



INSTITUTO POLITÉCNICO NACIONAL  
ESCUELA SUPERIOR DE CÓMPUTO

Tarea 7. Desarrollo de un cliente para un servicio web estilo  
REST

Unidad de aprendizaje: Desarrollo de Sistemas Distribuidos

Grupo: 4CV11

*Alumno:*  
Sanchez Mendez Edmundo Josue

*Profesor:*  
Pineda Guerrero Carlos

# Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Desarrollo</b>	<b>3</b>
2.1. Creación de la maquina virtual . . . . .	3
2.2. Cambios hechos en Servicio.java y Usuario.java . . . . .	8
2.3. Preparación del entorno tanto en servidor como en el cliente . . . . .	12
2.4. Pruebas . . . . .	15
2.4.1. Prueba 1 . . . . .	15
2.4.2. Prueba 2, 3 y 4 . . . . .	16
2.4.3. Prueba 5 . . . . .	18
2.4.4. Prueba 6 . . . . .	18
2.4.5. Finalizar programa cliente . . . . .	19
2.5. Creación de una imagen de la maquina virtual conservando el usuario . . . . .	20
<b>3. Conclusiones</b>	<b>22</b>

## 1. Introducción

Desarrollar un programa Java consola (modo carácter) cliente del servicio web REST que se implemento en la tarea anterior. Se deberá realizar las siguientes modificaciones al servicio web:

- Agregar el campo `id_usuario` a la clase `Usuario` (`Usuario.java`).
- Modificar el método web “alta\_usuario” (`Servicio.java`), de manera que al dar de alta un usuario el método web deberá regresar al cliente el id del usuario agregado. Se deberá desplegar el id del usuario dado de alta. El campo `id_usuario` es auto\_increment en la base de datos, por tanto se deberá recuperar el ID inmediatamente después de ejecutar la instrucción `INSERT`.
- Modificar el método web “consulta\_usuario” (`Servicio.java`), ahora la consulta se deberá realizar mediante el id del usuario no el email.
- Modificar el método web “modifica\_usuario” (`Servicio.java`), utilizar el id del usuario en el `WHERE` de las instrucciones SQL en lugar del email. No deberá modificar el id de un usuario ya que se trata de la llave primaria.
- Modificar el método web “borra\_usuario” (`Servicio.java`), utilizando como clave el id del usuario no el email.

El programa cliente deberá desplegar el siguiente menú:

MENU

- Alta usuario
- Consulta usuario
- Borra usuario
- Salir

Opción: \_

Las opciones deberán implementar la siguiente funcionalidad:

- La opción “Alta usuario” leerá del teclado el email, el nombre del usuario, el apellido paterno, el apellido materno, la fecha de nacimiento, el teléfono y el género (“M” o “F”). Entonces se invocará el método web “alta\_usuario”. Se deberá desplegar el id del usuario dado de alta, o bien, el mensaje de error que regresa el servicio web. Notar que el método web “alta\_usuario” recibe como parámetro una instancia de la clase `Usuario`, recordemos que esta clase se deberá definir de la siguiente manera:

```
class Usuario
{
    int id_usuario;
    String email;
    String nombre;
    String apellido_paterno;
    String apellido_materno;
    String fecha_nacimiento;
    String telefono;
    String genero;
    byte[] foto;
}
```

Para invocar el método web “alta\_usuario” desde el cliente Java es necesario crear una instancia de la clase Usuario y asignar los valores a los campos (en este caso el campo “foto” será null). Una vez que se tenga el objeto de tipo Usuario se deberá utilizar GSON para convertir el objeto a una string JSON, entonces se deberá codificar como URL y el resultado se utilizará como valor del parámetro.

- La opción “Consulta usuario” leerá del teclado el id de un usuario previamente dado de alta. Entonces se invocará el método web “consulta\_usuario”. Si el usuario existe, se desplegará en pantalla el nombre del usuario, el apellido paterno, el apellido materno, la fecha de nacimiento, el teléfono y el género. Debido a que el programa no es gráfico, la foto del usuario se ignorará. Notar que el método web “consulta\_usuario” regresa una string JSON la cual representa un objeto de tipo Usuario, por tanto será necesario utilizar GSON para convertir la string JSON a un objeto Java de tipo Usuario y posteriormente desplegar los campos del objeto (excepto el campo “foto”). Si hubo error, se desplegará el mensaje que regresa el servicio web. Una vez desplegados los datos del usuario se preguntará “¿Desea modificar los datos del usuario (s/n)¿”, si se responde con un caracter “s” entonces se leerá del teclado el email, el nombre del usuario, el apellido paterno, el apellido materno, la fecha de nacimiento, el teléfono y el género (“M” o “F”), si un campo se presiona solo Enter, entonces el campo queda sin modificar. Entonces se invocará el método web “modifica\_usuario”. Este método recibe como parámetro un objeto de tipo Usuario, por tanto se deberá utilizar GSON para convertir el objeto a una string JSON, entonces se deberá codificar como URL y el resultado se utilizará como valor del parámetro. Se deberá desplegar “El usuario ha sido modificado” si se pudo modificar el usuario, o bien, el mensaje de error que regresa el servicio web.
- La opción “Borra usuario” leerá del teclado el id de un usuario previamente dado de alta. Entonces se invocará el método “borra\_usuario” del servicio web. Se deberá desplegar “El usuario ha sido borrado” si se pudo borrar el usuario, o bien, el mensaje de error que regresa el servicio web.
- La opción “Salir” terminará el programa.

Se deberá realizar las siguientes pruebas:

- Dar de alta un nuevo usuario.
- Consultar el usuario dado de alta anteriormente.
- Modificar algún dato del usuario.
- Consultar el usuario modificado, para verificar que la modificación se realizó.
- Intentar borrar un usuario que no exista, se deberá desplegar el mensaje de error indicando que el id no existe.
- Borrar el usuario dado de alta en el paso 1.

## 2. Desarrollo

### 2.1. Creación de la maquina virtual

En esta parte veremos la creación de la maquina virtual en donde llevaremos acabo la practica, ademas en el punto final crearemos una imagen de la maquina virtual para usarla en practicas futuras. Recordar que en la practica anterior creamos una imagen de la maquina virtual utilizada, por lo que usaremos esa imagen para poder crear la maquina virtual que usaremos para esta practica.

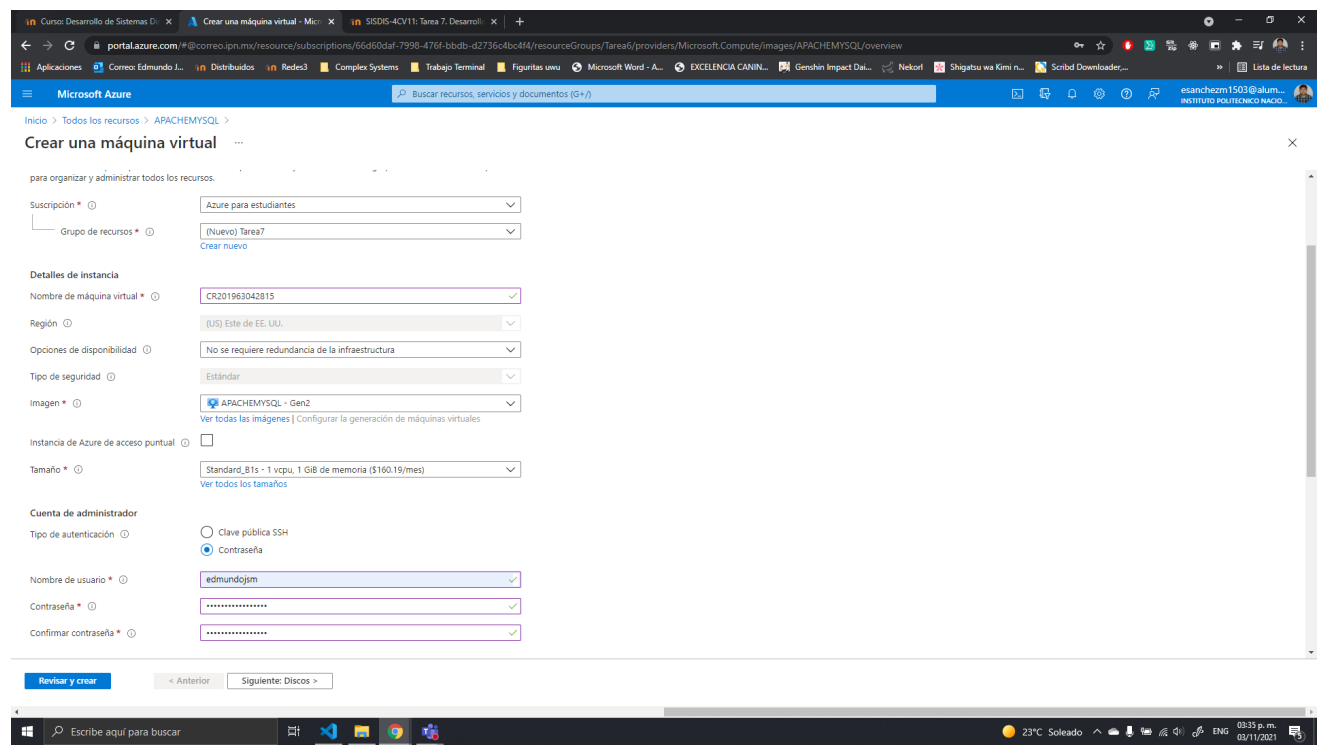


Figura 1: Datos básicos de la maquina virtual.

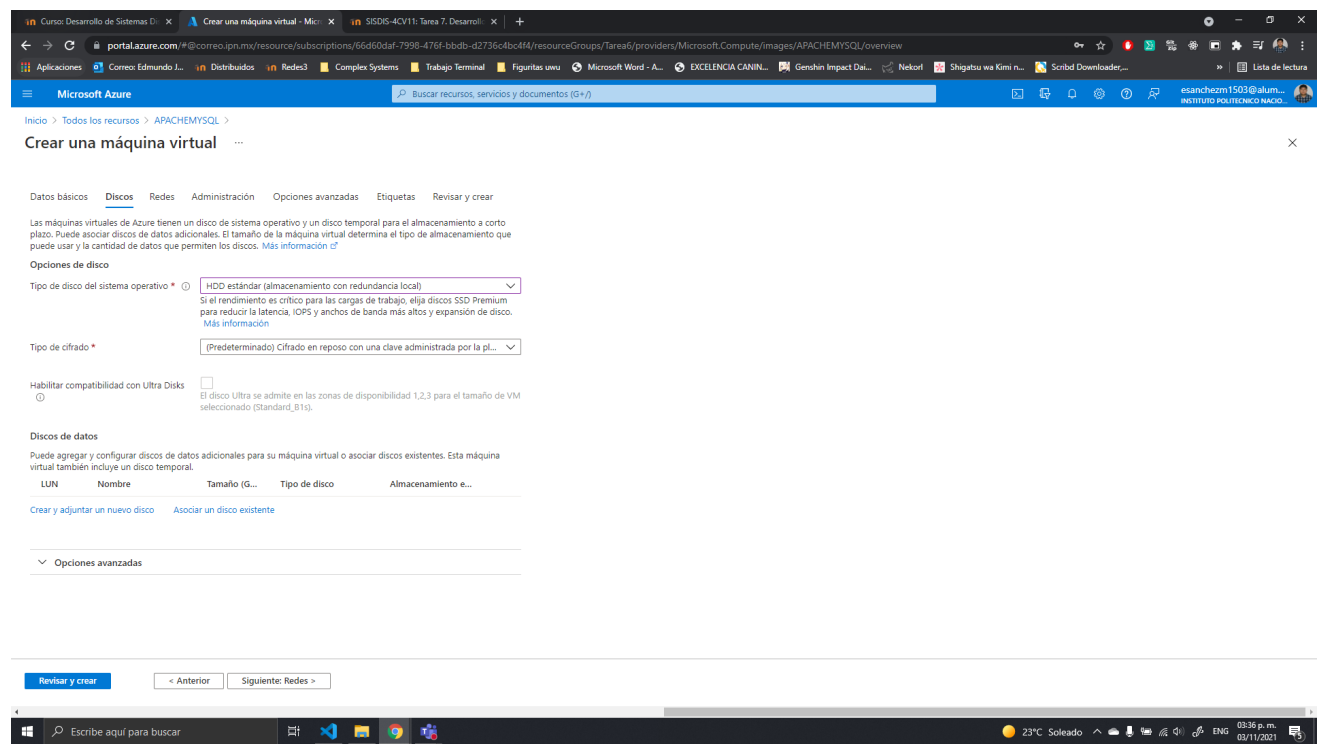


Figura 2: Configuración del tipo de disco de la maquina virtual.

