



Instituto Politécnico Nacional  
Escuela Superior de Computo

---



Sistemas Distribuidos

# Desarrollo de un sistema utilizando un servicio web

Profesor: Pineda Guerrero Carlos

Grupo: 3CV14

Bautista García Hadad

# Introducción

## ¿Qué es una API de REST?

Una API de REST, o API de RESTful, es una interfaz de programación de aplicaciones (API o API web) que se ajusta a los límites de la arquitectura REST y permite la interacción con los servicios web de RESTful. El informático Roy Fielding es el creador de la transferencia de estado representacional (REST).

Las API son conjuntos de definiciones y protocolos que se utilizan para diseñar e integrar el software de las aplicaciones. Suele considerarse como el contrato entre el proveedor de información y el usuario, donde se establece el contenido que se necesita por parte del consumidor (la llamada) y el que requiere el productor (la respuesta).

## GSON

GSON es un API en Java, desarrollada por Google, que se utiliza para convertir objetos Java a JSON (serialización) y JSON a objetos Java (deserialización).

Esta librería estructura los JSON de la siguiente manera:

- **JsonElement:** Esta clase representa cualquier elemento del Json que puede ser de alguno de los siguientes 4 tipos:
  1. **JsonObject:** Esta clase representa un objeto en el Json; es decir, un conjunto de pares clave-valor donde las claves son strings y los valores son cualquier otro tipo de JsonElement.
  2. **JsonArray:** Esta clase representa un array en el Json. Un array es una lista de JsonElements cada uno de los cuales puede ser de un tipo diferente. Se trata de una lista ordenada, por lo que el orden en que se añaden los elementos se conserva.
  3. **JsonPrimitive:** Esta clase representa un tipo de dato primitivo u objetos de datos simples (String, Integer, Double, etc.).
  4. **JsonNull:** Representa un objeto a null.

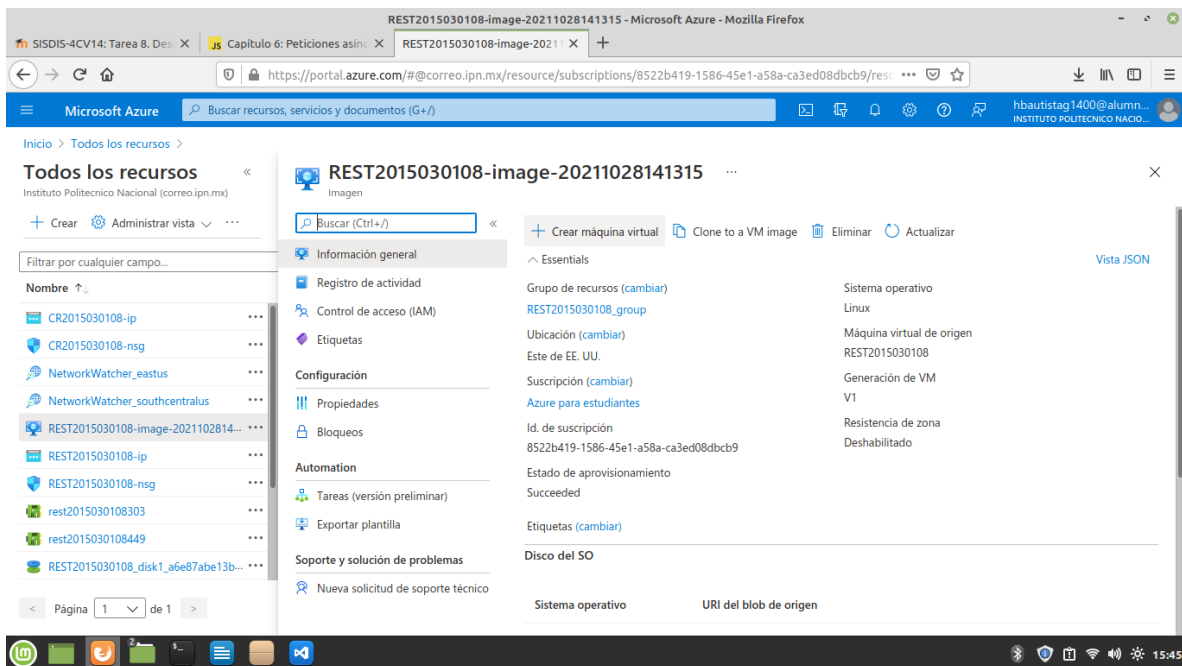
## Peticiones asíncronas

Una petición asíncrona es una operación que, mientras esté siendo procesada, deja libre al navegador para que pueda hacer otras operaciones. Llamaremos peticiones asíncronas a las operaciones que tengan que ver con realizadas llamadas a servidores; sin embargo, existen muchas más operaciones asíncronas en JavaScript, como las que se realizan para leer y escribir en archivos, obtener la geolocalización de un navegador, o manejar base de datos.

# Desarrollo

## Creación de la maquina virtual

En esta ocacion crearemos la maquina virtual a partir de una imagen creada en una practica anterior , con esto nos ahorraremos un par de pasos para la ejecucion de esta practica en la captura podemos observar como seleccionamos la imagen para desde ahi empezar a crear la maquina virtual



Una vez seleccionada la opción de crear maquina virtual se nos muestra un panel similar al panel que se nos muestra al crear una maquina virtual así que procedemos a crearla , inicialmente configuramos el campo de nombre de la maquina virtual

Crear una máquina virtual - Microsoft Azure - Mozilla Firefox

Inicio > Todos los recursos > REST2015030108-image-20211028141315 >

### Crear una máquina virtual

Cree una máquina virtual que ejecuta Linux o Windows. Seleccione una imagen de Azure Marketplace o use una imagen personalizada propia. Complete la pestaña Conceptos básicos y, después, use Revisar y crear para aprovisionar una máquina virtual con parámetros predeterminados o bien revise cada una de las pestañas para personalizar la configuración.

