

Instituto Politécnico Nacional
Escuela Superior de Cómputo

Desarrollo de un sistema utilizando un servicio web

Tarea 8

Alumno: Francisco Nicolas Sánchez García
Asignatura Desarrollo de Sistemas Distribuidos
Profesor: Carlos Pineda Guerrero
Grupo 4CV12

ÍNDICE

Introducción	1
Desarrollo	1
Conclusiones	21

INTRODUCCIÓN

El uso de servicios web para la automatización de tareas mediante la creación de sistemas se ha vuelto un conocimiento esencial en nuestra carrera, nos interesa conocer como es el diseño de estas independientemente de la tecnología a usar. Como vimos en clases pasadas y gracias a la retroalimentación de los compañeros que van más avanzados pudimos llegar a ciertos acuerdos como el número de métodos que se tienen en el servicio y la forma y tipo de datos que se tienen en la base de datos, gracias a esto y a conocimientos previos obtenidos en materias de desarrollo web y diseño orientado a objetos a continuación se presenta la solución al sistema, se muestra el código fuente del back end, la compilación de este y del front end para finalmente mostrar las pruebas en un dispositivo móvil.

DESARROLLO

Se presenta el código del servicio y de la base de datos, de igual forma, se pueden encontrar dentro de la carpeta zip con el código faltante.

Código fuente de Servicio.java

Se tienen cuatro métodos en el servicio web, el primero sirve para dar de alta un artículo, para consultar artículos usando LIKE de SQL, para ver el carrito y para agregar artículos al carrito.

```
package negocio;

import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Consumes;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.QueryParam;
import javax.ws.rs.FormParam;
import javax.ws.rs.core.Response;

import java.sql.*;
import javax.sql.DataSource;
import javax.naming.Context;
import javax.naming.InitialContext;

import java.util.ArrayList;
import com.google.gson.*;

// la URL del servicio web es http://localhost:8080/Servicio/rest/ws
// donde:
//     "Servicio" es el dominio del servicio web (es decir, el nombre de archivo Servicio.war)
//     "rest" se define en la etiqueta <url-pattern> de <servlet-mapping> en el archivo WEB-
INF\web.xml
//     "ws" se define en la siguiente anotación @Path de la clase Servicio
```

```

@Path("/ws")
public class Servicio {
    static DataSource pool = null;
    static {
        try {
            Context ctx = new InitialContext();
            pool = (DataSource) ctx.lookup("java:comp/env/jdbc/datasource_Servicio");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    static Gson j = new GsonBuilder().registerTypeAdapter(byte[].class, new AdaptadorGsonBase64())
        .setDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS").create();

    @POST
    @Path("alta_articulo")
    @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
    @Produces(MediaType.APPLICATION_JSON)
    public Response alta(@FormParam("articulo") Articulo articulo) throws Exception {
        Connection conexion = pool.getConnection();

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

        try {
            PreparedStatement stmt_1 = conexion
                .prepareStatement("select id_articulo from articulos where descripcion=?");

            try {
                stmt_1.setString(1, articulo.descripcion);
                ResultSet rs = stmt_1.executeQuery();
                try {
                    if (rs.next())
                        return Response.status(400)
                            .entity(j.toJson(new Error("ya hay un articulo con esa
descripcion"))).build();
                } finally {
                    rs.close();
                }
            } finally {
                stmt_1.close();
            }

            conexion.setAutoCommit(false);
            // alta
            PreparedStatement stmt_2 = conexion.prepareStatement("insert into articulos VALUES
(0,?, ?, ?)");

            try {
                stmt_2.setString(1, articulo.descripcion);
                stmt_2.setFloat(2, articulo.precio);
                stmt_2.setInt(3, articulo.cantidad);
                stmt_2.executeUpdate();
            } finally {
                stmt_2.close();
            }

            if (articulo.imagen != null) {
                PreparedStatement stmt_3 = conexion.prepareStatement(
                    "insert into imagenes_articulo values (0,?, (select id_articulo from
articulos where descripcion=?))");
                try {
                    stmt_3.setBytes(1, articulo.imagen);
                    stmt_3.setString(2, articulo.descripcion);
                    stmt_3.executeUpdate();
                } finally {
                    stmt_3.close();
                }
            }

            conexion.commit();
        } catch (Exception e) {
            conexion.rollback();
            return Response.status(400).entity(j.toJson(new Error(e.getMessage()))).build();
        } finally {
    
```

```

        conexion.setAutoCommit(true);
        conexion.close();
    }
    return Response.ok().build();
}

@POST
@Path("consulta_articulos")
@Consumes(MediaType.APPLICATION_FORM_URLENCODED)
@Produces(MediaType.APPLICATION_JSON)
public Response consulta(@FormParam("busqueda") String busqueda) throws Exception {
    Connection conexion = pool.getConnection();

    try {
        PreparedStatement stmt_1 = conexion.prepareStatement(
            "SELECT a.descripcion, a.precio, b.imagen FROM articulos a LEFT OUTER JOIN
imagenes_articulo b ON a.id_articulo=b.id_articulo WHERE descripcion LIKE ?");
        try {
            stmt_1.setString(1, '%' + busqueda + '%');
            ResultSet rs = stmt_1.executeQuery();
            try {
                ArrayList<Articulo> articulos = new ArrayList<Articulo>();
                while (rs.next()) {
                    Articulo art = new Articulo();
                    art.descripcion = rs.getString(1);
                    art.precio = rs.getFloat(2);
                    art.imagen = rs.getBytes(3);
                    articulos.add(art);
                }

                if (articulos.isEmpty()) {
                    return Response.status(201).entity(j.toJson("No se encontraron articulos
con esa descripcion"))
                        .build();
                } else {
                    return Response.ok().entity(j.toJson(articulos)).build();
                }
            } finally {
                rs.close();
            }
        } finally {
            stmt_1.close();
        }
    } catch (Exception e) {
        return Response.status(400).entity(j.toJson(new Error(e.getMessage()))).build();
    } finally {
        conexion.close();
    }
}

@POST
@Path("alta_carrito")
@Consumes(MediaType.APPLICATION_FORM_URLENCODED)
@Produces(MediaType.APPLICATION_JSON)
public Response altaCarrito(@FormParam("descripcion") String descripcion,
@FormParam("cantidad") int cantidad)
    throws Exception {
    Connection conexion = pool.getConnection();
    int cantidad_actual = -1;
    int id_articulo = -1;
    try {
        PreparedStatement stmt_1 = conexion
            .prepareStatement("select id_articulo, cantidad from articulos where
descripcion=?");
        try {
            stmt_1.setString(1, descripcion);
            ResultSet rs = stmt_1.executeQuery();
            try {
                if (rs.next()) {
                    id_articulo = rs.getInt(1);
                    cantidad_actual = rs.getInt(2);
                } else {
                    return Response.status(400)
                        .entity(j.toJson(new Error("No hay un articulo con esa
descripcion"))).build();
                }
            }
        }
    }
}