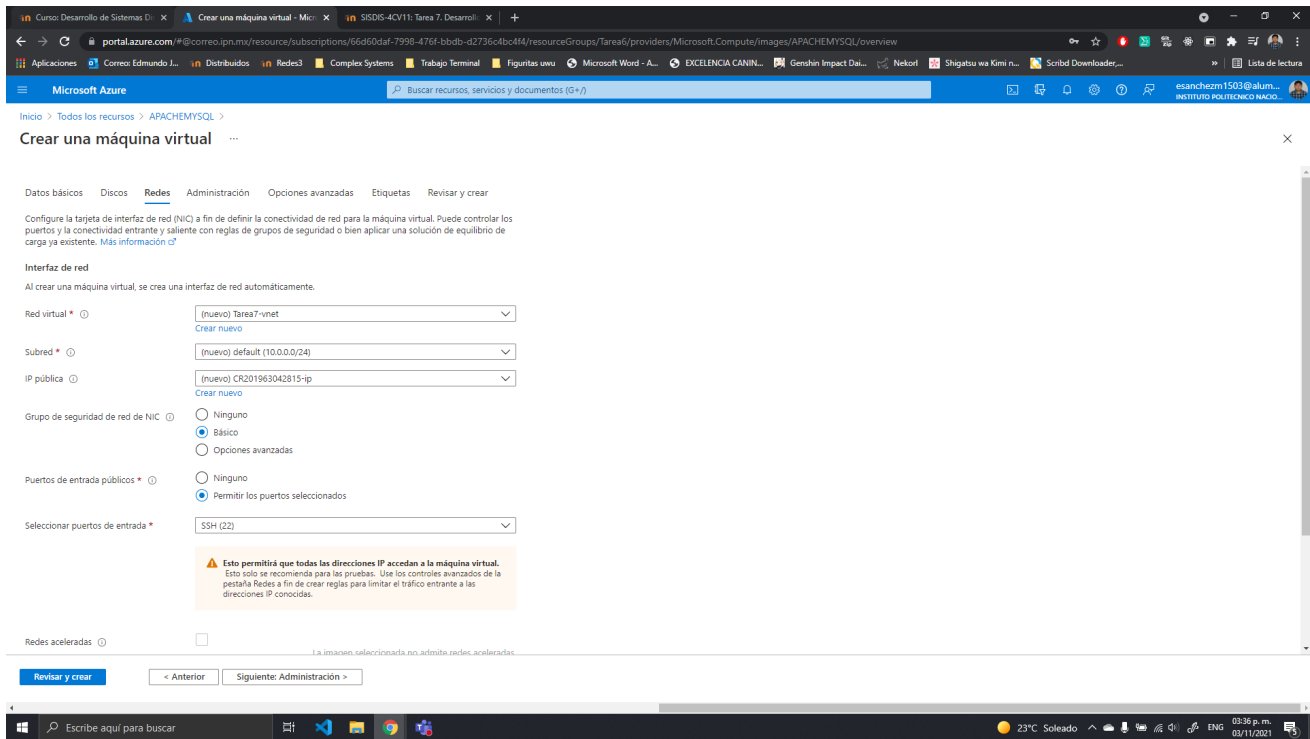


Figura 3: Información sobre la redes de la maquina virtual.

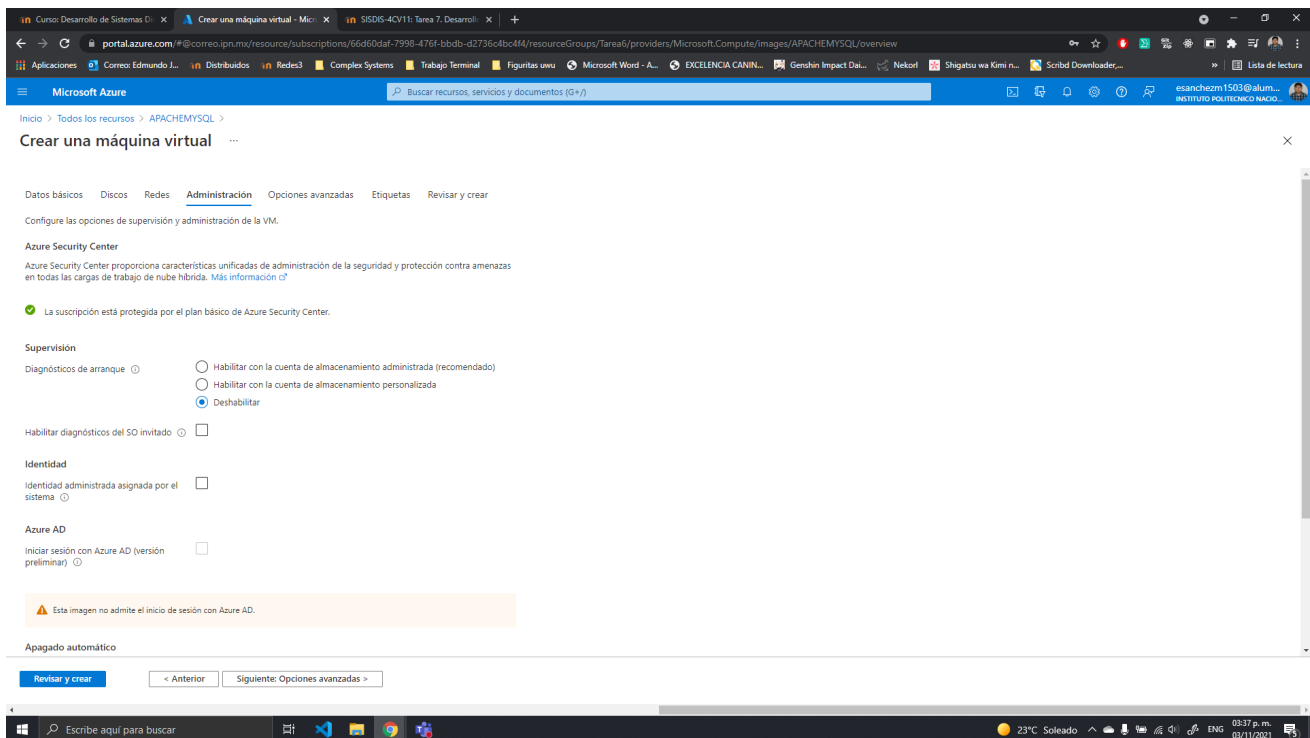


Figura 4: Configuración de la administración de la maquina virtual.

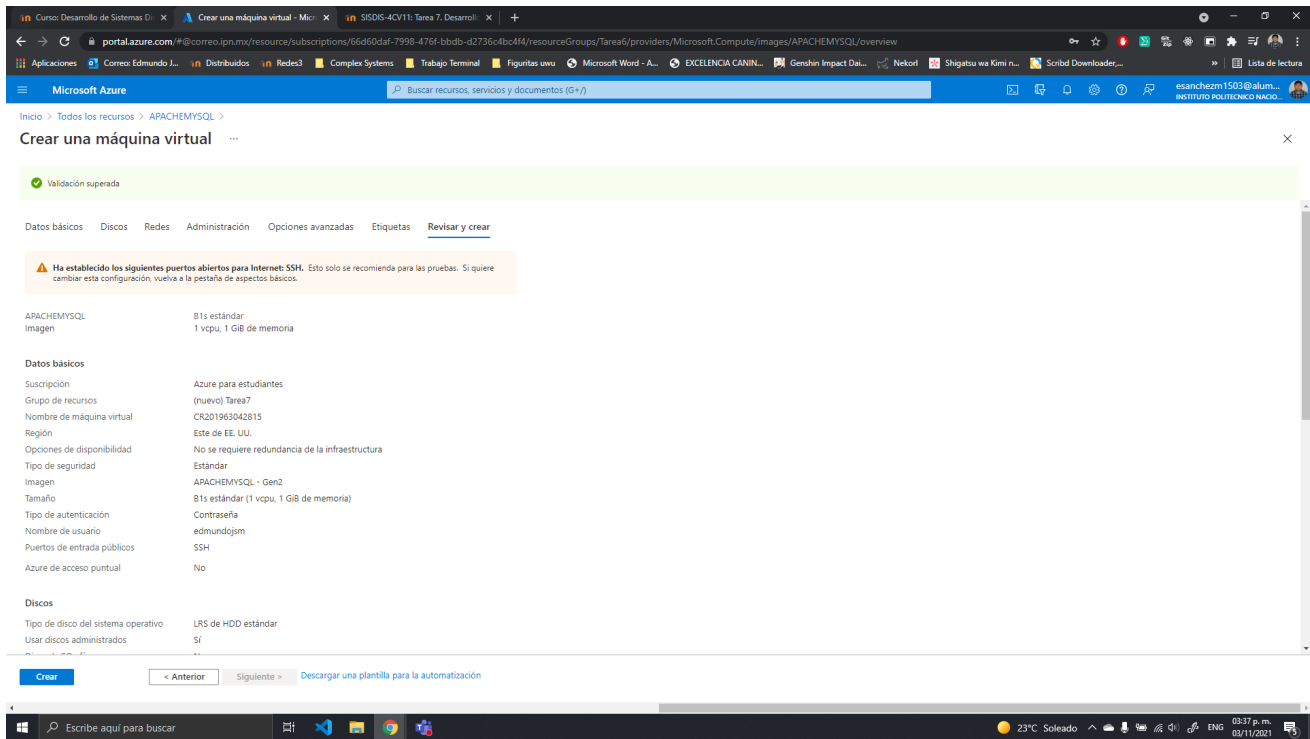


Figura 5: Creación de la maquina virtual.

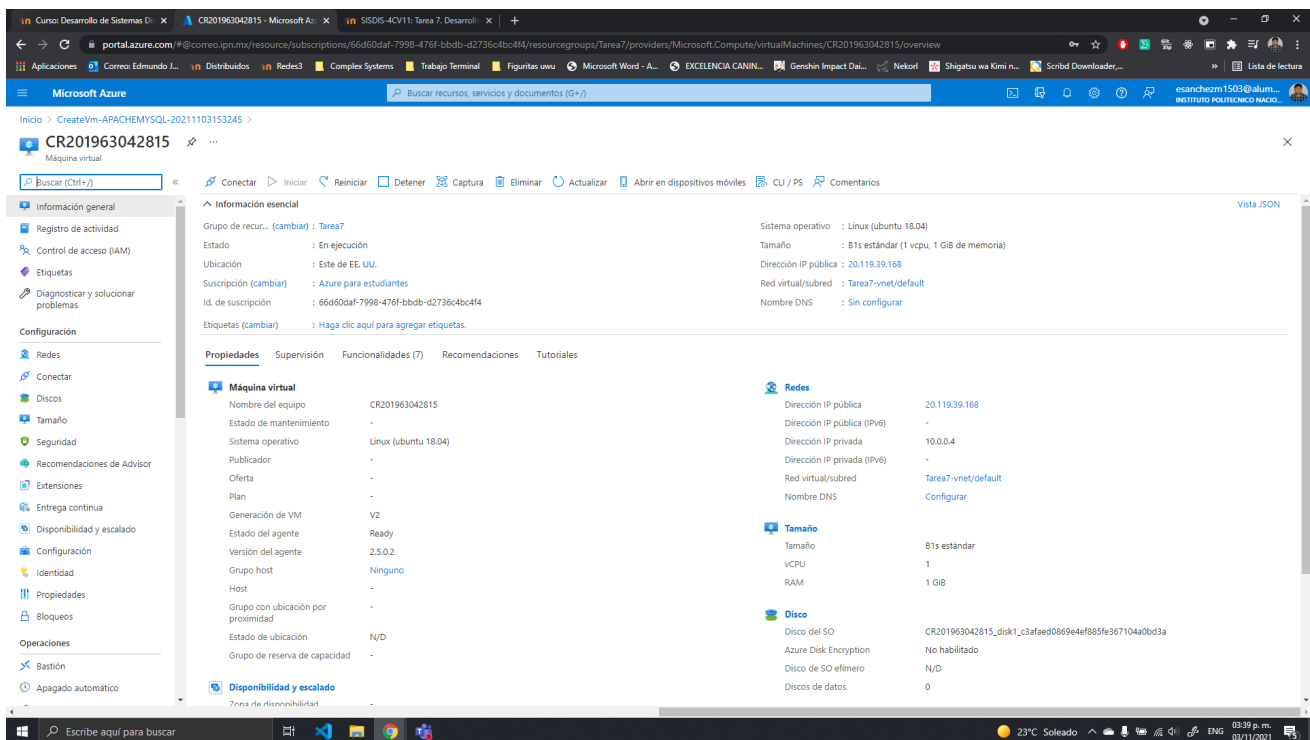


Figura 6: Panel de control de la maquina virtual.

