



Instituto Politécnico Nacional

Escuela Superior de Cómputo

Sistemas Distribuidos

Pineda Guerrero Carlos

Tarea 7

4CV13

Humberto Alejandro Ortega Alcocer (2016630495)

26 de Mayo del 2023

Índice

Índice	1
Desarrollo	3
Creación de Máquina Virtual	3
Prueba de Conexión	9
Apertura de puerto 8080 para Tomcat	10
Backend	12
Estructura de datos	12
Artículo	12
Carrito	12
Desarrollo del servicio web	13
Crear artículo	13
Obtener listado de artículos	15
Obtener artículos en el carrito	15
Añadir un artículo al carrito	16
Editar la cantidad de un artículo en el carrito	18
Eliminar un artículo en el carrito	20
Base de datos	21
Creación de la base de datos en la máquina virtual	22
Copia de archivos de servicio web a la máquina virtual	24
Compilación de servicio web en la máquina virtual	24
Frontend	25
Creación de páginas con ChatGPT	26
Página principal	27
Página de creación de artículo	31
Página de visualización de artículos	37
Página de carrito	43
Copia de archivos a la máquina virtual	51
Copia de archivos a directorio de Tomcat	51
Prueba de funcionamiento	52
Página principal	52
Creación de artículo	53
Creación de artículo exitosa	54
Catálogo de artículos	55
Filtrado de artículos por nombre	56
Detalles de un artículo	57
Agregar al carrito exitoso	58
Agregar al carrito fallido	59
Carrito de compras	60
Modificar la cantidad de un artículo en el carrito exitosamente	61

Modificar la cantidad de un artículo en el carrito fallida	62
Eliminar un elemento del carrito	63
Eliminar todos los elementos del carrito	66
Conclusión	70

Desarrollo

Para el desarrollo de esta tarea, primero se realizará la creación de la máquina virtual a partir de la imagen generada en la Tarea 6, posteriormente se desarrollará el backend y se realizarán pruebas del mismo, posteriormente se desarrollará el frontend y se realizarán las pruebas finales del mismo. Para el desarrollo del frontend y del backend se incluirán las capturas de las conversaciones sostenidas con ChatGPT que sean pertinentes para lograr los objetivos planteados.

Creación de Máquina Virtual

Para crear la máquina virtual, primero iremos a la imagen creada previamente en la Tarea 6 dentro del portal de Azure.

The screenshot shows the Azure portal interface for a VM image. The main title is "T6-2016630495-image-20230508195913". The left sidebar has sections for Overview, Activity log, Access control (IAM), Tags, Settings, Properties, Locks, Automation, Tasks (preview), Export template, Support + troubleshooting, and New Support Request. The Overview section is selected. The main content area shows the following details:

Essentials	
Resource group (move) :	SistemasDistribuidosResourceGroup
Location (move) :	East US
Subscription (move) :	Azure for Students
Subscription ID	: a5d9d675-3ae1-4e2b-95b5-02150d67b8d2
Provisioning state	: Succeeded
Tags (edit)	: Click here to add tags

Operating system : Linux
Source virtual machine : T6-2016630495
VM generation : V2
Zone resiliency : Disabled

OS disk

LUN	Source blob URI	Storage type	Caching
Linux	-	Standard HDD LRS	Read/write

Data disks

LUN	Source blob URI	Storage type	Caching
No data disk attached			

Haremos click en *Create VM* y llenaremos los campos con la información de la máquina virtual para esta tarea.

The screenshot shows the 'Create a virtual machine' wizard in Microsoft Azure. The 'Basics' tab is selected. The 'Subscription' dropdown is set to 'Azure for Students'. The 'Resource group' dropdown is set to 'SistemasDistribuidosResourceGroup'. Under 'Instance details', the 'Virtual machine name' is 'T7-2016630495', 'Region' is '(US) East US', 'Availability options' is 'No infrastructure redundancy required', 'Security type' is 'Standard', and the 'Image' dropdown is set to 'T6-2016630495-image-20230508195913 - x64 Gen2'. A note at the top states: 'Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image. Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization.' Below the note, a message says: 'This subscription may not be eligible to deploy VMs of certain sizes in certain regions.' At the bottom of the screen, there are buttons for 'Review + create' and 'Next : Disks >'. The status bar at the top right shows the user's email 'hortegaa1500@alumno...' and the institution 'INSTITUTO POLITECNICO NACIONAL'.

Dejaremos el tamaño que utilizamos en la máquina virtual de la tarea anterior a modo de que no tengamos problemas de ajuste de capacidades y seleccionaremos la llave SSH que creamos en la Tarea 1 para poder retener la autenticación cómo se ha hecho todo el semestre.