```

```

        } finally {
            rs.close();
        }
    } finally {
        stmt_1.close();
    }

    if (cantidad <= cantidad_actual) {
        conexion.setAutoCommit(false);
        // update de cantidad Articulos
        PreparedStatement stmt_2 = conexion
            .prepareStatement("UPDATE articulos SET cantidad=? WHERE id_articulo=?");
        try {
            stmt_2.setInt(1, cantidad_actual - cantidad);
            stmt_2.setInt(2, id_articulo);
            stmt_2.executeUpdate();
        } finally {
            stmt_2.close();
        }

        // agregar al carrito
        PreparedStatement stmt_3 = conexion.prepareStatement("insert into carrito_compra
VALUES (0,?,?)");
        try {
            stmt_3.setInt(1, id_articulo);
            stmt_3.setInt(2, cantidad);
            stmt_3.executeUpdate();
        } finally {
            stmt_3.close();
        }
        conexion.commit();
    } else {
        return Response.status(201).entity(j.toJson(cantidad_actual)).build();
    }
} catch (Exception e) {
    return Response.status(400).entity(j.toJson(new Error(e.getMessage()))).build();
} finally {
    conexion.setAutoCommit(true);
    conexion.close();
}
return Response.ok().build();
}

@POST
@Path("/consulta_carrito")
@Consumes(MediaType.APPLICATION_FORM_URLENCODED)
@Produces(MediaType.APPLICATION_JSON)
public Response consultaCarrito() throws Exception {
    Connection conexion = pool.getConnection();

    try {
        PreparedStatement stmt_1 = conexion.prepareStatement(
            "select a.descripcion, a.precio, b.imagen, c.cantidad from articulos a left
outer join imagenes_articulo b on a.id_articulo=b.id_articulo left outer join carrito_compra c on
a.id_articulo=c.id_articulo where c.cantidad is not null");
        try {
            ResultSet rs = stmt_1.executeQuery();
            try {
                ArrayList<Articulo> articulos = new ArrayList<Articulo>();
                while (rs.next()) {
                    Articulo art = new Articulo();
                    art.descripcion = rs.getString(1);
                    art.precio = rs.getFloat(2);
                    art.imagen = rs.getBytes(3);
                    art.cantidad = rs.getInt(4);
                    articulos.add(art);
                }

                if (articulos.isEmpty()) {
                    return Response.status(202).entity(j.toJson("No se encontraron articulos
en el carrito"))
                        .build();
                } else {
                    return Response.ok().entity(j.toJson(articulos)).build();
                }
            }
        }
    }
}

```

```

        } finally {
            rs.close();
        }
    } finally {
        stmt_1.close();
    }
} catch (Exception e) {
    return Response.status(400).entity(j.toJson(new Error(e.getMessage()))).build();
} finally {
    conexion.close();
}
}
}
}

```

Código fuente de la base de datos

Se tienen tres tablas, artículos, imágenes_articulo y carrito_compra como se plantea en los requerimientos funcionales.

```

--Como root
grant all on carrito.* to hugo@localhost;

--como hugo
create database carrito;
use carrito;

create table articulos(
    id_articulo integer auto_increment primary key,
    descripcion varchar(256) not null,
    precio float not null,
    cantidad integer not null
);

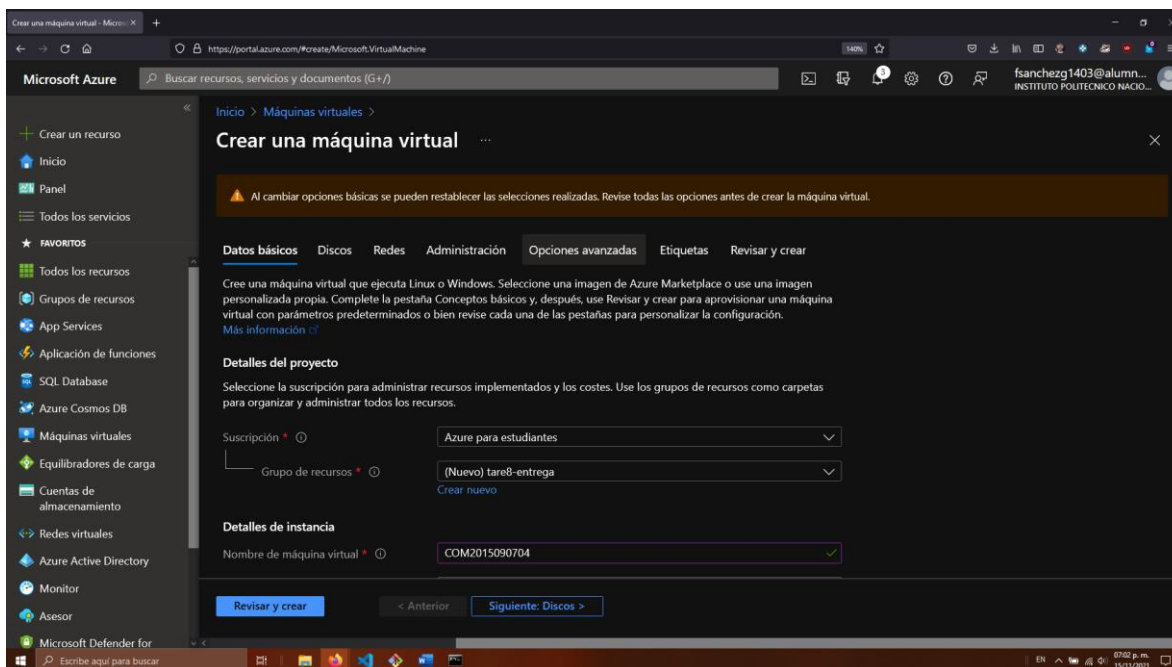
create table imagenes_articulo(
    id_imagen integer auto_increment primary key,
    imagen longblob,
    id_articulo integer not null
);

alter table imagenes_articulo add foreign key(id_articulo) references
articulos(id_articulo);
create unique index articulo_1 on articulos(descripcion);
-- alter table carrito_compra add foreign key(id_articulo) references
articulos(id_articulo);

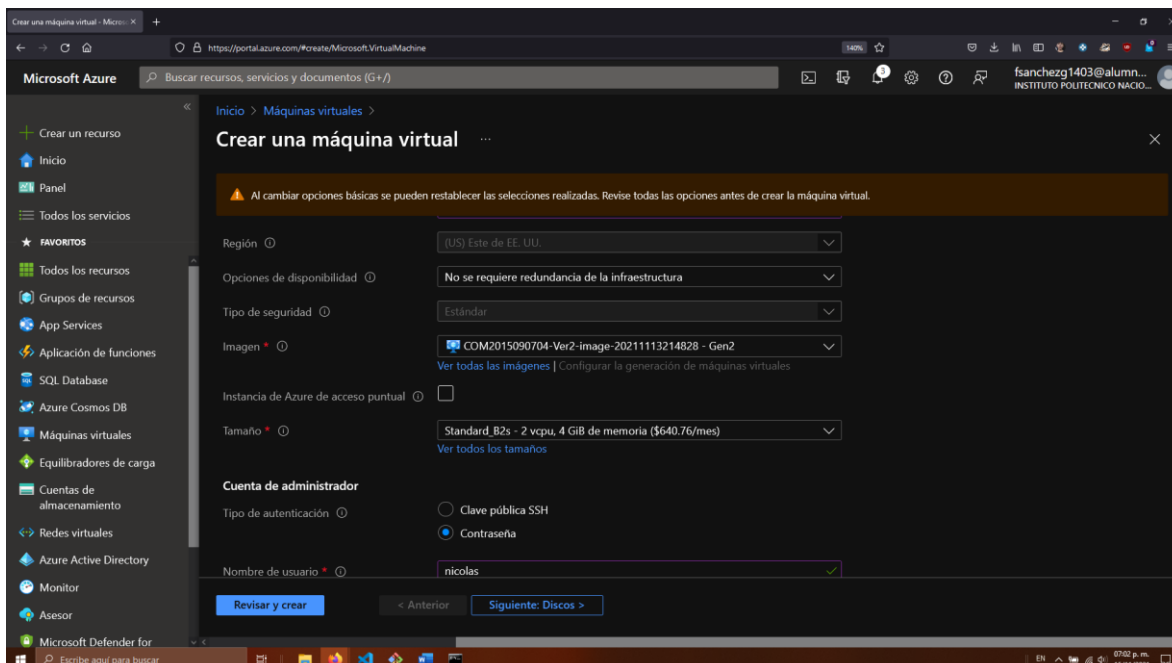
--manejo de articulos en carrito
create table carrito_compra(
    id_compra integer auto_increment primary key,
    id_articulo integer not null,
    cantidad integer not null
);
alter table carrito_compra add foreign key(id_articulo) references articulos(id_articulo);

