



Universidad Nacional Experimental de Guayana  
Área de Informática

# **Servicio Web del sistema de fotocopiado de la UNEG en servidor Prolog.**

Cuotto, J y Stalin, S

21 de julio de 2016

Sistemas Distribuidos. 2016-I



## **1. Introducción**

El crecimiento que vive la web hoy en día, trae consigo el incremento de servicios que se pueden disponer al público para realizar operaciones que se adaptan mejor a sus necesidades. Los usuarios que consumen y generan información lo hacen a través de buscadores web que permiten referenciar recursos o simplemente visualizar páginas web que a su vez les ofrezcan recursos de interés. Entendiendo a dichos buscadores web como clientes que necesitan comunicarse con servidores para obtener recursos, dando a lugar a una interacción cliente servidor.

Los servicios web, proveen una infraestructura para mantener una rica y mas estructurada forma de interoperabilidad entre clientes y servidores, proveyendo una base por la cual un cliente que puede estar en una localización, pueda interactuar con un servidor en otra localización sin necesidad de supervisión humana, siendo a su vez una colección de operaciones que pueden ser usados y están al alcance de cualquier cliente sobre todo el Internet [Coulouris and Dollimore and Kindberg and Blair (2012)].

Entre las publicaciones relacionadas con el objeto de estudio presentado, nos encontramos con [Navarro (2007)], que indica que el uso de SOAP es adecuado cuando se establece un contrato para la descripción de la interfaz que el servicio ofrece así como también que la arquitectura de los servicios web deben abordar requerimientos no funcionales y especificaciones como las transacciones y operaciones de recuperación e inserción de datos de en una base de datos, haciendo uso de una metodología descriptiva y comparativa.

También se encuentra a [Bermúdez e Ibáñez y González (2004)], los cuales utilizan un modelo de programación evolutiva y concluyen en su trabajo de estudio sobre la seguridad en servicios basados en XML, donde en una arquitectura cliente/servidor, el cliente debe ser capaz de realizar peticiones encriptados para luego el servidor, descryptarlas y procesar las operaciones.

## **1.1.Objetivos**

### ➤ Objetivo General

Desarrollar servicios web en Prolog para el sistema de fotocopiado de la Universidad Nacional Experimental de Guayana con la utilización de SOAP y su descripción de servicio WSDL.

### ➤ Objetivos específicos

1. Análisis de los requerimientos funcionales y de información de los servicios a ofrecer.
2. Diseñar diagramas, modelos y arquitectura de los servicios a proponer.
3. Desarrollar servicios web sobre operaciones del sistema de fotocopiado de la Universidad Nacional Experimental de Guayana.
4. Realizar pruebas de los resultados obtenidos del desarrollo de los servicios web.
5. Implementar los servicios web desarrollados en el laboratorio de Sistemas Distribuidos de la Universidad Nacional Experimental de Guayana.

## **1.2. Metodología de Desarrollo**

Para el desarrollo del presente proyecto, se hizo uso de la metodología en Cascada para desarrollo de software, la cual posee 5 fases de desarrollo durante el proceso que conlleve a la realización del mismo, las cuales se desarrollan de forma lineal o secuencial, y es imprescindible que se tengan los requerimientos bien claros y definidos [Nava (2006)].

Las fases de este modelo se presentan a continuación.

1. Análisis de requerimientos: Se debe realizar un análisis tanto de los requerimientos funcionales como de informaciones necesarias y requeridas para las operaciones y funciones que ha de cumplir el sistema a desarrollar.
2. Diseño: Realizar los diseños de los diagramas pertinentes que para el desarrollo del sistema así como también el diseño de la arquitectura en la que se basaran los servicios web, siendo este el paso anterior a la parte de codificación, por lo tanto

resulta necesario tener a disposición todos los diagramas y modelos en que estará basado el sistema.

3. Codificación: Fase donde se inicia el desarrollo del sistema propuesta en base a los requerimientos ya establecidos en conjunto con el diseño del mismo que permita tener como base plasmada el funcionamiento que ha de tener el sistema.
4. Pruebas: Donde se verifica que el comportamiento del sistema desarrollado cumple con los requerimientos establecidos con anterioridad en las fases iniciales, cumpliendo con el modelado para el sistema, de modo que sirve como comparación de calidad para establecer el funcionamiento de las operaciones que ha de cumplir el sistema.
5. Implementación: Una vez desarrollado el sistema y luego de haberle realizado las pruebas pertinentes, este ha de ser implementado en el lugar donde sea requerido, para ser usado por los usuarios finales destinado a darle provecho a las funcionalidades establecidas y alcanzadas.

El siguiente informe sigue una estructura en la cual en la sección 2 se tratara sobre los materiales y métodos utilizados para el desarrollo propuesto, en la cual se detalla la arquitectura del sistema denotando los distintos servicios que se ofrecen, así como se tratan temas sobre el modelo de interacción del sistema así como del modelo de fallas que este presenta para el tratamiento de irregularidades que se puedan presentar, en conjunto con la instalación y pruebas realizadas del sistema. Se continúa en la sección 3, referente a los resultados obtenidos del desarrollo de los servicios del sistema fotocopiado para luego, dar lugar a la sección 4, donde se indican las conclusiones de los objetivos logrados.

## **2. Materiales y métodos**

### **2. a. Análisis de requerimientos:**

Para poder aplicar y consumir un servicio web, es requerido hacer uso del protocolo SOAP, que es un protocolo ligero basado en XML y está diseñado para el intercambio de información en un ambiente de computación distribuido y no existe un concepto de servidor central, ya que todos los nodos pueden ser considerados iguales [Englander (2002)].

Al estar basado en XML, se define el mismo como lenguaje jerárquico basado en etiquetas, que permite describir contenido que es específico a nuestras propias aplicaciones en una manera estándar, y el protocolo SOAP, a través del XML, es transportado a con HTTP. [Englander (2002)].

El servicio se consume en conjunto con un servidor el cual recibirá las peticiones por parte de los clientes. Este servidor ha de ser en Prolog, el cual se conectara con otros servidores que contendrán la base de datos fragmentada en modo de distribuida, la cual a su vez es una colección de múltiples bases de datos lógicamente interrelacionados sobre una red de computadoras [Hernández (2013)].

Para evitar los problemas de interoperabilidad de servicios basados en SOAP, siendo interoperabilidad como la habilidad de dos sistemas para intercambiar información y utilizarla, contamos con WSDL, de Web Services Definition Lenguaje, o en español, Lenguaje de definición de servicios web, el cual proporciona un lenguaje estructurado para describir un servicio, su locación, los métodos del servicios, parámetros, tipos de datos y más. [Englander (2002)].

Finalmente, el servicio a suministrar, debe ser un conjunto de servicios o métodos que permitan realizar operaciones sobre el sistema de fotocopiado de la UNEG, de manera que permita conocer el número de copias que ha sacado, buscar estudiantes, registrar estudiantes y mostrar factura luego de efectuar la operación de sacar copias, utilizando los mecanismos antes mencionados. Por lo tanto, los requerimientos del sistema como tal son:

➤ **Requerimientos funcionales.**

- Servicio Web con servidor Prolog que permita realizar distintas operaciones.
- Utilización de SOAP para el intercambio de información y peticiones, basado en XML.
- Utilización de WSDL, para describir la estructura de un servicio, sus métodos y sus tipos de datos como de parámetros.
- Base De datos distribuidos.

➤ **Requerimientos de Información.**

- Manejar información sobre datos de los estudiantes.
- Manejar cantidad de copias que un estudiante que le quedan.

**2. b. Diseño:**

Para el desarrollo del servicio, hacemos uso de una Base de Datos relacional, con tres entidades que son:

• **Estudiante:**

Entidad principal, que brindara la información sobre los estudiantes para poder consumir las fotocopias.

Atributos:

- Cedula
- Nombre
- Apellido
- Copias
- Clave
- Sede

• **Fotocopia:**

Entidad que permite saber el número de copias y la fecha en que se sacaron las copias y sirve para realizar la factura de la transacción.

Atributos:

- Id\_fotocopias

- Copias
- Fecha

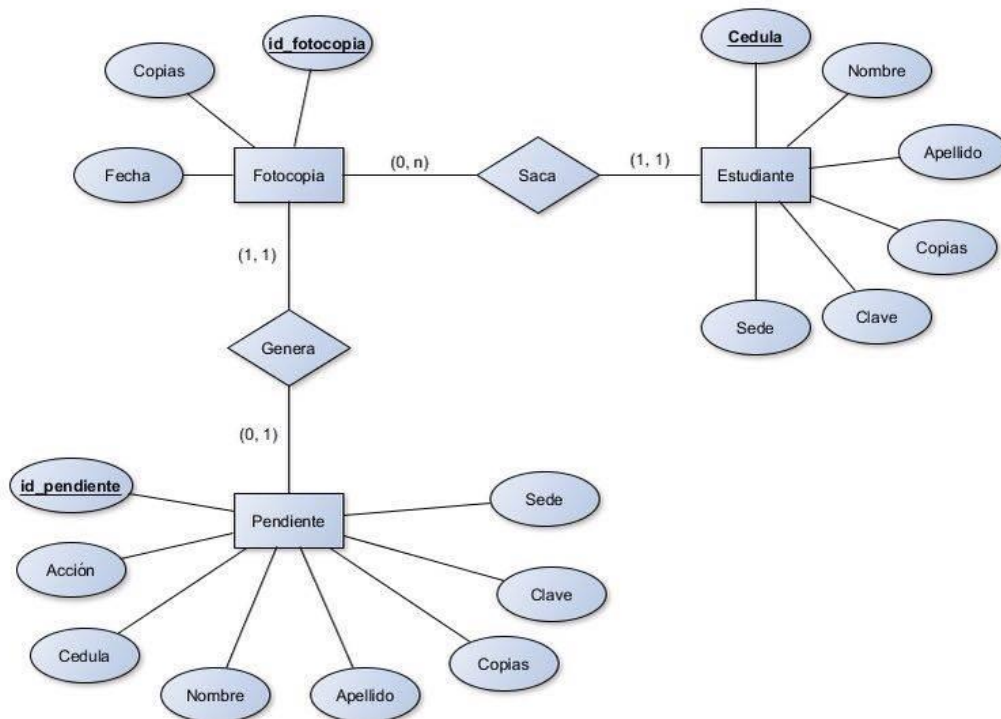
• **Pendiente:**

Entidad destinada a cubrir una de las fallas cuando se caiga el servidor secundario de base de datos.

Atributos:

- Id\_pendiente
- Acción
- Cedula
- Nombre
- Apellido
- Copias
- Clave
- Sede

Teniendo en cuenta estas entidades y sus atributos, se obtiene el siguiente Diagrama de Entidad/Relación.