Una vez creada la maquina virtual tenemos que abrir el puerto 8080, ya que este puerto es el que ocupa Apache Tomcat, así podemos conectarnos de manera remota desde cualquier dispositivo , también es importante mencionar que si no se configura de manera correcta el puerto 8080 la conexión con la

maquina virtual para poder visualizar el servicio web. En las figuras 7 y 8 podemos ver la configuración del puerto.

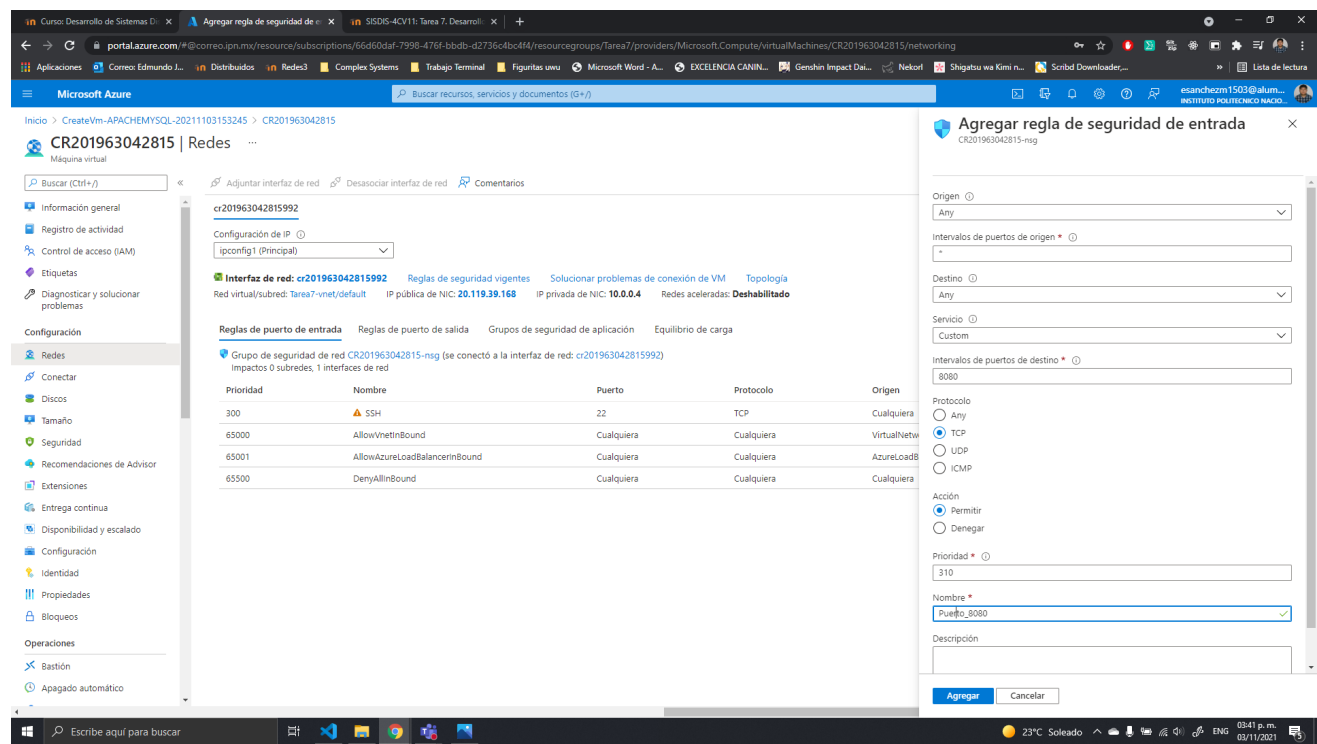


Figura 7: Configuración del puerto 8080.

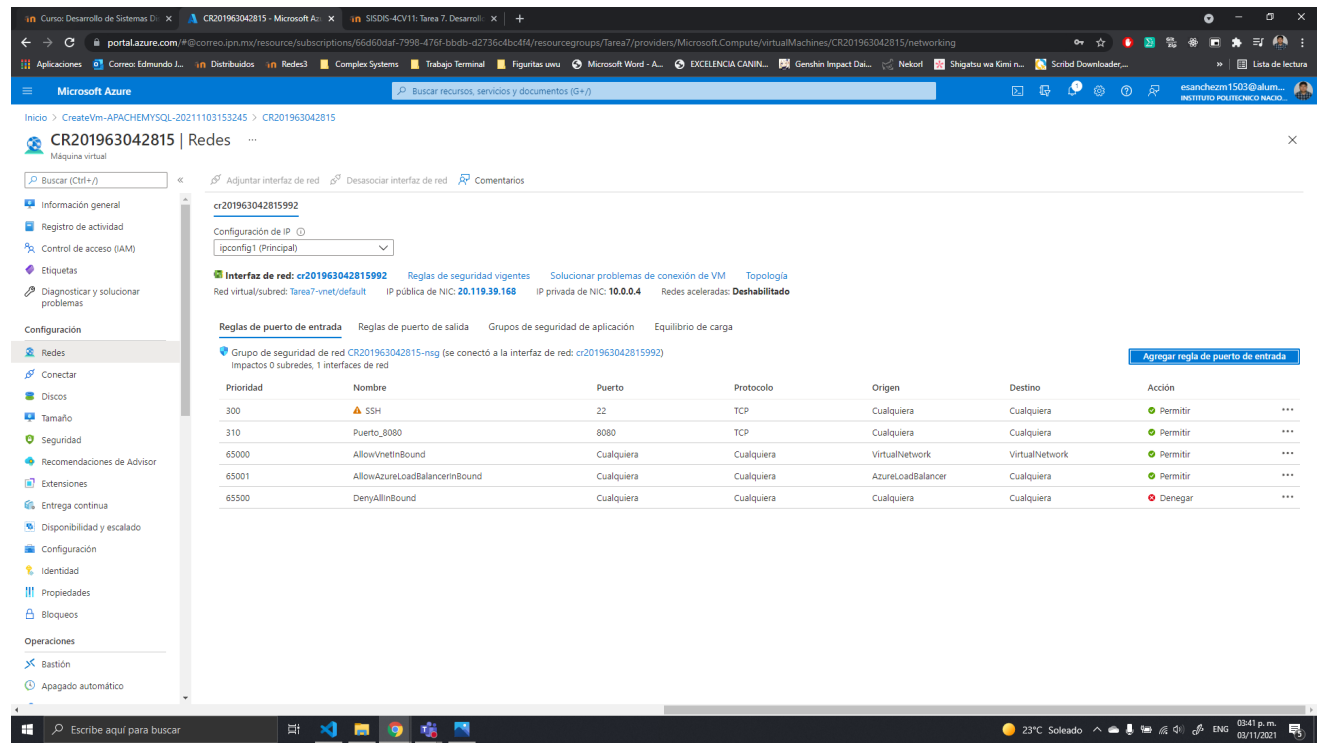


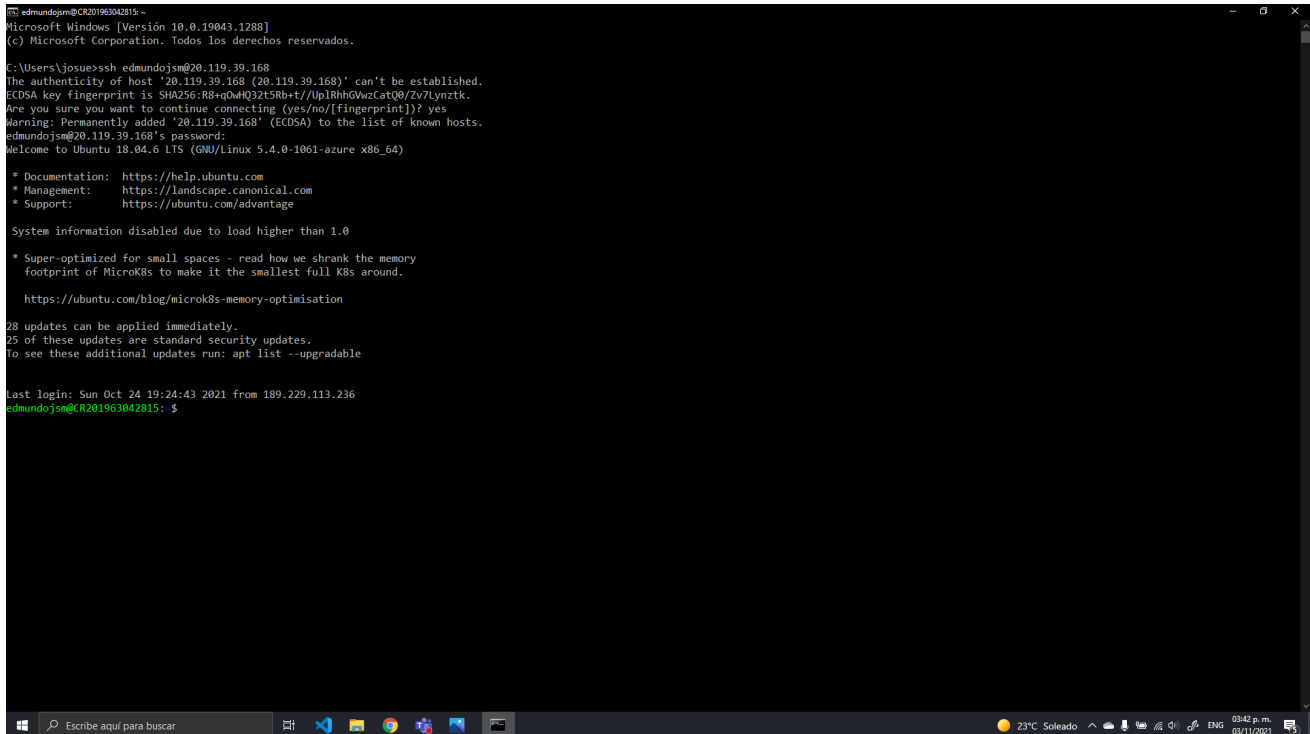
Figura 8: Puerto 8080 abierto correctamente.



Ya teniendo la maquina preparada, procederemos a realizar los cambios en Servicio.java para poder cumplir con los requerimientos de la practica.

## 2.2. Cambios hechos en Servicio.java y Usuario.java

Primero que nada nos tendremos que conectar por medio de ssh a la maquina virtual como vemos en la figura 9.



```
edmundojm@CR201963042815 -  
Microsoft Windows [Versi3n 10.0.19043.1288]  
(c) Microsoft Corporation. Todos los derechos reservados.  
  
C:\Users\josue>ssh edmundojm@20.119.39.168  
The authenticity of host '20.119.39.168 (20.119.39.168)' can't be established.  
ECDSA key fingerprint is SHA256:R8+qDwHQ32t5Rb+t//Up1RhhGWzCatQ0/Zv7LynztK.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '20.119.39.168' (ECDSA) to the list of known hosts.  
edmundojm@20.119.39.168's password:  
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1061-azure x86_64)  
  
 * Documentation:  https://help.ubuntu.com  
 * Management:    https://landscape.canonical.com  
 * Support:        https://ubuntu.com/advantage  
  
System information disabled due to load higher than 1.0  
  
 * Super-optimized for small spaces - read how we shrank the memory  
   footprint of MicroK8s to make it the smallest full K8s around.  
   https://ubuntu.com/blog/microk8s-memory-optimisation  
  
28 updates can be applied immediately.  
25 of these updates are standard security updates.  
To see these additional updates run: apt list --upgradable  
  
Last login: Sun Oct 24 19:24:43 2021 from 189.229.113.236  
edmundojm@CR201963042815:~$
```

Figura 9: Conexi3n ssh con la maquina virtual.

