Te va a pedir la contraseña de postgres, que es la misma de antes (postgres).

Al teclear la contraseña se abre inmediatamente, en la parte de abajo, un árbol de información entre los que vemos Databases

Se crea una BD llamada Central_Citas con la siguiente tabla cita_medica y posee las siguientes columnas:

Tabla 1: BD Central_Cita contiene la tabla cita_medica

Columna	Tipo	Descripción
id_cita	serial	Identificar de la cita medica
cedula_paciente	character varying	Cedula del paciente
nombre_paciente	character varying	Nombre del paciente
especialidad	character varying	Especialidad del doctor
nombre_doctor	character varying	Nombre del doctor
fecha_cita	date	Fecha de la cita medica

Se crea una BD llamada rayos_x con la siguiente tabla rayos_x y posee las siguientes columnas:

Tabla 2: BD rayos_x contiene la tabla rayos_x

Columna	Tipo	Descripción
id_rayos	serial	Identificador de rayos X
id_cita	integer	Identificador de la cita medica
doctor	character varying	Nombre del doctor que autorizo la cita medica
nombre	character varying	Nombre de la radiografía
descripcion	character varying	Descripción de la radiografía
fecha_rayos_x	date	Fecha que se le asigno la radiografia

Proyecto

Para ejecutar el sistema se crea una carpeta llamada Proyecto en la dirección C:\AppServ\www se abre la carpeta y se almacena todos los archivos necesarios para la ejecución de nuestra aplicación

Archivo: inicio.html

Contiene un elemento `<frameset>` y especifica que cuánta con 2 columnas, y cuánto porcentaje de espacio ocuparán cada uno de ellos.

- La primera columna muestra el archivo listado.html
- La segunda columna proporciona la interacción con los demás archivos html e inicia mostrando el archivo ventana_contenido.html

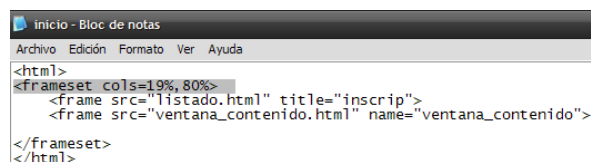


Figura 4: Archivo: inicio.html

Archivo: listado.html

Muestra un menú sencillo, vertical y desplegable empleando solo maquetación HTML y CSS. El menú permite dirigirse a los siguientes archivos crear.html, modificar.html y eliminar.html para la gestión de citas y la opción de Rayos X se dirige al archivo crear_rayos_x.html

