



Instituto Politécnico Nacional Escuela Superior de Cómputo

Tarea 8. Desarrollo de un cliente para un servicio web REST

PRESENTA

Vladimir Azpeitia Hernández

PROFESOR

Carlos Pineda Guerrero

ASGINATURA

Desarrollo de Sistemas Distribuidos - 4CV2

13 de mayo de 2021

1. Introducción

1.1. Planteamiento del problema

Cada alumno deberá desarrollar un programa Java consola (modo carácter) cliente del servicio web que creamos en la tarea 7.

Se deberá realizar las siguientes modificaciones al servicio web que creamos en la tarea 7:

1. Agregar el campo `id_usuario` a la clase `Usuario`.
2. Modificar el método web `'alta_usuario'`, 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`.
3. Modificar el método web `'consulta_usuario'`, ahora la consulta se deberá realizar mediante el id del usuario no el email.
4. Modificar el método web `'modifica_usuario'`, utilizar el id del usuario en el `WHERE` de la instrucción `UPDATE` en lugar del email. No deberá ser posible modificar el id de un usuario ya que se trata de la llave primaria.
5. Modificar el método web `'borra_usuario'`, utilizando como clave el id del usuario no el email.

El programa cliente deberá desplegar el siguiente menú:

MENU

- a. Alta usuario
- b. Consulta usuario
- c. Borra usuario
- d. Salir

Opcion:

Listing 1: Ejemplo del menú del cliente

2. Desarrollo

2.1. Creación de las máquinas virtuales

A continuación se muestran los pasos para configurar la máquina virtual de Ubuntu Server 18.04, la cual se utilizara para el desarrollo de esta práctica

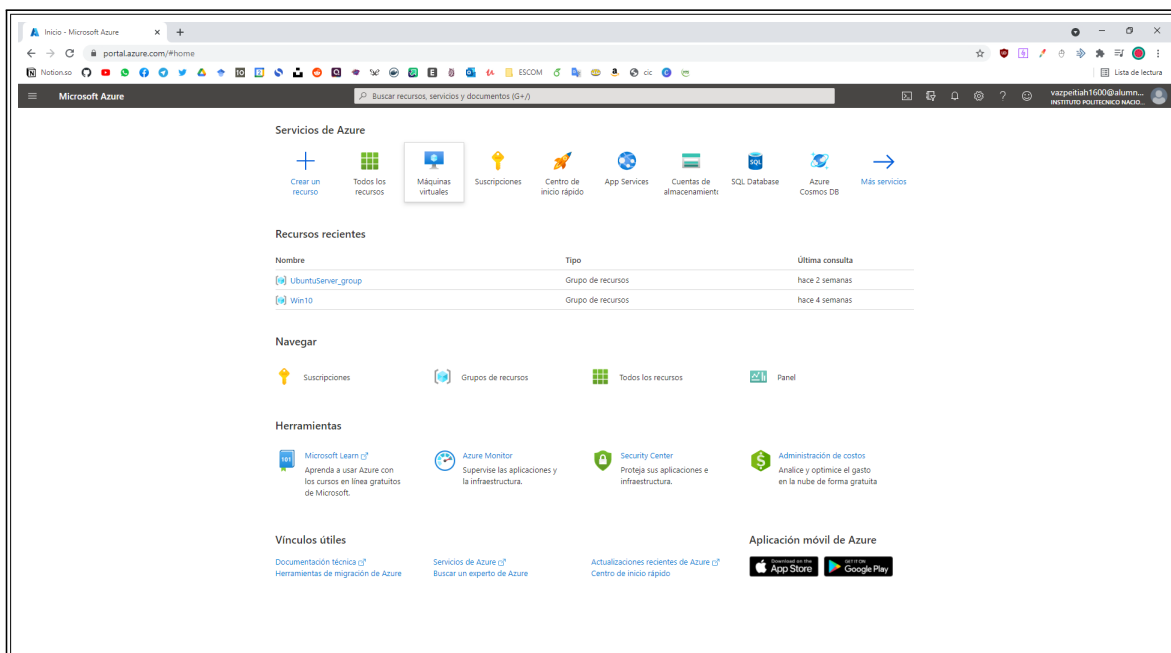


Figura 1: Creación de la máquina virtual: Portal de azure

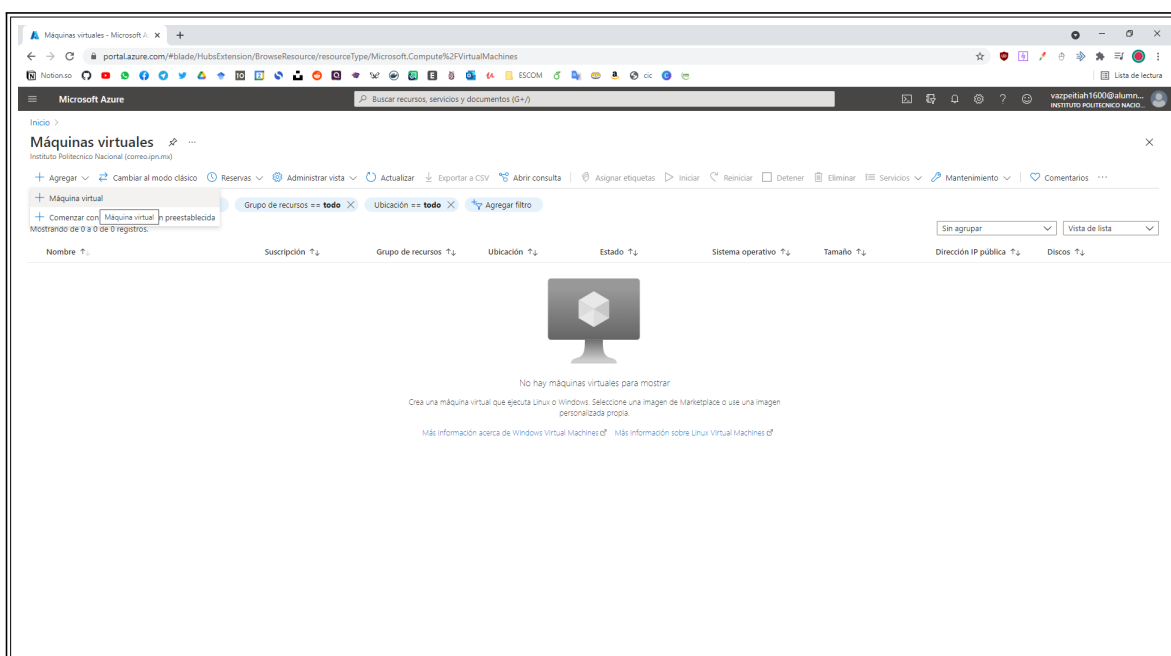


Figura 2: Creación de la máquina virtual: Máquinas virtuales

Crear una máquina virtual

Detalles del proyecto

Seleccione la suscripción para administrar recursos implementados y los costos. Use los grupos de recursos como carpetas para organizar y administrar todos los recursos.