De la cual se tienen las siguientes relaciones:

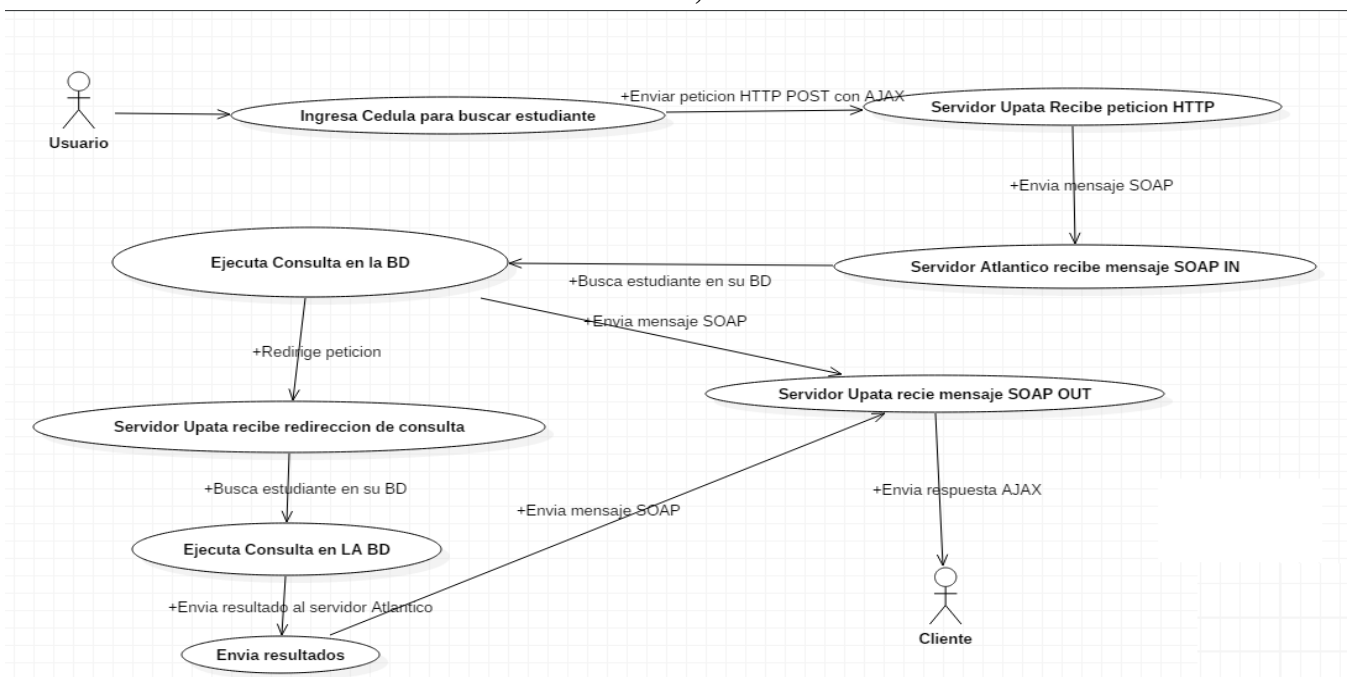
- Un estudiante, genera de [0,N] Copias
- 1 Fotocopia es sacada Solo por [1] Estudiante
- 1 Fotocopia genera de [0,1] Pendiente
- 1 Pendiente es generada solo por [1] Estudiante.

El diseño de la codificación, está basada en el Modelo Vista Controlador (MVC), el cual separa la codificación por su interés. Por un lado, se tiene toda la programación del modelo de negocios, por otro lado están las vistas que serán vistos por los usuarios y por el otro, los controladores de cada vista. En total, el diseño de la programación se dividirá en:

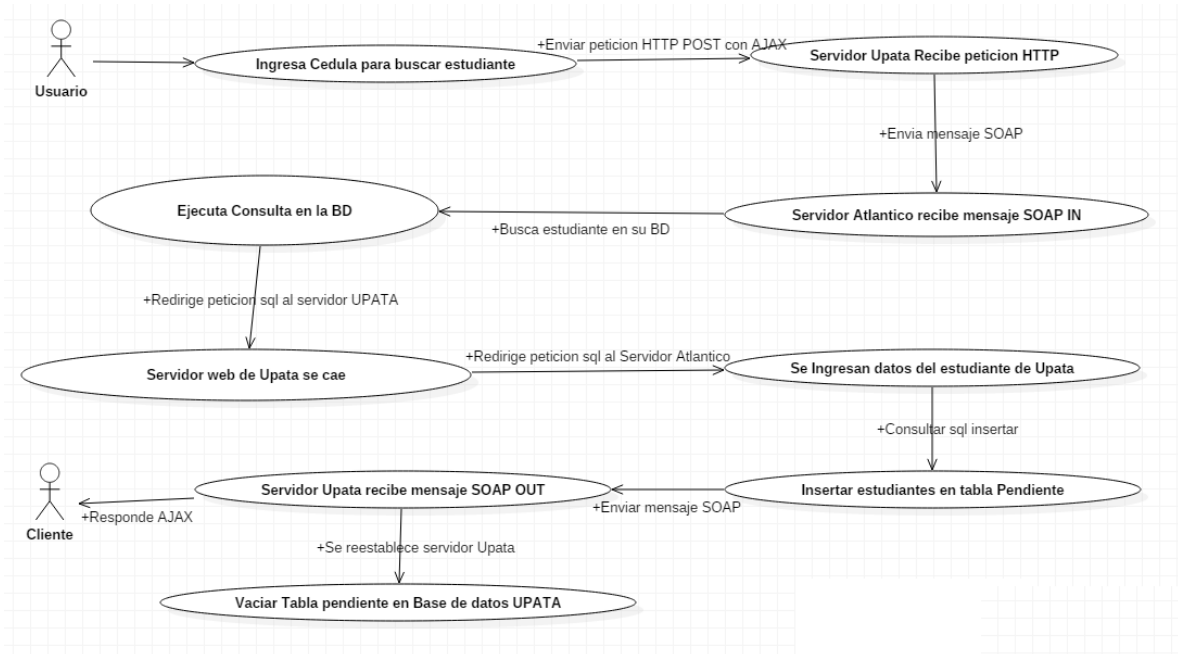
- Modelos: Todos los programas y códigos destinados al manejo del modelo de negocio del sistema.
- Vistas: Todo lo visual para el usuario, en donde se mostraran los resultados proporcionados por el servicio.
- Controladores: Todos los programas encargados de controlar las vistas, de manera que sirven para controlar los resultados obtenidos por los modelos y proporcionárselos a las vistas.

- **Casos de Uso**

- **Se busca estudiante, estando ambos servidores activos.**



## - Buscar estudiante cuando el servidor de base de datos de Upata se ha caído



## 2. c. Codificación

En este apartado, obtenemos la codificación necesaria para cumplir con los requerimientos. Para una mejor lectura de los códigos, han sido sombreados en gris.

### Vista Inicial del Cliente.

```

<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Servicio Web - Prolog</title>
  <link rel="stylesheet"
href="vistas/css/index.css">
</head>
<body>
  <div id="lista_servicios">
    <div>
      <h1>Servicio Web - Servidor Prolog</h1>
      <h4>Lista de los servicios que se
ofrecen</h4>
    </div>
    <div>
      <a
href="vistas/buscar_estudiante.html"><button
class="boton" name="btn_buscar_estudiante" >BUSCAR
ESTUDIANTE</button> </a>
    
```

```

        </div>
        <div>
            <a
href="vistas/registrar_estudiante.html"><button
class="boton" name="btn_registrar_estudiante" >REGISTRAR
ESTUDIANTE</button> </a>
        </div>
        <div>
            <a
href="vistas/actualizar_estudiante.html"><button
class="boton" name="btn_actualizar_estudiante"
>ACTUALIZAR ESTUDIANTE</button> </a>
        </div>
        <div>
            <a href="vistas/renovar_copias.html"><button
class="boton" name="btn_renovar_copias" >RENOVAR
COPIAS</button> </a>
        </div>
        <div>
            <a
href="vistas/consumir_copias.html"><button class="boton"
name="btn_consumir_copias" >CONSUMIR COPIAS</button>
</a>
        </div>
    </div>
</div>
</body>
</html>

```

### Vista de Buscar ESTUDIANTES:

```

<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Buscar Estudiante</title>
    <link rel="stylesheet" href="css/buscar_estudiante.css">
</head>
<body>
    <div id="buscar_estudiante-formulario">
        <div>
            <h1>Buscar Estudiante</h1>
            <h4>Ingrese la cÃ©dula del estudiante que
desea buscar.</h4>
        </div>

        <form method="post" action="#">
            <div>
                <label for="txt_cedula_buscar">
                    <span class="required">CÃ©dula: *</span>

```

```

        <input type="text"
id="txt_cedula_buscar" name="txt_cedula_buscar" value=""
placeholder="C dula del estudiante que desea buscar"
required="required" tabindex="1" autofocus="autofocus" />
    </label>
</div>
<div>
    <label for="txt_nombre">
        <span>Nombre:</span>
        <input type="text" id="txt_nombre"
name="txt_nombre" value="" readonly/>
    </label>
</div>
<div>
    <label for="txt_apellido">
        <span>Apellido:</span>
        <input type="text" id="txt_apellido"
name="txt_apellido" value="" readonly/>
    </label>
</div>
<div>
    <label for="txt_cedula">
        <span>C dula:</span>
        <input type="text" id="txt_cedula"
name="txt_cedula" value="" readonly/>
    </label>
</div>
<div>
    <label for="txt_copias">
        <span>Copias:</span>
        <input type="text" id="txt_copias"
name="txt_copias" value="" readonly/>
    </label>
</div>
<div>
    <label for="txt_sede">
        <span>Sede:</span>
        <input type="text" id="txt_sede"
name="txt_sede" value="" readonly/>
    </label>
</div>
</form>
<div>
    <button name="btn_buscar" id="btn_buscar">
BUSCAR</button>
</div>
<div>
    <button name="btn_limpiar" id="btn_limpiar">
LIMPIAR CAMPOS</button>
</div>
</div>

```

```

    <script type="text/javascript" src="js/jquery-
3.1.0.js"></script>
    <script type="text/javascript"
src="js/buscar_estudiante.js"></script>
</body>
</html>

```

## Código del servidor en PROLOG

```

:- module(uneg,[ buscar_estudiante/2,
insertar_estudiante/7, actualizar_estudiante/7,
renovar_copias/2, consumir_copias/4 ]).

:- set_prolog_flag(double_quotes, chars).

:- use_module(library(wsd1)).
:- use_module(library(xpath)).
:- use_module(library(soap)).

:- use_module(library(http/thread_httpd)).
:- use_module(library(http/http_dispatch)).
:- use_module(library(http/http_parameters)).
:- use_module(library(http/html_write)).
:- use_module(library(http/http_header)).

:- http_handler(root(uneg), uneg_servicios, []).

:- http_server(http_dispatch, [port(3000)]).

:- debug(soap).

:- multifile sgml_write:xmlns/2.

sgml_write:xmlns(uneg, 'http://localhost').

:- initialization wsd1_read('uneg.wsd1').

uneg_servicios(Request):-
    http_parameters(
        Request,
        [
            metodo(Metodo, []),
            cedula(Cedula, []),
            nombre(Nombre, []),
            apellido(Apellido, []),
            copias(Copias, []),
            clave(Clave, []),

```



```

insertar_estudiante(Cedula, Nombre, Apellido, Copias,
Clave, Sede, Reply) :-
    Operation = ('http://localhost':unegSoap) /
                ('http://localhost':'insertarEstudiante'),
    soap_call(Operation,
        [ 'cedula'=Cedula,
          'nombre'=Nombre,
          'apellido'=Apellido,
          'copias'=Copias,
          'clave'=Clave,
          'sede' =Sede
        ],
        Reply)
.