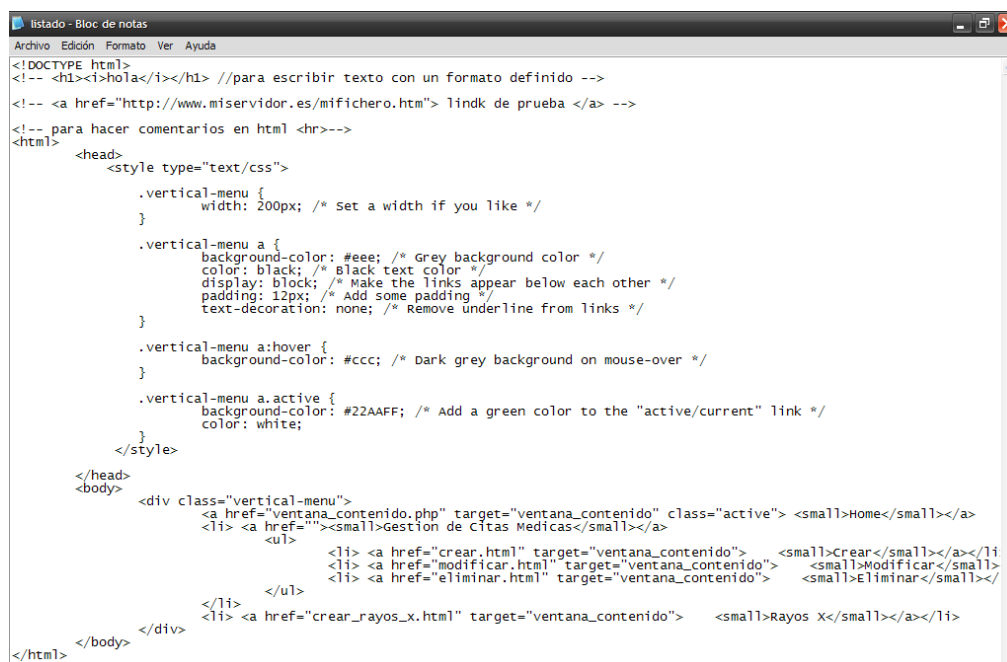
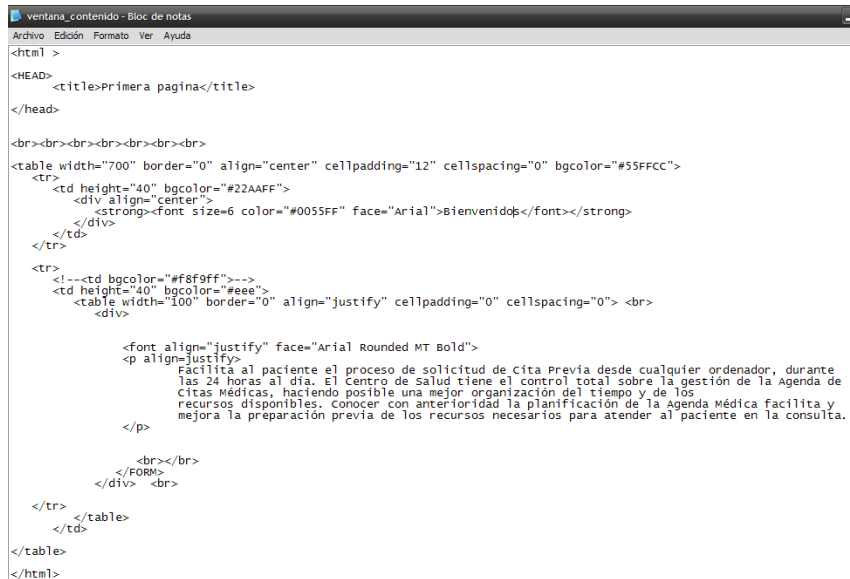


Figura 5: Archivo: listado.html

Archivo: ventana_contenido.html

Muestra una tabla con la bienvenida al sistema.



```
ventana_contenido - Bloc de notas
Archivo  Edición  Formato  Ver  Ayuda

<html>
<head>
<title>Primera pagina</title>
</head>

<br><br><br><br><br><br>
<table width="700" border="0" align="center" cellpadding="12" cellspacing="0" bgcolor="#55FFCC">
<tr>
<td height="40" bgcolor="#22AAFF">
<div align="center">
<strong><font size=6 color="#0055FF" face="Arial">Bienvenido</font></strong>
</div>
</td>
</tr>
<tr>
<!--td bgcolor="#f8f9ff"-->
<td height="40" bgcolor="#eee">
<table width="100" border="0" align="justify" cellpadding="0" cellspacing="0"> <br>
<div>

<font align="justify" face="Arial Rounded MT Bold">
<p align=justify>
Facilita al paciente el proceso de solicitud de cita Previa desde cualquier ordenador, durante
las 24 horas al día. El Centro de salud tiene el control total sobre la gestión de la Agenda de
Citas Médicas, haciendo posible una mejor organización del tiempo y de los
recursos disponibles. Conocer con anterioridad la planificación de la Agenda Médica facilita y
mejora la preparación previa de los recursos necesarios para atender al paciente en la consulta.
</p>
</div>
<br></br>
</FORM>
</div> <br>
</tr>
</table>
</td>
</table>
</html>
```

Figura 6: Archivo: ventana_contenido.html

Antes de continuar con los siguientes archivos crear.html, modificar.html, eliminar.html y crear_rayos_x.html se debe saber la funcionalidad de la function loadDoc()

Funcion: function loadDoc()

Todos los navegadores modernos tienen incorporado un objeto XMLHttpRequest para solicitar datos de un servidor.