Posteriormente realizaremos los cambios correspondientes, en primer lugar vamos a agregar el campo id\_usuario a la clase Usuario.java como se ve en la figura 10.

```

GNU nano 2.9.3                                Usuario.java
Carlos Pineda Guerrero, Octubre 2021

package negocio;

import com.google.gson.*;

public class Usuario
{
    int id_usuario;
    String email;
    String nombre;
    String apellido_paterno;
    String apellido_materno;
    String fecha_nacimiento;
    String telefono;
    String genero;
    byte[] foto;

    // @Param necesita un metodo que convierta una String al objeto de tipo Usuario
    public static Usuario valueOf(String s) throws Exception
    {
        Gson j = new GsonBuilder().registerTypeAdapter(byte[].class, new AdaptadorGsonBase64()).create();
        return (Usuario)j.fromJson(s, Usuario.class);
    }
}

```

Figura 10: Campo id\_usuario agregado a la clase Usuario.java.

Ahora en las figuras 11 y 12 podemos ver las modificaciones necesarias para cumplir el punto 2 de las modificaciones solicitadas.

```

GNU nano 2.9.3                                Servicio.java
Modified

stmt_1.setString(1, usuario.email);
ResultSet rs = stmt_1.executeQuery();
try {
    if (rs.next()) {
        return Response.status(400).entity(j.toJson(new Error("El email ya existe"))).build();
    }
} finally {
    rs.close();
}
} finally {
    stmt_1.close();
}

PreparedStatement stmt_2 = conexion.prepareStatement("INSERT INTO usuarios VALUES (0,?,?,?,?,?,?)", Statement.RETURN_GENERATED_KEYS);
ResultSet keys = null;
try {
    stmt_2.setString(1, usuario.email);
    stmt_2.setString(2, usuario.nombre);
    stmt_2.setString(3, usuario.apellido_paterno);
    stmt_2.setString(4, usuario.apellido_materno);
    stmt_2.setString(5, usuario.fecha_nacimiento);
    stmt_2.setString(6, usuario.telefono);
    stmt_2.setString(7, usuario.genero);
    stmt_2.executeUpdate();
    keys = stmt_2.getGeneratedKeys();
    keys.next();
    int id = keys.getInt(1);
    usuario.id_usuario = id;
} finally {
    stmt_2.close();
    keys.close();
}

if (usuario.foto != null) {
    PreparedStatement stmt_3 = conexion.prepareStatement(
        "INSERT INTO fotos_usuarios VALUES (0,?,(SELECT id_usuario FROM usuarios WHERE id_usuario=?))");
    try {
        stmt_3.setBytes(1, usuario.foto);
        stmt_3.setInt(2, usuario.id_usuario);
        stmt_3.executeUpdate();
    } finally {
        stmt_3.close();
    }
}

```

Figura 11: Cambios para cumplir el punto 2 de las modificaciones solicitadas. Parte 1.

```

stmt_2.executeUpdate();
keys = stmt_2.getGeneratedKeys();
keys.next();
int id = keys.getInt(1);
usuario.id_usuario = id;
} finally {
stmt_2.close();
keys.close();
}

if (usuario.foto != null) {
PreparedStatement stmt_3 = conexion.prepareStatement(
    "INSERT INTO fotos_usuarios VALUES (0,?,(SELECT id_usuario FROM usuarios WHERE id_usuario=?))");
try {
stmt_3.setBytes(1, usuario.foto);
stmt_3.setInt(2, usuario.id_usuario);
stmt_3.executeUpdate();
} finally {
stmt_3.close();
}
}
} catch (Exception e) {
return Response.status(400).entity(j.toJson(new Error(e.getMessage()))).build();
} finally {
conexion.close();
}
return Response.ok().entity(j.toJson(usuario.id_usuario)).build();
}

@POST
@Path("/consulta_usuario")
@Consumes(MediaType.APPLICATION_FORM_URLENCODED)
@Produces(MediaType.APPLICATION_JSON)
public Response consulta(@FormParam("id_usuario") int id_usuario) throws Exception {
Connection conexion = pool.getConnection();

try {
PreparedStatement stmt_1 = conexion.prepareStatement(
    "SELECT a.id_usuario,a.email,a.nombre,a.apellido_paterno,a.apellido_materno,a.fecha_nacimiento,a.telefono,a.genero,b.foto FROM usuarios a LEFT OUTER JOIN fotos_usuarios b ON a.id_usuario=b.id_usuario $");
stmt_1.setInt(1, id_usuario);
ResultSet rs = stmt_1.executeQuery();

try {
if (rs.next()) {
Usuario r = new Usuario();
r.id_usuario = rs.getInt(1);
r.email = rs.getString(2);
r.nombre = rs.getString(3);
r.apellido_paterno = rs.getString(4);
r.apellido_materno = rs.getString(5);
r.fecha_nacimiento = rs.getString(6);
r.telefono = rs.getString(7);
r.genero = rs.getString(8);
r.foto = rs.getBytes(9);
return Response.ok().entity(j.toJson(r)).build();
}
return Response.status(400).entity(j.toJson(new Error("No existe un usuario con ID " + id_usuario))).build();
} finally {
rs.close();
}
} finally {
stmt_1.close();
}
} catch (Exception e) {
return Response.status(400).entity(j.toJson(new Error(e.getMessage()))).build();
}
}

```

Figura 12: Cambios para cumplir el punto 2 de las modificaciones solicitadas. Parte 2.

Después llevamos acabo lo solicitado en el punto 3, esto lo podemos ver en la figura 12 al final cuando ponemos como parámetro el id del usuario y en la figura 13.

```

return Response.status(400).entity(j.toJson(new Error(e.getMessage()))).build();
} finally {
conexion.close();
}
return Response.ok().entity(j.toJson(usuario.id_usuario)).build();
}

@POST
@Path("/consulta_usuario")
@Consumes(MediaType.APPLICATION_FORM_URLENCODED)
@Produces(MediaType.APPLICATION_JSON)
public Response consulta(@FormParam("id_usuario") int id_usuario) throws Exception {
Connection conexion = pool.getConnection();

try {
PreparedStatement stmt_1 = conexion.prepareStatement(
    "SELECT a.id_usuario,a.email,a.nombre,a.apellido_paterno,a.apellido_materno,a.fecha_nacimiento,a.telefono,a.genero,b.foto FROM usuarios a LEFT OUTER JOIN fotos_usuarios b ON a.id_usuario=b.id_usuario $");
stmt_1.setInt(1, id_usuario);
ResultSet rs = stmt_1.executeQuery();

try {
if (rs.next()) {
Usuario r = new Usuario();
r.id_usuario = rs.getInt(1);
r.email = rs.getString(2);
r.nombre = rs.getString(3);
r.apellido_paterno = rs.getString(4);
r.apellido_materno = rs.getString(5);
r.fecha_nacimiento = rs.getString(6);
r.telefono = rs.getString(7);
r.genero = rs.getString(8);
r.foto = rs.getBytes(9);
return Response.ok().entity(j.toJson(r)).build();
}
return Response.status(400).entity(j.toJson(new Error("No existe un usuario con ID " + id_usuario))).build();
} finally {
rs.close();
}
} finally {
stmt_1.close();
}
} catch (Exception e) {
return Response.status(400).entity(j.toJson(new Error(e.getMessage()))).build();
}
}

```

Figura 13: Cambios para cumplir el punto 3 de las modificaciones solicitadas.

Después pasamos a cumplir el punto 4 como vemos en las figuras 14 y 15.

```

conexion.close();
}
}

@POST
@Path("/modifica_usuario")
@Consumes(MediaType.APPLICATION_FORM_URLENCODED)
@Produces(MediaType.APPLICATION_JSON)
public Response modifica(@FormParam("usuario") Usuario usuario) throws Exception {
    Connection conexion = pool.getConnection();

    if (usuario.email == null || usuario.email.equals(""))
        return Response.status(400).entity(j.toJson(new Error("Se debe ingresar el email"))).build();

    if (usuario.nombre == null || usuario.nombre.equals(""))
        return Response.status(400).entity(j.toJson(new Error("Se debe ingresar el nombre"))).build();

    if (usuario.apellido_paterno == null || usuario.apellido_paterno.equals(""))
        return Response.status(400).entity(j.toJson(new Error("Se debe ingresar el apellido paterno"))).build();

    if (usuario.fecha_nacimiento == null || usuario.fecha_nacimiento.equals(""))
        return Response.status(400).entity(j.toJson(new Error("Se debe ingresar la fecha de nacimiento"))).build();

    try {
        PreparedStatement stmt_1 = conexion.prepareStatement(
            "UPDATE usuarios SET nombre=?,apellido_paterno=?,apellido_materno=?,fecha_nacimiento=?,telefono=?,genero=? WHERE id_usuario=?");
        try {
            stmt_1.setString(1, usuario.nombre);
            stmt_1.setString(2, usuario.apellido_paterno);
            stmt_1.setString(3, usuario.apellido_materno);
            stmt_1.setString(4, usuario.fecha_nacimiento);
            stmt_1.setString(5, usuario.telefono);
            stmt_1.setString(6, usuario.genero);
            stmt_1.setInt(7, usuario.id_usuario);
            stmt_1.executeUpdate();
        } finally {
            stmt_1.close();
        }

        if (usuario.foto != null) {
            PreparedStatement stmt_2 = conexion.prepareStatement(
                "DELETE FROM fotos_usuarios WHERE id_usuario=(SELECT id_usuario FROM usuarios WHERE id_usuario=?)");
            try {
                stmt_2.setInt(1, usuario.id_usuario);
                stmt_2.executeUpdate();
            }
        }
    }
}

```

Figura 14: Cambios para cumplir el punto 4 de las modificaciones solicitadas. Parte 1.

```

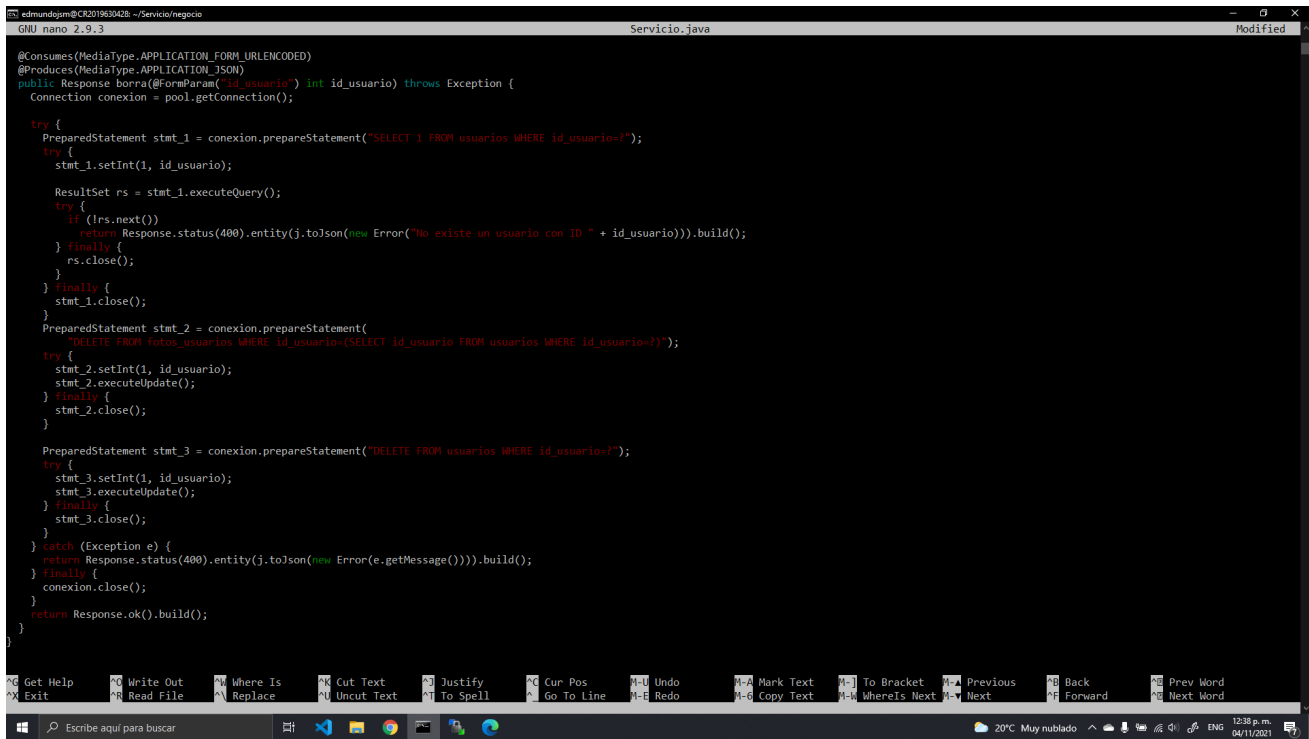
        PreparedStatement stmt_3 = conexion.prepareStatement(
            "INSERT INTO fotos_usuarios VALUES (0,?,(SELECT id_usuario FROM usuarios WHERE id_usuario=?))");
        try {
            stmt_3.setBytes(1, usuario.foto);
            stmt_3.setInt(2, usuario.id_usuario);
            stmt_3.executeUpdate();
        } finally {
            stmt_3.close();
        }
    } catch (Exception e) {
        return Response.status(400).entity(j.toJson(new Error(e.getMessage()))).build();
    } finally {
        conexion.close();
    }
    return Response.ok().build();
}

@POST

```

Figura 15: Cambios para cumplir el punto 4 de las modificaciones solicitadas. Parte 2.

