

Tarea 7

Servidor Tomcat

Con soporte REST

Viernes, 27 de noviembre de 2020

Joel Harim Hernández Javier

Descripción

Cada alumno ejecutará el procedimiento que vimos en clase, dónde instalamos Tomcat, instalamos MySQL y creamos un servicio web estilo REST.

Se deberá probar el servicio web utilizando la aplicación web prueba.html tal como se explicó en clase.

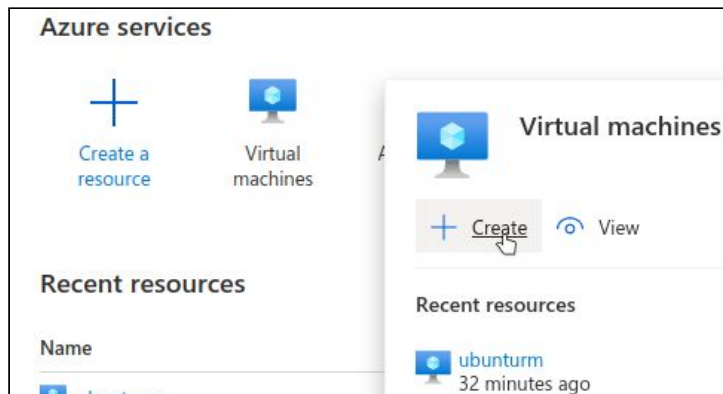
Se deberá entregar un reporte en formato PDF que incluya la descripción de cada paso y la captura de pantalla correspondiente. Además, el reporte deberá tener portada y conclusiones.

Desarrollo

Primero se necesita crear el servidor, que es donde se va a desplegar el servidor Tomcat.

Creación de la máquina virtual

En `portal.azure.com`, ingresando con nuestra cuenta, vamos al apartado de **Virtual machines > Crear**



De la creación de la máquina virtual, se configuran los siguientes campos:

→ El nombre de la máquina virtual

Instance details	
Virtual machine name *	<input type="text" value="ubuntvm"/>

→ El sistema operativo, Ubuntu Server

Image *	<input type="text" value="Ubuntu Server 18.04 LTS - Gen1"/>
Browse all public and private images	

→ Configurar el logging por password en lugar de ssh

Administrator account	
Authentication type ⓘ	<input type="radio"/> SSH public key <input checked="" type="radio"/> Password
Username *	<input type="text" value="joelcito"/>
Password *	<input type="password" value="....."/>
Confirm password *	<input type="password" value="....."/>

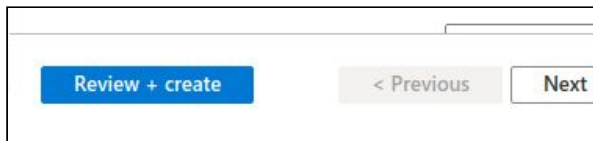
→ En **Disks**, configuramos el uso de discos duros HDD

Disk options	
OS disk type *	<input type="text" value="Standard HDD"/>
The selected VM size supports high IOPS workloads. Virtual 99.9% connectivity SLA.	
Encryption type *	<input type="text" value="(Default) Encryption at-rest"/>

→ En **Managment**, se deshabilita los diagnósticos de arranque



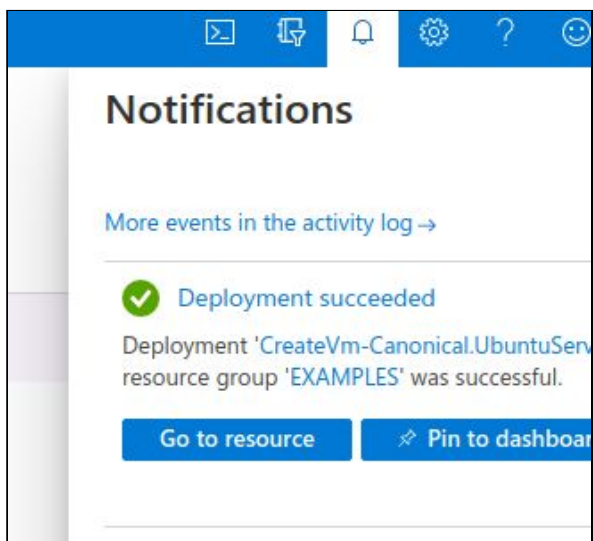
→ Terminamos con la configuración en **Review + create**