- Se crea la variable: var xhttp;
- Para asignar la creación de un objeto XMLHttpRequest:

xhttp = **new** XMLHttpRequest();

Las versiones anteriores de Internet Explorer (IE5 e IE6) usan un objeto ActiveX en lugar del objeto XMLHttpRequest:

xhttp = new ActiveXObject ("Microsoft.XMLHTTP");

Para manejar IE5 e IE6, verifique si el navegador admite el objeto XMLHttpRequest, o bien cree un objeto ActiveX:

Luego la propiedad readyState tiene el estado de XMLHttpRequest. El evento onreadystatechange se desencadena cada vez que readyState cambia y durante una solicitud del servidor, readyState cambia de 0 a 4:

- 0: solicitud no inicializada
- 1: conexión del servidor establecida
- 2: solicitud recibida
- 3: solicitud de procesamiento
- 4: solicitud finalizada y la respuesta está lista

En la propiedad onreadystatechange, especifique una función que se ejecutará cuando el readyState cambie: xmlhttp.onreadystatechange = function()

Cuando readyState es 4 y el estado es 200, la respuesta está lista:
if (this.readyState == 4 && this.status == 200)

La propiedad responseText devuelve la respuesta como una cadena.
document.getElementById("demo").innerHTML = this.responseText;

El parámetro url del método open () es una dirección a un archivo en un servidor: xmlhttp.open("POST", url, true); Para enviar la solicitud de forma asincrónica, el parámetro async del método open () debe establecerse en verdadero.

```
var xmlhttp;
if (window.XMLHttpRequest) {
    // code for modern browsers
    xmlhttp = new XMLHttpRequest();
} else {
    // code for IE6, IE5
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        document.getElementById("demo").innerHTML = this.responseText;
    }
};
xmlhttp.open("POST", url, true);
xmlhttp.send();
```

Figura 7: function loadDoc()

Archivo: crear.html

Se crea un formulario `<form></form>` y los campos `type="text"` y un campo `type="button"` con evento `onclick="loadDoc()"`

```
<body>
  <p id="demo"></p>
  <table width="0" border="0" align="center" cellpadding="0" cellspacing="0" >
    <tr>
      <td height="40" bgcolor="#22AAFF"><div align="center">
        <strong><font color="#333333" face="Verdana, Arial, Helvetica, sans-serif">Crear cita Medica</font></
      </tr>
      <tr>
        <td><table bgcolor="#eee" width="0%" border="0" align="center" cellpadding="0" cellspacing="30">
          <div align="center">
            <td>Cedula Paciente:</td>
            <td><input type="text" id="cedula" name="cedula" /></td>
          </tr>
          <tr>
            <td>Nombre y Apellido del Paciente:</td>
            <td><input type="text" id="paciente" name="paciente" /></td>
          </tr>
          <tr>
            <td>Especialidad del Doctor:</td>
            <td><input type="text" id="especialidad" name="especialidad" /></td>
          </tr>
          <tr>
            <td>Nombre y Apellido del Doctor:</td>
            <td><input type="text" id="doctor" name="doctor" /></td>
          </tr>
          <tr>
            <td>Fecha de la Cita Médica:</td>
            <td><input type="text" id="fecha" name="fecha" /></td>
          </tr>
        </table>
        <table bgcolor="#eee" width="100%" border="0" align="center" cellpadding="0" cellspacing="30">
          <div align="center">
            <tr>
              <td><button type="button" onclick="loadDoc()">Crear</button></td>
            </tr>
          </div>
        </table><br />
      </td>
    </tr>
  </table>
</body>
</html>
```