Suscripción *

Grupo de recursos *

Detalles de instancia

Nombre de máquina virtual *

Región *

Opciones de disponibilidad *

Imagen *

Instancia de Azure de acceso público * ☐

Tamaño *

Cuenta de administrador

Tipo de autenticación * ☐ ☒

Nombre de usuario *

Contraseña *

Confirmar contraseña *

Reglas de puerto de entrada

Seleccione los puertos de red de máquina virtual que son accesibles desde la red Internet pública. Puede especificar acceso de red más limitado o granular en la pestaña Red.

Puertos de entrada públicos * ☐ ☒

Seleccionar puertos de entrada *

Revisar y crear

Figura 3: Creación de la máquina virtual: Datos básicos

Crear una máquina virtual

Discos

Las máquinas virtuales de Azure tienen un disco de sistema operativo y un disco temporal para el almacenamiento a corto plazo. Puede asociar discos de datos adicionales. El tamaño de la máquina virtual determina el tipo de almacenamiento que puede usar y la cantidad de datos que permiten los discos. [Más información](#)

Opciones de disco

Tipo de disco del sistema operativo *

Tipo de cifrado *

Habilitar compatibilidad con Ultra Disks ☐

Discos de datos

Puede agregar y configurar discos de datos adicionales para su máquina virtual o asociar discos existentes. Esta máquina virtual también incluye un disco temporal.

LUN	Nombre	Tamaño (G...)	Tipo de disco	Almacenamiento e...
<input type="button" value="Crear y adjuntar un nuevo disco"/> <input type="button" value="Asociar un disco existente"/>				