Finalmente en la figura 16 podemos ver la modificación que se realizo para cumplir la ultima solicitada.



```
edmundojm@CR019630428: ~/Servicio/negocio
GNU nano 2.9.3 Servicio.java

@Consumes(MediaType.APPLICATION_FORM_URLENCODED)
@Produces(MediaType.APPLICATION_JSON)
public Response borra(@FormParam("id_usuario") int id_usuario) throws Exception {
    Connection conexion = pool.getConnection();

    try {
        PreparedStatement stmt_1 = conexion.prepareStatement("SELECT 1 FROM usuarios WHERE id_usuario=?");
        try {
            stmt_1.setInt(1, id_usuario);

            ResultSet rs = stmt_1.executeQuery();
            try {
                if (rs.next()) {
                    return Response.status(400).entity(j.toJson(new Error("No existe un usuario con ID " + id_usuario))).build();
                }
            } finally {
                rs.close();
            }
        } finally {
            stmt_1.close();
        }
        PreparedStatement stmt_2 = conexion.prepareStatement(
            "DELETE FROM fotos_usuarios WHERE id_usuario=(SELECT id_usuario FROM usuarios WHERE id_usuario=?)");
        try {
            stmt_2.setInt(1, id_usuario);
            stmt_2.executeUpdate();
        } finally {
            stmt_2.close();
        }

        PreparedStatement stmt_3 = conexion.prepareStatement("DELETE FROM usuarios WHERE id_usuario=?");
        try {
            stmt_3.setInt(1, id_usuario);
            stmt_3.executeUpdate();
        } finally {
            stmt_3.close();
        }
    } catch (Exception e) {
        return Response.status(400).entity(j.toJson(new Error(e.getMessage()))).build();
    } finally {
        conexion.close();
    }
    return Response.ok().build();
}
```

Figura 16: Cambios para cumplir el punto 4 de las modificaciones solicitadas. Parte 2.

## 2.3. Preparación del entorno tanto en servidor como en el cliente

Para empezar debemos de borrar el anterior Servicio de webapps ya que tenemos una actualización, esto lo podemos ver en la figura 17, después nos toca volver a configurar las variables de entorno de java jdk y CATALINA\_HOME y por supuesto correr Tomcat, esto se puede ver en la figura 18.

```

edmundojm@CR2019630428: ~/apache-tomcat-8.5.72/webapps
edmundojm@CR2019630428:~/apache-tomcat-8.5.72$ cd webapps/
edmundojm@CR2019630428:~/apache-tomcat-8.5.72/webapps$ ls
ROOT Servicio Servicio.war
edmundojm@CR2019630428:~/apache-tomcat-8.5.72/webapps$ rm Servicio.war
edmundojm@CR2019630428:~/apache-tomcat-8.5.72/webapps$ ls
ROOT Servicio
edmundojm@CR2019630428:~/apache-tomcat-8.5.72/webapps$ rm -r Servicio/
edmundojm@CR2019630428:~/apache-tomcat-8.5.72/webapps$ ls
^C
edmundojm@CR2019630428:~/apache-tomcat-8.5.72/webapps$ ls
ROOT
edmundojm@CR2019630428:~/apache-tomcat-8.5.72/webapps$

```

Figura 17: Carpeta Servicio y Servicio.war eliminado de webapps .

```

edmundojm@CR2019630428: -
edmundojm@CR2019630428:~/apache-tomcat-8.5.72$ cd
edmundojm@CR2019630428:~/apache-tomcat-8.5.72$ cd webapps/
edmundojm@CR2019630428:~/apache-tomcat-8.5.72/webapps$ ls
ROOT Servicio Servicio.war
edmundojm@CR2019630428:~/apache-tomcat-8.5.72/webapps$ rm Servicio.war
edmundojm@CR2019630428:~/apache-tomcat-8.5.72/webapps$ ls
ROOT Servicio
edmundojm@CR2019630428:~/apache-tomcat-8.5.72/webapps$ rm -r Servicio/
edmundojm@CR2019630428:~/apache-tomcat-8.5.72/webapps$ ls
^C
edmundojm@CR2019630428:~/apache-tomcat-8.5.72/webapps$ ls
ROOT
edmundojm@CR2019630428:~/apache-tomcat-8.5.72/webapps$ cd
edmundojm@CR2019630428:~/apache-tomcat-8.5.72$ cd
edmundojm@CR2019630428:~/apache-tomcat-8.5.72$ pwd
/home/edmundojm/apache-tomcat-8.5.72
edmundojm@CR2019630428:~/apache-tomcat-8.5.72$ cd
edmundojm@CR2019630428:~/apache-tomcat-8.5.72$ export CATALINA_HOME=/home/edmundojm/apache-tomcat-8.5.72
edmundojm@CR2019630428:~/apache-tomcat-8.5.72$ export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
edmundojm@CR2019630428:~/apache-tomcat-8.5.72$ sh $CATALINA_HOME/bin/catalina.sh start
Using CATALINA_BASE:   /home/edmundojm/apache-tomcat-8.5.72
Using CATALINA_HOME:   /home/edmundojm/apache-tomcat-8.5.72
Using CATALINA_TMPDIR: /home/edmundojm/apache-tomcat-8.5.72/temp
Using JRE_HOME:        /usr/lib/jvm/java-8-openjdk-amd64
Using CLASSPATH:       /home/edmundojm/apache-tomcat-8.5.72/bin/bootstrap.jar:/home/edmundojm/apache-tomcat-8.5.72/bin/tomcat-juli.jar
Tomcat started.
edmundojm@CR2019630428:~/apache-tomcat-8.5.72$

```

Figura 18: Variables de entorno asignadas y Tomcat corriendo.

Después nos toca compilar el servicio, generar el .war y copiarlo a la carpeta webapps para poder tener el servicio en línea usando Tomcat, todo esto se ve en la figura 19, finalmente abrimos la línea de comando de mysql con el usuario hugo para poder ver la información almacenada en la tabla usuarios y

comprobar el correcto funcionamiento de la tarea, esto lo podemos ver en la figura 20.

```

edmundojm@CR2019630428:~/apache-tomcat-8.5.72/webapps$ javac -cp $CATALINA_HOME/lib/javax.ws.rs-api-2.0.1.jar:$CATALINA_HOME/lib/gson-2.3.1.jar:. negocio/Servicio.java
javac: file not found: negocio/Servicio.java
Usage: javac <options> <source files>
Use -help for a list of possible options
edmundojm@CR2019630428:~/Service$ cd Servicio/
edmundojm@CR2019630428:~/Service$ cd Servicio/
-bash: cd: Servicio/: No such file or directory
edmundojm@CR2019630428:~/Service$ javac -cp $CATALINA_HOME/lib/javax.ws.rs-api-2.0.1.jar:$CATALINA_HOME/lib/gson-2.3.1.jar:. negocio/Servicio.java
edmundojm@CR2019630428:~/Service$ rm WEB-INF/classes/negocio/*
edmundojm@CR2019630428:~/Service$ cp negocio/*.class WEB-INF/classes/negocio/.
edmundojm@CR2019630428:~/Service$ jar cvf Servicio.war WEB-INF META-INF
adding manifest
adding: WEB-INF/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/web.xml(in = 672) (out= 296)(deflated 55%)
adding: WEB-INF/classes/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/negocio/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/negocio/Usuario.class(in = 924) (out= 532)(deflated 42%)
adding: WEB-INF/classes/negocio/AdaptadorGsonBase64.class(in = 1799) (out= 737)(deflated 59%)
adding: WEB-INF/classes/negocio/Servicio.class(in = 8191) (out= 3774)(deflated 53%)
adding: WEB-INF/classes/negocio/Error.class(in = 278) (out= 214)(deflated 23%)
ignoring entry META-INF/
adding: META-INF/context.xml(in = 306) (out= 210)(deflated 31%)
edmundojm@CR2019630428:~/Service$ ls
META-INF  Servicio.war  WEB-INF  negocio
edmundojm@CR2019630428:~/Service$ cp Servicio.war ../apache-tomcat-8.5.72/webapps/
edmundojm@CR2019630428:~/Service$ cd ../apache-tomcat-8.5.72/webapps/
edmundojm@CR2019630428:~/apache-tomcat-8.5.72/webapps$ ls
ROOT  Servicio  Servicio.war
edmundojm@CR2019630428:~/apache-tomcat-8.5.72/webapps$

```

Figura 19: Compilación de Servicio y copiado a la carpeta webapps .

```

edmundojm@CR2019630428:~/apache-tomcat-8.5.72/webapps$ ls
ROOT  Servicio  Servicio.war
edmundojm@CR2019630428:~/apache-tomcat-8.5.72/webapps$ cd
edmundojm@CR2019630428:~$ mysql -u hugo -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.36-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use servicio_web
-> ;
mysql> use servicio_web
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_servicio_web |
+-----+
| fotos_usuarios         |
| usuarios               |
+-----+
2 rows in set (0.00 sec)

mysql> select * from u
-> ;
+-----+
| usuarios |
+-----+
Empty set (0.00 sec)

mysql> select * from fotos_usuarios;
Empty set (0.00 sec)

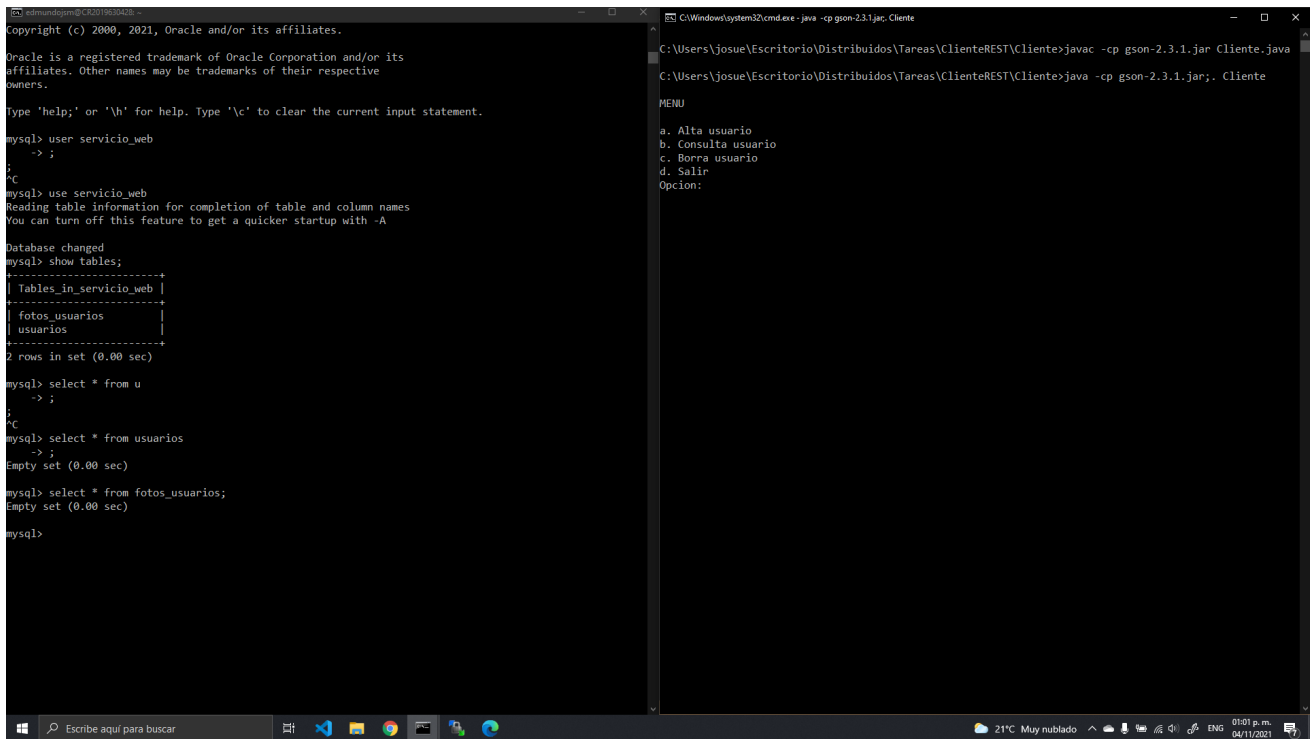
mysql>

```

Figura 20: Estado actual de la base de datos.