[Más información](#)

**Detalles del proyecto**

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

Suscripción \*

Grupo de recursos \*   
[Crear nuevo](#)

**Detalles de instancia**

Nombre de máquina virtual \*

Región

[Revisar y crear](#) < Anterior Siguiente: Discos >

Después seleccionamos el tamaño de nuestra maquina así como el tipo de autenticación , en este caso se realizara mediante contraseña.

Crear una máquina virtual - Microsoft Azure - Mozilla Firefox

Inicio > Todos los recursos > REST2015030108-image-20211028141315 >

### Crear una máquina virtual

Tipo de seguridad

Imagen \*   
[Ver todas las imágenes](#) | Configurar la generación de máquinas virtuales

Instancia de Azure de acceso puntual ☐

Tamaño \*   
[Ver todos los tamaños](#)

**Cuenta de administrador**

Tipo de autenticación ☐ Clave pública SSH ☒ Contraseña

Nombre de usuario \*

Contraseña \*

Confirmar contraseña \*

[Revisar y crear](#) < Anterior Siguiente: Discos >

Para finalizar esta seccion selccionamos permitir los puertos seleccionados y elegimos el puerto de ssh y pasamos a la seccion de discos

Crear una máquina virtual - Microsoft Azure - Mozilla Firefox

Inicio > Todos los recursos > REST2015030108-image-20211028141315 >

### Crear una máquina virtual

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 \* ☐ Ninguno ☒ Permitir los puertos seleccionados

Seleccionar puertos de entrada \*

**⚠ Esto permitirá que todas las direcciones IP accedan a la máquina virtual.**  
Esto solo se recomienda para las pruebas. Use los controles avanzados de la pestaña Redes a fin de crear reglas para limitar el tráfico entrante a las direcciones IP conocidas.

[Revisar y crear](#) [< Anterior](#) [Siguiente: Discos >](#)

15:51

Seleccionamos el disco que utilizaremos para nuestra maquina virtual en este caso sera un disco HDD estandar

Crear una máquina virtual - Microsoft Azure - Mozilla Firefox

Inicio > Todos los recursos > REST2015030108-image-20211028141315 >

### Crear una máquina virtual

Datos básicos **Discos** Redes Administración Opciones avanzadas Etiquetas Revisar y crear

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 \*

Si el rendimiento es crítico para las cargas de trabajo, elija discos SSD Premium para reducir la latencia, IOPS y anchos de banda más altos y expansión de disco. [Más información](#)

Tipo de cifrado \*

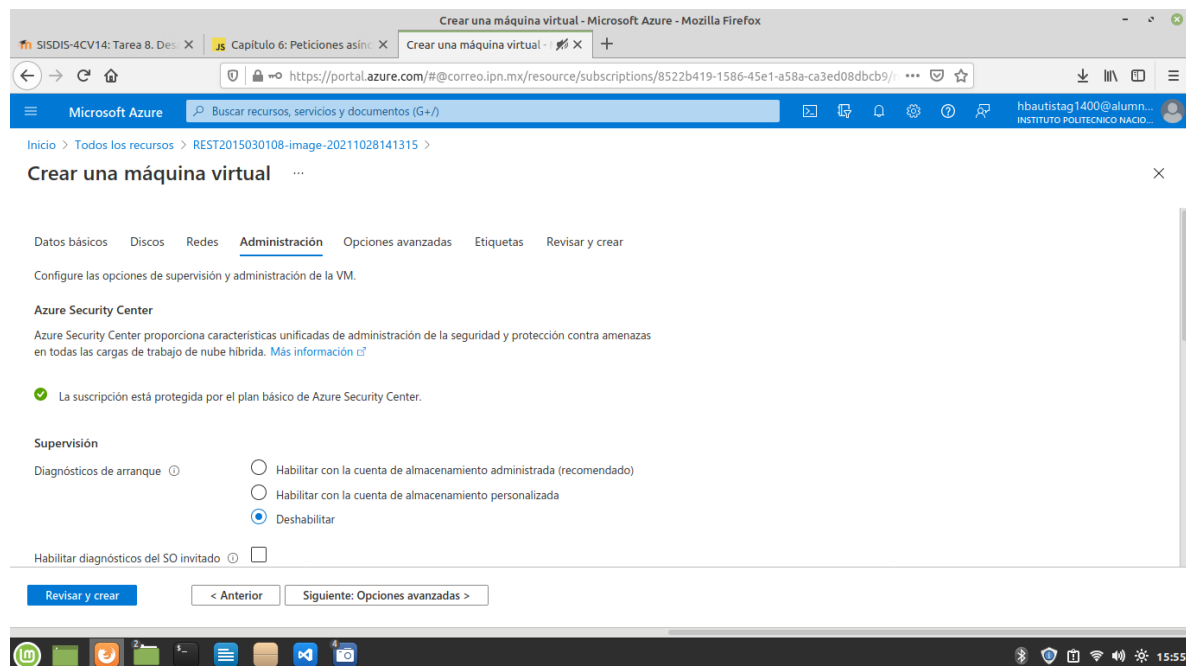
Habilitar compatibilidad con Ultra Disks ☐  
El disco Ultra se admite en las zonas de disponibilidad 1,2,3 para el tamaño de VM seleccionado (Standard\_B1s).