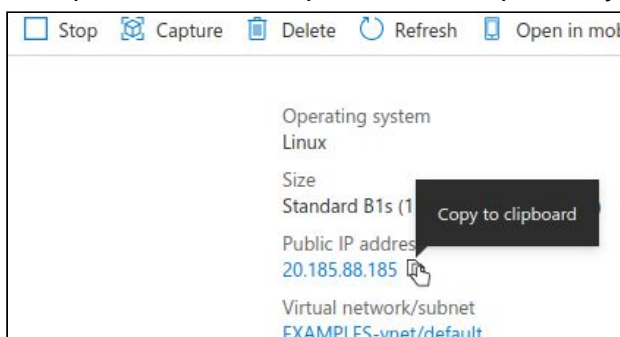
→ Revisamos la configuración y procedemos a crear la máquina virtual con **Create**



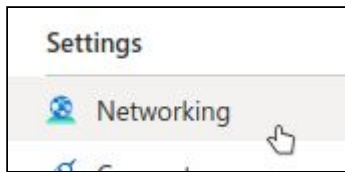
Una vez que se haya creado, ir al recurso (por ejemplo, desde notificaciones):



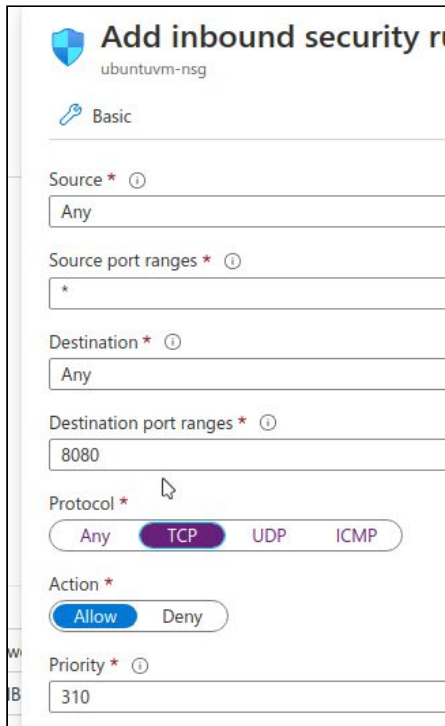
En la pantalla de la máquina virtual, copiamos y guardamos la dirección ip



Seleccionamos la pestaña **Networking**



Agregar regla de puerto de entrada, y configuramos el puerto 8080 por TCP:


A screenshot of the 'Add inbound security rule' configuration window for 'ubuntuvm-nsg'. The 'Basic' tab is selected. The configuration fields are as follows:

- Source: Any
- Source port ranges: *
- Destination: Any
- Destination port ranges: 8080
- Protocol: TCP (selected)
- Action: Allow (selected)
- Priority: 310

Carga de archivos


Para la carga de los archivos, se utilizó un script escrito en bash / shell que toma los archivos necesarios, tanto la aplicación en sí (`Servicio.zip` con `context.xml` ya modificado) y un archivo bash que realiza la configuración del servidor Tomcat y el deploy de la aplicación llamado `deploy.sh`.



En mi caso, configuré el archivo bash `upload.sh` para que subiera los archivos dentro del folder `files/`, se muestra la estructura del directorio.

```
distribuidos/Joel/Practicas/07_tomcat_server on  Joe
→ tree
.
├── credentials.txt
├── files
│   ├── db.sql
│   ├── deploy.sh
│   ├── prueba.html
│   ├── root.sql
│   ├── Servicio.zip
│   ├── sudo.sql
│   ├── usuario_sin_foto.png
│   └── WSClient.js
└── upload.sh

1 directory, 10 files
```

El archivo `credentials.txt` solo es ocupado para guardar de forma manual el y la dirección ip de la máquina remota, el usuario y la contraseña.


```
distribuidos/Joel/Practicas/07_tomcat_server on  Joe
→ cat credentials.txt
20.185.88.185
joelcito
PapasDeLimon.1234%


distribuidos/Joel/Practicas/07_tomcat_server on  Joe
→ 
```



Configuramos los permisos de ejecución del archivo:

```
distribuidos/Joel/Prac
→ chmod +x upload.sh
```

Ejecutamos el archivo `upload.sh`:

```
distribuidos/Joel/Practicas/07_tomcat_server on  Joe
→ ./upload.sh < credentials.txt

 Uploading files

distribuidos/Joel/Practicas/07_tomcat_server on  Joe
→ 
```