```

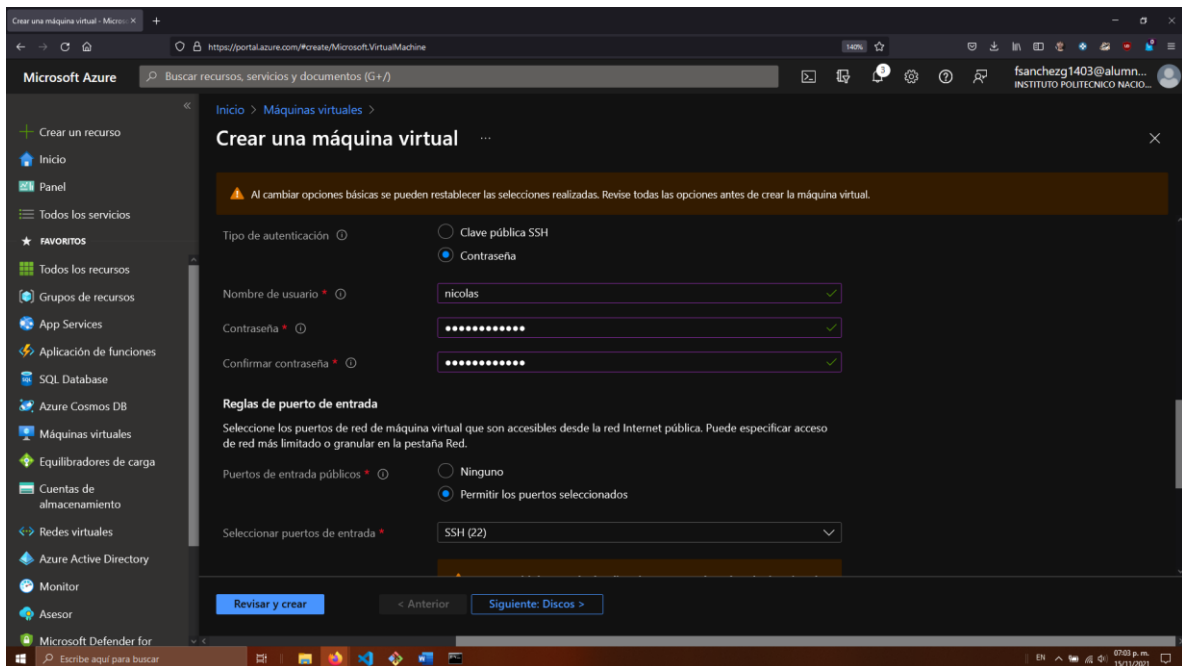
Creación de máquina virtual



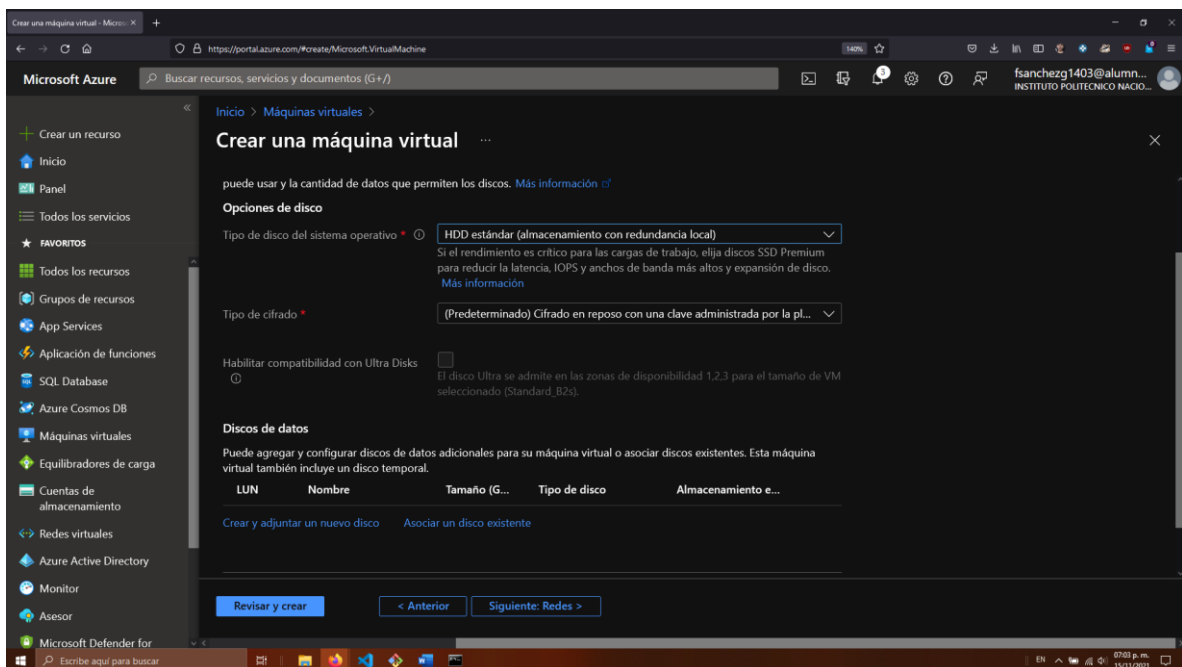
Para crear la máquina virtual se crea un nuevo grupo de recursos para la VM llamado “tarea8-entrega”, se agrega el nombre de la máquina siguiendo la nomenclatura de la tarea (COM2015090704).



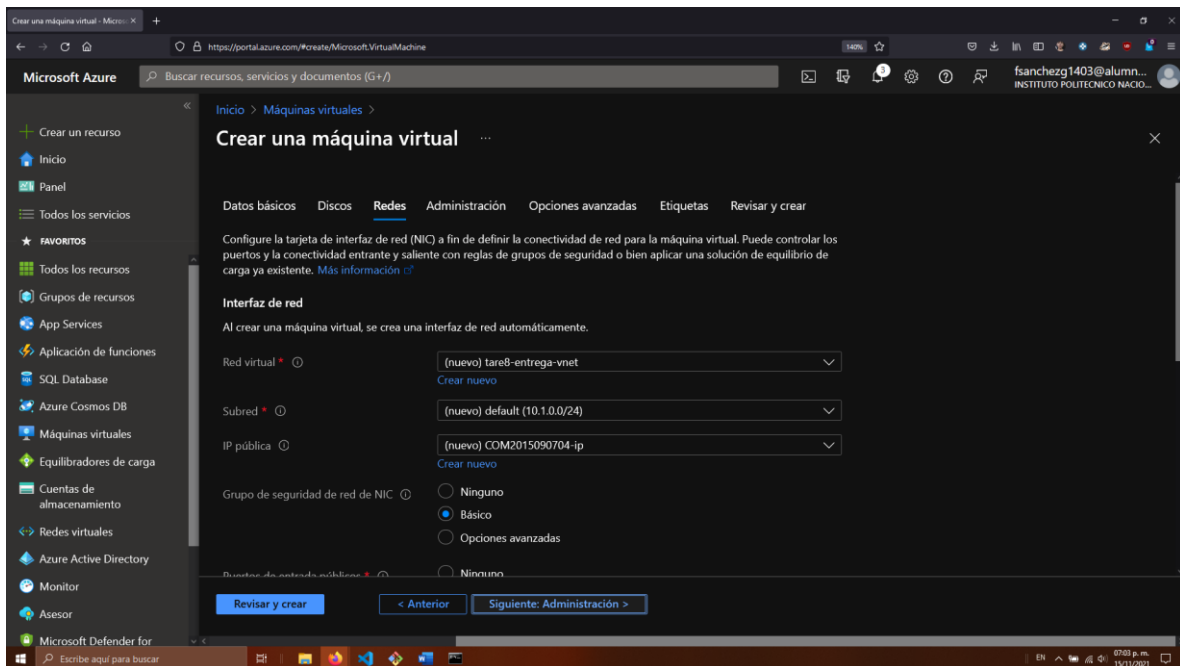
Se selecciona el tamaño B2s ya que tiene 4GB de memoria RAM. Se selecciona la imagen creada anteriormente con Tomcat, Java y toda la configuración hecha en clase.