Figura 8: Archivo: crear.html

Cuando presionan el botón se efectúa la función `loadDoc()`, obteniendo los datos ingresados en el formulario y creando el url `post_cita.php` con las variables a enviar.

```
function loadDoc() {
  var cedula = document.getElementById('cedula').value;
  var paciente = document.getElementById('paciente').value;
  var especialidad = document.getElementById('especialidad').value;
  var doctor = document.getElementById('doctor').value;
  var fecha = document.getElementById('fecha').value;

  ruta="post_cita.php";
  envio1="cedula="+cedula;
  envio2="paciente="+paciente;
  envio3="especialidad="+especialidad;
  envio4="doctor="+doctor;
  envio5="fecha="+fecha;
  url=ruta+"?"+"envio1+"&"+envio2+"&"+envio3+"&"+envio4+"&"+envio5;

  var xmlhttp;
  if (window.XMLHttpRequest) {
    // code for modern browsers
    xmlhttp = new XMLHttpRequest();
  } else {
    // code for IE6, IE5
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
  }
  xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xmlhttp.open("POST", url, true);
  xmlhttp.send();
}
```

Figura 9: funtion loadDoc() de archivo crear.html

Archivo: modificar.html

Se crea un formulario `<form></form>` y los campos `type="text"` y un campo `type="button"` con evento `onclick="loadDoc()"`

```

<p id="demo"></p>
<table width="0" border="0" align="center" cellpadding="0" cellspacing="0" >
  <tr>
    <td height="40" bgcolor="#22AAFF"><div align="center">
      <strong><font color="#333333" face="Verdana, Arial, Helvetica, sans-serif">Modificar Cita Medica</font>
    </div>
  </tr>
  <tr>
    <td><table bgcolor="#eee" width="0%" border="0" align="center" cellpadding="0" cellspacing="30">
      <div align="center">
        <td>Numero de Cita:</td>
        <td><input type="text" id="cita" name="cita" /></td>
      </tr>
      <tr>
        <td>Nombre y Apellido del Doctor:</td>
        <td><input type="text" id="doctor" name="doctor" /></td>
      </tr>
      <tr>
        <td>Fecha de la Cita Médica:</td>
        <td><input type="text" id="fecha" name="fecha" /></td>
      </tr>
    </table>
  </tr>
  <tr>
    <td><table bgcolor="#eee" width="100%" border="0" align="center" cellpadding="0" cellspacing="30">
      <div align="center">
        <td><button type="button" onclick="loadDoc()">Modificar</button></td>
      </div>
    </table>
  </tr>
</table><br />

```

Figura 10: Archivo: modificar.html

Cuando presionan el botón se efectúa la función loadDoc(), obteniendo los datos ingresados en el formulario y creando el url put_cita.php con las variables a enviar.

```

function loadDoc() {
  var cita = document.getElementById('cita').value;
  var doctor = document.getElementById('doctor').value;
  var fecha = document.getElementById('fecha').value;

  ruta="put_cita.php";
  envio1="cita="+cita;
  envio2="doctor="+doctor;
  envio3="fecha="+fecha;
  url=ruta+"?" +envio1+"&" +envio2+"&" +envio3;

  var xhttp;
  if (window.XMLHttpRequest) {
    // code for modern browsers
    xhttp = new XMLHttpRequest();
  } else {
    // code for IE6, IE5
    xhttp = new ActiveXObject("Microsoft.XMLHTTP");
  }
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("PUT", url, true);
  xhttp.send();
}

```

Figura 11: funtion loadDoc() de archivo modificar.html

Archivo: eliminar.html

Se crea un formulario <form></form> y los campos type="text" y un campo type="button" con evento onclick="loadDoc()"

```

<p id="demo"></p>
<table width="0" border="0" align="center" cellpadding="0" cellspacing="0" >
  <tr>
    <td height="40" bgcolor="#22AAFF"><div align="center">
      <strong><font color="#333333" face="Verdana, Arial, Helvetica, sans-serif">Eliminar Cita Medica</font></div>
    </tr>
    <tr>
      <td><table bgcolor="#eee" width="0%" border="0" align="center" cellpadding="0" cellspacing="30">
        <div align="center">
          <td>Numero de cita:</td>
          <td><input type="text" id="cita" name="cita" /></td>
        </tr>
      </table>
    </tr>
    <tr>
      <td><table bgcolor="#eee" width="100%" border="0" align="center" cellpadding="0" cellspacing="30">
        <div align="center">
          <td><button type="button" onclick="loadDoc()">Eliminar</button></td>
        </div>
      </table>
    </tr>
  </table><br />