**Discos de datos**

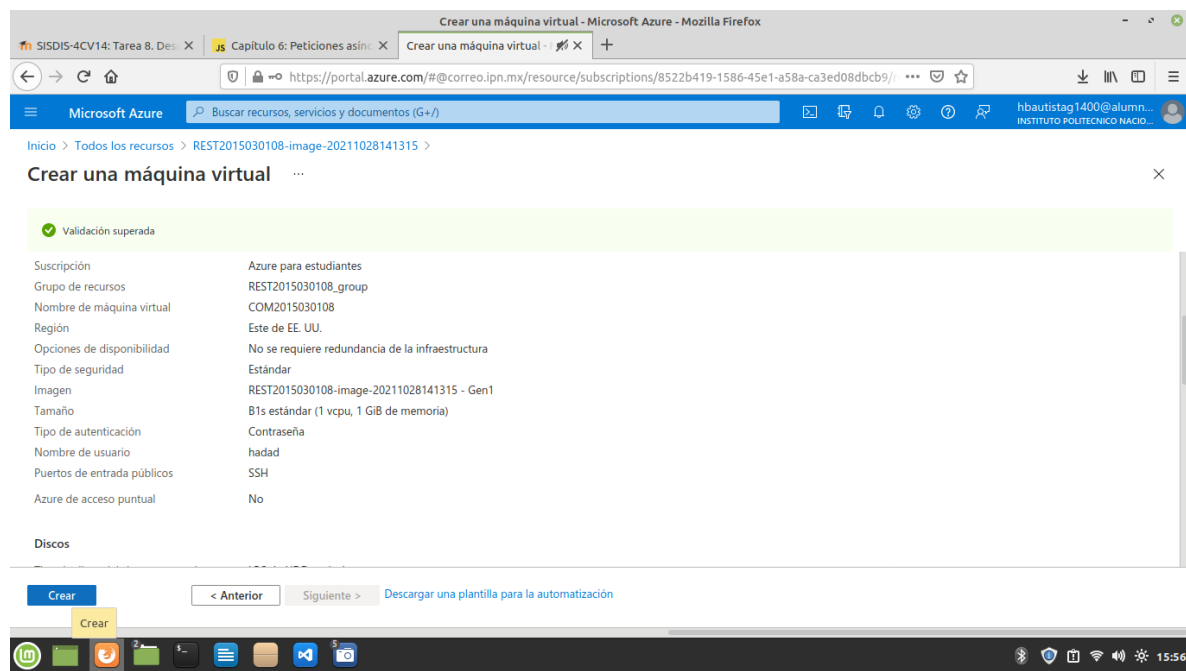
[Revisar y crear](#) [< Anterior](#) [Siguiente: Redes >](#)

15:53

Posteriormente nos vamos a la pestaña de administración para deshabilitar el diagnostico de arranque damos click en revisar y crear.



Finalmente revisamos los detalles de nuestra maquina a crear y le damos click en crear



## Compilación:

En primera instancia compilamos nuestro código fuente del front-end para generar nuestros archivos css ,js y html a partir de nuestro proyecto de react , como dependencias tenemos a node js que se puede instalar desde su pagina principal , yarn que puede ser instalado con npm install yarn -g y webpack el cual también puede ser instalado mediante npm install webpack -g , una vez con todas las dependencias podemos correr nuestro comando yarn build

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
hadad@cliente:~$ cd tienda-java/
hadad@cliente:~/tienda-java$ yarn
yarn install v1.22.17
(1/4) Resolving packages...
(2/4) Fetching packages...
(3/4) Linking dependencies...
warning " > eslint-plugin-react@7.26.1" has incorrect peer dependency "eslint@^3 || ^4 || ^5 || ^6 || ^7".
warning " > styled-components@5.3.3" has unmet peer dependency "react-is@>= 16.8.0".
warning " > eslint-plugin-promise@5.1.1" has incorrect peer dependency "eslint@^7.0.0".
(4/4) Building fresh packages...

Done in 52.01s.
hadad@cliente:~/tienda-java$ yarn build
yarn run v1.22.17
$ webpack
asset assets/images/ 3.83 KiB
asset assets/images/4fe75d5f9adb18067b85.ico 3.78 KiB [emitted] [immutable] [from: public/favicon.ico]
asset assets/images/b88d04fba731603756b1.css 50 bytes [emitted] [immutable] [from: node_modules/sweetalert2/dist/sweetalert2.min.css]
asset main.527319ce55e1bb56aa28.js 388 KiB [emitted] [immutable] [minimized] [big] (name: main) 1 related asset
asset ./index.html 656 bytes [emitted]
orphan modules 1.68 MiB [orphan] 771 modules
runtime modules 663 bytes 3 modules
cacheable modules 889 KiB
  modules by path ./node_modules/react/ 7.63 KiB 4 modules
  modules by path ./node_modules/prop-types/ 2.58 KiB 3 modules
  modules by path ./node_modules/hoist-non-react-statics/ 5.36 KiB 3 modules
  modules by path ./node_modules/react-dom/ 119 KiB
    ./node_modules/react-dom/index.js 1.33 KiB [built] [code generated]
    ./node_modules/react-dom/cjs/react-dom.production.min.js 118 KiB [built] [code generated]
  modules by path ./node_modules/scheduler/ 4.91 KiB
    ./node_modules/scheduler/index.js 198 bytes [built] [code generated]
    ./node_modules/scheduler/cjs/scheduler.production.min.js 4.72 KiB [built] [code generated]
  modules by path ./node_modules/react-is/ 2.48 KiB
    ./node_modules/react-is/index.js 196 bytes [built] [code generated]
    ./node_modules/react-is/cjs/react-is.production.min.js 2.29 KiB [built] [code generated]

