MANUAL TENICO WEB SERVICE
SIGMA INGENIERIA

1. **Instalación de software requerido y librerías en el servidor**
2. **Instalación de Phusion Passenger para Apache**
3. **Instalación de la aplicación y configuración de acceso a la base de datos**
4. **Configuración general del Web Service**
5. **Configuración de nombres de campos para servicios Vital**
6. **Descarga de actualizaciones desde repositorio**
7. **Acceso a métodos del Web Service**
8. **Descripción de métodos de consulta**
9. **Descripción de métodos de inserción**
10. **Documentación y web links.**

Marco Emilio Montes Botero
Ingeniero de Sistemas y Telecomunicaciones

email: marcomontes@gmail.com
Skype: marcomontes
Teléfono Celular: 301 267 9613
Teléfono Oficina: 8 86 21 22

1. Instalación de software requerido y librerías en el servidor

Instalando Ruby 1.8.7 con Rails 2.3.5 en CentOS 5

Tomado de http://blog.rockdeveloper.com/2010/01/09/instalando-ruby-y-rails-en-centos-5/ y actualizado a las versiones requeridas

CentOS puede instalar una versión de Ruby por defecto, es recomendable omitirla, ya que de cualquier modo lo que necesitaremos será la versión mas estable de ruby donde las Gems si compilan. En este momento esa versión es la 1.8.7. Instalar Ruby con YUM tampoco es una opción ya que descarga la versión 1.8.5 –Siempre puedes ejecutar yum erase ruby, si ya instalaste esta versión.

En este tutorial deberás estar logeado como root o tener permisos de superusuario para poder ejecutar algunas instrucciones.

Manos a la obra

1. Instalar GCC

```
sudo yum install -y gcc
```

Gcc es la herramienta que usaremos para compilar ruby. Puedes omitir el sudo si estas logeado como root. El modificar -y le indica a YUM que asuma SI a todas las preguntas, esto te ahorra tiempo, si quieres interactuar puedes omitirlo y estar atento a la instalación, incluso usar el modificador -v para que YUM te platique todo lo que esta haciendo.

2. Descargar Ruby 1.8.7-p72 desde ruby-lang.org

```
cd /usr/src
```

```
sudo curl -O ftp://ftp.ruby-lang.org/pub/ruby/1.8/ruby-1.8.7-p72.tar.gz
```

Asegurate de descargar el paquete de ruby marcado con -p72, de lo contrario algo podría no funcionar correctamente.

Una vez que el paquete de ruby 1.8.7 fue descargado procedemos a descomprimirlo, configurar, compilar e instalar.

Copilando ruby

```
tar xzvf ruby-1.8.7-p72.tar.gz
```

```
cd ruby-1.8.7-p72
```

```
./configure
```

```
make
```

```
make install
```

En este momento, si todo ha salido bien, podemos ejecutar el comando which ruby para ver si tenemos ruby instalado correctamente y ruby -v para enterarnos de la versión de este.

Comprobando la instalación

```
which ruby
```

```
ruby -v
```

El primer comando nos devuelve la ruta donde ruby esta instalado, y el segundo la versión de este.

3. Descargar las Gems

El procedimiento es muy similar, usamos curl para descargar y tar para descomprimir. Despúes procedemos a compilar GEM con ruby setup.rb

```
cd ..
```

```
curl -O
http://files.rubyforge.vm.bytemark.co.uk/rubygems/rubygems-
1.3.7.tgz
```

```
tar xzvf rubygems-1.3.7.tgz
```

```
cd rubygems-1.3.7
```

```
ruby setup.rb
```

En este momento si corremos el comando **`gem list`** nos debe devolver una lista vacía de las gemas disponibles, esto es porque aun nos falta instalar rails, mysql y mongrel.

4. Instalando Rails y Mongrel

```
cd /
```

```
gem install rails -v 2.3.8
```

Despues de haber corrido el comando para instalar rails, 8 gemas deben haberse instalado para ruby, si corremos el comando **`gem list`**, nos devolvera la lista de gemas instaladas, en este momento debemos poder ver rails version 2.3.8, rake 0.8.7 y compañía.