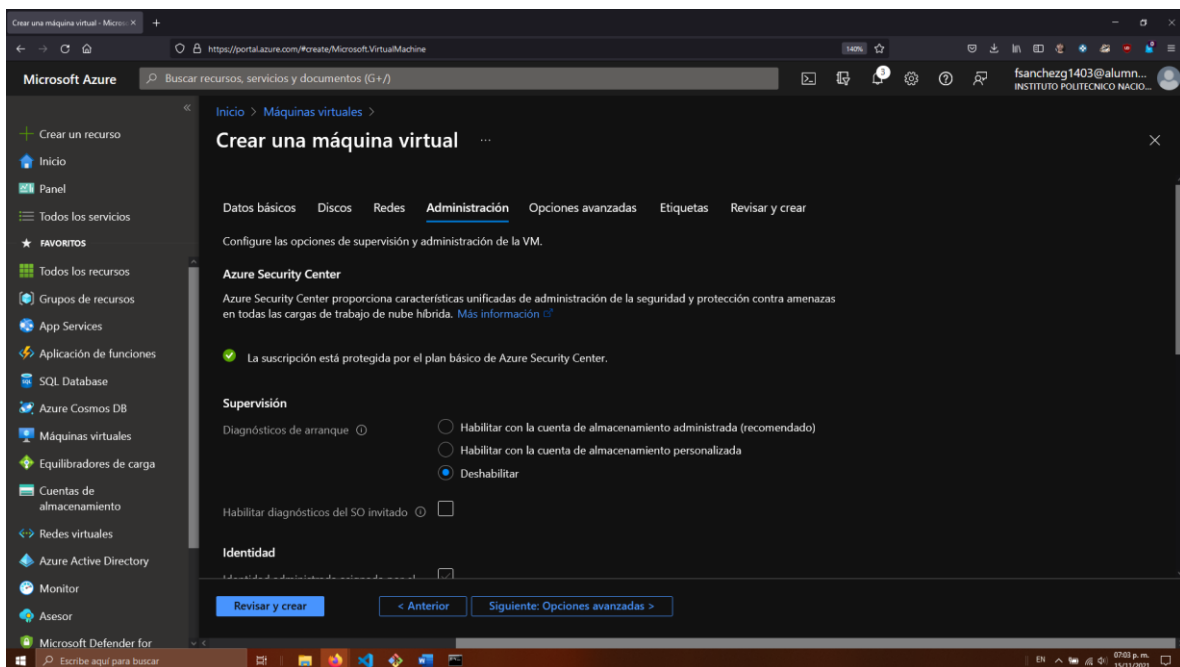
Se agrega usuario y contraseña y se deja el puerto SSH para conectarnos y pasamos a “Discos”.



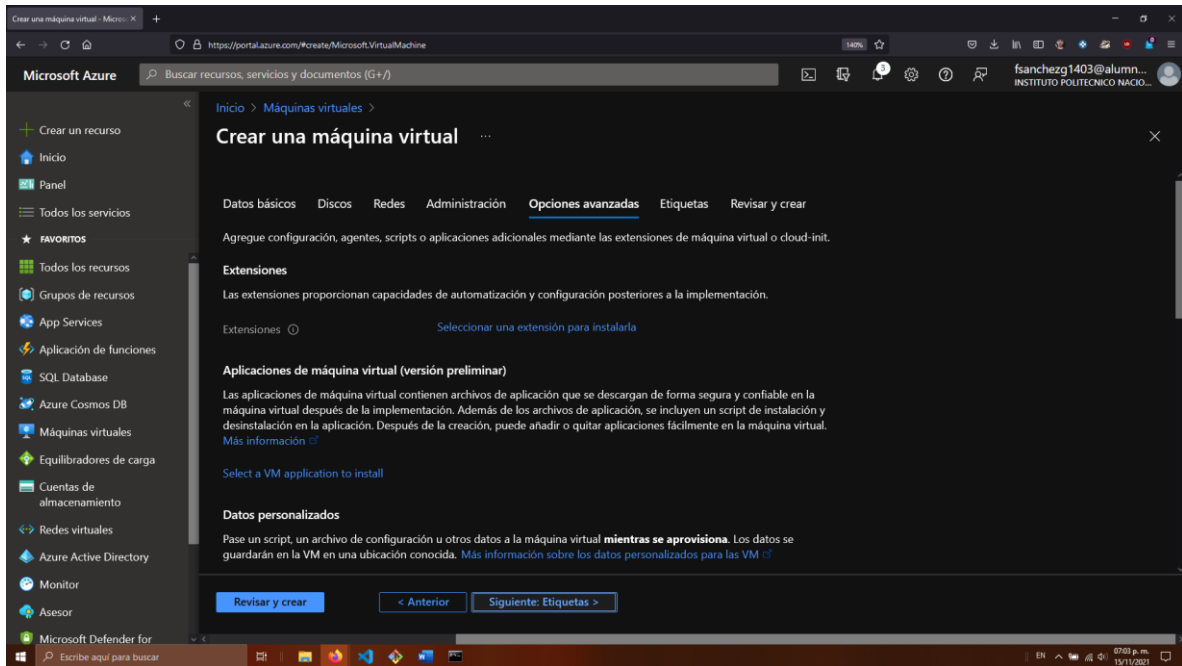
Cambiamos el tipo de disco del sistema operativo a un “HDD estándar”. No se modifica nada de “Opciones avanzadas” y pasamos al apartado de “Redes”.



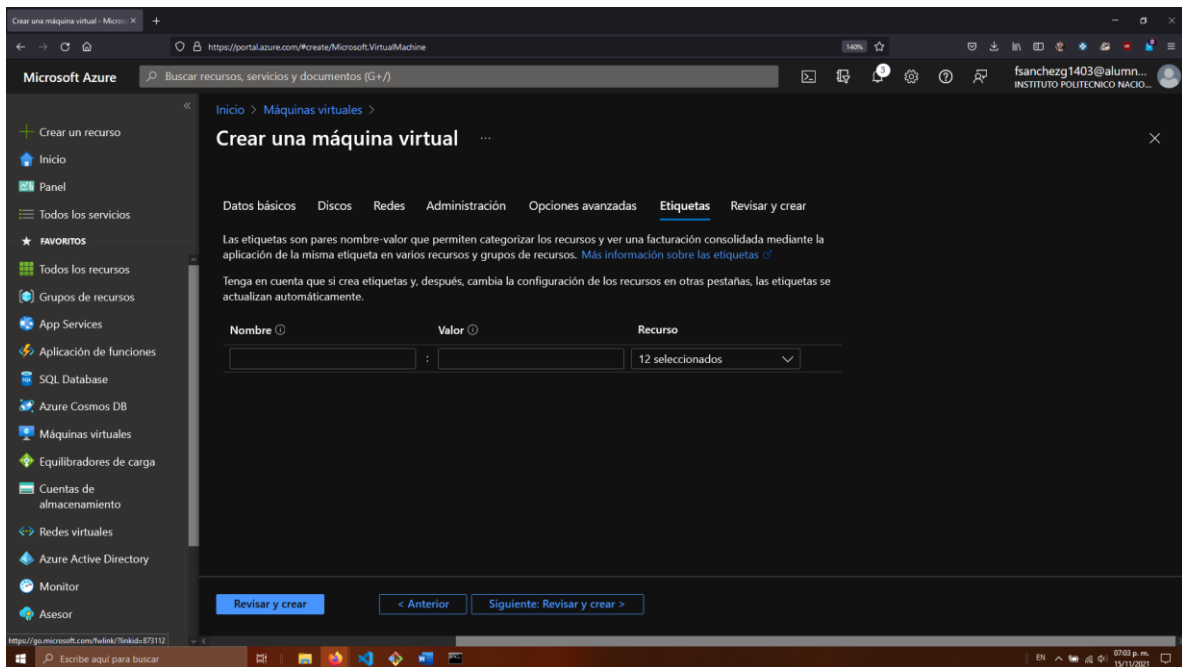
Dejamos los valores generados por defecto en el apartado de “Redes”. Pasamos al apartado de “Administración”.