```

```

actualizar_estudiante(Cedula, Nombre, Apellido, Copias,
Clave, Sede, Reply) :-
    Operation = ('http://localhost':unegSoap) /
                ('http://localhost':'actualizarEstudiante'),
    soap_call(Operation,
        [ 'cedula'=Cedula,
          'nombre'=Nombre,
          'apellido'=Apellido,
          'copias'=Copias,
          'clave'=Clave,
          'sede' =Sede
        ],
        Reply)
.

```

```

renovar_copias(Sede, Reply) :-
    Operation = ('http://localhost':unegSoap) /
                ('http://localhost':'renovarCopias'),
    soap_call(Operation,
        [
          'sede'=Sede
        ],
        Reply)
.

```

```

consumir_copias(Cedula, Clave, Copias, Reply) :-
    Operation = ('http://localhost':unegSoap) /
                ('http://localhost':'consumirCopias'),
    soap_call(Operation,
        [ 'cedula'=Cedula,
          'clave' =Clave,
          'copias'=Copias
        ],
        Reply)
.

```

## 2. d. Prueba

Se procede a realizar las pruebas ejecutando el programa, para ver si maneja los requerimientos operacionales y de información.

## 2. e. Implementación





## 2.1.Arquitectura del sistema

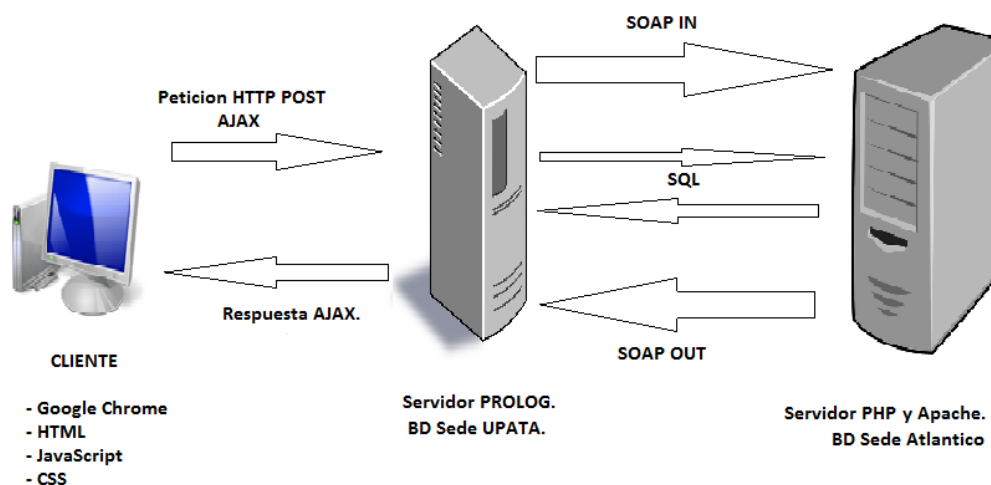


Figura N: Arquitectura Implementada.

Se tiene una arquitectura Cliente/Servidor, de manera que un servidor pueda proporcionar servicio a los clientes. Este cliente, esta dado por Google Chrome, a través de una página HTML con JavaScript para la noción dinámica de la página. Estos clientes, realizaran peticiones a un servidor a través de un método POST del protocolo HTTP con AJAX.

Este servidor que ha recibido dicha petición, debe tomar la petición y enviar una petición SOAP al servidor Principal, y si se busca un estudiante y este pertenece a la base de datos de dicho servidor, es decir de la Sede Atlántico, este servidor principal (PHP), enviara un SOAP OUT, o mensaje SOAP de salida al servidor Prolog, y este se encargara de enviar un objeto AJAX al cliente. En caso de que el estudiante no pertenezca a la Sede Atlantico, el servidor principal ha de comunicarse con sql con el servidor Prolog que es el que posee las fragmentaciones de la Sede Upata, y este buscara el estudiante en su base de datos, y luego envía sql al servidor principal de vuelta, y este, envía el SOAP OUT, o SOAP de salida, y este servidor en PROLOG se encargara de proporcionar el objeto AJAX en respuesta para el cliente.

Por lo tanto, entendemos que la comunicación sql entre ambos servidores llega a ser opcional, y además se deduce que la fragmentación es por Relación, ya que un estudiante pertenecerá al fragmento que contenga una base de datos, según su sede.

La arquitectura requiere poseer dos servidores para la base de datos fragmentada, de modo que se tenga una fragmentación en un servidor y el resto de la fragmentación, se localice en el servidor que resta. La comunicación entre servidores, puede verse como si el servidor de Prolog, es decir el servidor destinado a la Sede UPATA, pasa a ser un cliente del Servidor PHP, en caso de que un estudiante no pertenezca a dicha sede UPATA, por lo tanto este servidor PROLOG deberá pedir información al Servidor PHP de Atlántico.

El uso de un servidor en PHP es para manejar las peticiones SOAP, ya que el servidor PROLOG, por sí solo, solo puede enviar peticiones SOAP, mas no puede mantenerse activo escuchando peticiones SOAP, es decir, para que un servidor PROLOG pueda recibir un mensaje SOAP, es que este sea una respuesta a una petición que el mismo había hecho.

Ya sabiendo la comunicación entre los servidores, el intercambio de mensajes se realiza sobre XML puesto que es en lo que está basado SOAP. El servidor en PROLOG, procesara el XML, para mandar como respuesta al cliente un objeto AJAX, ya que el cliente envía su petición a través de POST sobre AJAX.

### **2.1.1 Modelo de Fallas**

Para el comportamiento de fallas que pueda ocasionar irregularidades en el sistema, tenemos dos enfoques.

#### **2.1.1.1 TCP:**

Como las consultas a Base de Datos están sobre TCP, y esto se logra a través de la librería que proporciona PHP para trabajar con bases de datos mySQL, por lo tanto, se asegura establecer una conexión entre puntos a punto a la hora de comunicación, y de esta manera siempre se tiene asegurado la recepción de los mensajes.

#### **2.1.1.2 Caída de servidor de Base de Datos:**

En cada Computadora, hay dos servidores, un Servidor Web y un Servidor para Base de datos. Ya se dijo que, para que se pueda enviar una respuesta al cliente, se debe enviar un mensaje SOAP entre servidores, (Del servidor Prolog un SOAP IN, al servidor en PHP, y un SOAP OUT, del servidor PHP al servidor Prolog de vuelta), por lo tanto, si se cae el servidor de Base de datos de

UPATA, no resulta de mayor problema ya que de igual manera se enviarán los SOAP, y el servicio de sacar copia se realiza en ATLANTICO. Como no se encuentra el estudiante en la base de datos de Upata, ya que el servidor de UPATA está caído, lo importante sería que se dejara el estudiante igual pudiera sacar sus Fotocopias.

Por lo tanto, si el estudiante a buscar se encuentra en la base de datos que se cayó y esta es la base de datos de UPATA, nos encontraríamos con la situación de no poder actualizar los datos del estudiante, o consumir sus copias, lo que se hace es colocar al estudiante en una tabla PENDIENTE en la base de datos. En esta tabla, se agregaran los datos pertinentes del estudiante y las copias que saco, de manera que cuando el servidor de UPATA se restablezca, se puedan insertar los datos del estudiante de UPATA a la base de datos del servidor correspondiente a esa Sede.

Resumiendo:

- Se consume el servicio de sacar copias en ATLANTICO.
- Si ambos servidores web y de base de datos están activos, no hay problema.
- Si se cae el servidor de Upata, y se intenta consumir el servicio con un estudiante que pertenece a la sede Upata, los datos del estudiante se guardaran en una tabla Pendiente, y se permitirá sacar sus copias.
- Del punto anterior, cuando se restablezca el servidor de Upata, todos los registros de la Tabla Pendiente, se vaciaran en su respectiva base de datos de Upata, y se descontaran las copias que le queden.

En el caso de que el servidor de base de datos de atlántico se caiga, no se podrá consumir el servicio.

### **2.1.1.3 Caída del servidor web cualquiera:**

En el caso de que algún servidor Web se caiga, tampoco se podrá consumir el servicio, ya que no se podrá realizar las transacciones de mensajes SOAP, por lo que caeríamos, al igual que los bancos, que si se cae el sistema principal, no cuentan con sistema global.

## 2.2.Instalación

Programas necesarios:

- Apache
- PHP
- MySQL
- Swi-Prolog
- Chromium
- Allow-Control-Allow-Origin (Extensión de Google Chrome)

Antes de comenzar la instalación vamos a definir nombres para cada una de las PC, ya que los paquetes a instalar y las configuraciones varían según la función que cumplirá cada PC, teniendo esto en cuenta llamaremos a la primera computadora PC1 (En esta PC se estará ejecutando el servidor en Prolog y estará la base de datos que simula una sede en Upata), la segunda computadora la llamaremos PC2 (En esta PC se estará ejecutando el servidor Apache y estará la base de datos que simula la sede principal en este caso atlántico), y por último la tercera computadora la llamaremos PC3 (Esta PC se usará como cliente para consumir los servicios). En cuanto a los archivos necesarios para la configuración y ejecución estarán dentro una carpeta llamada rootcs comprimida en un archivo .zip con el mismo nombre.

Instalación y configuración de paquetes en PC1:

-Swi-Prolog:

1. Para instalar este paquete ejecutamos desde la terminal como superusuario el siguiente comando:

```
# apt-get install swi-prolog
```

2. Añadir las librerías necesarias para trabajar con SOAP y WSDL, para ello nos dirigimos al directorio rootcs/libreriaspl/ y copiaremos los 3 archivos .pl contenidos aquí (soap.pl, wsdl.pl y xml\_schema.pl ), estos archivos debemos pegarlos en el directorio /usr/lib/swi-prolog/library/

## -MySQL:

1. Para instalar este paquete ejecutamos desde la terminal como superusuario el siguiente comando:

```
# apt-get install mysql-server mysql-client
```

2. Durante la instalación procedemos a seguir los pasos de configuración que aparecerán en pantalla (Aquí definiremos entre otras cosas la contraseña para el usuario root).