Finalmente la compilación y ejecución del programa Cliente en la computadora local, mencionar que

se usa `BufferedReader` en lugar de `Scanner` ya que este ultimo me arrojaba errores y es que no me dejaba escribir el primer parámetro al momento de querer de dar de alta al usuario, me pasaba inmediatamente al otro campo, también mencionar que es importante compilar pasando el `.jar` de `gson-2.3.1` ya que sin este nos generaría errores y nuestro programa no funcionaria. Todo esto se puede ver en la figura 21.



```
Copyright (c) 2000, 2021, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '?' for help. Type '?' to clear the current input statement.

mysql> user servicio_web
-> ;
^C
mysql> use servicio_web
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_servicio_web |
+-----+
| fotos_usuarios         |
| usuarios                |
+-----+
2 rows in set (0.00 sec)

mysql> select * from u
-> ;
^C
mysql> select * from usuarios
-> ;
Empty set (0.00 sec)

mysql> select * from fotos_usuarios;
Empty set (0.00 sec)

mysql>

C:\Windows\system32\cmd.exe - java -cp gson-2.3.1.jar Cliente
C:\Users\josue\Escritorio\Distribuidos\Tareas\ClienteREST\Cliente>javac -cp gson-2.3.1.jar Cliente.java
C:\Users\josue\Escritorio\Distribuidos\Tareas\ClienteREST\Cliente>java -cp gson-2.3.1.jar;. Cliente

MENU
a. Alta usuario
b. Consulta usuario
c. Borra usuario
d. Salir
Opcion:
```

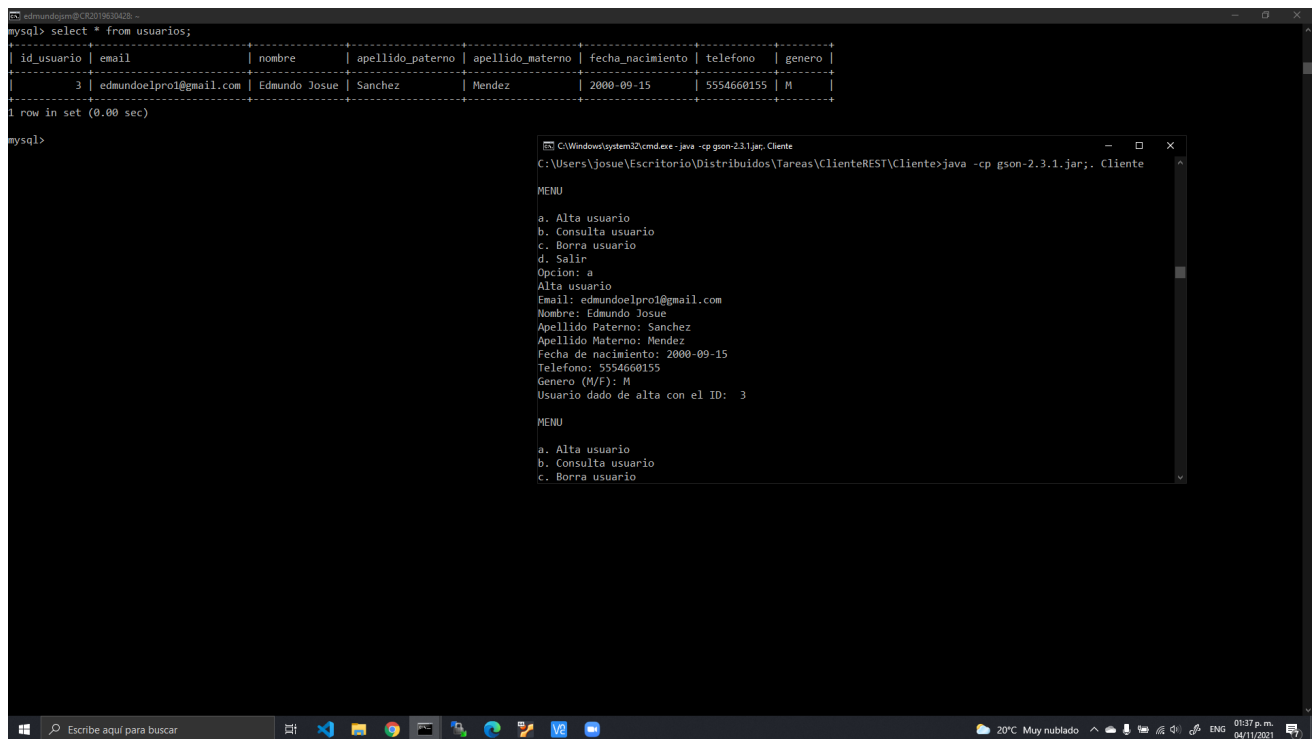
Figura 21: Estado actual de la base de datos.

## 2.4. Pruebas

### 2.4.1. Prueba 1

Daremos de alta un nuevo usuario con la información que podemos ver en la figura 22, observar como ahora en la base de datos tenemos un usuario almacenado.





The screenshot shows a terminal window with a MySQL command prompt and a Java application window. The MySQL prompt shows a query to select all data from the 'usuarios' table, returning one row for user ID 3. The Java application window shows a menu with options 'a. Alta usuario', 'b. Consulta usuario', and 'c. Borra usuario'. Option 'a' is selected, and the application displays the details of the user with ID 3: Email: edundoelpro1@gmail.com, Nombre: Edmundo Josue, Apellido Paterno: Sanchez, Apellido Materno: Mendez, Fecha de nacimiento: 2000-09-15, Telefono: 5554660155, Genero (M/F): M. It also shows the user ID 3.

```
mysql> select * from usuarios;
+-----+-----+-----+-----+-----+-----+-----+
| id_usuario | email | nombre | apellido_paterno | apellido_materno | fecha_nacimiento | telefono | genero |
+-----+-----+-----+-----+-----+-----+-----+
| 3 | edundoelpro1@gmail.com | Edmundo Josue | Sanchez | Mendez | 2000-09-15 | 5554660155 | M |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

```
C:\Windows\system32\cmd.exe - java -cp gson-2.3.1.jar; Cliente
C:\Users\josue\Escritorio\Distribuidos\Tareas\ClienteREST\Cliente>java -cp gson-2.3.1.jar;. Cliente

MENU
a. Alta usuario
b. Consulta usuario
c. Borra usuario
d. Salir
Opcion: a
Alta usuario
Email: edundoelpro1@gmail.com
Nombre: Edmundo Josue
Apellido Paterno: Sanchez
Apellido Materno: Mendez
Fecha de nacimiento: 2000-09-15
Telefono: 5554660155
Genero (M/F): M
Usuario dado de alta con el ID: 3

MENU
a. Alta usuario
b. Consulta usuario
c. Borra usuario
```

Figura 22: Prueba 1. Dar de alta a usuario.

#### 2.4.2. Prueba 2, 3 y 4

Consultaremos al usuario dado de alta en la prueba anterior y cambiaremos algún dato de este, el dato a cambiar sera el teléfono, notar que en la figura 23 vemos con el la base de datos se actualizo el campo, indicando un correcto funcionamiento de la tarea, el la figura 22 vemos los datos que nos regresa el servicio con la modificación del campo teléfono.

```

C:\Windows\system32\cmd.exe - java -cp gson-2.3.1.jar. Cliente
MENU
a. Alta usuario
b. Consulta usuario
c. Borra usuario
d. Salir
Opcion: b
Consulta usuario
Ingresa el ID de usuario:
3
Email : edundoelpro1@gmail.com
Nombre: Edmundo Josue
Apellido Paterno: Sanchez
Apellido Materno: Mendez
Telefono: 5554660155
Fecha: 2000-09-15
Genero: M
Desea modificar los datos del usuario (s/n)?
s
Email:
Nombre:
Apellido Paterno:
Apellido Materno:
Fecha de nacimiento:
Telefono:
5557329497
Genero (M/F):
El usuario ha sido modificado
MENU
a. Alta usuario
b. Consulta usuario
c. Borra usuario
d. Salir
Opcion: b
Consulta usuario
Ingresa el ID de usuario:
3
Email : edundoelpro1@gmail.com
Nombre: Edmundo Josue
Apellido Paterno: Sanchez
Apellido Materno: Mendez
Telefono: 5557329497

```

Figura 23: Prueba 2, 3 y 4. Consultar y modificar usuario.

```

mysql> select * from usuarios;
+----+-----+-----+-----+-----+-----+-----+
| id_usuario | email                | nombre | apellido_paterno | apellido_materno | fecha_nacimiento | telefono | genero |
+----+-----+-----+-----+-----+-----+-----+
| 3 | edundoelpro1@gmail.com | Edmundo Josue | Sanchez | Mendez | 2000-09-15 | 5554660155 | M |
+----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select * from usuarios;
+----+-----+-----+-----+-----+-----+-----+
| id_usuario | email                | nombre | apellido_paterno | apellido_materno | fecha_nacimiento | telefono | genero |
+----+-----+-----+-----+-----+-----+-----+
| 3 | edundoelpro1@gmail.com | Edmundo Josue | Sanchez | Mendez | 2000-09-15 | 5557329497 | M |
+----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

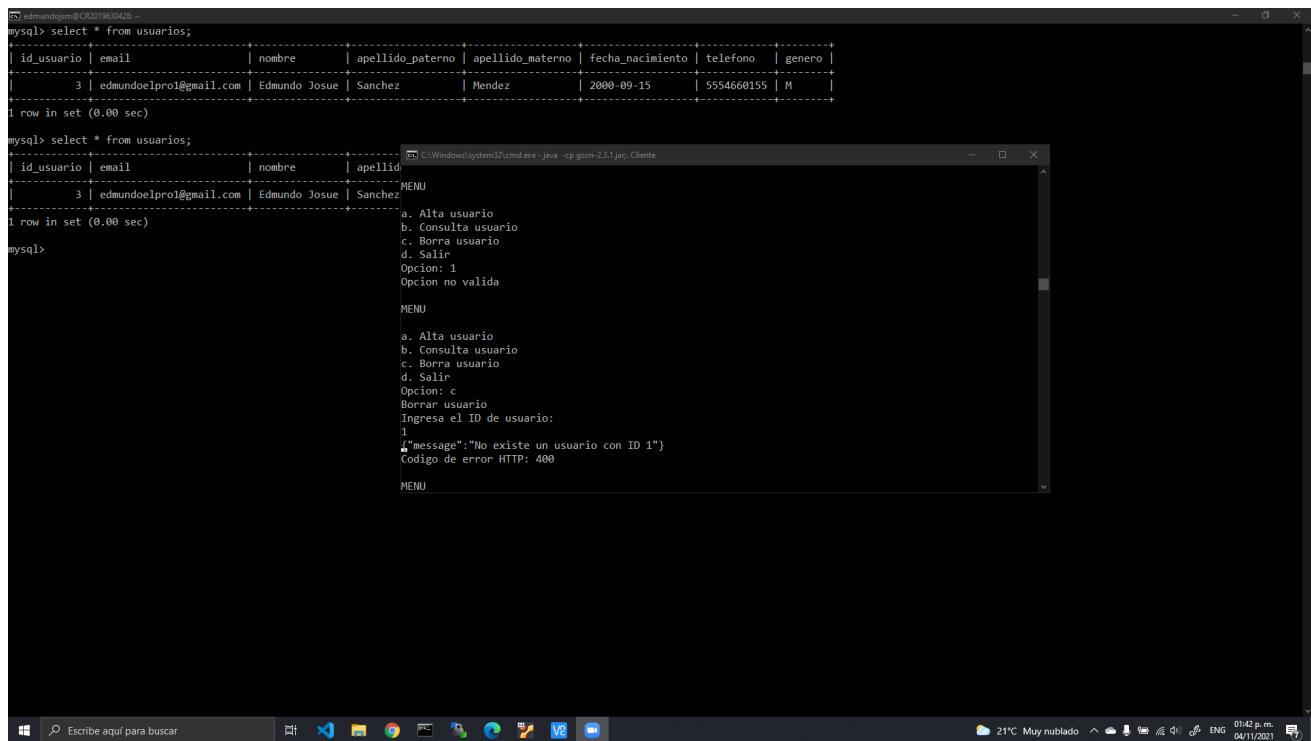
mysql>

```

Figura 24: Prueba 2, 3 y 4. Base de datos con los cambios correspondientes.

### 2.4.3. Prueba 5

Intentar borrar un usuario que no exista, como vemos se despliega el mensaje de error indicando que el id no existe.



The screenshot shows a terminal window with two parts. The top part is a MySQL command-line interface where the user has executed two queries. The first query, `select * from usuarios;`, returns one row with the following data: `id_usuario: 3, email: edmundolpro1@gmail.com, nombre: Edmundo Josue, apellido_paterno: Sanchez, apellido_materno: Mendez, fecha_nacimiento: 2000-09-15, telefono: 5554660155, genero: M`. The second query, `select * from usuarios;`, returns the same row. The bottom part of the terminal shows a Java application window titled `C:\Windows\system32\cmd.exe - java -cp gson-2.1.jar Cliente`. The application displays a menu with options: `a. Alta usuario, b. Consulta usuario, c. Borra usuario, d. Salir`. The user has selected option `c` and entered `1` as the ID. The application displays an error message: `["message": "No existe un usuario con ID 1"]` and `Codigo de error HTTP: 400`.