```

Figura 12: Archivo: eliminar.html

Cuando presionan el botón se efectúa la función loadDoc(), obteniendo los datos ingresados en el formulario y creando el url delete_cita.php con las variables a enviar.

```
function loadDoc() {
    var cita = document.getElementById('cita').value;

    ruta="delete_cita.php";
    envio1="cita="+cita;
    url=ruta+"?"+"envio1;

    var xhttp;
    if (window.XMLHttpRequest) {
        // code for modern browsers
        xhttp = new XMLHttpRequest();
    } else {
        // code for IE6, IE5
        xhttp = new ActiveXObject("Microsoft.XMLHTTP");
    }
    xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            document.getElementById("demo").innerHTML = this.responseText;
        }
    };
    xhttp.open("DELETE", url, true);
    xhttp.send();
}
```

Figura 13: funtion loadDoc() de archivo eliminar.html

Archivo: crear_rayos_x.html

Se crea un formulario <form></form> y los campos type="text" y un campo type="button" con evento onclick="loadDoc()"

```
<p id="demo"></p>
<table width="0" border="0" align="center" cellpadding="0" cellspacing="0" >
    <tr>
        <td height="40" bgcolor="#22AAFF"><div align="center">
            <strong><font color="#333333" face="Verdana, Arial, Helvetica, sans-serif">Crear Cita Medica</font></div>
        </tr>
        <tr>
            <td><table bgcolor="#eee" width="0%" border="0" align="center" cellpadding="0" cellspacing="30">
                <div align="center">
                    <td>Numero de Cita del Paciente:</td>
                    <td><input type="text" id="cita" name="cita" /></td>
                </tr>
                <tr>
                    <td>Nombre y Apellido del Doctor:</td>
                    <td><input type="text" id="doctor" name="doctor" /></td>
                </tr>
                <tr>
                    <td>Rayos X:</td>
                    <td><input type="text" id="rayos" name="rayos" /></td>
                </tr>
                <tr>
                    <td>Descripcion de Rayos X:</td>
                    <td><input type="text" id="descripcion" name="descripcion" /></td>
                </tr>
                <tr>
                    <td>Fecha:</td>
                    <td><input type="text" id="fecha" name="fecha" /></td>
                </tr>
            </table>
            <table bgcolor="#eee" width="100%" border="0" align="center" cellpadding="0" cellspacing="30">
                <div align="center">
                    <tr>
                        <td><button type="button" onclick="loadDoc()">Crear</button></td>
                    </tr>
                </div>
            </table><br />
```

Figura 14: Archivo: crear_rayos_x.html

Cuando presionan el botón se efectúa la función loadDoc(), obteniendo los datos ingresados en el formulario y creando el url post_rayos_x.php con las variables a enviar.

```

function loadDoc() {
    var cita = document.getElementById('cita').value;
    var doctor = document.getElementById('doctor').value;
    var rayos = document.getElementById('rayos').value;
    var descripcion = document.getElementById('descripcion').value;
    var fecha = document.getElementById('fecha').value;

    ruta="post_rayo_x.php";
    envio1="cita="+cita;
    envio2="doctor="+doctor;
    envio3="rayos="+rayos;
    envio4="descripcion="+descripcion;
    envio5="fecha="+fecha;
    url=ruta+"?"+"envio1+"&"+envio2+"&"+envio3+"&"+envio4+"&"+envio5;

    var xhttp;
    if (window.XMLHttpRequest) {
        // code for modern browsers
        xhttp = new XMLHttpRequest();
    } else {
        // code for IE6, IE5
        xhttp = new ActiveXObject("Microsoft.XMLHTTP");
    }
    xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            document.getElementById("demo").innerHTML = this.responseText;
        }
    };
    xhttp.open("POST", url, true);
    xhttp.send();
}