3. Accedemos a la shell de MySQL como usuario como root y con la clave que especificamos en el paso anterior a través del siguiente comando:

```
# mysql -u root -p
```

4. Creamos un usuario para esta base de datos llamado uneg\_upata y con la clave uneg\_upata a través del siguiente comando:

```
mysql> CREATE USER uneg_upata IDENTIFIED BY  
'uneg_upata';
```

5. Otorgamos los privilegios de conexión y administración al usuario recién creado sobre la base de datos llamada uneg\_upata (Aún no creada) a través del siguiente comando:

```
mysql> GRANT ALL ON uneg_upata.* TO  
'uneg_upata'@'%';
```

6. Creamos una base de datos llamada uneg\_upata a través del siguiente comando:

```
mysql> CREATE DATABASE uneg_upata;
```

7. Importamos la base de datos ubicada en rootcs/bbdd/uneg\_upata.sql a través del siguiente comando:

```
# mysql -u root -p uneg_upata < rootcs/bbdd/uneg_upata.sql
```

8. Configuramos MySQL para permitir conexiones remotas desde cualquier host, para ello vamos a abrir el archivo `/etc/mysql/my.cnf` y buscaremos las siguientes líneas y las descomentamos en caso de estar comentadas:

- `skip-external-locking`
- `bind-address = 127.0.0.1`  
Cambiamos 127.0.0.1 por 0.0.0.0 de tal manera que en el archivo `my.cnf` quede así:
- `bind-address = 0.0.0.0`  
Podemos abrir y editar el archivo `my.cnf` a través del siguiente comando:

```
# gedit /etc/mysql/my.cnf
```

9. Reiniciamos el servicio `mysql` a través del siguiente comando:

```
# service mysql restart
```

Instalación y configuración de paquetes en PC2:

-Apache:

1. Para instalar este paquete ejecutamos desde la terminal como superusuario el siguiente comando:

```
# apt-get install apache2
```

-PHP:

1. Para instalar este paquete ejecutamos desde la terminal como superusuario el siguiente comando:

*# apt-get install php5 libapache2-mod-php5 php5-mcrypt-*  
MySQL:

1. Para instalar este paquete ejecutamos desde la terminal como superusuario el siguiente comando:

*# apt-get install mysql-server php5-mysql*

2. Durante la instalación procedemos a seguir los pasos de configuración que aparecerán en pantalla (Aquí definiremos entre otras cosas la contraseña para el usuario root).

3. Accedemos a la shell de MySQL como usuario como root y con la clave que especificamos en el paso anterior a través del siguiente comando:

*# mysql -u root -p*

4. Creamos un usuario para esta base de datos llamado uneg y con la clave uneg a través del siguiente comando:

**mysql> CREATE USER uneg IDENTIFIED BY 'uneg';**

5. Otorgamos los privilegios de conexión y administración al usuario recién creado sobre la base de datos llamada uneg\_upata (Aún no creada) a través del siguiente comando:

**mysql> GRANT ALL ON uneg.\* TO 'uneg'@'%';**

6. Creamos una base de datos llamada uneg\_upata a través del siguiente comando:

**mysql> CREATE DATABASE uneg;**

7. Importamos la base de datos ubicada en rootcs/bbdd/uneg.sql a través del siguiente comando:

```
# mysql -u root -p uneg < rootcs/bbdd/uneg.sql
```

8. Configuramos MySQL para permitir conexiones remotas desde cualquier host, para ello vamos a abrir el archivo `/etc/mysql/my.cnf` y buscaremos las siguientes líneas y las descomentamos en caso de estar comentadas:

- `skip-external-locking`
- `bind-address = 127.0.0.1`

Cambiaremos `127.0.0.1` por `0.0.0.0` de tal manera que en el archivo `my.cnf` quede así:

- `bind-address = 0.0.0.0`

Podemos abrir y editar el archivo `my.cnf` a través del siguiente comando:

```
# gedit /etc/mysql/my.cnf
```

9. Reiniciamos el servidor mysql a través del siguiente comando:

```
# service mysql restart
```

\* (Opcional):

\*10. Instalamos PHPMyAdmin ejecutando desde la terminal como superusuario el siguiente comando:

```
# apt-get install phpmyadmin
```

\*11. Durante la instalación procedemos a seguir los pasos de configuración que aparecerán en pantalla teniendo en cuenta lo siguiente:

- Seleccionaremos **apache2**, cuando se nos pregunte por el servidor web en el que queremos ejecutar Phpmyadmin.
- Seleccionaremos **Sí**, cuando se nos pregunte si deseamos configurar la base de datos para phpmyadmin con `<<dbconfig-common>>`
- Introduciremos y confirmaremos la contraseña de administrador **root de Mysql**.



\*12. Incluimos phpmyadmin dentro de la configuración de apache, para ello vamos a abrir el archivo `/etc/apache2/apache2.conf` y nos vamos al final e introducimos el siguiente texto:

- `# phpMyAdmin Configuración`
- `Include /etc/phpmyadmin/apache.conf`

Podemos abrir y editar el archivo `apache2.conf` a través del siguiente comando:

```
# gedit /etc/apache2/apache2.conf
```

\*13. Reiniciamos el servidor apache a través del siguiente comando:

```
# service apache2 restart
```

Instalación y configuración de paquetes en PC3:

Chromium:

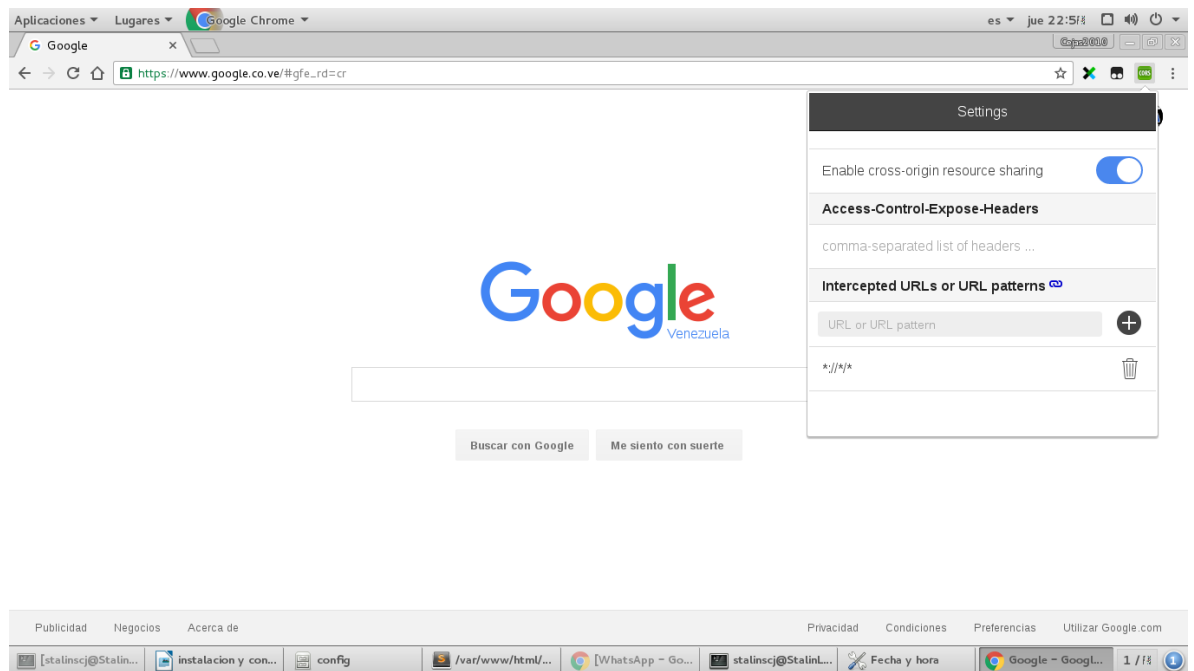
1. Para instalar este paquete ejecutamos desde la terminal como superusuario el siguiente comando:

```
# apt-get install chromium
```

2. Instalamos la extensión de Google Chrome (también es compatible con chromium) llamada Allow-Control-Allow-Origin, para ello nos dirigimos a la url que se muestra a continuación y seleccionamos la opción añadir a chrome:

<https://chrome.google.com/webstore/detail/allow-control-allow-origi/nlfbmbojpeacfgbkpbjhddihlkkiljbi>

3. Activamos esta extensión pulsando el icono llamado CORS y moviendo el botón de **enable** a la derecha, nos quedará así:



Una vez finalizada la instalación de todos los paquetes necesarios en todas las computadoras procedemos a copiar los siguientes archivos y carpetas que se encuentran dentro de la carpeta **rootcs** y los pegamos en **/var/www/html/** tal como se especifica a continuación:

En la **PC1** copiamos todos los archivos y carpetas contenidos en la carpeta **rootcs** a excepción de la carpeta **prolog/** para ello nos ubicamos dentro de la carpeta **rootcs** y como superusuario ejecutamos los siguientes comandos:

```
# cp -r * /var/www/html/  
  
# rm -r /var/www/html/prolog
```

En la **PC2** copiamos solo la carpeta **prolog/** para ello nos ubicamos dentro de la carpeta **rootcs** y como superusuario ejecutamos los siguientes comandos:

```
# cp -r prolog/ /var/www/html/
```

## Configuración de archivos en PC1

1. WSDL: Vamos a modificar el archivo `/var/www/html/controladores/uneg_servicios.php` para que apunte al wsdl que contiene la descripción de los servicios prestados, para ello abrimos el archivo con el siguiente comando:

```
# gedit /var/www/html/controladores/uneg_servicios.php
```

Ahora reemplazamos localhost por la IP de la PC2 (Para ver la IP ejecute `# ifconfig` en la PC2) en la siguiente línea:

```
$server = new SoapServer('http://localhost/prolog/uneg.wsdl');
```

Si la IP de la PC2 es 192.168.0.101 entonces nos quedaría de la siguiente manera:

```
$server = new SoapServer('http://192.168.0.101/prolog/uneg.wsdl');
```

2. Base de Datos externa: Modificamos el archivo `/var/www/html/config/dataBaseConfig2.php` de tal forma que este apunte al servidor MySQL de la PC2, para ello abriremos el archivo ejecutando como superusuario el siguiente comando:

```
# gedit /var/www/html/config/dataBaseConfig2.php
```

Ahora le asignamos la IP de la PC2 a "hostname" (Para ver la IP ejecute `# ifconfig` en la PC2) en la siguiente línea:

```
"hostname" => "192.168.0.111",
```

Si la IP de la PC2 es 192.168.0.101 entonces nos quedaría de la siguiente manera:

```
"hostname" => "192.168.0.101",
```

3. Peticiones a la PC2 (Servidor Prolog): Ahora modificaremos cada uno de los archivos .js ubicados en /var/www/html/vistas/js/ para que realicen las peticiones al servidor prolog ubicado en la PC2, para ello abrimos cada uno de los archivos .js ejecutando como superusuario el siguiente comando:

```
# gedit /var/www/html/vistas/js/buscar_estudiante.js
```

Ahora reemplazamos localhost por la IP de la PC2 (Para ver la IP ejecute # ifconfig en la PC2) en la siguiente linea:

```
url: "http://localhost:3000/uneg",
```

Si la IP de la PC2 es 192.168.0.101 entonces nos quedaría de la siguiente manera:

```
url: "http://192.168.0.101:3000/uneg",
```

\* Repetimos este proceso para cada uno de los archivos .js solo seamos cambiar el nombre del archivo en el comando por ejemplo para el archivo llamado consumir\_copias.js sería:

```
# gedit /var/www/html/vistas/js/consumir_copias.js
```

#### Configuración de archivos en PC2

1. Peticiones a la PC1 (Servidor Apache): Vamos a modificar el archivo /var/www/html/prolog/uneg.pl para que realice las llamadas SOAP a la PC1, para ello abrimos el archivo con el siguiente comando:

```
# gedit /var/www/html/prolog/uneg.pl
```

Ahora reemplazamos localhost por la IP de la PC1 (Para ver la IP ejecute # ifconfig en la PC1), podemos ayudarnos con el editor de texto para que busque y reemplace todas las palabras "localhost" y la reemplace por la IP de la PC1, para ello pulsamos CTRL+H e ingresamos la palabra a buscar que queremos reemplazar en este caso "localhost" y la palabra que la reemplazará en este caso la IP de la PC1.