```
mysql> select * from usuarios;
+----+-----+-----+-----+-----+-----+-----+-----+
| id_usuario | email                | nombre      | apellido_paterno | apellido_materno | fecha_nacimiento | telefono | genero |
+----+-----+-----+-----+-----+-----+-----+-----+
| 3          | edmundolpro1@gmail.com | Edmundo Josue | Sanchez          | Mendez           | 2000-09-15       | 5554660155 | M      |
+----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select * from usuarios;
+----+-----+-----+-----+
| id_usuario | email                | nombre      | apellido_paterno |
+----+-----+-----+-----+
| 3          | edmundolpro1@gmail.com | Edmundo Josue | Sanchez          |
+----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>

C:\Windows\system32\cmd.exe - java -cp gson-2.1.jar Cliente

MENU
a. Alta usuario
b. Consulta usuario
c. Borra usuario
d. Salir
Opcion: 1
Opcion no valida

MENU
a. Alta usuario
b. Consulta usuario
c. Borra usuario
d. Salir
Opcion: c
Borrar usuario
Ingresa el ID de usuario:
1
["message": "No existe un usuario con ID 1"]
Codigo de error HTTP: 400

MENU
```

Figura 25: Prueba 5. Dar de baja a un ID que no existe.

### 2.4.4. Prueba 6

Borrar el usuario dado de alta en la prueba 1, notar que en la base de datos al momento de seleccionar todos los elementos de la tabla usuario nos arroja que esta esta vacía.

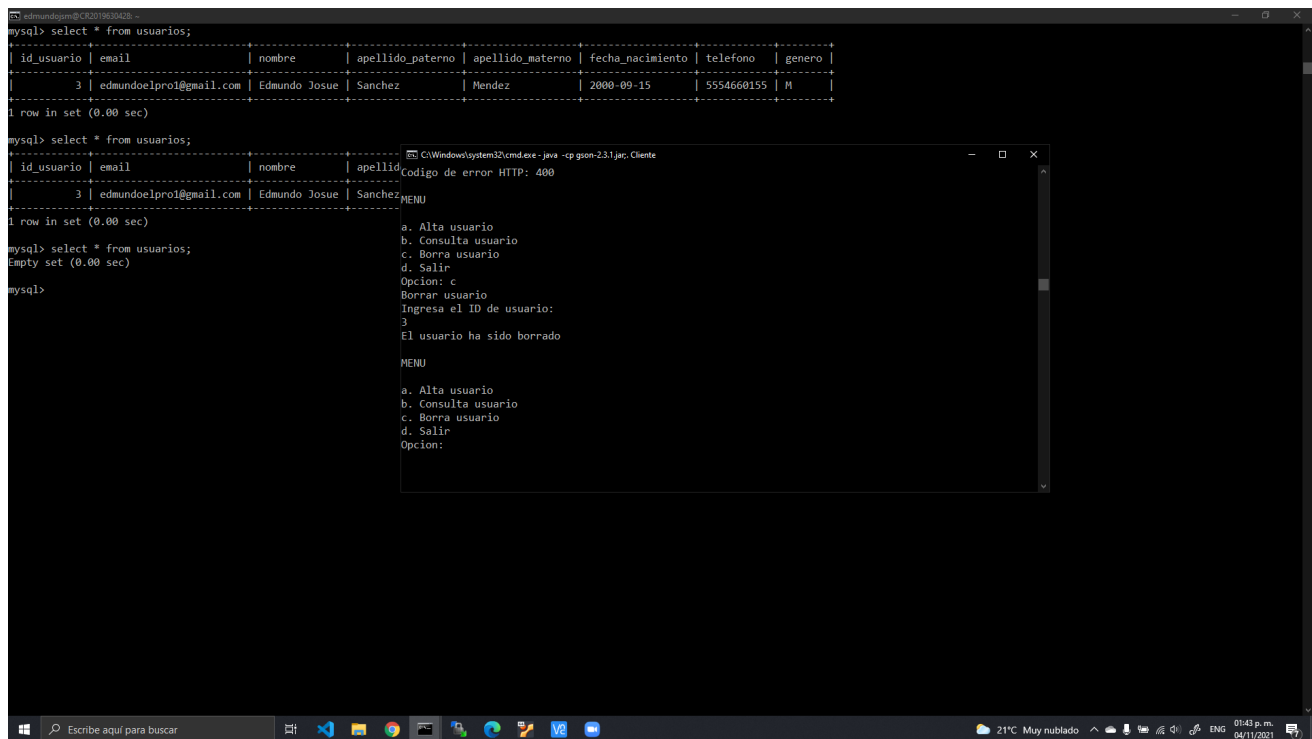


Figura 26: Prueba 6. Usuario dado de baja.

### 2.4.5. Finalizar programa cliente

Se finaliza la ejecución del programa.

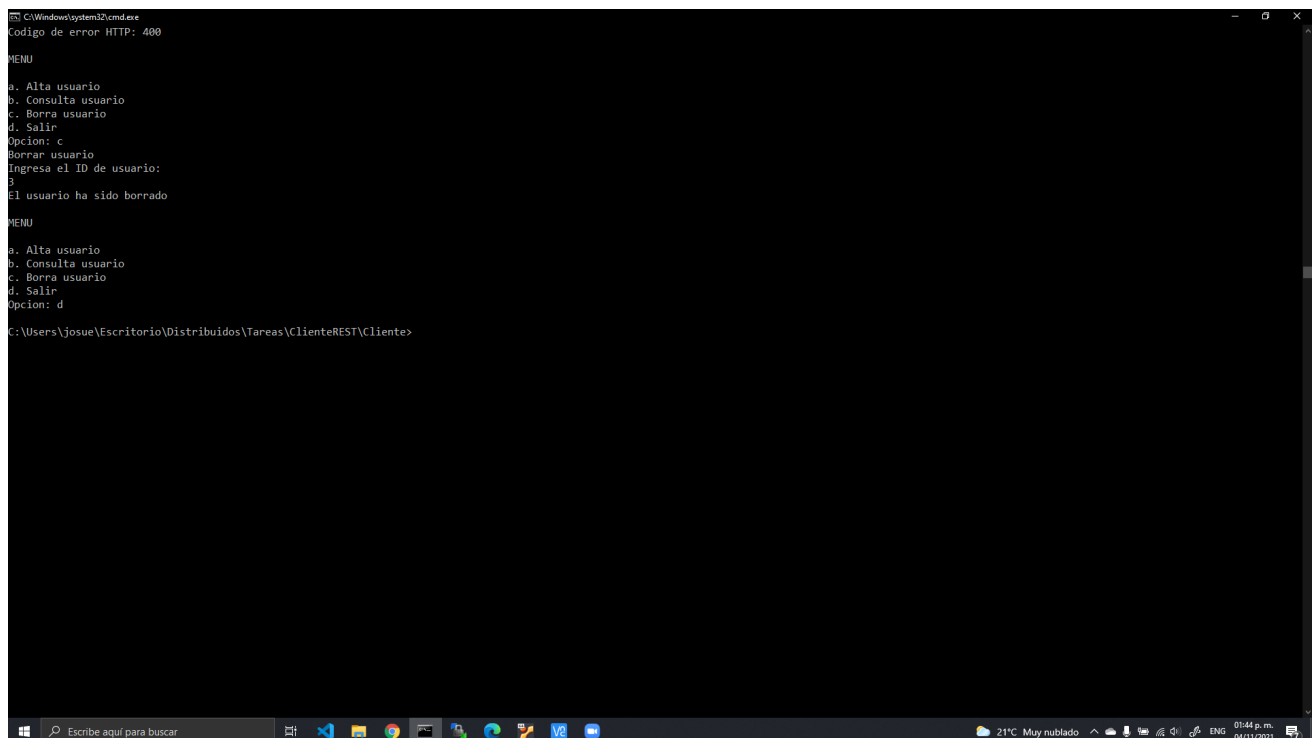
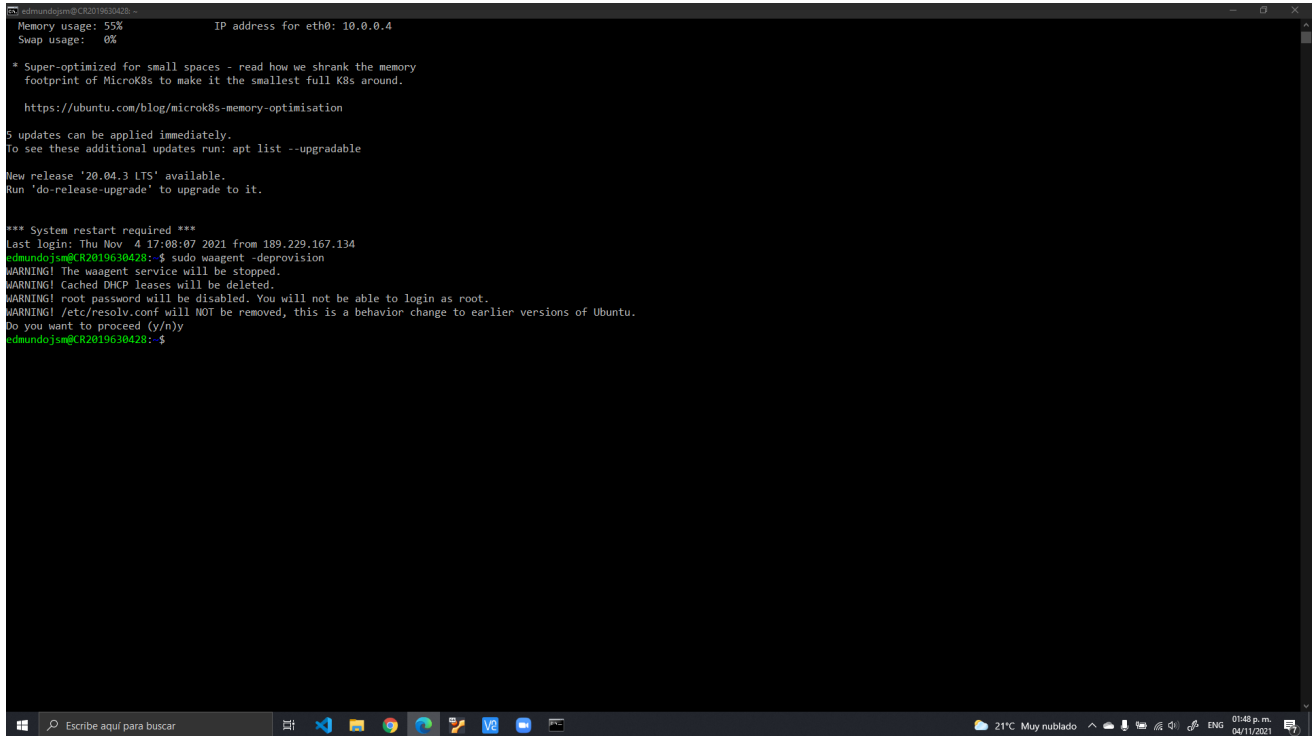


Figura 27: Programa finalizado.

## 2.5. Creación de una imagen de la maquina virtual conservando el usuario

En este punto tendremos que crear una imagen de la maquina virtual ya que esta tendrá todo lo que instalamos y usamos para esta practica, esto se hace con el fin de usar la imagen para las tareas posteriores, este proceso lo vemos en las figuras 28 a la 31.