```

Figura 15: funtion loadDoc() de archivo crear_rayos_x.html

Archivo: post_cita.php

Se obtiene los valores de las variables pasadas por el url a través de variable = \$_GET["variableURL"]; Se efectúa la conexión en postgresSQL y se ejecuta la sentencia insertar cita_medica

```

<?php
include 'bdcentral_Citas.php';

$cedula=$_GET["cedula"];
$paciente=$_GET["paciente"];
$especialidad=$_GET["especialidad"];
$doctor=$_GET["doctor"];
$fecha=$_GET["fecha"];

$con = pg_connect("user=".DB_USER." password=".DB_PASS." port=".DB_PORT." dbname=".DB_NAME." host=".DB_HOST);
if (!$con) {
    die('could not connect: ' . pg_error($con));
}
$sql="INSERT INTO cita_medica(cedula_paciente,nombre_paciente,especialidad,nombre_doctor,fecha_cita) VALUES('".$cedula
$result = pg_query($con,$sql);
pg_close($con);
?>

```

Figura 16: Archivo: post_cita.php

Archivo: put_cita.php

Se obtiene los valores de las variables pasadas por el url a través de variable = \$_GET["variableURL"]; Se efectua la conexión en postgresSQL y se ejecuta la sentencia modificar cita_medica

```

<?php
include 'bdcentral_Citas.php';

$cita=$_GET["cita"];
$doctor=$_GET["doctor"];
$fecha=$_GET["fecha"];

$con = pg_connect("user=".DB_USER." password=".DB_PASS." port=".DB_PORT." dbname=".DB_NAME." host=".DB_HOST);
if (!$con) {
    die('could not connect: ' . pg_error($con));
}
$sql="UPDATE cita_medica SET fecha_cita='".$fecha."', nombre_doctor='".$doctor.'" WHERE id_cita = ".$cita."";
$result = pg_query($con,$sql);
pg_close($con);
?>

```

Figura 17: Archivo: put_cita.php

Archivo: delete_cita.php

Se obtiene los valores de las variables pasadas por el url a través de `variable = $_GET["variableURL"]`; Se efectúa la conexión en PostgreSQL y se ejecuta la sentencia eliminar `cita_medica`

```
k?php
include 'bdCentral_Citas.php';
$cita=$_GET["cita"];

$con = pg_connect("user=".DB_USER." password=".DB_PASS." port=".DB_PORT." dbname=".DB_NAME." host=".DB_HOST);
if (!$con) {
    die('Could not connect: ' . pg_error($con));
}
$sql="DELETE FROM cita_medica WHERE id_cita = ".$cita."";
$result = pg_query($con,$sql);

pg_close($con);
?>
```

Figura 18: Archivo: delete_cita.php

Archivo: post_rayos_x.php

Se obtiene los valores de las variables pasadas por el url a través de `variable = $_GET["variableURL"]`; Se efectúa la conexión en PostgreSQL con la BD `rayos_x`, crea una instancia de la clase `Datos` la cual está ubicada en el archivo: `getDatos.php`, se llama a la función y se obtiene el numero de cita, se comprueba la igualdad y se ejecuta la sentencia insertar `rayos_x`

```
k?php
include 'bdRayos_X.php';
include 'getDatos.php';

$cita=$_GET["cita"];
$doctor=$_GET["doctor"];
$rayos=$_GET["rayos"];
$descripcion=$_GET["descripcion"];
$fecha=$_GET["fecha"];

$con = pg_connect("user=".DB_USER." password=".DB_PASS." port=".DB_PORT." dbname=".DB_NAME." host=".DB_HOST);
if (!$con) {
    die('Could not connect: ' . pg_error($con));
}

$dato = new Datos();
$verificar = $dato->get_cita($cita);

if($verificar==$cita){
    $sql="INSERT INTO rayos_x(id_cita,doctor,nombre,descripcion,fecha_rayos_x) VALUES('".$cita."', '".$doctor."', '".$rayos."', '".$descripcion."', '".$fecha."')";
    $result = pg_query($con,$sql);
}

pg_close($con);
?>
```