Si la IP de la PC1 es 192.168.0.100 entonces nos quedarían todas las url de la siguiente manera:

```
'http://192.168.0.100'
```

2. WSDL: Modificamos el archivo `/var/www/html/prolog/uneg.wsdl` de tal forma que este apunte al archivo que gestiona los llamados SOAP llamado `uneg_servicios.php` ubicados en la PC1, para ello abriremos el archivo ejecutando como superusuario el siguiente comando:

```
# gedit /var/www/html/prolog/uneg.wsdl
```

Ahora reemplazamos `localhost` por la IP de la PC1 (Para ver la IP ejecute `# ifconfig` en la PC1), podemos ayudarnos con el editor de texto para que busque y reemplace todas las palabras `"localhost"` y la reemplace por la IP de la PC1, para ello pulsamos `CTRL+H` e ingresamos la palabra a buscar que queremos reemplazar en este caso `"localhost"` y la palabra que la reemplazará en este caso la IP de la PC1, tal como lo hicimos en el paso anterior.

### 2.3.Pruebas realizadas

Probando ambos servidores de MySQL caídos:

Para simular la caída de los servidores (es decir que no responden), configuraremos las IP's de estos de tal modo que no apunten a ningún servidor de MySQL siguiendo las instrucciones descritas en el Paso 2 de la configuración de la PC1 sabiendo que en la IP 192.168.0.222 no hay ningún servidor de base de datos, modificaremos los archivos `dataBaConfig.php` y `dataBaConfig2.php` quedando de la siguiente manera ambos archivos:

```
"hostname" => "192.168.0.222",
```

foto

1. Compilamos y ejecutamos el archivo `uneg.pl` en la PC2 para activar el servidor `prolog`, para ello nos posicionamos en el directorio `/var/www/html/prolog/` con el siguiente comando:

```
$ cd /var/www/html/prolog
```

Ahora procedemos a ejecutar swi-prolog con el siguiente comando:

```
$ swipl
```

Luego compilamos y ejecutamos el archivo uneg.pl con el siguiente comando:

```
[uneg].
```

Nos debería quedar así:

foto

foto

2. Intentamos consumir el servicio que buscar un estudiante dada su cédula, para ello abrimos chromium y escribimos en la barra de direcciones la IP de la PC1, si la IP es 192.168.0.100 nos quedaría así:

foto

Ingresamos el número de cédula y pulsamos el botón buscar:

foto

Podemos ver el archivo SOAP que se envía en la terminal de la PC2:

(FOTO)

Vemos la respuesta que devuelve el servidor

(FOTO)

Probando un servidores de MySQL caído:

Para simular la caída de un servidor (es decir que no responde), configuraremos la IP de este, de tal modo que no apunte a ningún servidor de MySQL siguiendo las instrucciones descritas en el Paso 2 de la configuración de la PC1 sabiendo que en la IP 192.168.0.222 no hay ningún servidor de base de datos, modificaremos el archivo dataBaConfig2.php quedando de la siguiente manera:

```
"hostname" => "192.168.0.222",
```

(FOTO)

1. Compilamos y ejecutamos el archivo uneg.pl en la PC2 para activar el servidor prolog, para ello nos posicionamos en el directorio /var/www/html/prolog/ con el siguiente comando:

```
$ cd /var/www/html/prolog
```

Ahora procedemos a ejecutar swi-prolog con el siguiente comando:

```
$ swipl
```

Luego compilamos y ejecutamos el archivo uneg.pl con el siguiente comando:

```
[uneg].
```

Nos debería quedar así:

(FOTO1)

(FOTO2)

2. Intentamos consumir el servicio que buscar un estudiante dada su cédula, para ello abrimos chromium y escribimos en la barra de direcciones la IP de la PC1, si la IP es 192.168.0.100 nos quedaría así:

(FOTO)

Ingresamos el número de cédula y pulsamos el botón buscar:

(FOTO)

Podemos ver el archivo SOAP que se envía en la terminal de la PC2:

(FOTO)

Vemos la respuesta que devuelve el servidor

(FOTO)

Probando los dos servidores de MySQL funcionando normalmente:

1. Compilamos y ejecutamos el archivo uneg.pl en la PC2 para activar el servidor prolog, para ello nos posicionamos en el directorio /var/www/html/prolog/ con el siguiente comando:  
`$ cd /var/www/html/prolog`

Ahora procedemos a ejecutar swi-prolog con el siguiente comando:  
`$ swipl`

Luego compilamos y ejecutamos el archivo uneg.pl con el siguiente comando:  
`[uneg].`

Nos debería quedar así:

(FOTO1)

(FOTO2)

2. Intentamos consumir el servicio que busca un estudiante dada su cédula, para ello abrimos chromium y escribimos en la barra de direcciones la IP de la PC1, si la IP es 192.168.0.100 nos quedaría así:

(FOTO)

Ingresamos el número de cédula y pulsamos el botón buscar:



(FOTO)

Podemos ver el archivo SOAP que se envía en la terminal de la PC2:

(FOTO)

Vemos la respuesta que devuelve el servidor

(FOTO)

### **3. Conclusión**

La Universidad Nacional Experimental de Guayana cuenta con un servicio de Fotocopiado, el cual cuenta con irregularidades que corresponder a problemas con la replicación, ya que un estudiante puede consumir fotocopias en una sede, y alguien con su misma cedula puede consumir copias del mismo estudiante en otra sede, ya que la replicación no es instantánea, por lo tanto resulta necesario una alternativa para resolver dicho problema, por lo que se desarrollo un servicio web en Prolog para el sistema de fotocopiado de la Universidad Nacional Experimental de Guayana con la utilización de SOAP y su descripción de servicio WSDL.

Como todo desarrollo de software, es necesario seguir una metodología que permita el desarrollo del proyecto bajo en un enfoque esquemático, por lo tanto:

Se tiene que el análisis de los requerimientos funcionales y de información de los servicios a ofrecer viene siendo el punto más importante del desarrollo del servicio a proponer, ya que de esta manera, se puede entender los problemas que el sistema existente pueda tener, y sus características de funcionamiento y poder detectar mejoras posibles que se pueden implementar.

Con los requerimientos una vez captados, esto nos permite poder diseñar los programas, diagramas y modelos que nos den un enfoque de cómo debería ser la arquitectura a utilizar en el sistema. Estos diseños son la base para la codificación, porque nos identifica el funcionamiento y la manera en que se debe manejar la información que se está procesando y consumiendo con los servicios.

Estos diseños, nos sirven como marco de trabajo para la codificación, entendiendo la manera en que debe funcionar los códigos que se han desarrollar las operaciones que el sistema ha de realizar. Estos modelos sirven también para verificar que lo que se está desarrollando es lo que en realidad se debe programar, y de esta manera, no sirve también para realizar las pruebas al sistema, ya que como

se dijo, sirven para verificar que se está programando lo que se estableció en la fase de diseño, dando veracidad de esta manera a las pruebas realizadas.

Y ya con las pruebas realizadas, y entendiendo que están correctas, sin necesidad de volver a modificar mas nada, se puede implementar el servicio desarrollado para la Universidad Nacional Experimental de Guayana, como alternativa para contrarrestar las irregularidades que presenta el sistema actual.

## **ANEXOS FIGURAS**

```
% library(url) compiled into url 0.01 sec, 323 clauses
% library(memfile) compiled into memory_file 0.00 sec, 19 clauses
% library(broadcast) compiled into broadcast 0.00 sec, 13 clauses
% library(arithmetic) compiled into arithmetic 0.00 sec, 115 clauses
% library(settings) compiled into settings 0.01 sec, 214 clauses
% library(dcg/basics) compiled into dcg_basics 0.00 sec, 74 clauses
% library(pure_input) compiled into pure_input 0.00 sec, 20 clauses
% library(quasi_quotations) compiled into quasi_quotations 0.00 sec, 44 c
% html_quasiquotations compiled into html_quasi_quotations 0.00 sec, 60 cl
% html_write compiled into html_write 0.02 sec, 411 clauses
% http_exception compiled into http_exception 0.00 sec, 21 clauses
% mimetype compiled into mimetype 0.00 sec, 59 clauses
% mimepack compiled into mime_pack 0.00 sec, 23 clauses
% library(http/http_header) compiled into http_header 0.07 sec, 1,635 clause
% library(ssl) compiled into ssl 0.00 sec, 43 clauses
% library(rbtrees) compiled into rbtrees 0.01 sec, 190 clauses
% library(thread_pool) compiled into thread_pool 0.01 sec, 238 clauses
% http_stream compiled into http_stream 0.00 sec, 17 clauses
% http_wrapper compiled into httpd_wrapper 0.00 sec, 86 clauses
% thread_httpd compiled into thread_httpd 0.02 sec, 425 clauses
% library(http/http_ssl_plugin) compiled into http_ssl_plugin 0.03 sec, 480
% library(soap) compiled into soap 0.13 sec, 2,655 clauses
% library(time) compiled into time 0.00 sec, 25 clauses
% library(http/http_host) compiled into http_host 0.00 sec, 28 clauses
% library(http/http_path) compiled into http_path 0.00 sec, 77 clauses
% library(http/http_dispatch) compiled into http_dispatch 0.01 sec, 236 claus
% http_client compiled into http_client 0.00 sec, 92 clauses
% library(mime) compiled into mime 0.00 sec, 8 clauses
% http_mime_plugin compiled into http_mime_plugin 0.00 sec, 23 clauses
% http_hook compiled into http_hook 0.00 sec, 1 clauses
% library(http/http_parameters) compiled into http_parameters 0.01 sec, 199 c
% uneg compiled into uneg 0.22 sec, 5,257 clauses
true.
```

? -



stalinscj@StalinLX: ~



instalacion y conf.odt - Li...



[config]



/var/

Archivo Editar Ver Buscar Terminal Ayuda

```
stalinscj@StalinLX:~$ cd /var/www/html/prolog/
stalinscj@StalinLX:/var/www/html/prolog$ swipl
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 6.6.6)
Copyright (c) 1990-2013 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.
```

```
For help, use ?- help(Topic). or ?- apropos(Word).
```

```
?- [uneg].
```



stalinscj@StalinLX: ~



instalacion y conf.odt - Li...