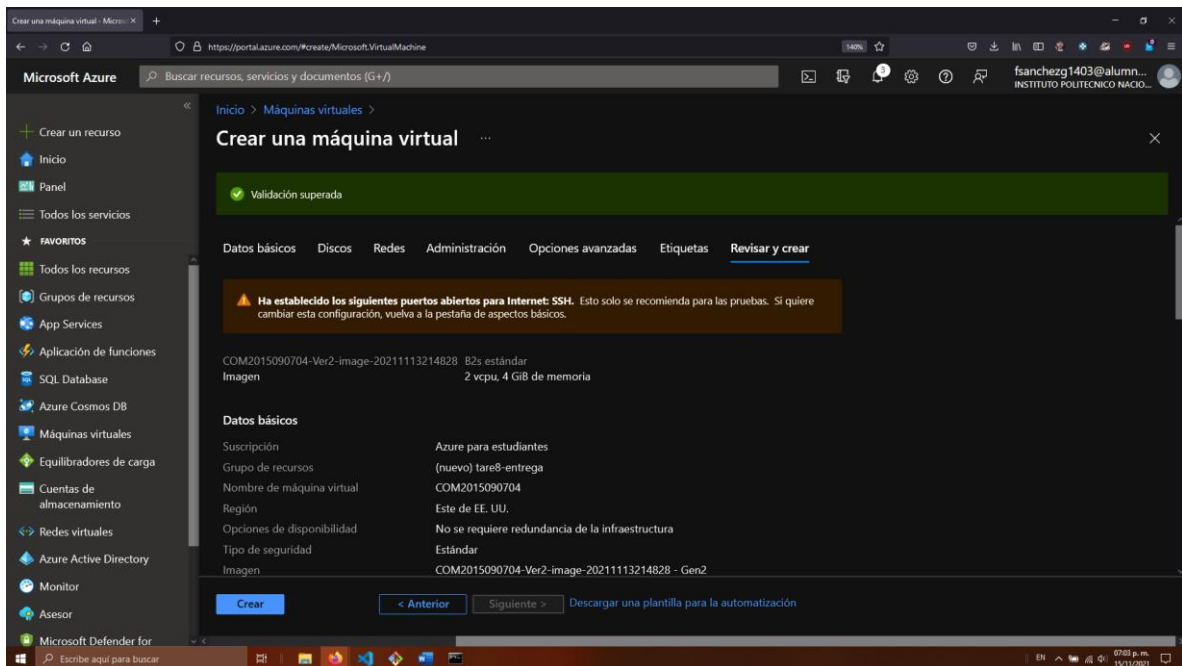
Se deshabilita el diagnóstico de arranque de la máquina virtual. No se modifica ninguna otra opción y pasamos al apartado de “Opciones avanzadas”.



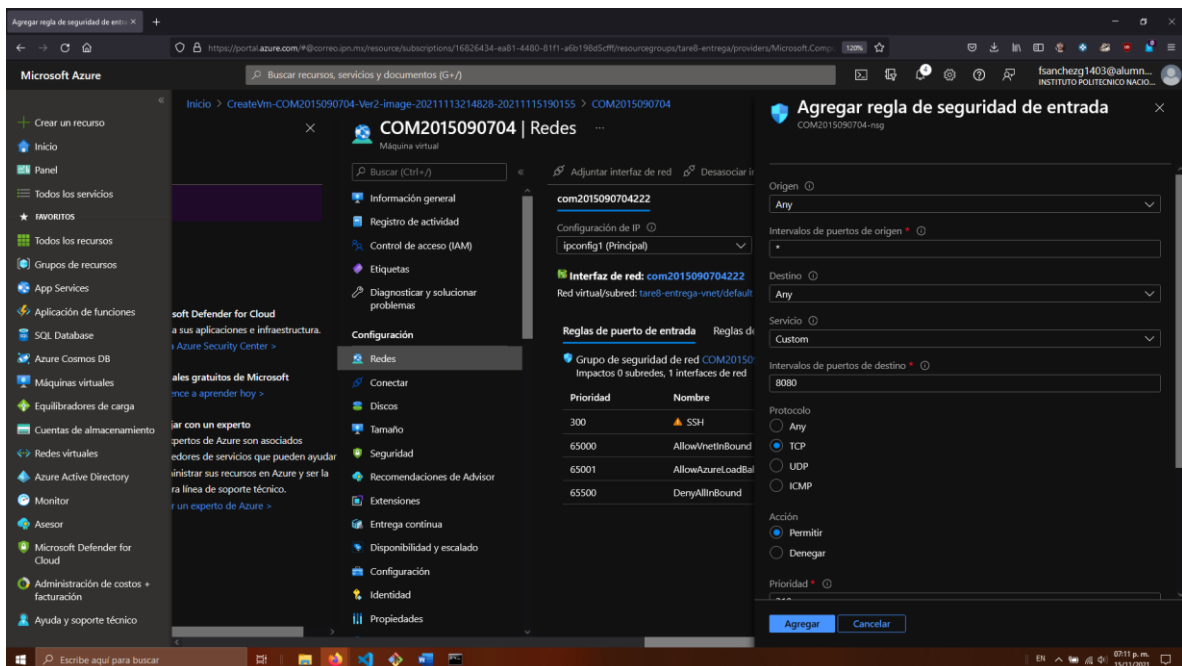
No se modifica nada del apartado de “Opciones avanzadas”. Pasamos al apartado de “Etiquetas”.



No se agrega ninguna etiqueta. Pasamos al apartado de “Revisar y crear”.



Una vez que la validación ha sido superada, confirmamos los datos anteriormente ingresados. Todos los datos de la máquina virtual son los que ingresamos anteriormente. Una vez revisado todos los campos, damos clic en “Crear”.



Agregamos la regla para abrir el puerto 8080 para el servicio de Tomcat.