Configuración del servidor y deploy de la aplicación

Iniciamos una sesión ssh con la máquina remota:

```
distribuidos/Joel/Practicas/07_tomcat_server on  Joel [!]  
→ ssh joelcito@20.185.88.185  
The authenticity of host '20.185.88.185 (20.185.88.185)' can't be established.  
ECDSA key fingerprint is SHA256:wnJqJ2x7ZdB+9BBfRcI8exj8Ky8027k1EkT3cI5q1r4.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

El anterior script nos creó una carpeta llamada `server/`, cambios a esa carpeta:

```
joelcito@ubuntuvms:~$ ls  
server  
joelcito@ubuntuvms:~$ cd server/  
joelcito@ubuntuvms:~/server$
```

Ejecutamos el archivo `server.sh`, el cual ejecutará la instalación de las utilidades necesarias, la instalación del servidor tomcat, el deploy de la aplicación y la configuración de mysql, la cual NO es automática, y tendremos que ingresar las instrucciones manualmente:

```
joelcito@ubuntuvms:~/server$ ./deploy.sh  
  
:: Updating system and installing utilities
```

Configuración manual del mysql (N, root, root, Y, Y, Y, Y):

```
joelcito@ubuntuvms:~/server$ ./deploy.sh  
  
✔ Updating system and installing utilities  
✔ Installing tomcat  
🔧 Configuring MySQL:  
Please enter 'root' as root's password and 'hugo' as hugo password.  
  
Securing the MySQL server deployment.  
  
Connecting to MySQL using a blank password.  
  
VALIDATE PASSWORD PLUGIN can be used to test passwords  
and improve security. It checks the strength of password  
and allows the users to set only those passwords which are  
secure enough. Would you like to setup VALIDATE PASSWORD plugin?  
  
Press y|Y for Yes, any other key for No:
```

Cuando el script finaliza, el servidor junto con la aplicación estarán desplegados:

```
made so far will take effect immediately.

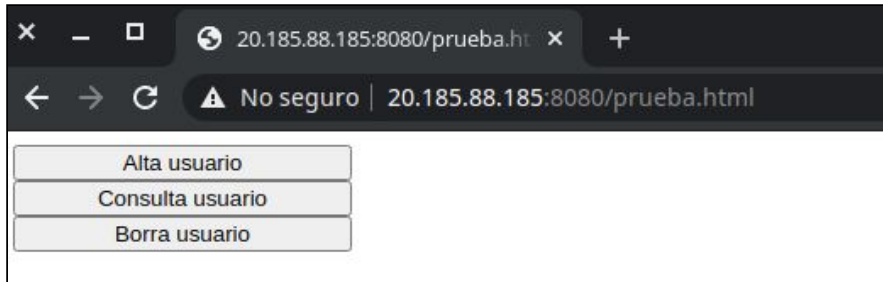
Reload privilege tables now? (Press y|Y for Yes,
Success.

All done!
  Enter root's password: (root)
Enter password:
  Enter hugo's password: (hugo)
Enter password:
  🍷 Configured MySQL
  ✅ Deploying app