WARNING in asset size limit: The following asset(s) exceed the recommended size limit (244 KiB).
This can impact web performance.
Assets:
  main.527319ce55e1bb56aa28.js (388 KiB)
```

Con nuestros archivos listos (carpeta dist), los podemos pasar por medio de sftp a nuestro servidor

```
Archivo Editar Ver Buscar Terminal Ayuda
hadad@cliente: ~/tienda-java
This can impact web performance.
Assets:
  main.527319ce55e1bb56aa28.js (388 KiB)

WARNING in entrypoint size limit: The following entrypoint(s) combined asset size exceeds the recommended limit (244 KiB). This can impact web performance.
Entrypoints:
  main (388 KiB)
    main.527319ce55e1bb56aa28.js

WARNING in webpack performance recommendations:
You can limit the size of your bundles by using import() or require.ensure to lazy load some parts of your application.
For more info visit https://webpack.js.org/guides/code-splitting/

webpack 5.63.0 compiled with 3 warnings in 13105 ms
Done in 14.22s.
hadad@cliente:~/tienda-java$ ls
dist  node_modules  package.json  public  src  webpack.config.dev.js  webpack.config.js  yarn.lock
hadad@cliente:~/tienda-java$ ssh hadad@20.120.96.123
The authenticity of host '20.120.96.123 (20.120.96.123)' can't be established.
ECDSA key fingerprint is SHA256:9H2pS4TtmNGzR9JfFeXNaJo4t0Sxw9dUP0rJ6G2a2V0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '20.120.96.123' (ECDSA) to the list of known hosts.
hadad@20.120.96.123's password:
hadad@cliente:~/tienda-java$ sftp hadad@20.120.96.123
hadad@20.120.96.123's password:
Connected to 20.120.96.123.
sftp> put -r dist/
Uploading dist/ to /home/hadad/dist
Entering dist/
dist/index.html
dist/main.527319ce55e1bb56aa28.js
Entering dist/assets
Entering dist/assets/images
dist/assets/images/b8d04fba731603756b1.css
dist/assets/images/4fe75d5f9adb18067b05.ico
dist/main.527319ce55e1bb56aa28.js.LICENSE.txt
sftp>

100% 656 10.8KB/s 00:00
100% 388KB 636.4KB/s 00:00

100% 50 0.8KB/s 00:00
100% 3870 57.8KB/s 00:00
100% 1694 27.9KB/s 00:00
```

Posteriormente eliminamos los archivos que tengamos en nuestro servidor de apache dentro de la carpeta webapps/ROOT y colocamos ahí nuestros archivos generados

```
Archivo Editar Ver Buscar Terminal Ayuda
hadad@COM2015030108: ~
hadad@cliente:~/tienda-java$ ssh hadad@20.120.96.123
hadad@20.120.96.123's password:
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1063-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Tue Nov 16 01:10:13 UTC 2021

System load:  0.0          Processes:      109
Usage of /:   8.8% of 28.90GB Users logged in:    0
Memory usage: 59%         IP address for eth0: 10.0.0.6
Swap usage:   0%

 * Super-optimized for small spaces - read how we shrink the memory
   footprint of MicroK8s to make it the smallest full K8s around.
   https://ubuntu.com/blog/microk8s-memory-optimisation

6 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.