[config]



/var/www/html/config/d...



Atajos de teclado ó

dataBaseConfig.php •

```
1 <?php
2     return array(
3         "hostname" => "192.168.0.222"
4         "username" => "uneg",
5         "password" => "uneg",
6         "dbname"   => "uneg"
7     );
8 ?>
```

dataBaseConfig2.php x

```
1 <?php
2     return array(
3         "hostname"
4         "username"
5         "password"
6         "dbname"
7     );
8 ?>
```

Line 8, Column 3



stalinscj@StalinLX: ~



instalacion y conf.odt - Li...



config



/var/www/html/config/d...




Atajos de teclado ú

```
dataBaseConfig.php x
1 <?php
2
3     return array(
4         "hostname" => "192.168.0.222"
5         "username" => "uneg",
6         "password" => "uneg",
7         "dbname"   => "uneg"
8     );
9 ?>
```

```
dataBaseConfig2.php •
1 <?php
2     return array(
3         "hostname" => "192.168.22
4         "username" => "uneg_upata
5         "password" => "uneg_upata
6         "dbname"   => "uneg_upata
7     );
8 ?>
```

Aplicaciones ▾ Lugares ▾ gedit ▾

Abrir ▾ 

uned.p  
/var/www/html

```
:- module(uneg,[ buscar_estudiante/2, insertar_estudiante/7, actualizar_estudi

:- set_prolog_flag(double_quotes, chars).

:- use_module(library(wsd)).
:- use_module(library(xpath)).
:- use_module(library(soap)).

:- use_module(library(http/thread_httpd)).
:- use_module(library(http/http_dispatch)).
:- use_module(library(http/http_parameters)).
:- use_module(library(http/html_write)).
:- use_module(library(http/http_header)).

:- http_handler(root(uneg), uneg_servicios, []).

:- http_server(http_dispatch, [port(3000)]).


:- debug(soap).


:- multifile sgml_write:xmlns/2.


sgml_write:xmlns(uneg, 'http://localhost').


:- initialization wsd_read('uned.wsd').

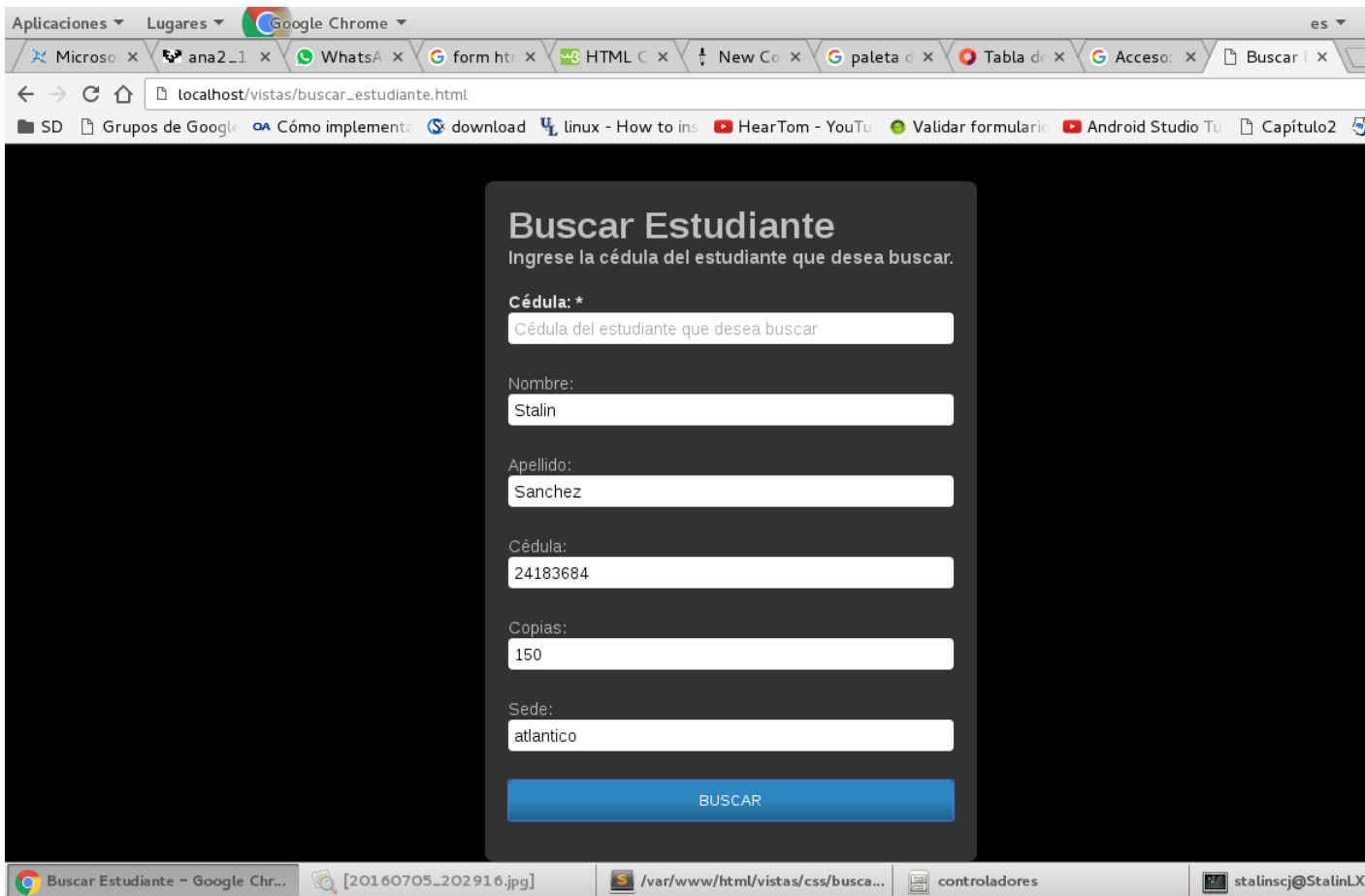
uned_servicios(Request):-
    http_parameters(
        Request,
        [
            metodo(Metodo, []),
            cedula(Cedula, []),
            nombre(Nombre, []),
```

 stalinscj@StalinLX: ~

 instalacion y conf.od...

 prolog

 [/var/www/html/pr





Aplicaciones ▾ Lugares ▾ Google Chrome ▾

Microso x ana2\_1 x WhatsA x form htr x HTML C x New Co x paleta d x Tabla d x Acceso: x Buscar | x

localhost/vistas/buscar\_estudiante.html

SD Grupos de Google Cómo implementa download linux - How to ins HearTom - YouTu Validar formulario Android Studio Tu Capítulo2

# Buscar Estudiante

Ingrese la cédula del estudiante que desea buscar.

**Cédula: \***

Nombre:

Apellido:

Cédula:

Copias:

Sede:

BUSCAR

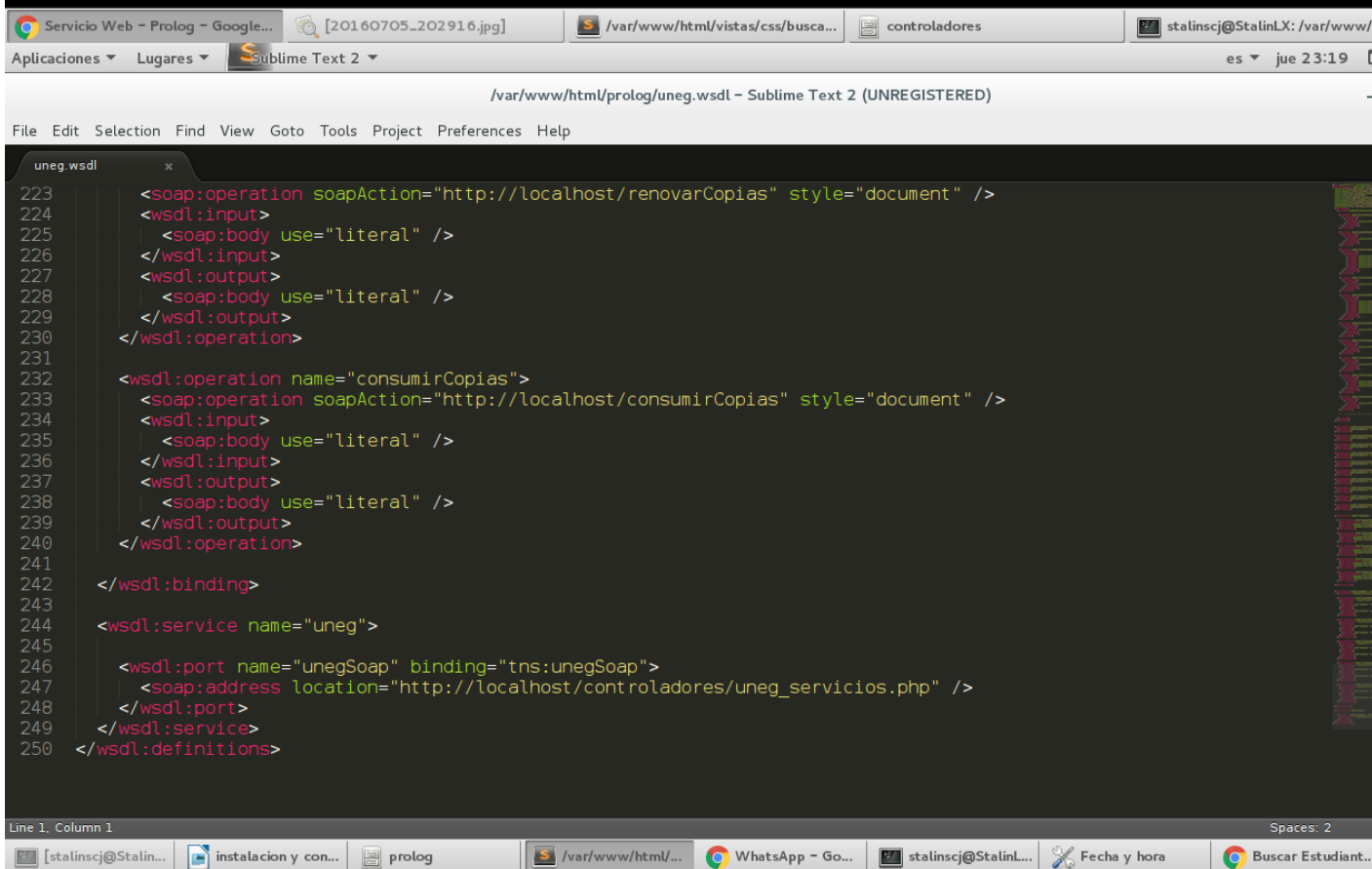
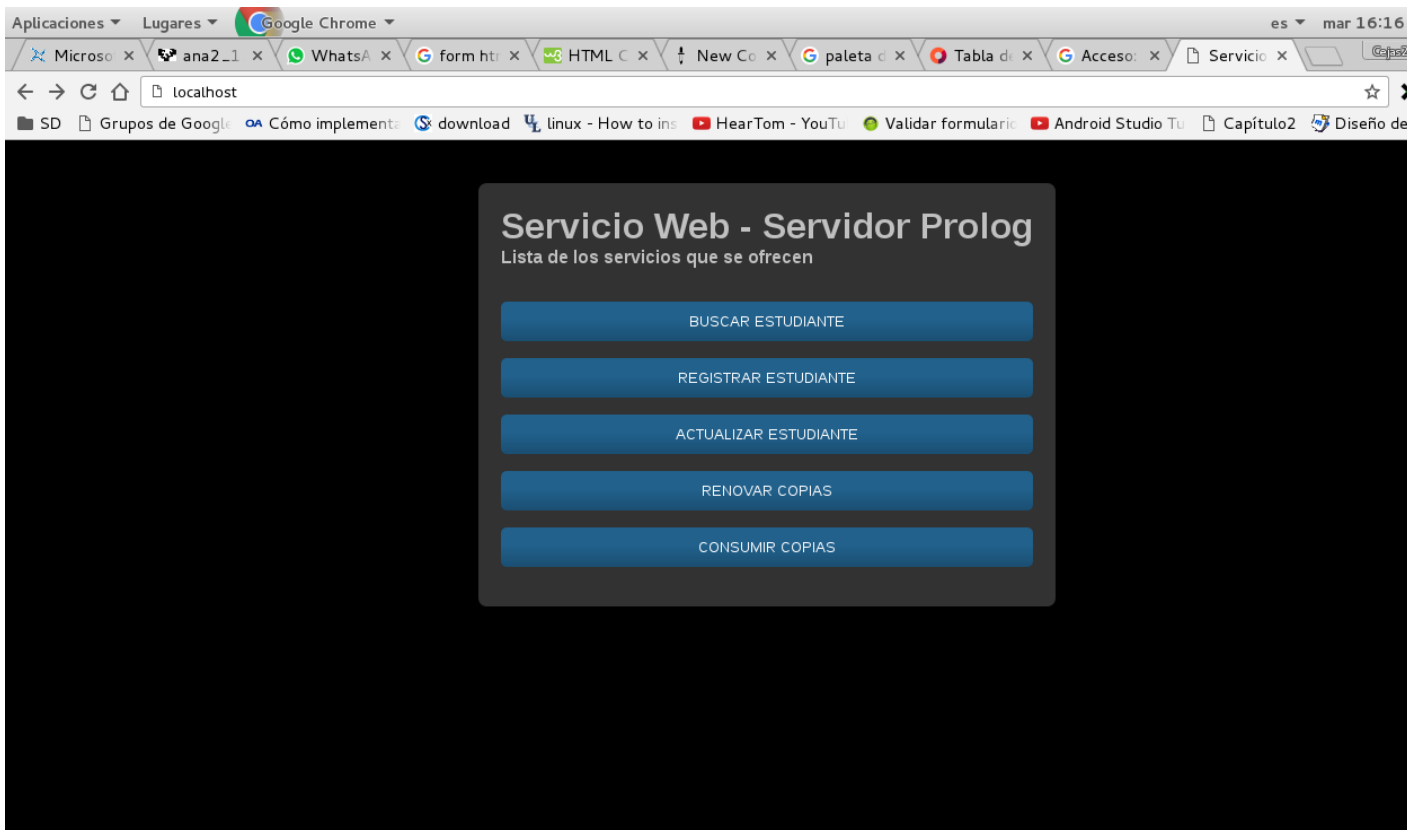
Buscar Estudiante - Google Chr...