2. Instalación de Phusion Passenger para Apache

Tomado de: http://www.modrails.com/install.html

Ver esta URL en caso de mas detalles para la instalación

1. Instalar la gema passenger

```
gem install passenger
```

2. Instalar el modulo passenger para apache 2

```
passenger-install-apache2-module
```

3. Seguir las instrucciones de el asistente para la instalación de este componente

4. Configuración del host virtual en apache

Bloque de código para la creación de un nuevo host virtual es opcional, puede adaptarse a su configuración personalizada.

La ruta /webservice hace referencia al directorio o ruta absoluta donde se copiaran los archivos del webservice.

/webservice/public. 'public' es un directorio ya existente dentro de /webservice/

```
<VirtualHost *:80>
    ServerName webservice.dev
    DocumentRoot /webservice/public
    RailsEnv production
    <Directory /webservice/public>
         Options ExecCGI FollowSymLinks
         AllowOverride all
         Allow from all
    </Directory>
</VirtualHost>
```

3. Instalación de la aplicación y configuración de acceso a la base de datos

Instrucciones para la descarga de la aplicación desde el repositorio

1. Crear una cuenta en www.github.com

2. Enviar al administrador del repositorio el nombre de usuario y correo electrónico con el que quedó creada la cuenta en github.com (Para concederle acceso al repositorio privado)

3. Ingresar al repositorio https://github.com/espiral/webservice_sigma para verificar que se tiene acceso

5. Instalar GIT en el servidor

http://shakaran.net/blog/2010/07/como-instalar-git-en-un-servidor-centos/

6. Descargar el proyecto con el comando:

git@github.com:espiral/webservice_sigma.git

Este comando se ejecuta en la ruta donde quedará ubicado el proyecto, por ejemplo desde /var/www y como resultado creará una carpeta llamada webservice_sigma

Finalmente la ruta completa sería /var/www/webservice_sigma

7. Verificar que la ruta anterior concuerde con la ruta del virtual host de apache. Ejemplo:

```
<VirtualHost *:80>
    ServerName webservice.dev
    DocumentRoot /var/www/webservice_sigma/public
    RailsEnv production
    <Directory /var/www/webservice_sigma/public>
         Options ExecCGI FollowSymLinks
         AllowOverride all
         Allow from all
    </Directory>
</VirtualHost>
```

Esta línea 'ServerName webservice.dev' se configura u omite de acuerdo a la configuración local de su propio servidor.

Instrucciones para la configuración de acceso a la base de datos

En el archivo `config/database.yml` se establecen los parámetros de conexión para la base de datos en producción

```
production:
  adapter: postgresql
  username: consultacorpo
  password: consultacorpo
  database: corpocaldas
  encoding: utf8
  pool: 5
  host: 200.58.206.186
```

4. Configuración general del Web Service

En el archivo `config/webservice.yml` se establecen parámetros de acceso y configuración para el webservice.

```
webservice_root: http://200.58.206.186:8686/
output:
  download: true
  xml: false
```

El valor para **webservice_root** hace referencia a la URL desde donde se tiene acceso a el webservice

El valor booleano para **output** hace referencia a el tipo de salida que se generara para la respuesta enviada por los metodos de Vital.

download genera un archivo xml descargable
xml renderiza en el navegador la estructura xml de la respuesta enviada

5. Configuración de nombres de campos para servicios Vital

En el archivo `config/vital.yml` se establecen los nombres de los campos usados por cada método en vital y agrupados por cada servicio.

Se permite la configuración de nombres en este archivo debido al estado incompleto de definición de nombres de campos y métodos en la especificación de Vital.