Opciones avanzadas

Revisar y crear

Figura 4: Creación de la máquina virtual: Discos

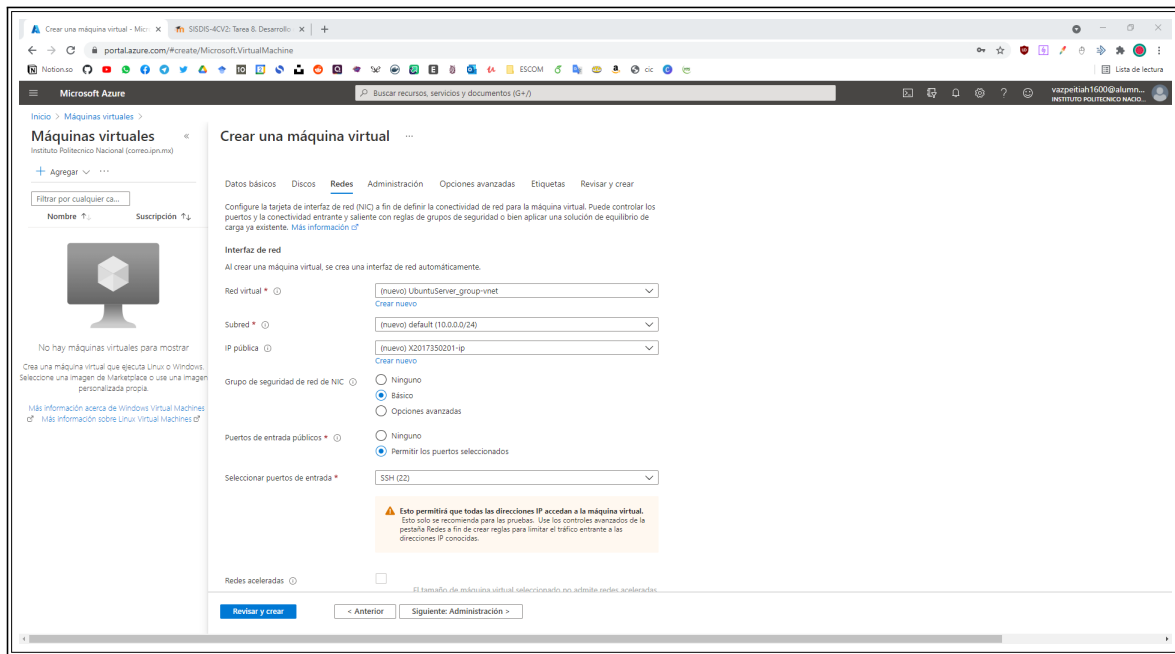


Figura 5: Creación de la máquina virtual: Redes

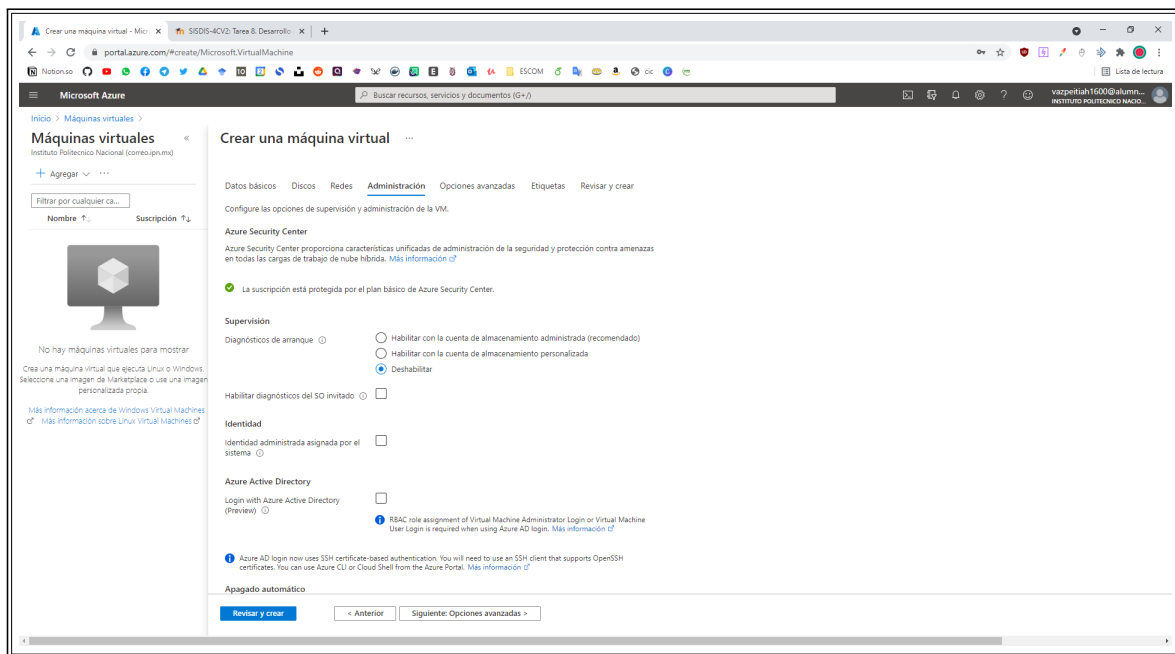


Figura 6: Creación de la máquina virtual: Administración

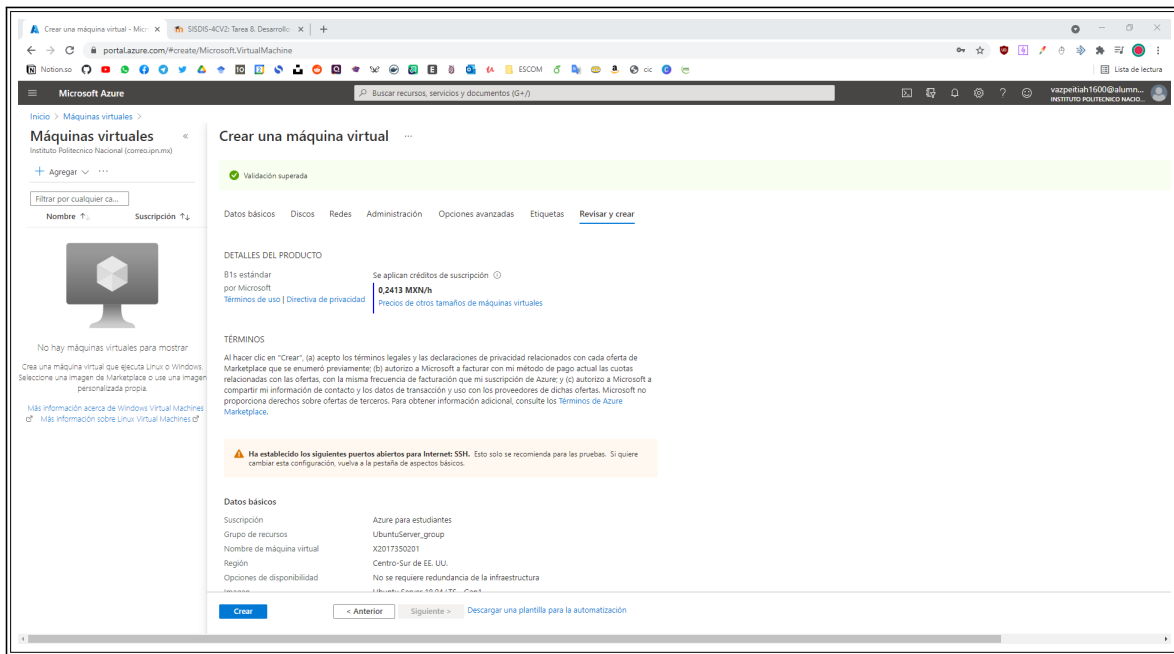


Figura 7: Creación de la máquina virtual: Validación superada

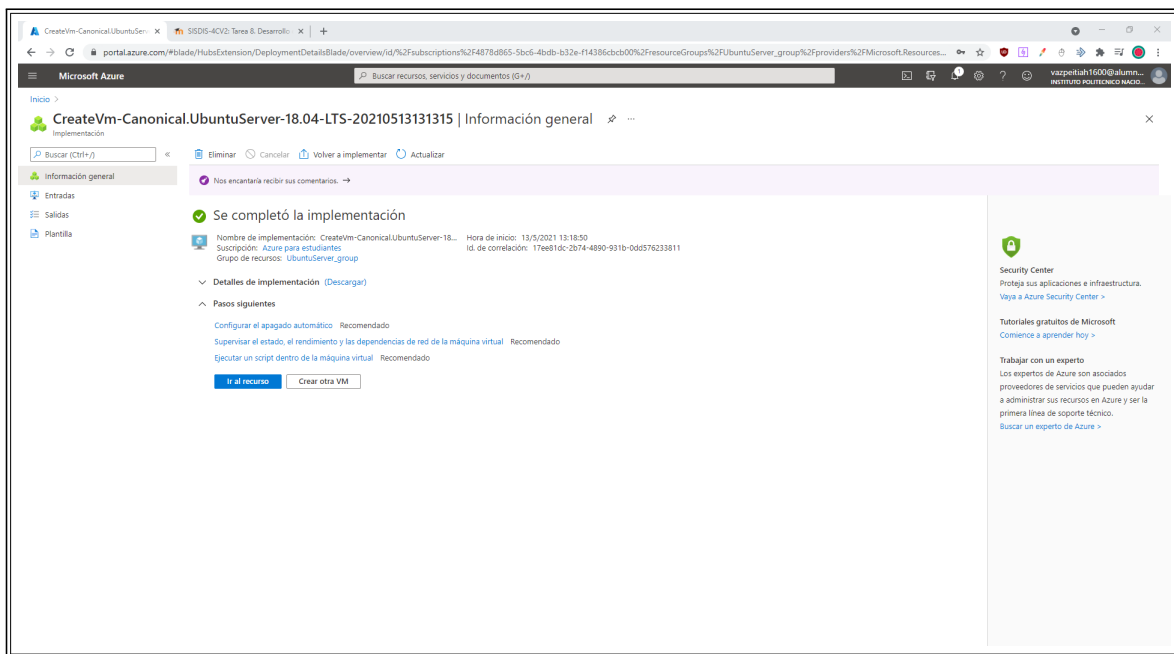


Figura 8: Creación de la máquina virtual: Implementación completada

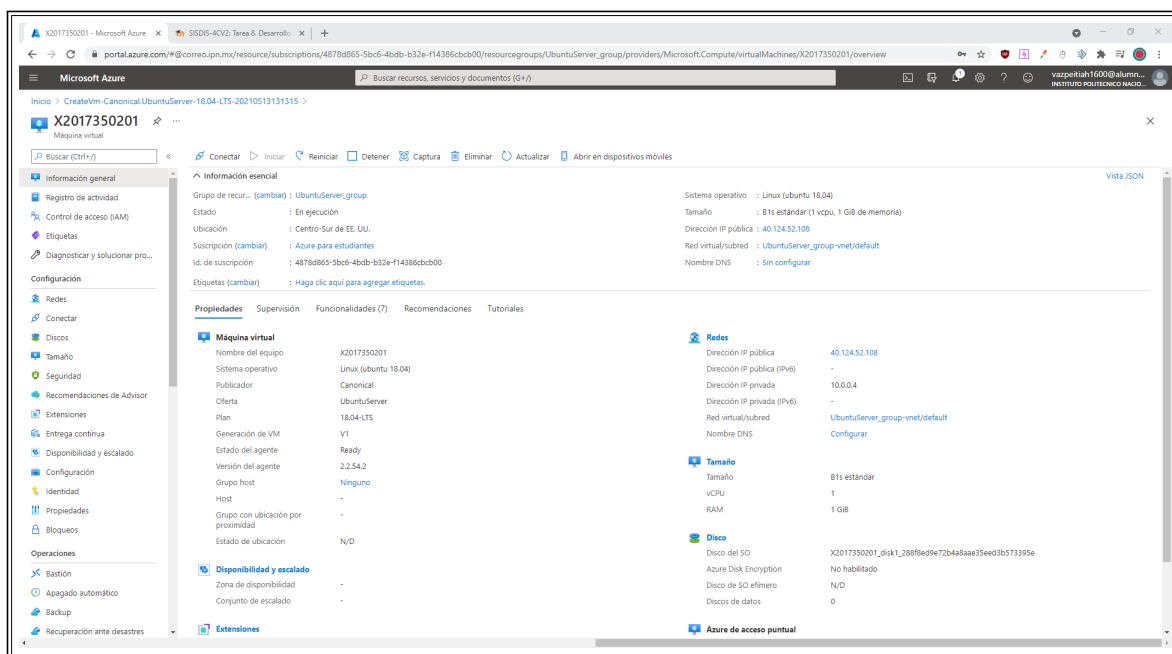


Figura 9: Creación de la máquina virtual: Panel de control de la máquina virtual

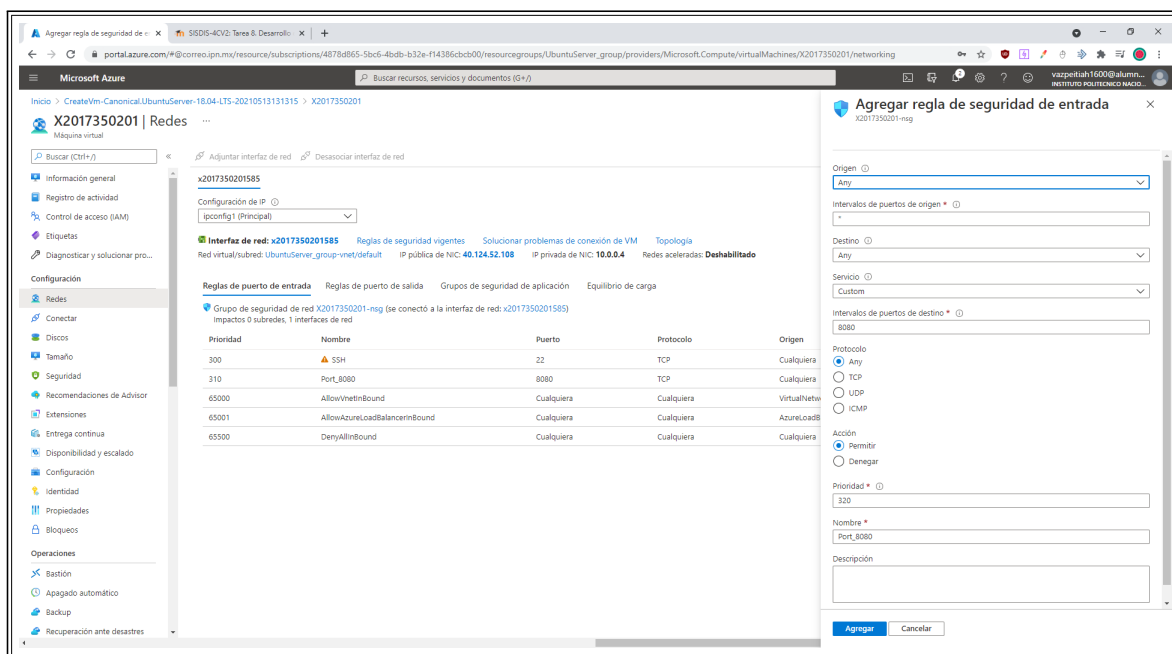


Figura 10: Creación de la máquina virtual: Configurando puerto 8080

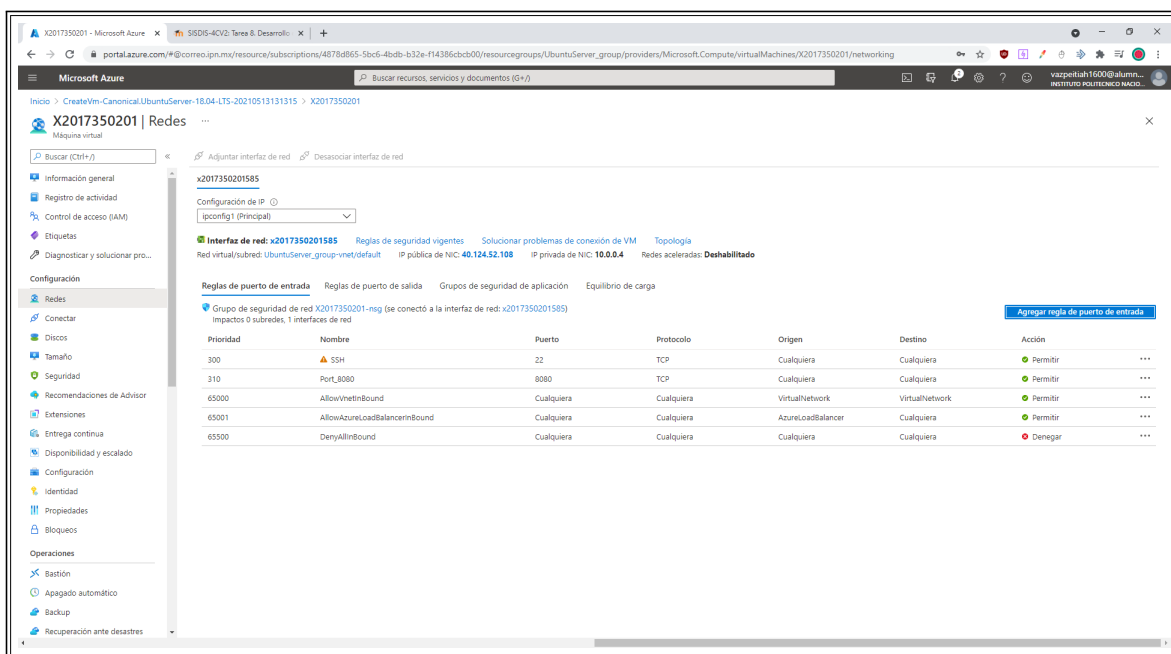


Figura 11: Creación de la máquina virtual: Puerto 8080 abierto para TCP

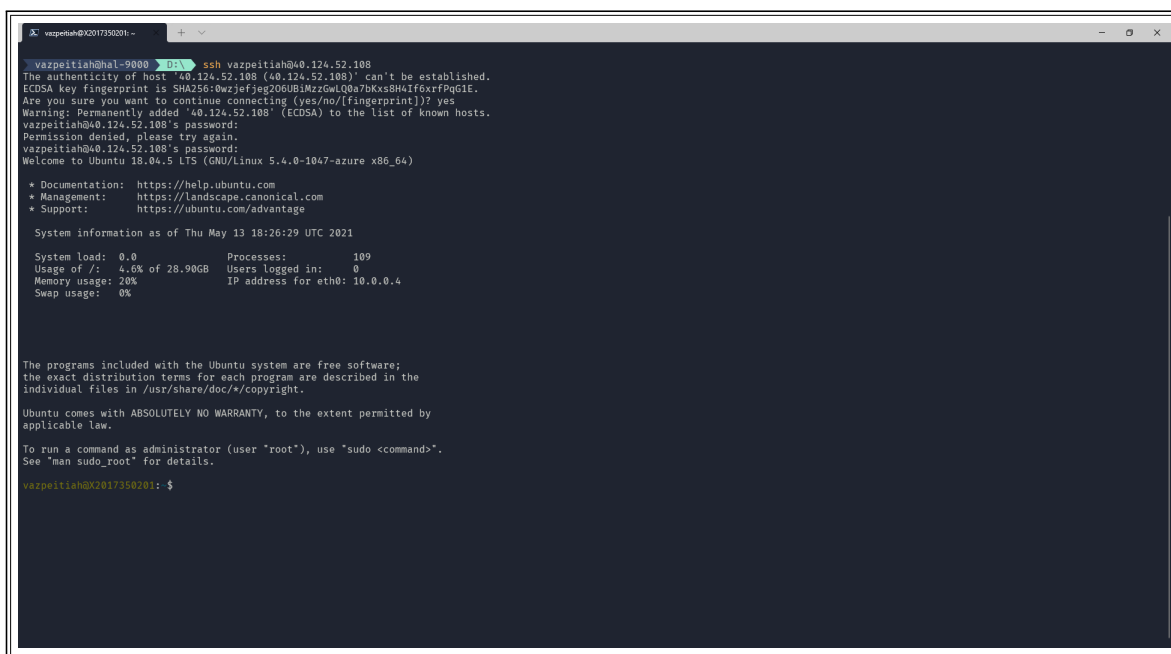
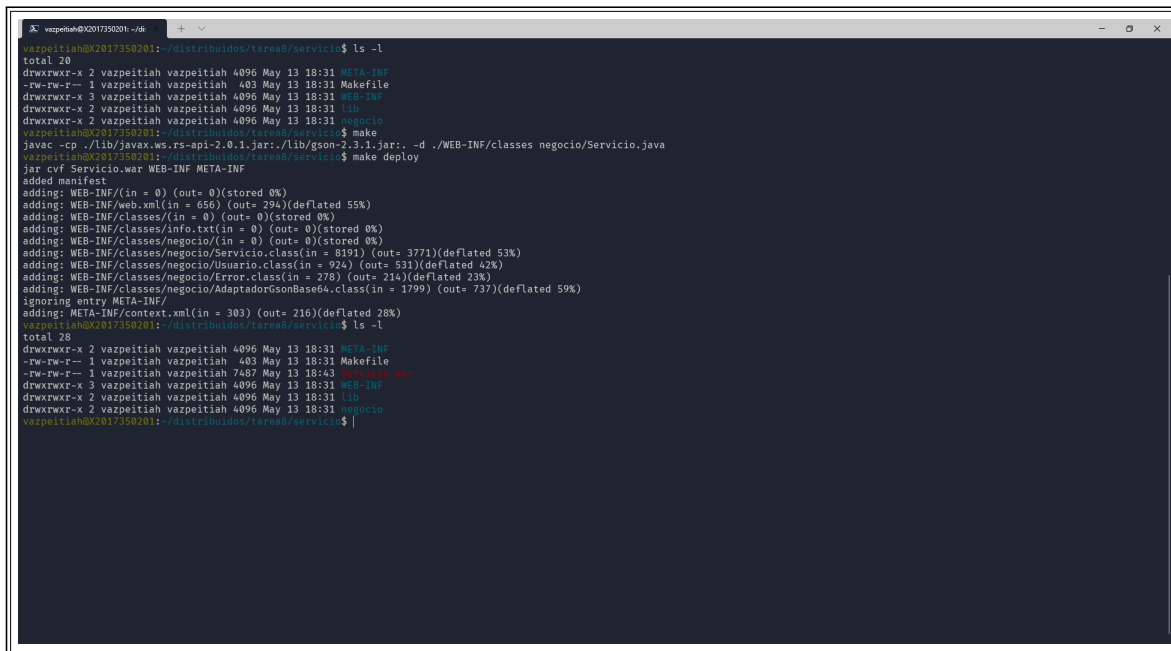


Figura 12: Creación de la máquina virtual: Conexión a través de ssh a la máquina virtual

Finalmente tendremos que instalar Java en la máquina virtual para poder compilar el código fuente y generar el archivo .war. Además debemos instalar Apache Tomcat.