[20160705\_202916.jpg]

/var/www/html/vistas/css/busca...

controladores

stalinscj@Stalin...



```
uneg.wsdl x
154 <wsdl:portType name="unegSoap">
155
156   <wsdl:operation name="buscarEstudiante">
157     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Busc
158     <wsdl:input message="tns:buscarEstudianteSoapIn" />
159     <wsdl:output message="tns:buscarEstudianteSoapOut" />
160   </wsdl:operation>
161
162   <wsdl:operation name="insertarEstudiante">
163     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">inse
164     <wsdl:input message="tns:insertarEstudianteSoapIn" />
165     <wsdl:output message="tns:insertarEstudianteSoapOut" />
166   </wsdl:operation>
167
168   <wsdl:operation name="actualizarEstudiante">
169     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">actu
170     <wsdl:input message="tns:actualizarEstudianteSoapIn" />
171     <wsdl:output message="tns:actualizarEstudianteSoapOut" />
172   </wsdl:operation>
173
174   <wsdl:operation name="renovarCopias">
175     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Renov
176     <wsdl:input message="tns:renovarCopiasSoapIn" />
177     <wsdl:output message="tns:renovarCopiasSoapOut" />
178   </wsdl:operation>
179
180   <wsdl:operation name="consumirCopias">
181     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">cons
182     <wsdl:input message="tns:consumirCopiasSoapIn" />
183     <wsdl:output message="tns:consumirCopiasSoapOut" />
184   </wsdl:operation>
```

Line 1, Column 1



[stalinscj@Stalin...



instalacion y con...



prolog



/var/www/html/...



WhatsApp - Go...

```
uneg.wsdl
112
113 <wsdl:message name="buscarEstudianteSoapIn">
114   <wsdl:part name="parameters" element="tns:buscarEstudiante" />
115 </wsdl:message>
116
117 <wsdl:message name="buscarEstudianteSoapOut">
118   <wsdl:part name="parameters" element="tns:buscarEstudianteResponse" />
119 </wsdl:message>
120
121 <wsdl:message name="insertarEstudianteSoapIn">
122   <wsdl:part name="parameters" element="tns:insertarEstudiante" />
123 </wsdl:message>
124
125 <wsdl:message name="insertarEstudianteSoapOut">
126   <wsdl:part name="parameters" element="tns:insertarEstudianteResponse" />
127 </wsdl:message>
128
129 <wsdl:message name="actualizarEstudianteSoapIn">
130   <wsdl:part name="parameters" element="tns:actualizarEstudiante" />
131 </wsdl:message>
132
133 <wsdl:message name="actualizarEstudianteSoapOut">
134   <wsdl:part name="parameters" element="tns:actualizarEstudianteResponse" />
135 </wsdl:message>
136
137 <wsdl:message name="renovarCopiasSoapIn">
138   <wsdl:part name="parameters" element="tns:renovarCopias" />
139 </wsdl:message>
140
141 <wsdl:message name="renovarCopiasSoapOut">
142   <wsdl:part name="parameters" element="tns:renovarCopiasResponse" />
```

uneg.wSDL

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
3   xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
4   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
5   xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
6   xmlns:tns="http://localhost"
7   xmlns:s="http://www.w3.org/2001/XMLSchema"
8   xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
9   xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
10  targetNamespace="http://localhost"
11  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
12  <wsdl:types>
13    <s:schema elementFormDefault="qualified" targetNamespace="http://localhost">
14      <s:element name="buscarEstudiante">
15        <s:complexType>
16          <s:sequence>
17            <s:element minOccurs="0" maxOccurs="1" name="cedula" type="s:string" />
18          </s:sequence>
19        </s:complexType>
20      </s:element>
21
22      <s:element name="buscarEstudianteResponse">
23        <s:complexType>
24          <s:sequence>
25            <s:element minOccurs="0" maxOccurs="1" name="buscarEstudianteResult" type="s:string" />
26          </s:sequence>
27        </s:complexType>
28      </s:element>
29
30      <s:element name="insertarEstudiante">
```

Line 1, Column 1



[stalinscj@Stalin...



instalacion y con...



prolog



/var/www/html/...



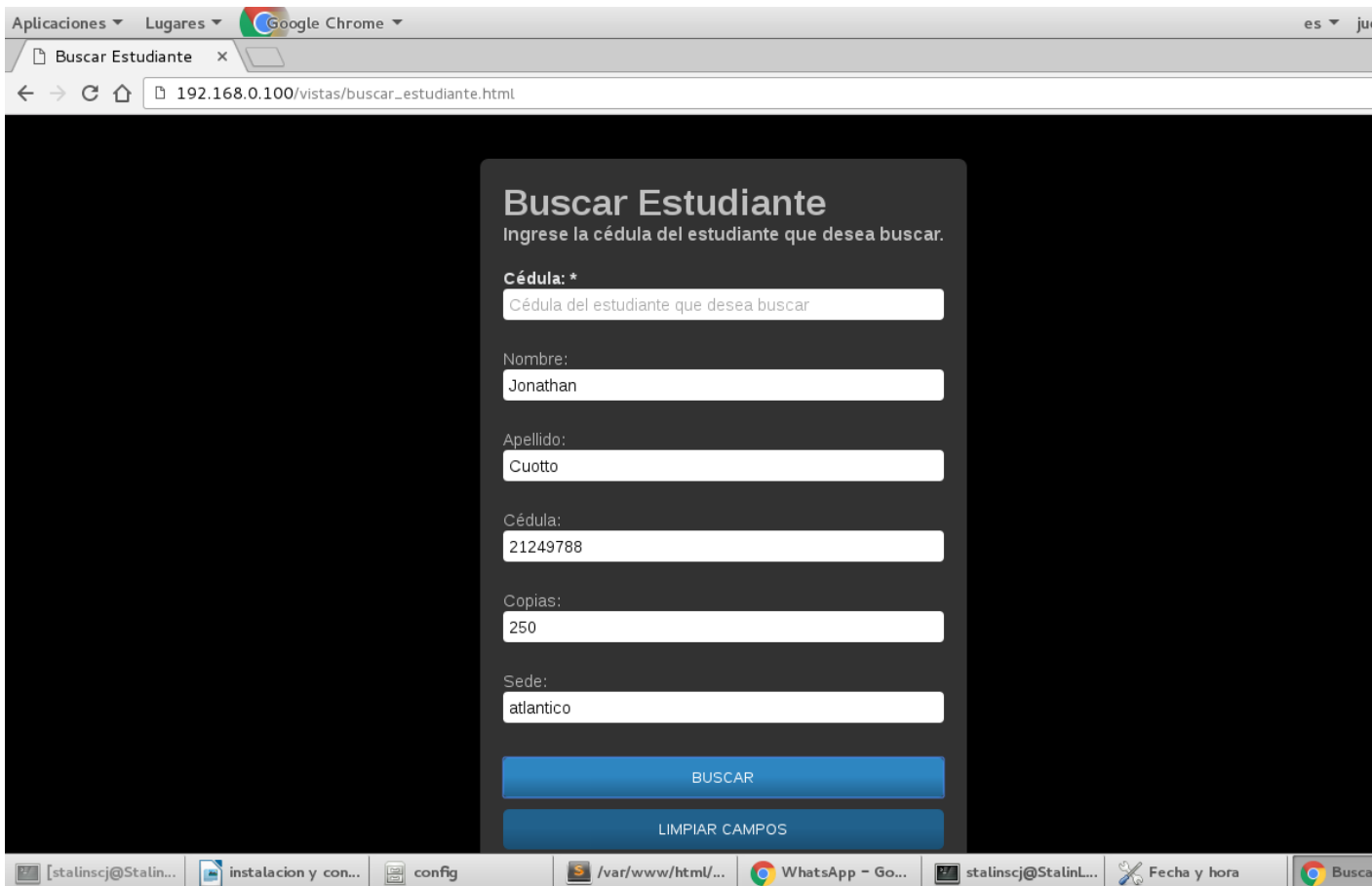
WhatsApp - Go...



stalinscj@Stalin...



Fec



Aplicaciones ▾ Lugares ▾ Terminal ▾ es ▾ ju

stalinscj@StalinLX: /var/www/html/prolog

Archivo Editar Ver Buscar Terminal Ayuda

?- % [Thread httpd@3000\_1] soap11: URL='http://localhost/controladores/uneg\_servicios.php'  
% [Thread httpd@3000\_1] SOAPAction: "http://localhost/buscarEstudiante"  
Content-Length: 270  
Content-Type: text/xml; charset=UTF-8  
  
<?xml version="1.0" encoding="UTF-8"?>  
  
<soap11:Envelope xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/">  
 <soap11:Body>  
 <buscarEstudiante xmlns="http://localhost">  
 <cedula>21249788</cedula>  
 </buscarEstudiante>  
 </soap11:Body>  
</soap11:Envelope>  
% [Thread httpd@3000\_1] Status = 200; content = 'text/xml; charset=utf-8'  
█

[stalinscj@Stalin... instalacion y con... config /var/www/html/... WhatsApp - Go... stalinscj@StalinL... Fecha y hora Busca

Aplicaciones ▾ Lugares ▾ Google Chrome ▾ es ▾

Buscar Estudiante x

192.168.0.100/vistas/buscar\_estudiante.html

## Buscar Estudiante

Ingrese la cédula del estudiante que desea buscar.

Cédula: \*  
21249788

Nombre:

Apellido:

Cédula:

Copias:

Sede:

BUSCAR

LIMPIAR CAMPOS

[stalinscj@Stalin...]

instalacion y con... config /var/www/html/... WhatsApp - Go... stalinscj@Stalin... Fecha y hora

Aplicaciones ▾ Lugares ▾ Google Chrome ▾ es ▾ jue 23:12

Buscar Estudiante x

192.168.0.100/vistas/buscar\_estudiante.html

192.168.0.100 dice:

Error: El estudiante no se encuentra en la sede principal y no se pudo conectar a la Base de Datos de upata.

☐ Evita que esta página cree cuadros de diálogo adicionales.

Aceptar

## B

Ing

Cédula: \*  
214

Nombre:

Apellido:

Cédula:

Copias:

Sede:

BUSCAR

LIMPIAR CAMPOS

[stalinscj@Stalin...]

instalacion y con... config /var/www/html/... WhatsApp - Go... stalinscj@Stalin... Fecha y hora

Buscar Estudiant... 1 / 14



## Buscar Estudiante

Ingrese la cédula del estudiante que desea buscar.

Cédula: \*

21489789

Nombre:

Apellido:

Cédula:

Copias:

Sede:

BUSCAR

LIMPIAR CAMPOS

Archivo Editar Ver Buscar Terminal Ayuda