Ejemplo:

```
wsaud:
  recibir_datos_audiencia_publica:
    - numero_silpa
    - fecha_expedicion
    - id_solicitante
    - ids_entidades
    - nombre_proyecto
    - numero_expediente
    - id_persona_interesada
    - fecha_audiencia_publica
    - lugar_audiencia_publica
    - lugares_inscripcion
    - lugares_consulta
    - fecha_reunion
    - lugar_reunion
    - convocatoria_ponentes
    - edicto_adjunto
    - id_autoridad_ambiental
```

wsaud hace referencia a uno de los nombres de servicio incluidos en Vital

recibir_datos_audiencia_publica hace referencia a uno de los nombres de los metodos incluidos en el servicio **wsaud**

- numero_silpa
- fecha_expedicion
- …
hacen referencia a los nombres de los campos usados en el método
recibir_datos_audiencia_publica

6. Descarga de actualizaciones desde repositorio

El software permite ser actualizado a nuevas versiones de archivos simplemente utilizando el comando:

```
git pull
```

Este comando descarga las ultimas versiones de los archivos desde el repositorio y debe ser ejecutado desde la carpeta donde se encuentra instalado el software.

Para su correcto funcionamiento es requerido que se haya instalado el acceso al repositorio especificado en el paso *3. Instalación de la aplicación y configuración de acceso a la base de datos.*

7. Acceso a Métodos del Web Service

El software ha sido desarrollado bajo la arquitectura REST permitiendo un fácil acceso a los recursos a través de su URI

```
http://nombre_del_host/servicio/nombre_del_metodo
```

Ejemplo:

```
http://host/wsaud/recibir_datos_audiencia_publica
```

En el archivo `doc/app/index.html` se encuentra la documentación, especificación y referencia de los métodos usados en el webservice.

Por defecto todos los métodos del web service usan el método HTTP GET y estos están especificados en el archivo `config/routes.rb`

Ejemplo:

```
map.resources :wsaud, :collection => {
  :recibir_datos_audiencia_publica => :get
}
```

puede ser adapatado al metodo HTTP POST simplemente cambiando `:get` por `:post`

```
map.resources :wsaud, :collection => {
  :recibir_datos_audiencia_publica => :post
}
```

8. Descripción de métodos de consulta

La estructura general de los métodos de consulta es la siguiente:

```
  def nombre_del_metodo
    method = get_method_name
    vital = VitalAdapter.get_fields(get_service, method)

    sql = ""

    sql_result = Corpocaldas.query(sql)
    send_file(method, sql_result)
  end
```

Donde **sql =** " " contiene la consulta en formato SQL que se ejecutará en la base de datos.

Los parámetros para las condiciones tendrán el siguiente formato:

```
"select campo1, campo2
 from tabla
 where campo1 = '#{vital['nombre_del_campo']}'"
```

nombre_del_campo hace referencia y debe ser reemplazado por el nombre del campo especificado en el archivo de configuración **vital.yml**

9. Descripción de métodos de inserción

La estructura general de los métodos de inserción es la siguiente:

```
def nombre_metodo_insertar
  solicitud = Intermedia.create_record(params)
  method = get_method_name
  send_file(method, {"insert" => solicitud})
end
```

Este método recibirá los parámetros especificados en el archivo **“Documentación Metodos”** y procesará automáticamente la inserción de datos en la tabla **t_intermedia** a traves del método **create_record(parameters)** incluido en el archivo **app/models/intermedia.rb**

Retornará un valor booleano (falso o verdadero) de acuerdo al éxito o fracaso en la inserción de datos.

8. Documentación y Web Links

- Instalación Ruby on Rails en CentOS
http://blog.rockdeveloper.com/2010/01/09/instalando-ruby-y-rails-en-centos-5/

- Instalación Phusion Passenger
http://www.modrails.com/install.html

- Instalación GIT en CentOS
http://shakaran.net/blog/2010/07/como-instalar-git-en-un-servidor-centos/

- URL del repositorio
https://github.com/espiral/webservice_sigma

- Ruta de documentación
doc/app/index.html

- Ruby on Rails
http://es.wikipedia.org/wiki/Ruby_on_Rails

- REST
http://es.wikipedia.org/wiki/Representational_State_Transfer

- GIT (Control de Versiones)
http://es.wikipedia.org/wiki/Git

- Formato YAML
http://es.wikipedia.org/wiki/YAML