Last login: Mon Nov 15 22:31:15 2021 from 187.190.153.241
hadad@COM2015030108:~$ cd dist/
hadad@COM2015030108:~/dist$ ls
assets  index.html  main.527319ce55e1bb56aa28.js  main.527319ce55e1bb56aa28.js.LICENSE.txt
hadad@COM2015030108:~/dist$ cd ..
hadad@COM2015030108:~$ rm -R apache-tomcat-8.5.72/webapps/ROOT/
assets/
index.html
main.7b013d01ceadd5ce2926.js
main.7b013d01ceadd5ce2926.js.LICENSE.txt
hadad@COM2015030108:~$ rm -R apache-tomcat-8.5.72/webapps/ROOT/*
hadad@COM2015030108:~$ cp -R dist/* apache-tomcat-8.5.72/webapps/ROOT/*
cp: target 'apache-tomcat-8.5.72/webapps/ROOT/*' is not a directory
hadad@COM2015030108:~$ cp -R dist/* apache-tomcat-8.5.72/webapps/ROOT/
cp: target 'apache-tomcat-8.5.72/webapps/ROOT/' is not a directory
hadad@COM2015030108:~$
```

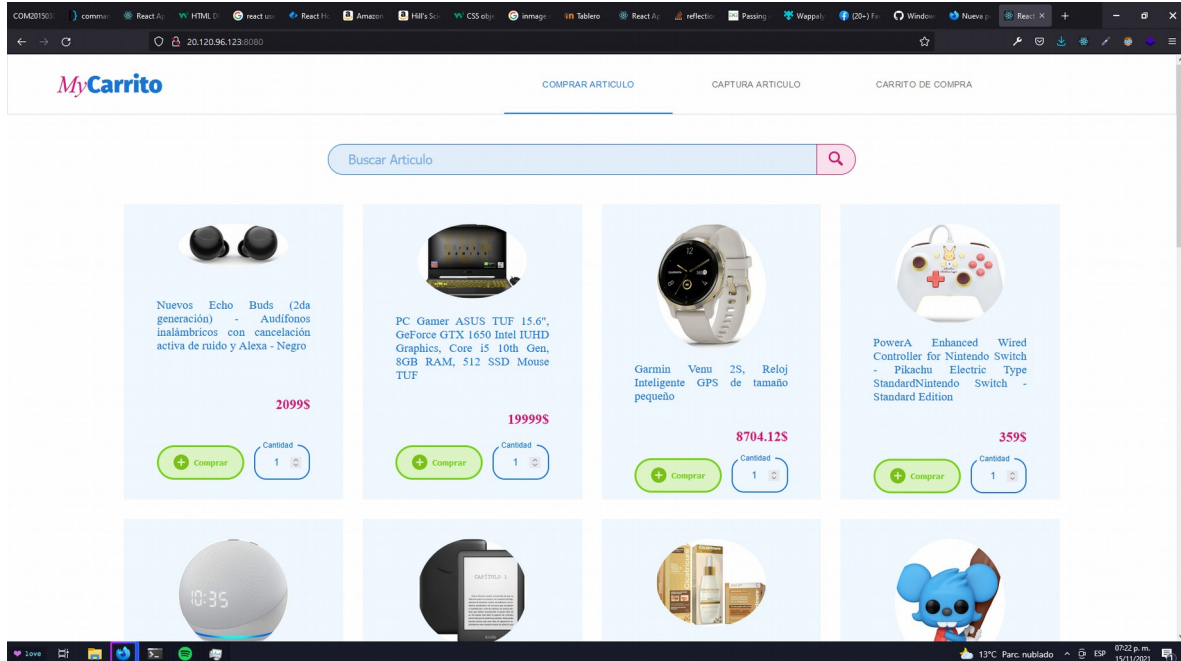


En la captura se puede observar la compilación desde la parte del servidor con el archivo servicio.java, en primera instancia compilamos el servicio java por medio del comando javac pasandole como argumentos el programa y las dependencias necesarias ,posteriormente copiamos todos los archivos .class del directorio negocio a WEB-INF/classes/negocio, empaquetamos nuestro servicio en un archivo .war, eliminamos los archivos anteriores del servicio y copiamos nuestro paquete en la carpeta webapps de nuestro tomcat , finalmente reiniciamos nuestro servidor