```
?- % [Thread httpd@3000_3] soap11: URL='http://localhost/controladores/uneg_se
% [Thread httpd@3000_3] SOAPAction: "http://localhost/buscarEstudiante"
Content-Length: 270
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>

<soap11:Envelope xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/">
  <soap11:Body>
    <buscarEstudiante xmlns="http://localhost">
      <cedula>21249789</cedula>
    </buscarEstudiante>
  </soap11:Body>
</soap11:Envelope>
```

dataBaseConfig2.php •

```
1 <?php
2     return array(
3         "hostname"    => "192.168.0.222",
4         "username"    => "uneg_upata",
5         "password"    => "uneg_upata",
6         "dbname"      => "uneg_upata"
7     );
8 ?>
```

Line 8, Column 3



[stalinscj@Stalin...



instalacion y con...



config



/var/www/html/...



Wha

Aplicaciones ▾ Lugares ▾ Terminal ▾

stalinscj@StalinLX: /var/www/html/prolog

es ▾ jue 22:58

Archivo Editar Ver Buscar Terminal Ayuda

```
?- % [Thread httpd03000_5] soap11: URL='http://localhost/controladores/uneg_servicios.php'
% [Thread httpd03000_5] SOAPAction: "http://localhost/buscarEstudiante"
Content-Length: 270
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>

<soap11:Envelope xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/">
  <soap11:Body>
    <buscarEstudiante xmlns="http://localhost">
      <cedula>24183684</cedula>
    </buscarEstudiante>
  </soap11:Body>
</soap11:Envelope>
% [Thread httpd03000_5] Status = 200; content = 'text/xml; charset=utf-8'
```

stalinscj@Stalin... instalacion y con... config /var/www/html/... [WhatsApp - Go... stalinscj@Stalin... Fecha y hora Buscar Estudiant...

Aplicaciones ▾ Lugares ▾ Google Chrome ▾

es ▾ jue 22:58

Buscar Estudiante x

192.168.0.100/vistas/buscar\_estudiante.html

192.168.0.100 dice:  
Error: No se pudo conectar a la Base de Datos de la sede principal ni a la de upata.  
Aceptar

B  
Ing  
Céd  
24183684

Nombre:

Apellido:

Cédula:

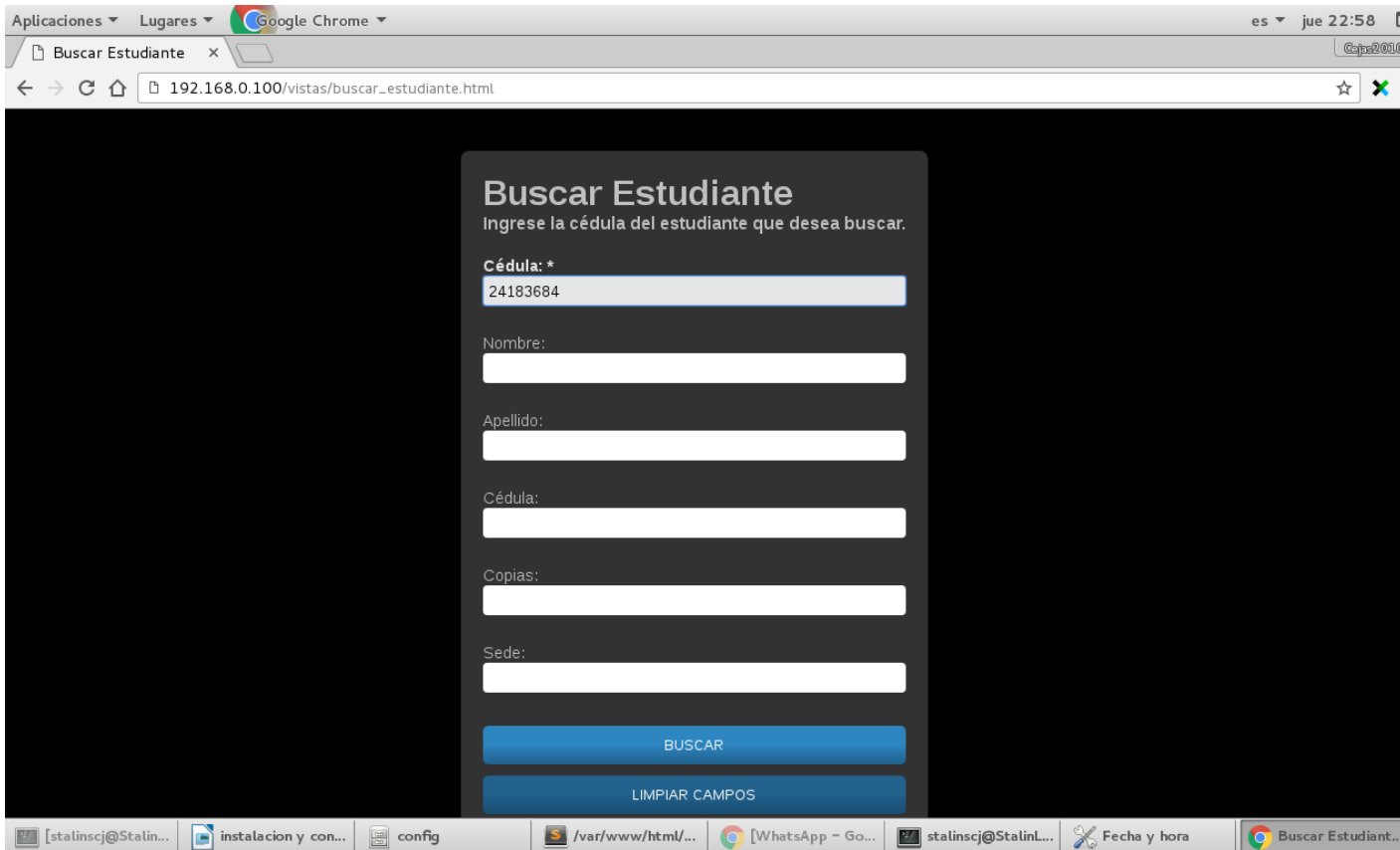
Copias:

Sede:

BUSCAR

LIMPIAR CAMPOS

stalinscj@Stalin... instalacion y con... config /var/www/html/... [WhatsApp - Go... stalinscj@Stalin... Fecha y hora Buscar Estudiant...



## Servicio Web - Servidor Prolog

Lista de los servicios que se ofrecen

BUSCAR ESTUDIANTE

REGISTRAR ESTUDIANTE

ACTUALIZAR ESTUDIANTE

RENOVAR COPIAS

CONSUMIR COPIAS

192.168.0.100/vistas/buscar\_estudiante.html



Buscar con Google

Me siento con suerte

### Settings

Enable cross-origin resource sharing ☒

#### Access-Control-Expose-Headers

comma-separated list of headers ...

#### Intercepted URLs or URL patterns <sup>CO</sup>

URL or URL pattern +

\*/\*/\*\* 🗑️

## Bibliografía

- N. Bermúdez y A. Ibáñez y J. González. *Modelo de seguridad para una arquitectura de Servicios Web XML*. Madrid, 2007.
- G. Coulouris and J. Dollimore and T. Kindberg and G. Blair. *DISTRIBUTED SYSTEM. Concepts and Design. Fifth Edition*. Number ISBN: 978-0-13-214301-1. Addison Wesley, 2012.
- R. Englander. *Java and SOAP*. Number ISBN: 0-596-00175-4. O'Reilly. Mayo 2002.
- R. Hernández. *Fundamentos de los sistemas de Bases de Datos Distribuidas*. Universidad Central de Venezuela, Caracas. 2013.
- R. Navarro. *Rest vs Web Services*. 2007
- E. Nava. *Modelo de evaluación de metodologías para el desarrollo de software*. Caracas, Venezuela. 2006.