Compilación del servidor

Código fuente del script.

```
#!/bin/bash
#agregar en /usr/bin/
#chmod +x /usr/bin/archivo.sh
#alias compilar='/usr/bin/compilar_servicio.sh'
#Compilacion
echo "Compilando clase Servicio..."
# export CATALINA_HOME

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 WEB-INF/classes/negocio/*
cp negocio/*.class WEB-INF/classes/negocio/.
echo "Creando archivo war..."
jar cvf Servicio.war WEB-INF META-INF

#apagar servidor
echo "apagando tomcat..."
sh $CATALINA_HOME/bin/catalina.sh stop

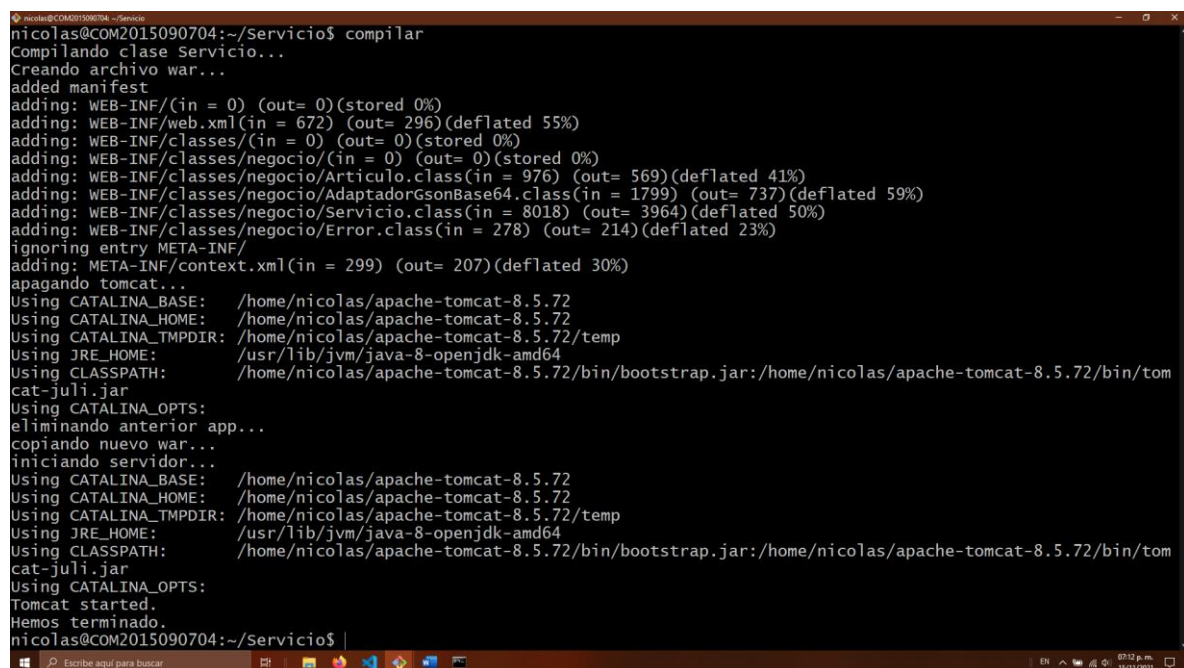
echo "eliminando anterior app..."
#limpiar lo del servidor
rm $CATALINA_HOME/webapps/Servicio.war
rm -R $CATALINA_HOME/webapps/Servicio

#copia a tomcat
echo "copiando nuevo war..."
cp Servicio.war $CATALINA_HOME/webapps/

#iniciar servidor
echo "iniciando servidor..."
sh $CATALINA_HOME/bin/catalina.sh start

echo "Hemos terminado."
```

Se crea un alias llamado compilar para no tener que estar escribiendo toda la ruta del script.



```
nicolas@COM2015090704:~/Servicio$ compilar
Compilando clase Servicio...
Creando archivo war...
added 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/Articulo.class(in = 976) (out= 569)(deflated 41%)
adding: WEB-INF/classes/negocio/AdaptadorGsonBase64.class(in = 1799) (out= 737)(deflated 59%)
adding: WEB-INF/classes/negocio/Servicio.class(in = 8018) (out= 3964)(deflated 50%)
adding: WEB-INF/classes/negocio/Error.class(in = 278) (out= 214)(deflated 23%)
ignoring entry META-INF/
adding: META-INF/context.xml(in = 299) (out= 207)(deflated 30%)
apagando tomcat...
Using CATALINA_BASE:   /home/nicolas/apache-tomcat-8.5.72
Using CATALINA_HOME:   /home/nicolas/apache-tomcat-8.5.72
Using CATALINA_TMPDIR: /home/nicolas/apache-tomcat-8.5.72/temp
Using JRE_HOME:        /usr/lib/jvm/java-8-openjdk-amd64
Using CLASSPATH:       /home/nicolas/apache-tomcat-8.5.72/bin/bootstrap.jar:/home/nicolas/apache-tomcat-8.5.72/bin/tomcat-juli.jar
Using CATALINA_OPTS:
eliminando anterior app...
copiando nuevo war...
iniciando servidor...
Using CATALINA_BASE:   /home/nicolas/apache-tomcat-8.5.72
Using CATALINA_HOME:   /home/nicolas/apache-tomcat-8.5.72
Using CATALINA_TMPDIR: /home/nicolas/apache-tomcat-8.5.72/temp
Using JRE_HOME:        /usr/lib/jvm/java-8-openjdk-amd64
Using CLASSPATH:       /home/nicolas/apache-tomcat-8.5.72/bin/bootstrap.jar:/home/nicolas/apache-tomcat-8.5.72/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
Hemos terminado.
nicolas@COM2015090704:~/Servicio$
```

Escribí un script para automatizar la compilación del lado del servidor, el cuál consta de la compilación de la clase Servidor.java, el borrado de las clases anteriores en las carpetas del servicio, la copia de los nuevos archivos compilados, la generación del archivo .war, el apagado del servidor, la eliminación del anterior archivo war dentro de tomcat y la carpeta generada por este, luego, copiamos el nuevo war y se inicia el servidor de Tomcat de nuevo.

Compilación del cliente

```
C:\Users\fnico\Documents\github\desarrolloDeSistemasDistribuidos\tareas\carrito\cliente>npm run build

> cliente@0.1.0 build C:\Users\fnico\Documents\github\desarrolloDeSistemasDistribuidos\tareas\carrito\cliente
> react-scripts build

Creating an optimized production build...
Compiled with warnings.

src\components\Carrito.js
  Line 4:8:  'Menu' is defined but never used  no-unused-vars
  Line 6:10: 'useRef' is defined but never used  no-unused-vars

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.

File sizes after gzip:

  51.99 KB  build\static\js\2.13ce1cf6.chunk.js
   1.82 KB  build\static\js\main.0bad815d.chunk.js
    774 B   build\static\js\runtime-main.0adf6f3d.js
    531 B   build\static\css\main.8c8b27cf.chunk.css

The project was built assuming it is hosted at /.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.
You may serve it with a static server:

  npm install -g serve
```