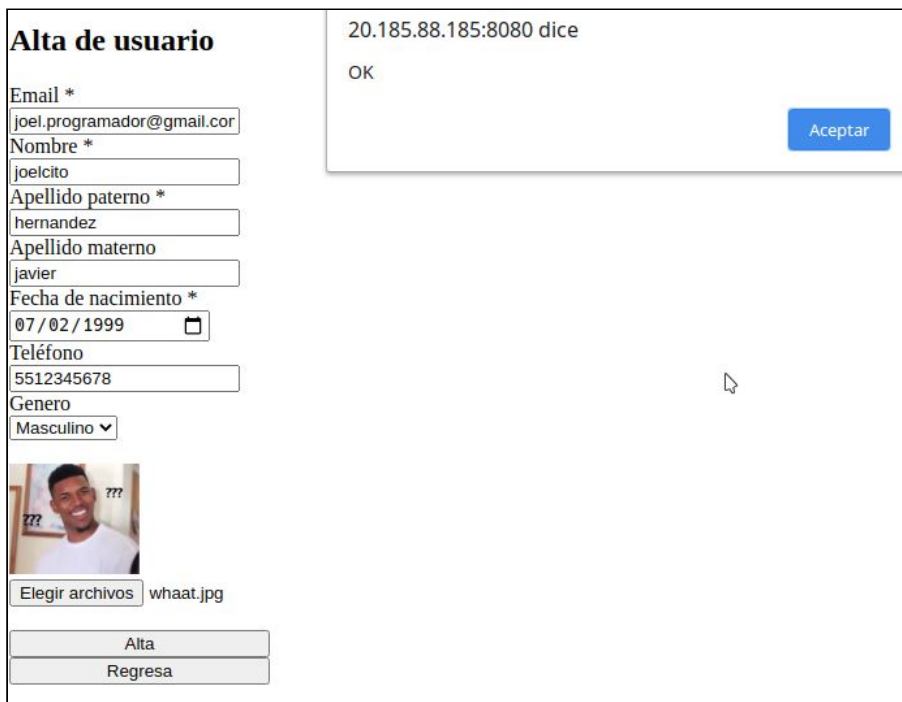
  Tomcat server is running at port 8080. Stop w
/bin/catalina.sh stop
joelcito@ubuntuvms:~/server$
```


Uso de la aplicación

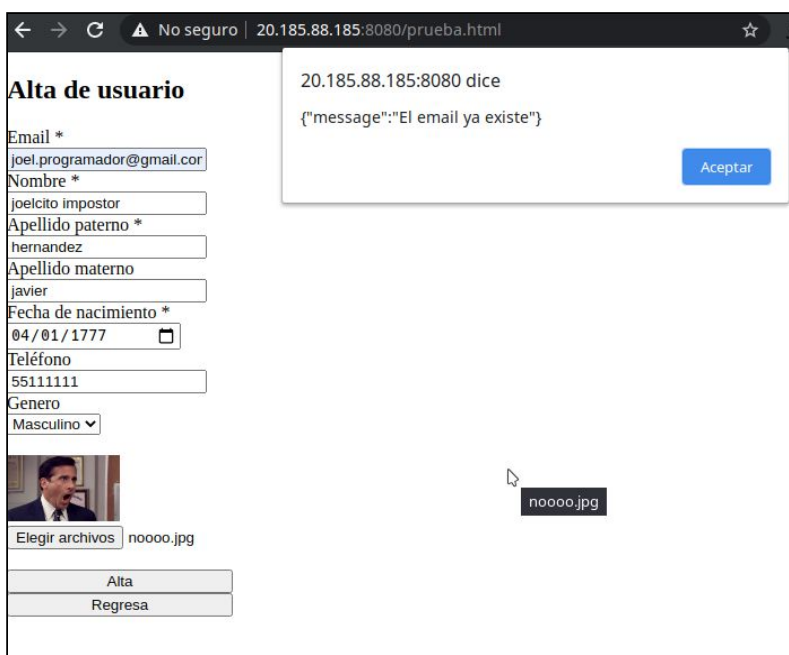
En cualquier navegador, ingresamos a `http://<ip>:8080/prueba.html`:



Dando de alta un usuario:

A screenshot of the 'Alta de usuario' form. The form fields are filled with: Email: joel.programador@gmail.com, Nombre: joelcito, Apellido paterno: hernandez, Apellido materno: javier, Fecha de nacimiento: 07/02/1999, Teléfono: 5512345678, Genero: Masculino. A success message box at the top right says '20.185.88.185:8080 dice OK' with an 'Aceptar' button. At the bottom, there is a file upload section with a placeholder image and the text 'Elegir archivos' followed by 'whaat.jpg'. Below that are 'Alta' and 'Regresa' buttons.

Intentando dar de alta al mismo usuario:

A screenshot of the 'Alta de usuario' form. The form fields are filled with: Email: joel.programador@gmail.com, Nombre: joelcito impostor, Apellido paterno: hernandez, Apellido materno: javier, Fecha de nacimiento: 04/01/1777, Teléfono: 55111111, Genero: Masculino. An error message box at the top right says '20.185.88.185:8080 dice {"message": "El email ya existe"}' with an 'Aceptar' button. At the bottom, there is a file upload section with a placeholder image and the text 'Elegir archivos' followed by 'nooooo.jpg'. Below that are 'Alta' and 'Regresa' buttons.

Consulta:

Consulta/Modifica usuario

Email *
joel.programador@gmail.cor

Nombre *
joelcito

Apellido paterno *
hernandez

Apellido materno
javier

Fecha de nacimiento *
07/02/1999

Teléfono
5512345678

Genero
Masculino



Elegir archivos No se eligió archivo

Consulta
Modifica
Regresa

Modificación:

← → ↻ ⚠ No seguro | 20.185.88.185:8080/prueba.html

Consulta/Modifica usua

Email *
joel.programador@gmail.cor

Nombre *
joelcito modificado


Apellido paterno *
javier

Apellido materno
hernandez

Fecha de nacimiento *
09/09/1999

Teléfono
5512345678

Genero
Masculino



Elegir archivos contexto.jpeg

Consulta
Modifica
Regresa

20.185.88.185:8080 dice
OK

Comprobación de modificación

Consulta/Modifica usuario

Email *

joel.programador@gmail.cor

Nombre *

joelcito modificado

Apellido paterno *

javier

Apellido materno

hernandez

Fecha de nacimiento *


09/09/1999

Teléfono

5512345678

Genero

Masculino



Elegir archivos

No se eligió archivo

Consulta

Modifica

Regresa

Borrado de usuario

← → ↻ ⚠ No seguro | 20.185.88.185:8080/prueba

Borra usuario

Email *

joel.programador@gmail.cor

Borra

Regresa

20.185.88.185:8080

OK

Prueba en un dispositivo móvil

10:25 [íconos de notificación] [íconos de estado]

20.185.88.185:8080/prueba.html [33] [menú]

Alta de usuario

Email *

Nombre *

Apellido paterno *

Apellido materno

Fecha de nacimiento *

Teléfono

Genero


 123840841_...81587_n.jpg

Conclusiones

Esta práctica fue particularmente difícil por la automatización. Intenté el desarrollo del despliegue de forma remota usando expect en bash y luego usando pexpect en python, pero ninguno de los dos lograba de forma satisfactoria (y breve) el despliegue de la aplicación, muchas veces los comandos se perdían y el script no llegaba a su conclusión de forma correcta.

Me decanté entonces por ejecutar el script directamente en el servidor, lo cual funcionó estupendamente bien y el uso de la aplicación web fue un paseo en el parque a comparación del desarrollo del script.

Anexos

upload.sh

(Sin la definición de spinner.sh)