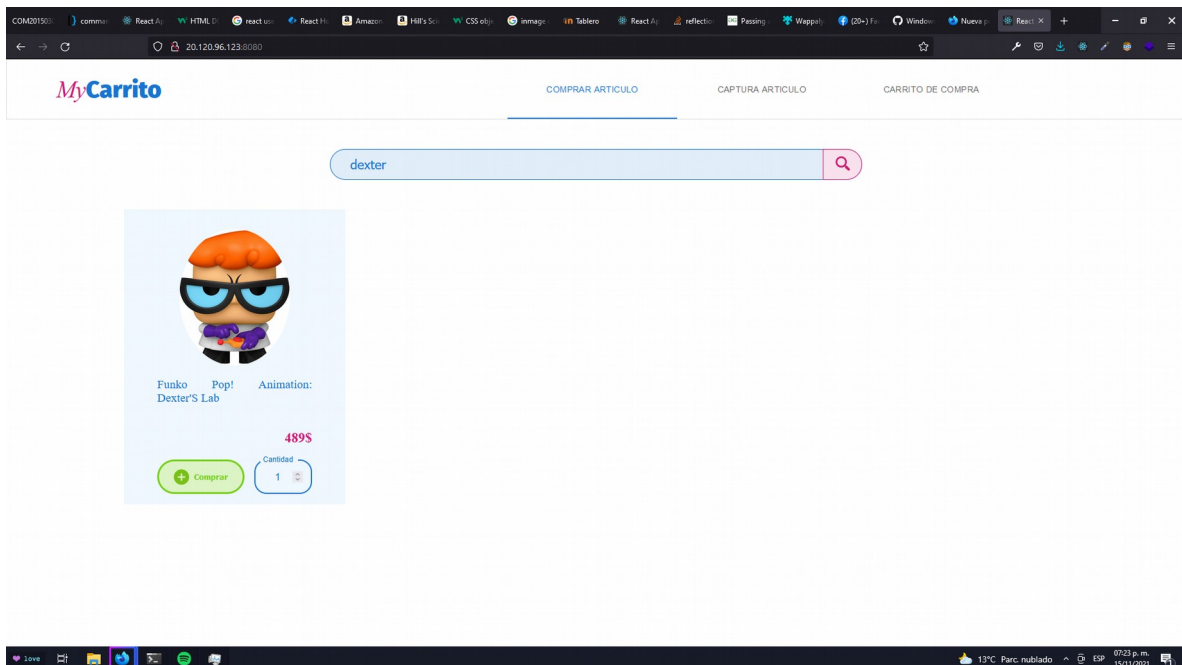
```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
hk2-locator-2.5.0-b05.jar      jersey-container-servlet.jar  tomcat-i18n-de.jar            websocket-api.jar
hk2-utils-2.5.0-b05.jar       jersey-guava-2.24.jar         tomcat-i18n-es.jar
hadad@COM2015030108:~/Servicio$ javac -cp /home/hadad/apache-tomcat-8.5.72/lib/gson-2.3.1.jar:/home/hadad/apache-tomcat-8.5.72/lib/javax.ws.rs-api-2.0.1.jar:. negocio/Servicio.java
hadad@COM2015030108:~/Servicio$ rm WEB-INF/classes/negocio/*
hadad@COM2015030108:~/Servicio$ cp negocio/*.class WEB-INF/classes/negocio/
hadad@COM2015030108:~/Servicio$ 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 = 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 = 899) (out= 518)(deflated 42%)
adding: WEB-INF/classes/negocio/Articulo.class(in = 866) (out= 505)(deflated 41%)
adding: WEB-INF/classes/negocio/AdaptadorGsonBase64.class(in = 1799) (out= 737)(deflated 59%)
adding: WEB-INF/classes/negocio/Servicio.class(in = 8606) (out= 4196)(deflated 51%)
adding: WEB-INF/classes/negocio/Error.class(in = 278) (out= 214)(deflated 23%)
adding: WEB-INF/classes/negocio/Servicio$Mensaje.class(in = 771) (out= 474)(deflated 38%)
ignoring entry META-INF/
adding: META-INF/context.xml(in = 325) (out= 231)(deflated 28%)