3. Compilación y ejecución del código

Primero compilamos el archivo Servicio.java, con los archivos .jar necesarios. Y una vez compilado, podremos generar el archivo .war, para poder desplegar nuestra aplicación en Apache Tomcat.



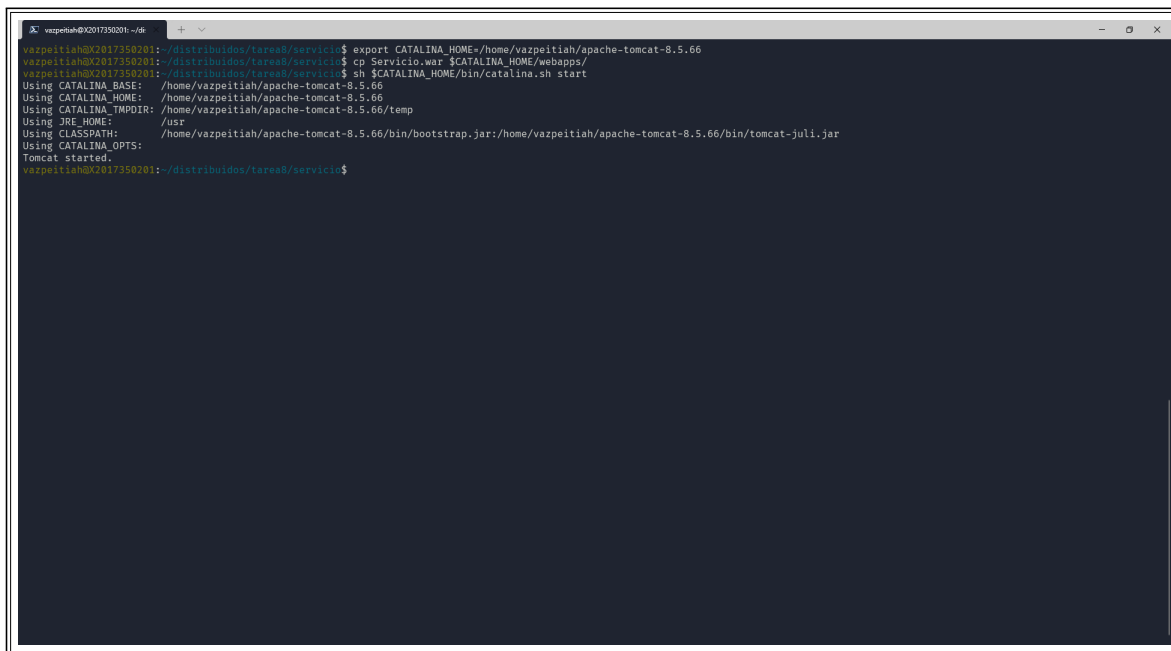
```

vazpeitiah@X2017350201:~$ cd /distribuidos/tarea8/servicio
vazpeitiah@X2017350201:~/distribuidos/tarea8/servicio$ ls -l
total 20
drwxrwxr-x 2 vazpeitiah vazpeitiah 4096 May 13 18:31 META-INF
-rw-rw-r-- 1 vazpeitiah vazpeitiah 403 May 13 18:31 MakeFile
drwxrwxr-x 3 vazpeitiah vazpeitiah 4096 May 13 18:31 WEB-INF
drwxrwxr-x 2 vazpeitiah vazpeitiah 4096 May 13 18:31 lib
drwxrwxr-x 2 vazpeitiah vazpeitiah 4096 May 13 18:31 negocio
vazpeitiah@X2017350201:~/distribuidos/tarea8/servicio$ make
javac -cp ./lib/javaweb-api-2.0.1.jar:./lib/gson-2.3.1.jar:./WEB-INF/classes negocio/Servicio.java
vazpeitiah@X2017350201:~/distribuidos/tarea8/servicio$ make deploy
jar cvf Servicio.war WEB-INF META-INF
added manifest
adding: WEB-INF/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/web.xml(in = 650) (out= 294)(deflated 55%)