```
read -p "Enter your machine's ip: " IP
read -p "Enter your username: " USERNAME
read -p "Enter your password: " -s pw
printf "\n\n"
start_spinner 'Uploading files'
sshpass -p "${pw}" ssh "${USERNAME}@${IP}" "mkdir -p server"
sshpass -p "${pw}" scp files/prueba.html files/sudo.sql files/deploy.sh
files/root.sql files/usuario_sin_foto.png files/db.sql files/Servicio.zip
files/WSClient.js "${USERNAME}@${IP}:~/server"
sshpass -p "${pw}" ssh "${USERNAME}@${IP}" "chmod +x server/deploy.sh"
stop_spinner $? '✓'
```

deploy.sh

(Sin la definición de spinner.sh)

```
start_spinner 'Updating system and installing utilities'
sudo apt -qq update 2>/dev/null > /dev/null
sudo apt -qq -o Dpkg::Use-Pty=0 install openjdk-8-jdk-headless mysql-server unzip
-y -qq 2>/dev/null > /dev/null
stop_spinner $? '🚀'

start_spinner 'Installing tomcat'
wget -q
https://downloads.apache.org/tomcat/tomcat-8/v8.5.60/bin/apache-tomcat-8.5.60.zip
unzip -qq apache*.zip
rm apache*.zip
cd apache*/
rm webapps -r
mkdir webapps
mkdir webapps/ROOT
wget -qq
https://repo1.maven.org/maven2/org/glassfish/jersey/bundles/jaxrs-ri/2.24/jaxrs-ri-
2.24.zip
unzip -qq jax*.zip
rm jax*.zip
cp jaxrs-ri/api/*.jar lib
cp jaxrs-ri/ext/*.jar lib
cp jaxrs-ri/lib/*.jar lib
rm jaxrs-ri/ -r
rm lib/javax.servlet-api-3.0.1.jar
```