hadad@COM2015030108:~/Servicio$ rm -R /home/hadad/apache-tomcat-8.5.72/webapps/Servicio
hadad@COM2015030108:~/Servicio$ rm -R /home/hadad/apache-tomcat-8.5.72/webapps/Servicio.war
hadad@COM2015030108:~/Servicio$ cp Servicio.war /home/hadad/apache-tomcat-8.5.72/webapps/
hadad@COM2015030108:~/Servicio$ sh /home/hadad/apache-tomcat-8.5.72/bin/catalina.sh stop
Using CATALINA_BASE:   /home/hadad/apache-tomcat-8.5.72
Using CATALINA_HOME:   /home/hadad/apache-tomcat-8.5.72
Using CATALINA_TMPDIR: /home/hadad/apache-tomcat-8.5.72/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /home/hadad/apache-tomcat-8.5.72/bin/bootstrap.jar:/home/hadad/apache-tomcat-8.5.72/bin/tomcat-juli.jar
Using CATALINA_OPTS:
hadad@COM2015030108:~/Servicio$ sh /home/hadad/apache-tomcat-8.5.72/bin/catalina.sh start
Using CATALINA_BASE:   /home/hadad/apache-tomcat-8.5.72
Using CATALINA_HOME:   /home/hadad/apache-tomcat-8.5.72
Using CATALINA_TMPDIR: /home/hadad/apache-tomcat-8.5.72/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /home/hadad/apache-tomcat-8.5.72/bin/bootstrap.jar:/home/hadad/apache-tomcat-8.5.72/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
hadad@COM2015030108:~/Servicio$
```

## Ejecución:

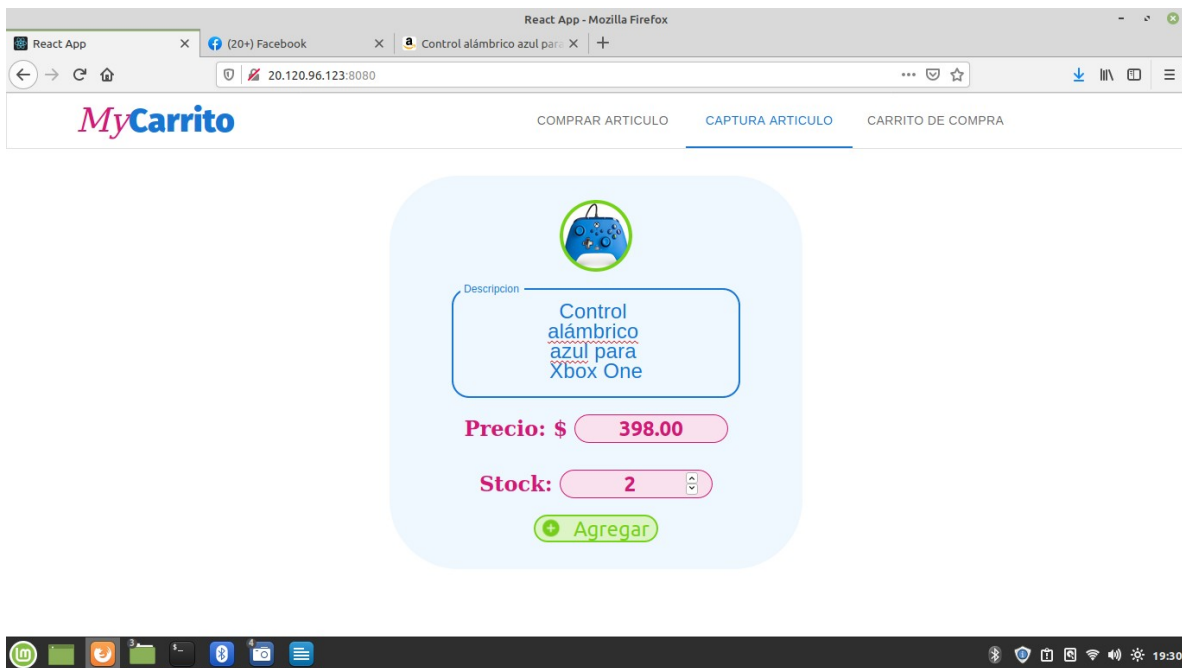