adding: WEB-INF/classes/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/info.txt(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/negocio/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/negocio/Servicio.class(in = 8191) (out= 3771)(deflated 53%)
adding: WEB-INF/classes/negocio/Usuario.class(in = 924) (out= 531)(deflated 42%)
adding: WEB-INF/classes/negocio/Error.class(in = 278) (out= 214)(deflated 23%)
adding: WEB-INF/classes/negocio/AdaptadorGsonBase64.class(in = 1799) (out= 737)(deflated 59%)
ignoring entry META-INF/
adding: META-INF/context.xml(in = 303) (out= 216)(deflated 28%)
vazpeitiah@X2017350201:~/distribuidos/tarea8/servicio$ ls -l
total 28
drwxrwxr-x 2 vazpeitiah vazpeitiah 4096 May 13 18:31 META-INF
-rw-rw-r-- 1 vazpeitiah vazpeitiah 403 May 13 18:31 MakeFile
-rw-rw-r-- 1 vazpeitiah vazpeitiah 7487 May 13 18:43 Servicio.war
drwxrwxr-x 3 vazpeitiah vazpeitiah 4096 May 13 18:31 WEB-INF
drwxrwxr-x 2 vazpeitiah vazpeitiah 4096 May 13 18:31 lib
drwxrwxr-x 2 vazpeitiah vazpeitiah 4096 May 13 18:31 negocio
vazpeitiah@X2017350201:~/distribuidos/tarea8/servicio$