Figura 19: Archivo: post_rayos_x.php

Archivo: getDatos.php

Se efectúa la conexión en PostgreSQL con la BD `Central_Citas` y se ejecuta la selección para retornar el valor suministrado

```

<?php
include 'bdCentral_Citas.php';
class Datos
{
    function get_cita($cita)
    {
        $conn = pg_connect("user=".DB_USER." password=".DB_PASS." port=".DB_PORT." dbname=Central_Citas host=
        $verificar=null;
        if ($conn) {
            $result = pg_query($conn, "select id_cita from cita_medica WHERE id_cita=$cita");
            if (pg_num_rows($result)>0)
            {
                while ($row = pg_fetch_row($result)) {
                    $verificar=$row[0];
                }
            }
            else
            {
                echo "Conexión Erronea";
                exit;
            }
            return $verificar;
        }
    }
}
?>

```

Figura 20: Archivo: getDatos.php

Archivo: bdCentral_Citas.php

Contiene definida las variables para acceder a la BD Central_Citas

```

<?php
define ('DB_HOST','localhost'); //Host de postgresql (puede ser otro)
define ('DB_USER','postgres'); //Usuario de postgresql (puede ser otro)
define ('DB_PASS','1234'); //Password de postgresql (puede ser otro)
define ('DB_NAME','Central_Citas'); //Database de postgresql (puede ser otra)
define ('DB_PORT','5432'); //Puerto de postgresql (puede ser otro)
?>

```

Figura 21: bdCentral_Citas.php

Archivo: bdRayos_X.php

Contiene definida las variables para acceder a la BD rayos_x

```

<?php
define ('DB_HOST','localhost'); //Host de postgresql (puede ser otro)
define ('DB_USER','postgres'); //Usuario de postgresql (puede ser otro)
define ('DB_PASS','1234'); //Password de postgresql (puede ser otro)
define ('DB_NAME','rayos_x'); //Database de postgresql (puede ser otra)
define ('DB_PORT','5432'); //Puerto de postgresql (puede ser otro)
?>

```

Figura 22: bdRayos_X.php

Análisis y Resultados

La figura 23. se visualiza como enviar y recibir los datos.

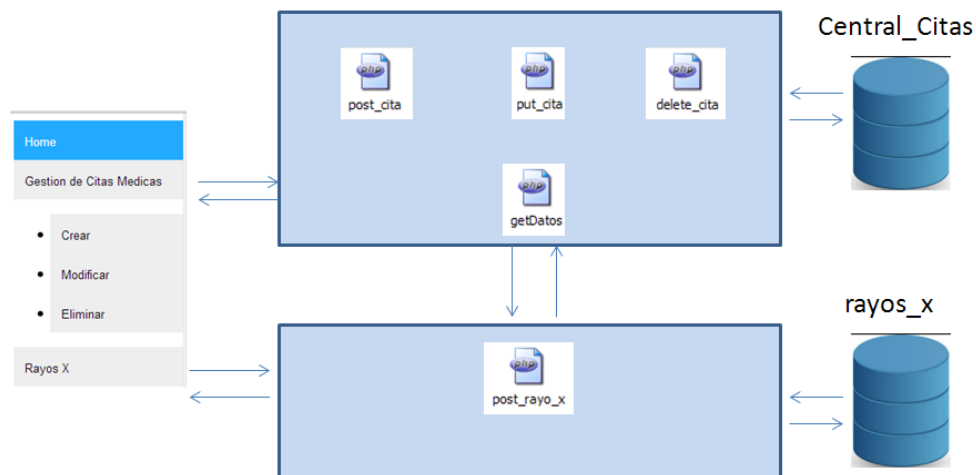


Figura 23: Diagrama de enviar y recibir datos

Para ingresar al servicio debe abrir su navegador favorito y en la barra de direcciones escribir <http://localhost/Proyecto/inicio.html>