Para ver el comportamiento de nuestro programa debemos ingresar a la url formada por la ip de nuestro servidor junto con el puerto a utilizar en este caso el 8080



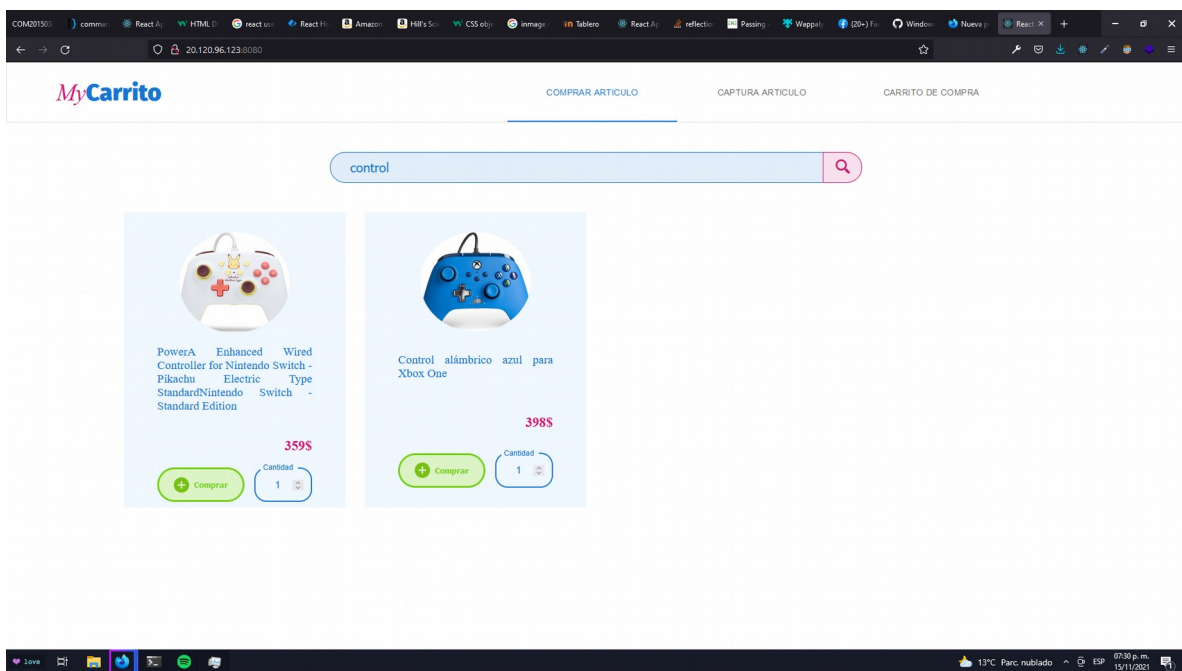
Para la primera prueba vamos a buscar un artículo que tenga la cadena "dexter"



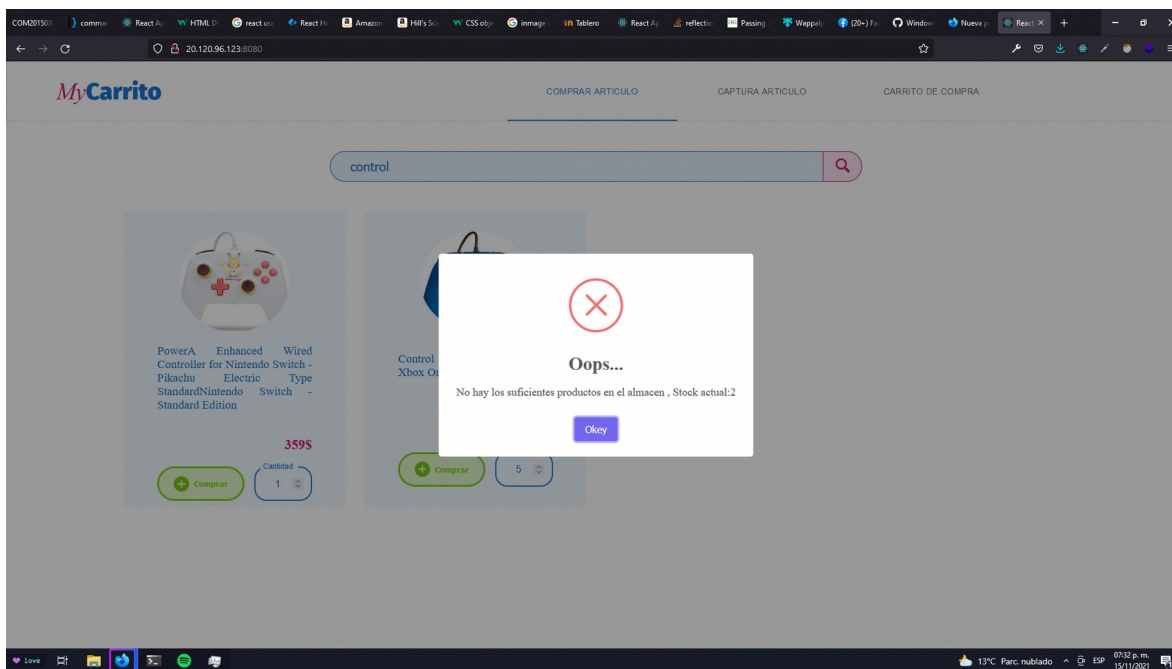
Posteriormente vamos añadir un artículo con su respectiva foto



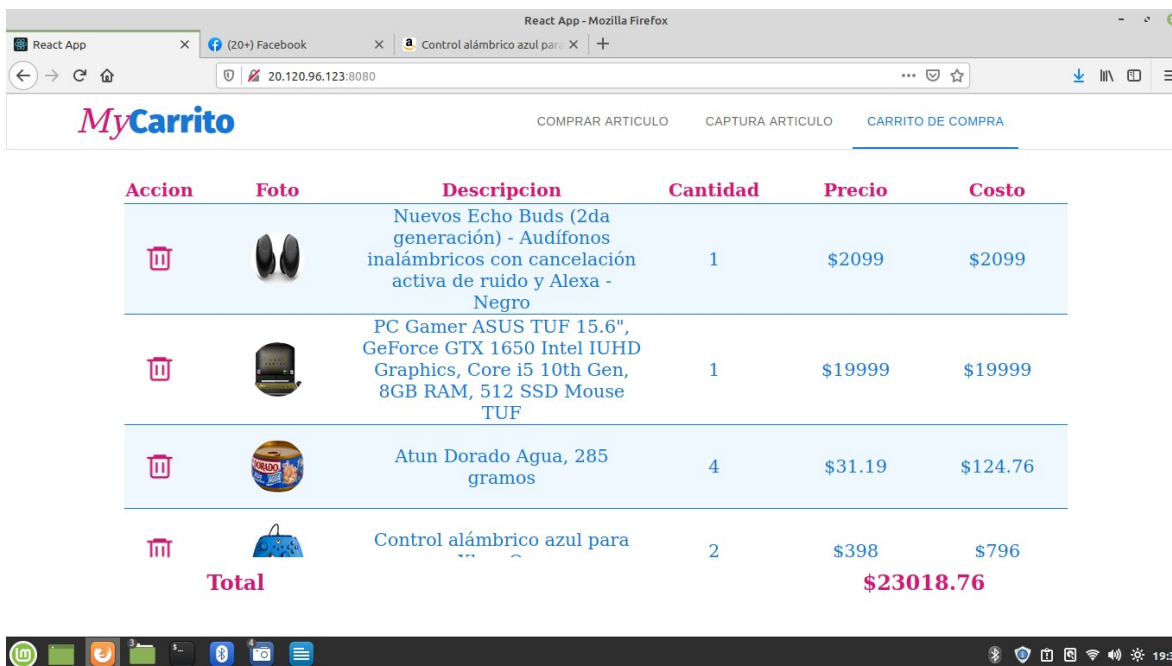
Posteriormente buscamos nuestro producto con la palabra clave control



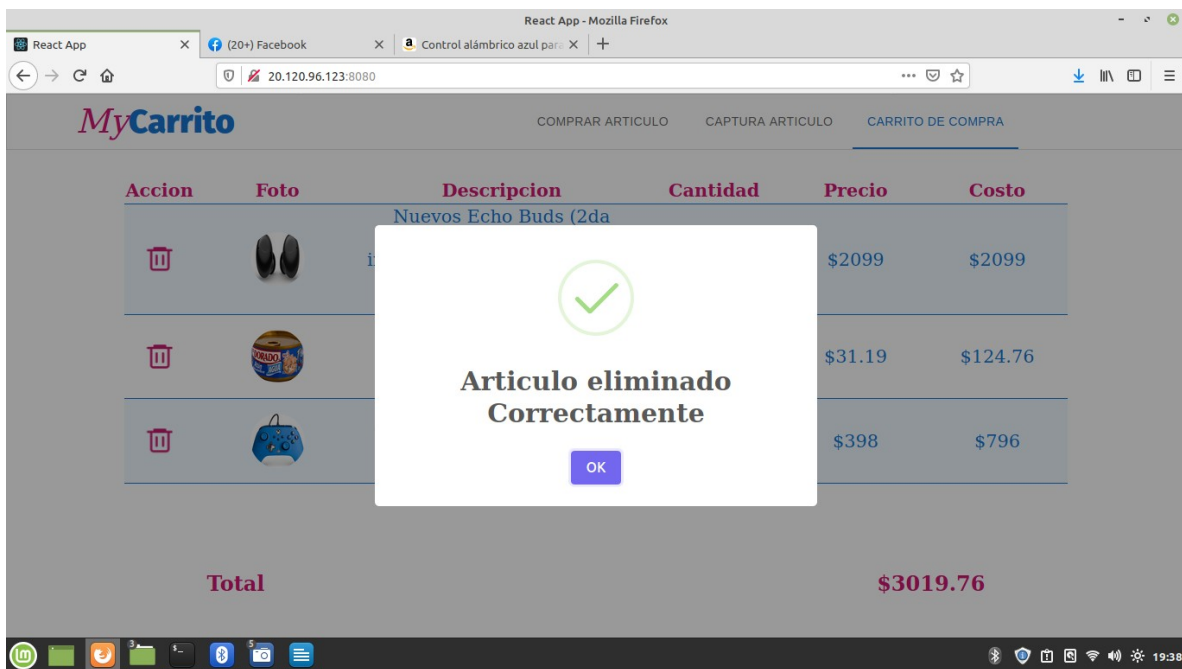
Después intentaremos agregar 5 controles a nuestro carrito para hacer saltar el error



Agregaremos unos cuantos articulos a nuestro carrito para ver la siguiente demostracion, y poder ver nuestro carrito



Finalmente vamos a eliminar un objeto de nuestro carrito y podemos ver como el total se actualiza automáticamente



## Conclusiones

Esta practica fue bastante entretenida de realizar ya que me permitió utilizar conocimientos adquiridos de otras tecnologías para poder implementarlo con un servidor de java rest el cual recibe y envía peticiones , hubo varios inconvenientes al realizar la practica desde el planteamiento del diseño del front-end y la estructura de la base de datos pero debido a que se dio un tiempo suficiente para concluir la practica por lo que se pudo realizar con éxito la practica