```

Figura 13: Compilación y ejecución: Compilando el servicio y generando archivo .war

Luego copiamos el archivo Servicio.war a la carpeta webapps de Tomcat e iniciamos el Apache Tomcat.



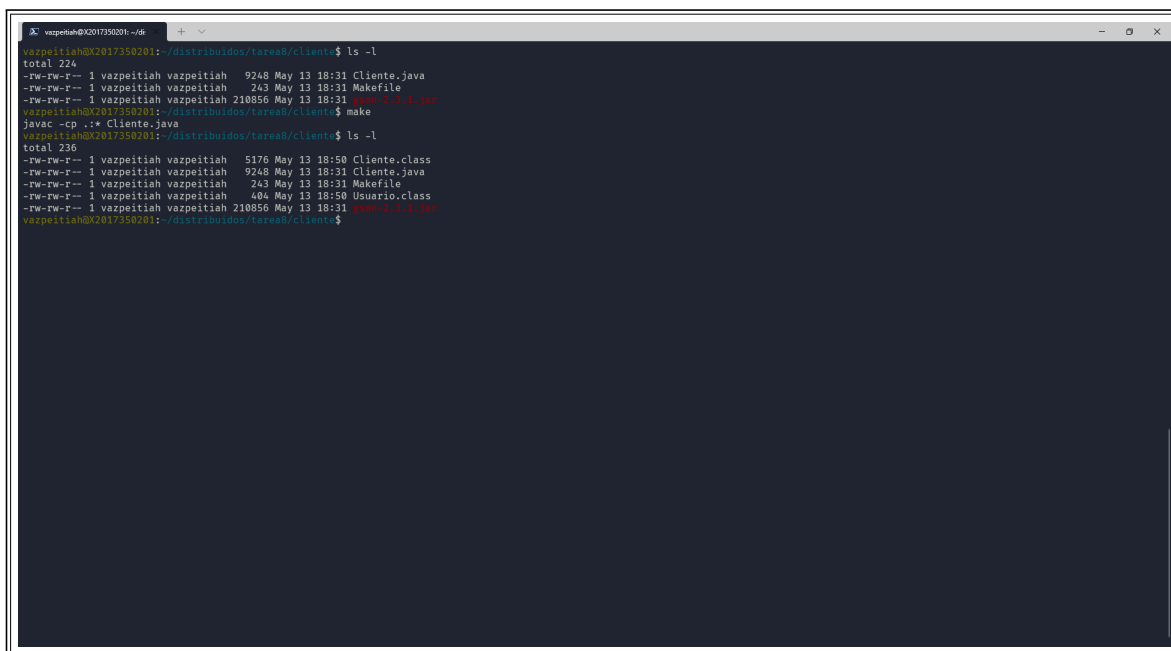
```

vazpeitiah@X2017350201:~$ cd /distribuidos/tarea8/servicio
vazpeitiah@X2017350201:~/distribuidos/tarea8/servicio$ export CATALINA_HOME=/home/vazpeitiah/apache-tomcat-8.5.66
vazpeitiah@X2017350201:~/distribuidos/tarea8/servicio$ cp Servicio.war $CATALINA_HOME/webapps/
vazpeitiah@X2017350201:~/distribuidos/tarea8/servicio$ sh $CATALINA_HOME/bin/catalina.sh start
Using CATALINA_BASE:   /home/vazpeitiah/apache-tomcat-8.5.66
Using CATALINA_HOME:   /home/vazpeitiah/apache-tomcat-8.5.66
Using CATALINA_TMPDIR: /home/vazpeitiah/apache-tomcat-8.5.66/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /home/vazpeitiah/apache-tomcat-8.5.66/bin/bootstrap.jar:/home/vazpeitiah/apache-tomcat-8.5.66/bin/tomcat-juli.jar
Tomcat started.
vazpeitiah@X2017350201:~/distribuidos/tarea8/servicio$