```

cd lib
wget -q
https://repo1.maven.org/maven2/com/google/code/gson/gson/2.3.1/gson-2.3.1.jar
wget -q
https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-8.0.22.zip
unzip -qq mysql*.zip
cp mysql*/mysql*.jar .
rm mysql*/ -r
rm mysql*.zip
stop_spinner $? '🐱'

cd ../
export CATALINA_HOME=$(pwd)
cd ../
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
printf " 📦 Configuring MySQL:\n"
printf "      Please enter 'root' as root's password and 'hugo' as hugo
password\n"
sudo mysql_secure_installation
sudo mysql < sudo.sql
printf "      Enter root's password: (root)\n"
mysql -u root -p < root.sql
printf "      Enter hugo's password: (hugo)\n"
mysql -u hugo -p < db.sql
printf " 📦 Configured MySQL\n"

start_spinner 'Deploying app'
unzip -qq Servicio.zip
rm Servicio.zip
cd Servicio
javac -cp
$CATALINA_HOME/lib/javax.ws.rs-api-2.0.1.jar:$CATALINA_HOME/lib/gson-2.3.1.jar:.
negocio/Servicio.java
rm -f WEB-INF/classes/negocio/*
cp negocio/*.class WEB-INF/classes/negocio/
jar cvf Servicio.war WEB-INF META-INF > /dev/null
rm -f $CATALINA_HOME/webapps/Servicio.war
rm -f -r $CATALINA_HOME/webapps/Servicio
cp Servicio.war $CATALINA_HOME/webapps
cd ../
mv usuario_sin_foto.png $CATALINA_HOME/webapps/ROOT/
mv WSClient.js $CATALINA_HOME/webapps/ROOT/
mv prueba.html $CATALINA_HOME/webapps/ROOT/
sh $CATALINA_HOME/bin/catalina.sh start > /dev/null
stop_spinner $? '🐱'
printf "\n  Tomcat server is running at port 8080. Stop with
${CATALINA_HOME}/bin/catalina.sh stop\n"

```

spinner.sh

```
# Pretty output
# Author: Tasos Latsas
# Modified: Joel Hernández

function _spinner() {
    # $1 start/stop
    #
    # on start: $2 display message
    # on stop : $2 process exit status
    #          $3 spinner function pid (supplied from stop_spinner)

    local on_success="DONE"
    local on_fail="FAIL"
    local white="\e[1;37m"
    local green="\e[1;32m"
    local red="\e[1;31m"
    local nc="\e[0m"

    case $1 in
        start)
            printf "    #  ${2}"
            # start spinner
            i=1
            sp='#####'
            delay=${SPINNER_DELAY:-0.05}

            while :
            do
                j=1
                while [ $j -le $(( ${#2} + 3 )) ]
                do
                    ((j++))
                    printf "\b"
                done

                printf "${sp:i++:${#sp}:1}  "
                echo -ne ${2}
                sleep $delay
            done
            ;;
        stop)
            if [[ -z ${3} ]]; then
                echo "spinner is not running.."
                exit 1
            fi
    esac
}
```

```

        kill $3 > /dev/null 2>&1

        let column=$(tput cols)
        j=1
        while [ $j -le $column ]
        do
            ((j++))
            printf "\b"
        done

        # inform the user upon success or failure
        echo -en "    "
        if [[ $2 -eq 0 ]]; then
            echo -en "✓"
        else
            echo -en "✗"
        fi
        echo -e " ${4}"
        ;;
    *)
        echo "invalid argument, try {start/stop}"
        exit 1
        ;;
esac
}

message=""

function start_spinner {
    # $1 : msg to display
    message=`echo ${1}`
    _spinner "start" "${1}" &
    # set global spinner pid
    _sp_pid=$!
    disown
}

function stop_spinner {
    # $1 : command exit status
    _spinner "stop" $1 $_sp_pid "$message" "${2}"
    unset _sp_pid
}

printf "\n"

```


db.sql

```
create database servicio_web;
use servicio_web;
create table usuarios
(
    id_usuario integer auto_increment primary key,
    email varchar(256) not null,
    nombre varchar(100) not null,
    apellido_paterno varchar(100) not null,
    apellido_materno varchar(100),
    fecha_nacimiento date not null,
    telefono varchar(20),
    genero char(1)
);
create table fotos_usuarios
(
    id_foto integer auto_increment primary key,
    foto longblob,
    id_usuario integer not null
);
alter table fotos_usuarios add foreign key (id_usuario) references
usuarios(id_usuario);
create unique index usuarios_1 on usuarios(email);
quit
```

root.sql

```
create user hugo@localhost identified by 'hugo';
grant all on servicio_web.* to hugo@localhost;
quit
```

sudo.sql

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'root';
FLUSH PRIVILEGES;
quit
```