El cliente o front end fue desarrollado usando la librería React JS.

```
531 B   build\static\css\main.8c8b27cf.chunk.css

The project was built assuming it is hosted at /.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.
You may serve it with a static server:

  npm install -g serve
  serve -s build

Find out more about deployment here:

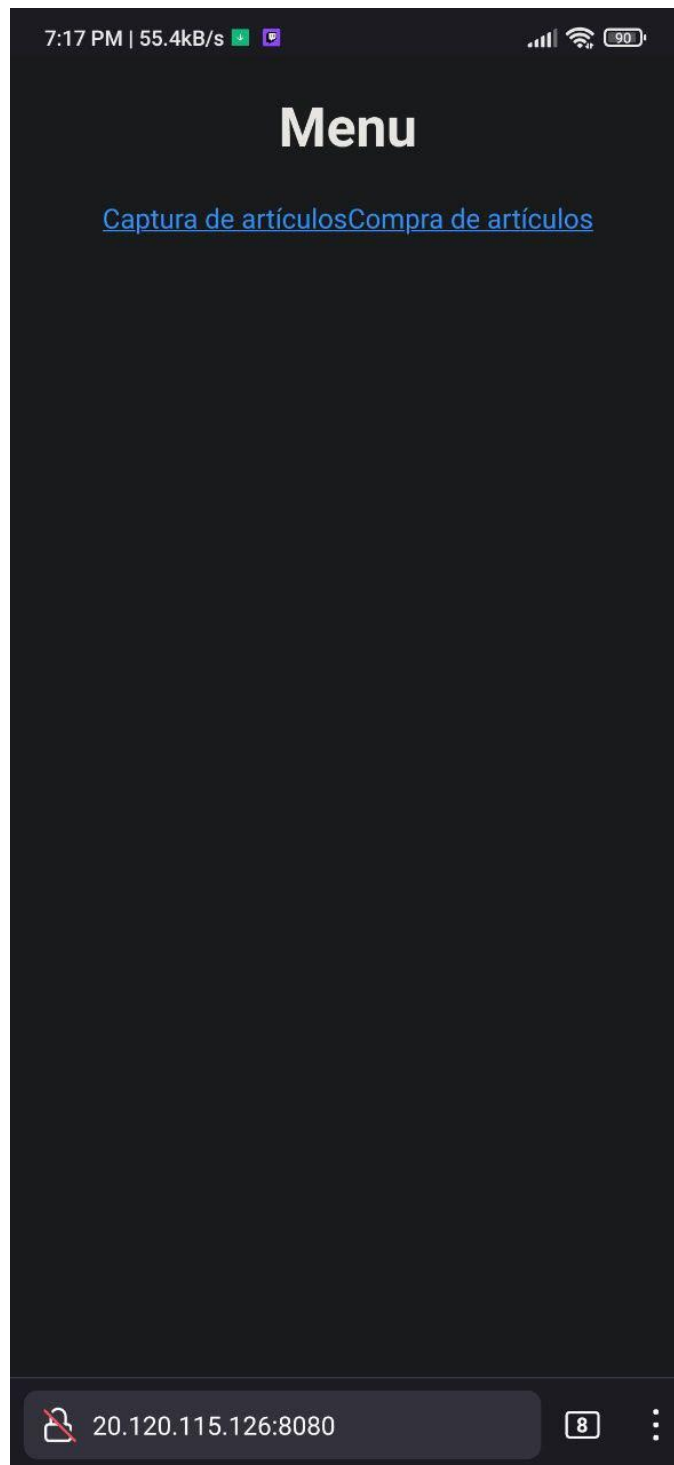
  https://cra.link/deployment

C:\Users\fnico\Documents\github\desarrolloDeSistemasDistribuidos\tareas\carrito\cliente>




npm install -g serve
```

Se muestra como se hace el build que genera los archivos necesarios que se agregan en la carpeta de ROOT/ de Tomcat.

Pruebas



Se muestra el menú de inicio con las opciones para agregar artículos y para comprar artículos.

7:18 PM | 0.4kB/s   

Menu


[Captura de artículos](#)[Compra de artículos](#)



Añadir artículo

Descripción del artículo:

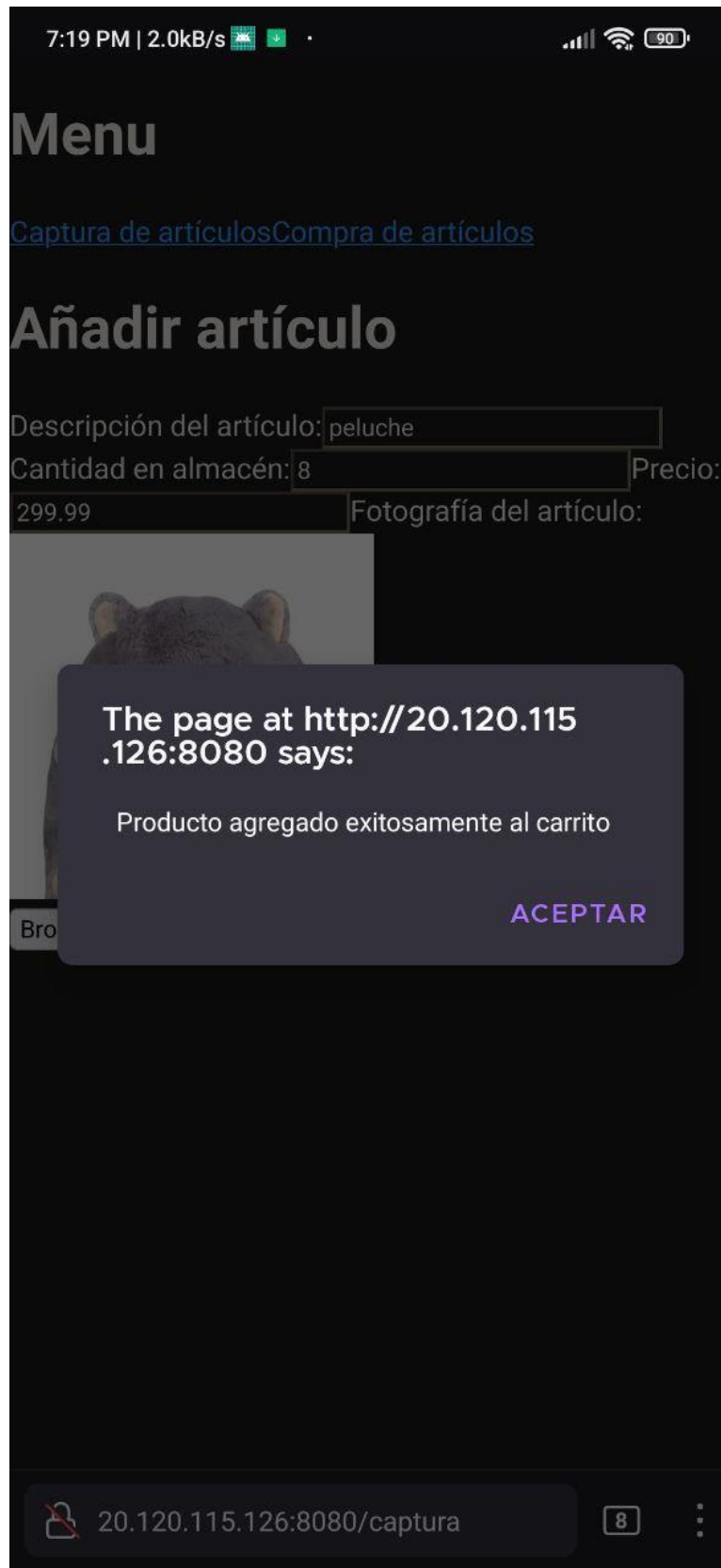
Cantidad en almacén: Precio:

Fotografía del artículo:



 20.120.115.126:8080/captura 8 

Se Añade un artículo a la tabla artículos.