```

Figura 14: Compilación y ejecución: Desplegar el servicio con apache tomacat

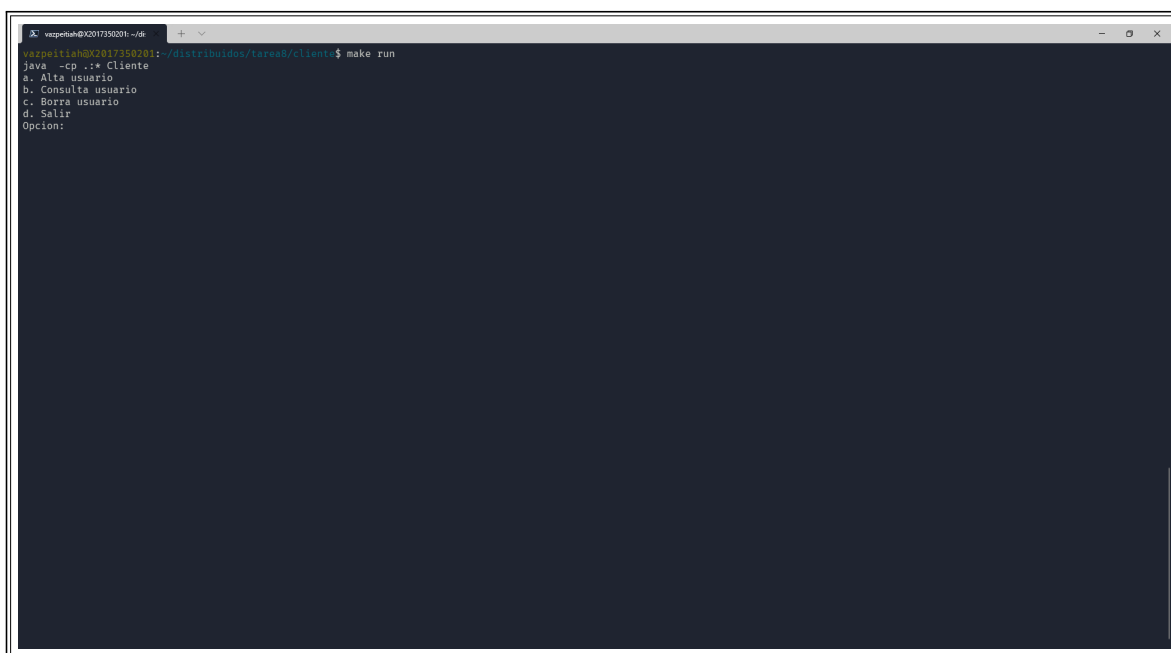
En mi caso el cliente se ejecutará en la misma máquina virtual que el servicio. Simplemente lo compilamos con el archivo .jar de gson.



```
varpeitiah@MX2017350201:~/distribuidos/tarea8/cliente$ ls -l
total 224
-rw-rw-r-- 1 varpeitiah varpeitiah 9248 May 13 18:31 Cliente.java
-rw-rw-r-- 1 varpeitiah varpeitiah 243 May 13 18:31 Makefile
-rw-rw-r-- 1 varpeitiah varpeitiah 210856 May 13 18:31 gson-2.8.1.jar
varpeitiah@MX2017350201:~/distribuidos/tarea8/cliente$ make
javac -cp .: gson-2.8.1.jar Cliente.java
varpeitiah@MX2017350201:~/distribuidos/tarea8/cliente$ ls -l
total 226
-rw-rw-r-- 1 varpeitiah varpeitiah 5176 May 13 18:50 Cliente.class
-rw-rw-r-- 1 varpeitiah varpeitiah 9248 May 13 18:31 Cliente.java
-rw-rw-r-- 1 varpeitiah varpeitiah 243 May 13 18:31 Makefile
-rw-rw-r-- 1 varpeitiah varpeitiah 484 May 13 18:50 Usuario.class
-rw-rw-r-- 1 varpeitiah varpeitiah 210856 May 13 18:31 gson-2.8.1.jar
varpeitiah@MX2017350201:~/distribuidos/tarea8/cliente$
```

Figura 15: Compilación y ejecución: Compilando el cliente del servicio REST

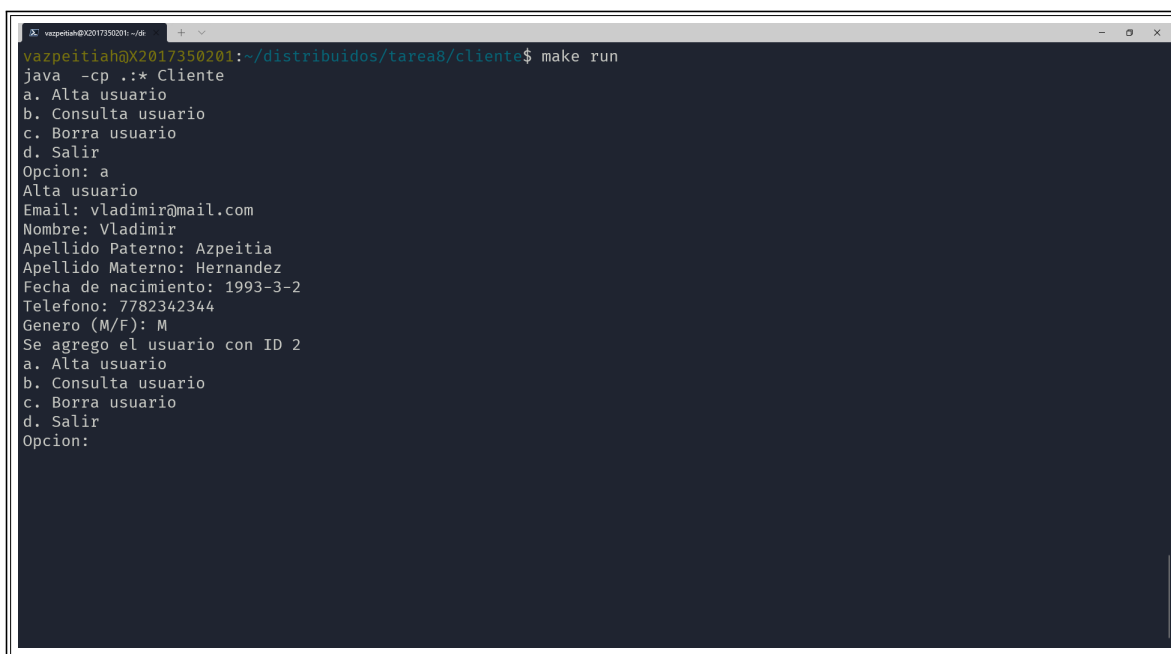
Ahora ejecutamos el archivo, y nos mostrará el siguiente menú de opciones.



```
varpeitiah@MX2017350201:~/distribuidos/tarea8/cliente$ make run
java -cp .: Cliente
a. Alta usuario
b. Consulta usuario
c. Borra usuario
d. Salir
Opcion:
```