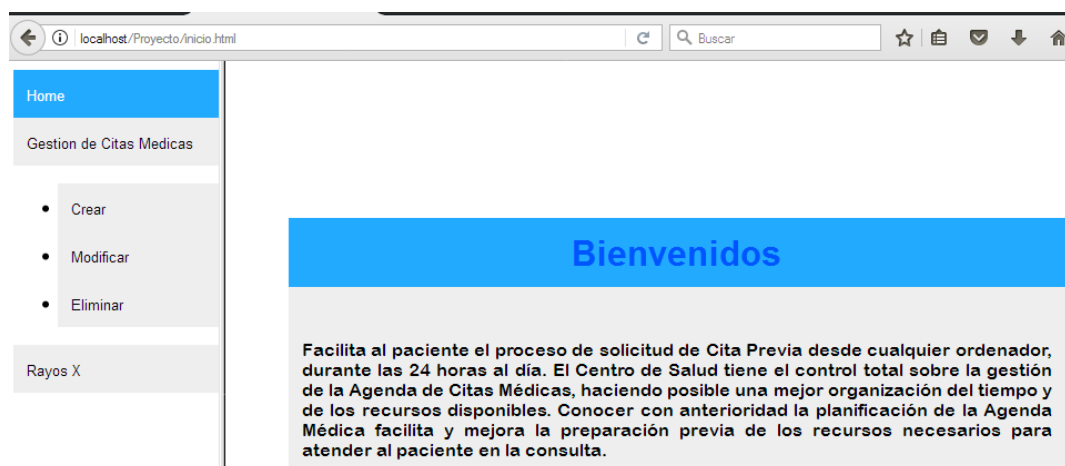


Figura 24: Servicio de gestión de citas y Rayos X

Crear Cita Paciente

Para crear una cita ingresamos al modulo Gestión de Citas Medicas opción Crear

Crear Cita Medica

Cedula Paciente:

Nombre y Apellido del Paciente:

Especialidad del Doctor:

Nombre y Apellido del Doctor:

Fecha de la Cita Médica:

Figura 25: Crear cita medica

Ingresamos los datos que solicita el formulario Crear Cita Medica las cuales son: cedula del paciente, nombre del paciente, especialidad del doctor, nombre del doctor y la fecha la cual se le va asignar la cita médica. Al rellenar los datos presiona el botón Crear.

Modificar Cita Paciente

Se desea modificar la cita ingresamos al modulo Gestión de Citas Medicas opción Modificar

Modificar Cita Medica

Numero de Cita:

Nombre y Apellido del Doctor:

Fecha de la Cita Médica:

Figura 26: Modificar cita medica

Ingresamos los datos que solicita el formulario Modificar Cita Medica las cuales son: Número de cita previamente creada, el nombre del doctor el cual se le asigno la cita y la fecha de la cita. Al rellenar los datos presiona el botón Modificar.

Eliminar Cita Paciente

Se desea eliminar la cita ingresamos al modulo Gestión de Citas Medicas opción Eliminar

Eliminar Cita Medica

Numero de cita:

Figura 27: Eliminar cita medica

Ingresamos los datos que solicita el formulario Eliminar Cita Medica las cuales son: Número de cita previamente creada. Al rellenar los datos presiona el botón Eliminar.

Solicitar Rayos X

Se desea solicitar Rayos X ingresamos al modulo de Rayos X

Solicitar Rayos X

Numero de Cita del Paciente:

Nombre y Apellido del Doctor:

Rayos X:

Descripcion de Rayos X:

Fecha:

Figura 28: Solicitar Rayos X

Ingresamos los datos que solicita el formulario Solicitar Rayos X, las cuales son: Número de cita del paciente permite establecer que el rayos x se solicito específicamente en dicha cita. El nombre del doctor que autorizo la solicitud de los rayos x, el nombre de la radiografía y la descripción de la misma. Al rellenar los datos presiona el botón Crear.