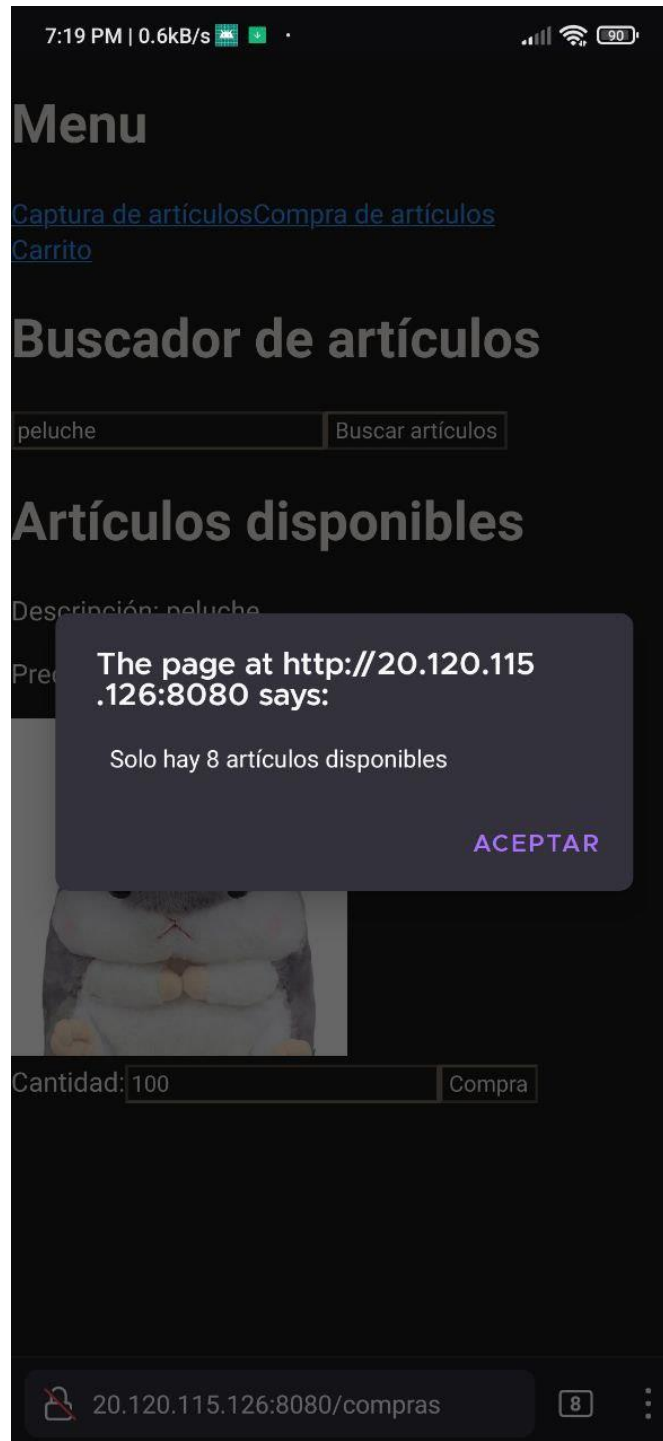
Se muestra el mensaje de que se agregó exitosamente.



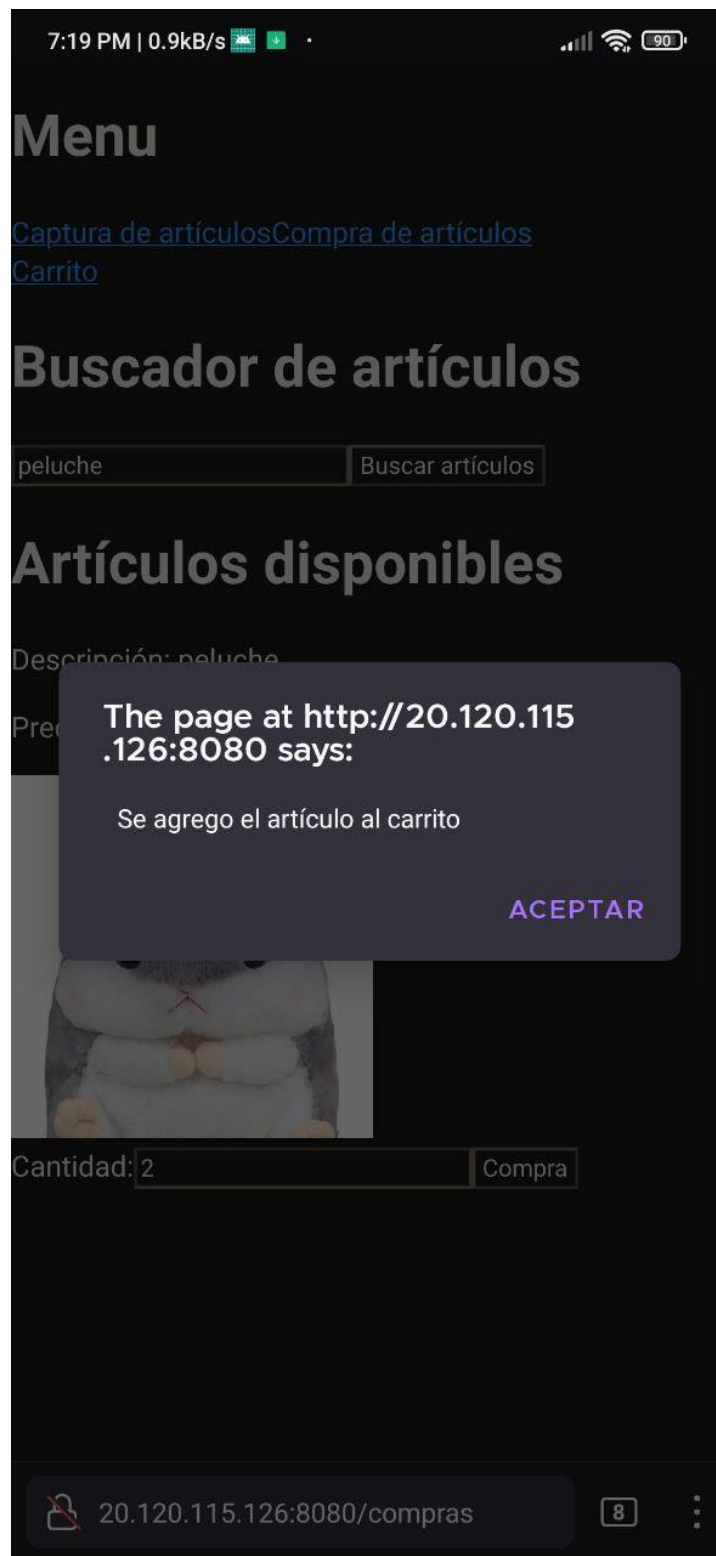
Si damos clic en “Compra de artículos” nos muestra un buscador y los artículos disponibles, como no se ha buscado ningún artículo no se muestra nada.






Si buscamos nuestro artículo nos aparece su descripción, precio, imagen agregada y un input para escribir la cantidad que se quiere comprar, además del botón de compra.



Si solicitamos una cantidad mayor a la que hay en almacén se muestra con una alerta el número máximo que hay disponible.











Ahora, seleccionamos dos artículos del peluche y damos clic en compra, recibimos un mensaje de que el artículo fue agregado al carrito.

7:19 PM | 1.1MB/s   

[Compra de artículos](#)

Carrito de compras

Imagen	Descripción	Cantidad	Precio total
	bocina triple	3	299.96999999999997
	botella de agua	3	35.97
	bocina	1	29.99
	cubo rubik	2	265
	peluche	2	599.98

 20.120.115.126:8080/carrito  8 

Si damos clic en el carrito se nos muestra los artículos en el carrito en forma de tabla, primero la imagen, luego la descripción, cantidad y precio total del artículo por su cantidad.

CONCLUSIONES

Una de las principales dificultades que pensé que tendría a la hora de desarrollar el front end era con las políticas de CORS que no permiten enviar información entre diferentes sitios, para solucionar esto, para realizar pruebas en el front end usaba respuestas reales que enviaba el back end, de forma que el build del front end solo lo hacía para revisar el correcto funcionamiento. Gracias a mi experiencia previa usando React JS en proyectos escolares y personal me facilitó la creación del lado del cliente. Finalmente, me apoyé del código mostrado por el profesor para el manejo de imágenes a su base64 y la creación de la solicitud a partir de un objeto JSON.

Como tal, no encontré una lista de pruebas a realizar, por ende, traté de que todos los puntos funcionales se mostraran en las capturas de pantalla.