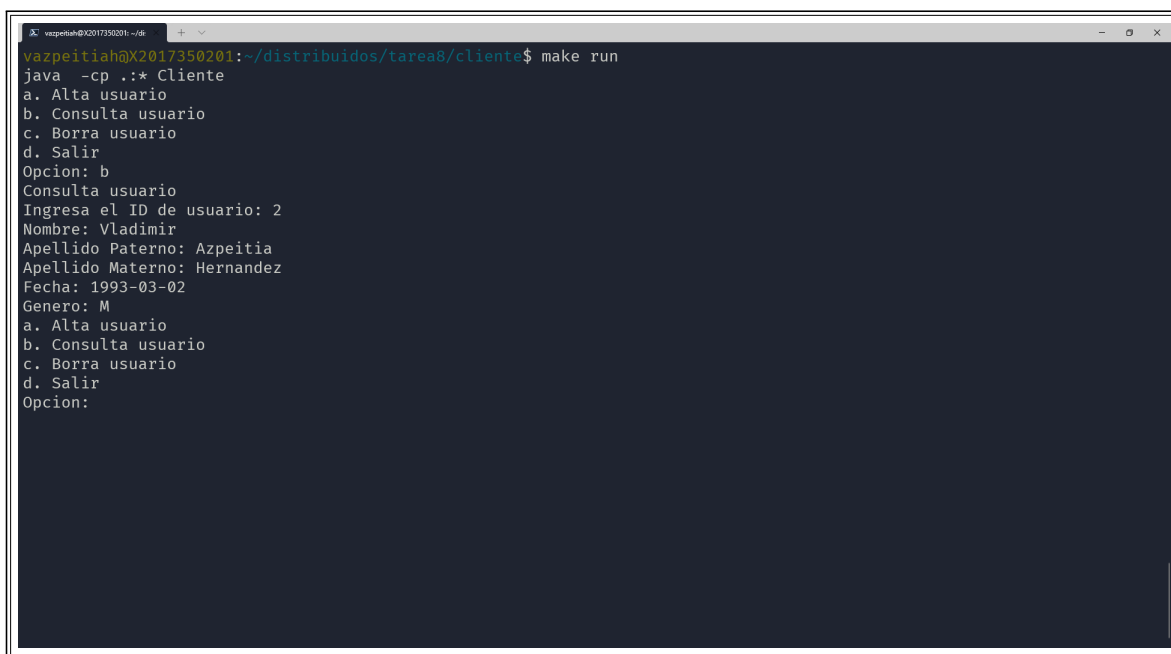
Figura 16: Compilación y ejecución: Menú de opciones del cliente

Ahora probaremos cada una de las opciones del menú:



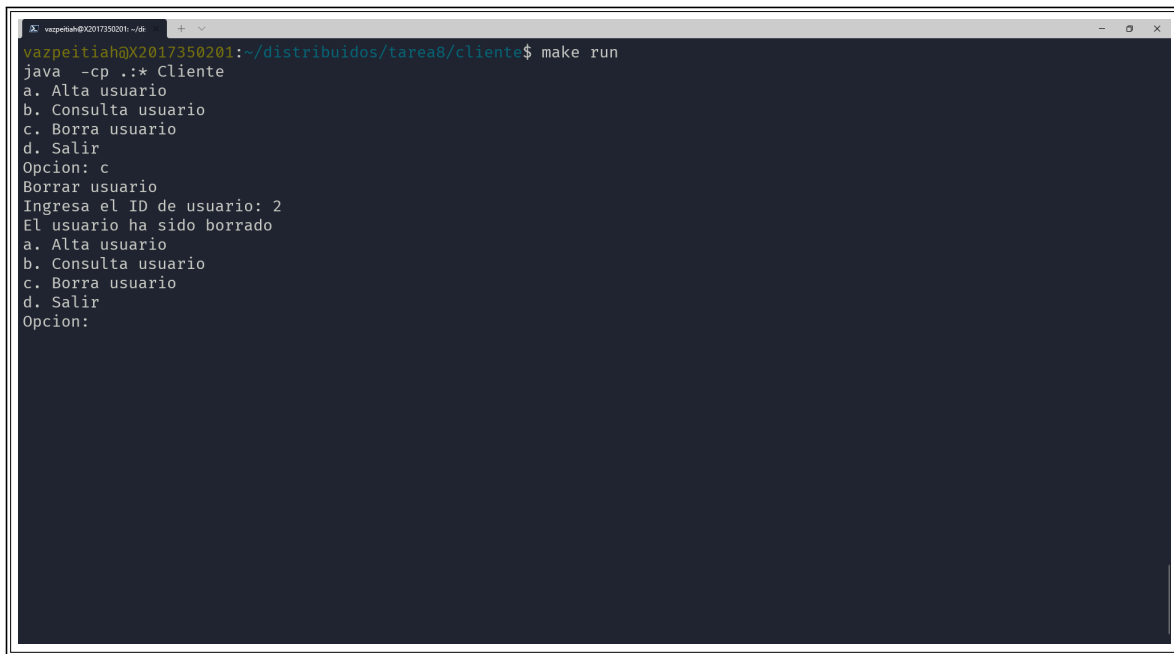
```
vazpeitia@X2017350201:~/distribuidos/tarea8/cliente$ make run
java -cp .:* Cliente
a. Alta usuario
b. Consulta usuario
c. Borra usuario
d. Salir
Opcion: a
Alta usuario
Email: vladimir@mail.com
Nombre: Vladimir
Apellido Paterno: Azpeitia
Apellido Materno: Hernandez
Fecha de nacimiento: 1993-3-2
Telefono: 7782342344
Genero (M/F): M
Se agrego el usuario con ID 2
a. Alta usuario
b. Consulta usuario
c. Borra usuario
d. Salir
Opcion:
```

Figura 17: Compilación y ejecución: Opción a. Alta usuario



```
vazpeitia@X2017350201:~/distribuidos/tarea8/cliente$ make run
java -cp .:* Cliente
a. Alta usuario
b. Consulta usuario
c. Borra usuario
d. Salir
Opcion: b
Consulta usuario
Ingresa el ID de usuario: 2
Nombre: Vladimir
Apellido Paterno: Azpeitia
Apellido Materno: Hernandez
Fecha: 1993-03-02
Genero: M
a. Alta usuario
b. Consulta usuario
c. Borra usuario
d. Salir
Opcion:
```

Figura 18: Compilación y ejecución: Opción b. Consulta usuario



```
vazpeitiah@X2017350201: ~/distribuidos/tarea8/cliente$ make run
java -cp .:* Cliente
a. Alta usuario
b. Consulta usuario
c. Borra usuario
d. Salir
Opcion: c
Borrar usuario
Ingresa el ID de usuario: 2
El usuario ha sido borrado
a. Alta usuario
b. Consulta usuario
c. Borra usuario
d. Salir
Opcion:
```

Figura 19: Compilación y ejecución: Opción a. Borra usuario

4. Conclusiones

En esta práctica se observa el comportamiento de un servicio web y un cliente que lo consume en una máquina local en cualquier parte del mundo. Es una fuerte herramienta porque nos muestra como es el mundo real de las aplicaciones que consumen los servicios web en la nube y en un servidor. Por lo que también se pueden hacer aplicaciones para android que requieren consumir servicios web REST y que por consiguiente tienen un alcance bastante amplio en el mundo de la web. Hablando del código, solo tenemos que acoplarnos y saber qué es lo que se está mandando y qué está recibiendo el servidor, ya que si no son compatibles los tipos de datos es posible que haya una ruptura o no funcione correctamente la aplicación que consume el servicio.