```
edmundojm@CR2019630428:~$  
Memory usage: 55%      IP address for eth0: 10.0.0.4  
Swap usage:  0%  
  
* Super-optimized for small spaces - read how we shrank the memory  
  footprint of MicroK8s to make it the smallest full K8s around.  
  
https://ubuntu.com/blog/microk8s-memory-optimisation  
  
5 updates can be applied immediately.  
To see these additional updates run: apt list --upgradable  
  
New release '20.04.3 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
*** System restart required ***  
Last login: Thu Nov  4 17:08:07 2021 from 189.229.167.134  
edmundojm@CR2019630428:~$ sudo waagent -deprovision  
WARNING! The waagent service will be stopped.  
WARNING! Cached DHCP leases will be deleted.  
WARNING! root password will be disabled. You will not be able to login as root.  
WARNING! /etc/resolv.conf will NOT be removed, this is a behavior change to earlier versions of Ubuntu.  
Do you want to proceed (y/n)y  
edmundojm@CR2019630428:~$
```

Figura 28: Ejecución del comando `sudo waagent -deprovision`, esto para crear la imagen con todo y el usuario.

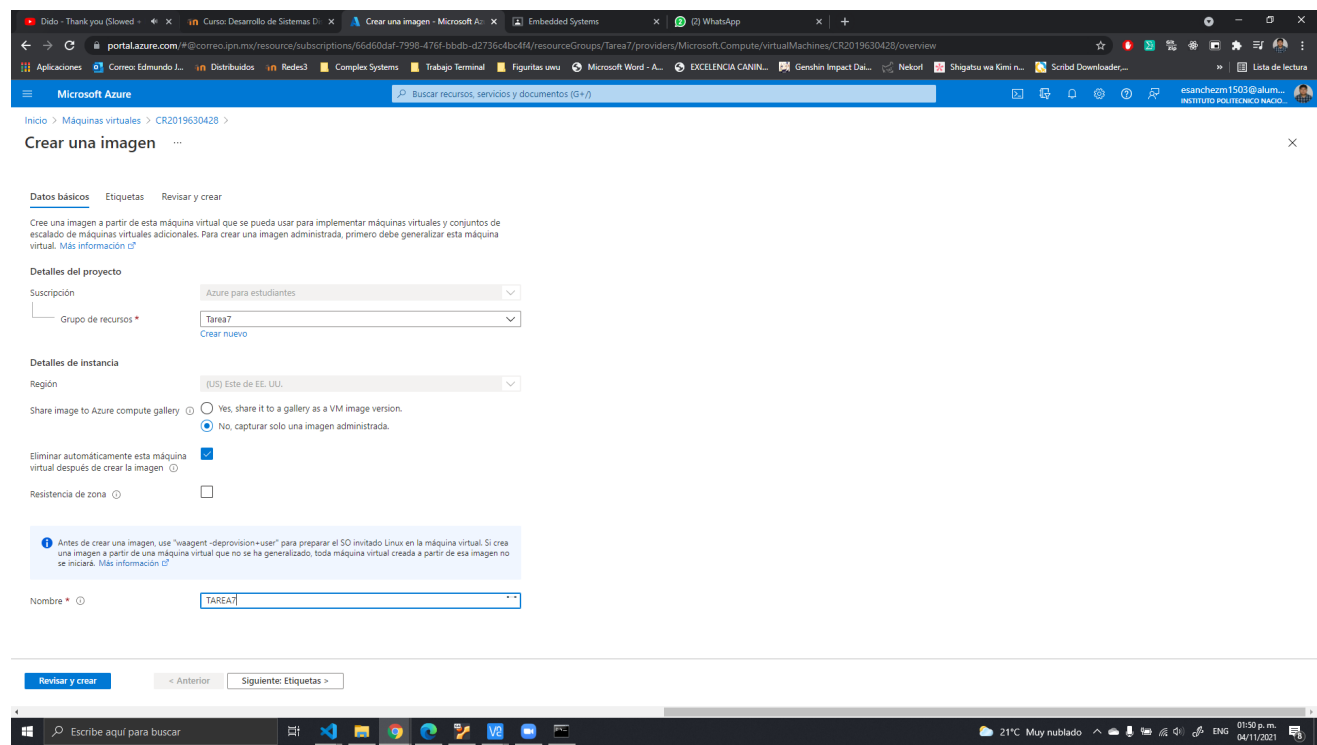


Figura 29: Opción seleccionada de eliminar automáticamente esta maquina virtual después de crear la imagen, también elegimos un nombre el cual es TAREA7.

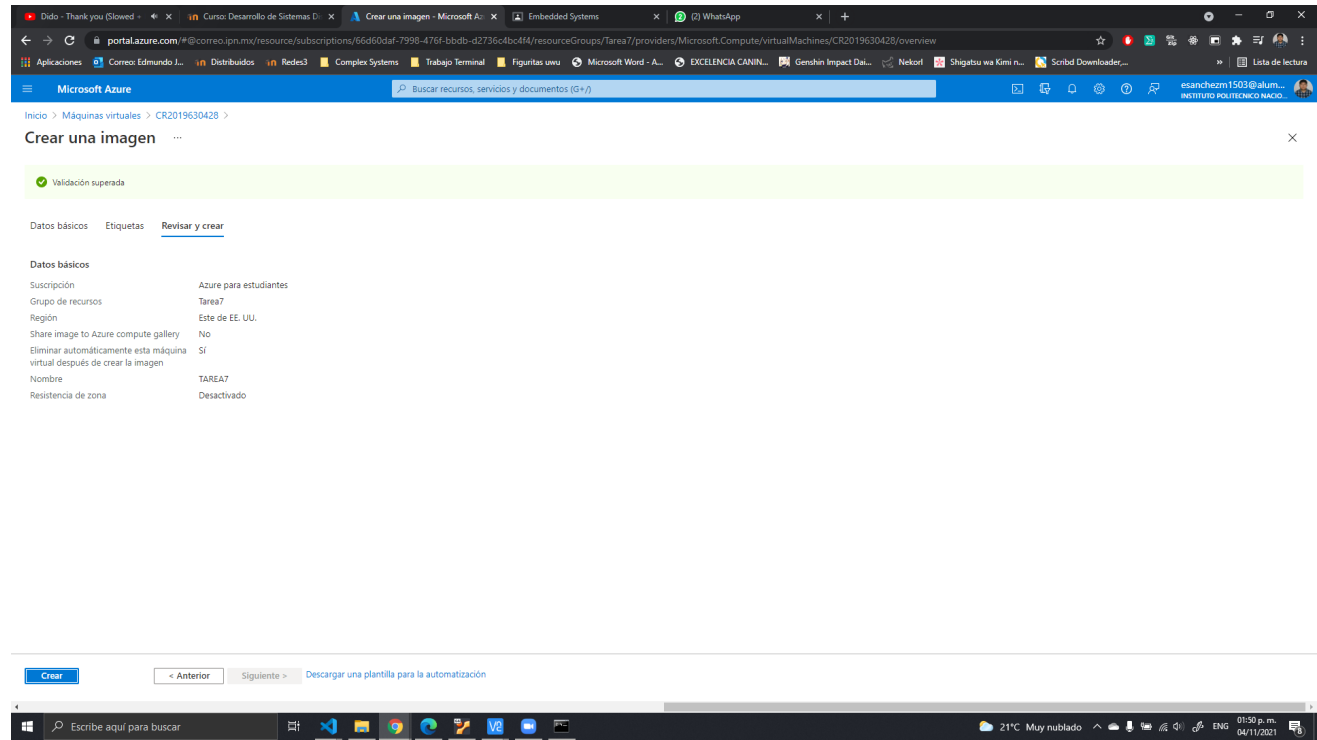


Figura 30: Información de la imagen.

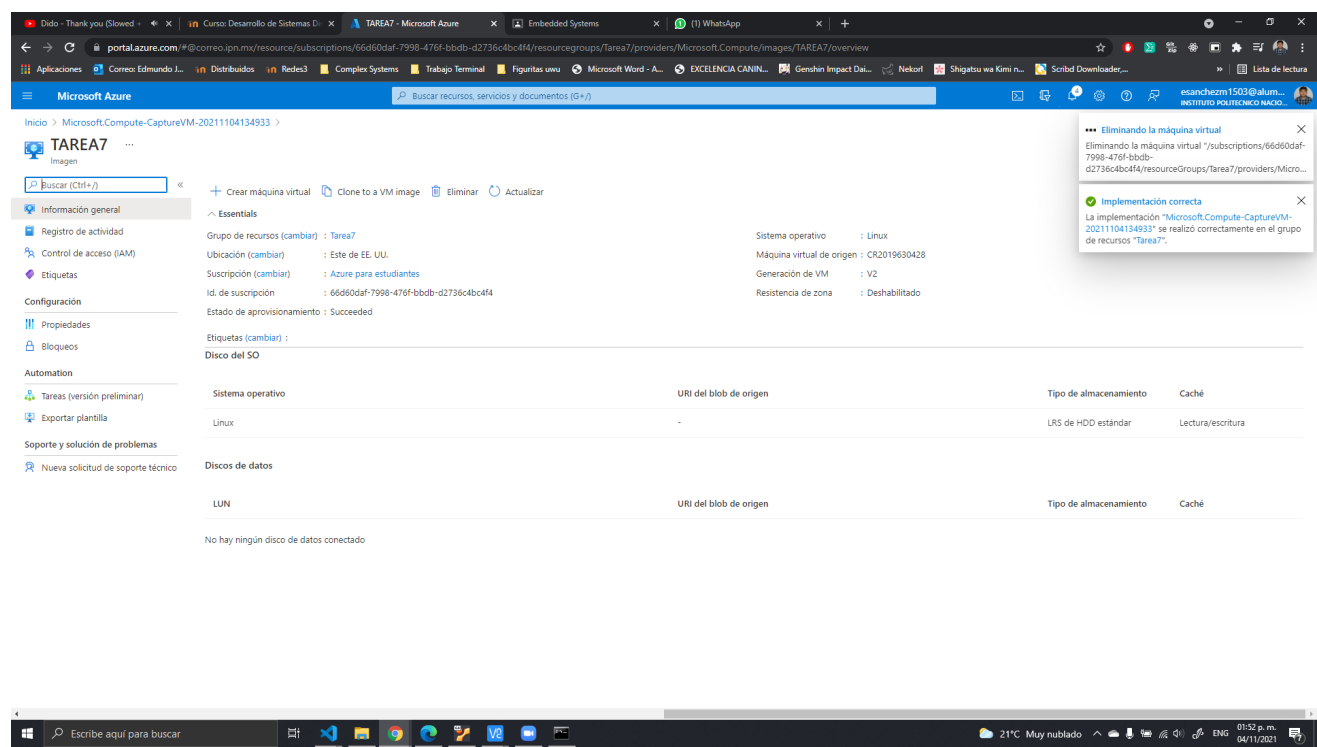


Figura 31: Imagen creada de manera exitosa.

### 3. Conclusiones

Al igual que en la tarea anterior observamos el funcionamiento de una aplicación REST, y su comportamiento básico. Solo que en esta ocasión la aplicación que consume REST es una aplicación desarrollada en Java y no es una aplicación WEB, esto nos permite ver la versatilidad de las aplicaciones REST ya que funciona tanto para aplicaciones de escritorio como para aplicaciones en la WEB, esto trae como consecuencia el uso muy sencillo y rápido de crear diferentes aplicaciones para que puedan consumir la aplicación REST. En lo que respecta al desarrollo del código para el cliente fue muy sencillo ya que durante la ESCOM se hacen programas de manera local con la misma lógica, así que solo fue traer los conocimientos ya obtenidos y vaciarlos en la tarea, evidentemente apegándonos a lo solicitado para la entrega de esta.