The screenshot shows the 'Create a virtual machine' wizard in Microsoft Azure. The 'Disks' tab is selected. Under 'VM architecture', 'x64' is selected, while 'Arm64' is noted as unsupported. Under 'Size', 'Standard_B2s - 2 vcpus, 4 GiB memory (\$30.37/month)' is selected. Under 'Administrator account', 'SSH public key' is chosen as the authentication type. A note explains that Azure generates an SSH key pair for the user. The 'Username' field is 'humbertowoody', and the 'SSH public key source' is 'Use existing key stored in Azure', with 'humbertowoody-azure' listed in the dropdown. At the bottom, 'Public inbound ports' is set to 'None'. The status bar at the top right shows the user's email 'hortegaa1500@alumno...' and the institution 'INSTITUTO POLITECNICO NACIONAL'.

También, permitiremos el tráfico en el puerto 22 para poder conectarnos de manera remota por SSH (más adelante se abrirán los puertos requeridos para Tomcat).

The screenshot shows the 'Create a virtual machine' wizard on the 'Inbound port rules' step. Under 'Authentication type', 'SSH public key' is selected. The 'Username' field contains 'humbertowoody'. Under 'SSH public key source', 'Use existing key stored in Azure' is selected, with 'humbertowoody-azure' listed in the dropdown. In the 'Inbound port rules' section, 'Allow selected ports' is chosen, and 'SSH (22)' is selected from the dropdown. A warning message states: '⚠ This will allow all IP addresses to access your virtual machine. This is only recommended for testing. Use the Advanced controls in the Networking tab to create rules to limit inbound traffic to known IP addresses.' At the bottom, there are 'Review + create', '< Previous', 'Next : Disks >', and 'Give feedback' buttons.

En cuanto al disco, usaremos un HDD en lugar de un SSD para ahorrar un poco de créditos.

The screenshot shows the 'Create a virtual machine' wizard on the 'Disks' step. The 'Disks' tab is selected. Under 'OS disk', 'Standard HDD (locally-redundant storage)' is chosen. A note says: 'The selected VM size supports premium disks. We recommend Premium SSD for high IOPS workloads. Virtual machines with Premium SSD disks qualify for the 99.9% connectivity SLA.' There are options for 'Delete with VM', 'Key management' (set to 'Platform-managed key'), and 'Enable Ultra Disk compatibility'. A note for 'Ultra disk' says: 'Ultra disk is supported in Availability Zone(s) 1,2,3 for the selected VM size Standard_B2s.' At the bottom, there are 'Review + create', '< Previous', 'Next : Networking >', and 'Give feedback' buttons.

Respecto a la configuración de red, dejaremos que se cree una red virtual (vnet) nueva para la tarea, así como el default para la subred y una IP pública puesto que vamos a acceder a la máquina virtual de forma remota (sin usar el CLI de Azure).

The screenshot shows the 'Networking' tab of the Azure VM creation wizard. It includes fields for Virtual network, Subnet, Public IP, NIC network security group, and Public inbound ports. A warning message states: "⚠️ This will allow all IP addresses to access your virtual machine. This is only recommended for testing. Use the Advanced controls in the Networking tab to create rules to limit inbound traffic to known IP addresses." Navigation buttons at the bottom include 'Review + create', '< Previous', 'Next : Management >', and 'Give feedback'.

Finalmente, este es el resumen de la máquina virtual a crear:

The screenshot shows the final summary page of the Azure VM creation wizard. It lists the selected configuration: Image (T6-2016630495-image-20230508195913), Size (Standard B2s), and Public inbound ports (SSH). The 'Basics' section details the subscription, resource group, virtual machine name, region (East US), availability options, security type, image, size, authentication type (SSH public key), username (humbertwoody), key pair name (humbertwoody-azure), and public inbound ports (SSH). A note says: "⚠️ You have set SSH port(s) open to the internet. This is only recommended for testing. If you want to change this setting, go back to Basics tab." Navigation buttons at the bottom include 'Create', '< Previous', 'Next >', 'Download a template for automation', and 'Give feedback'.

Microsoft Azure Search resources, services, and docs (G+/)

Home > T6-2016630495-image-20230508195913 > Create a virtual machine ...

Validation passed

Azure Spot No

Disks

OS disk type	Standard HDD LRS
Use managed disks	Yes
Delete OS disk with VM	Disabled
Ephemeral OS disk	No

Networking

Virtual network	(new) T7-2016630495-vnet
Subnet	(new) default (10.0.0.0/24)
Public IP	(new) T7-2016630495-ip
Accelerated networking	Off
Place this virtual machine behind an existing load balancing solution?	No
Delete public IP and NIC when VM is deleted	Disabled

Management

Microsoft Defender for Cloud	Basic (free)
System assigned managed identity	Off
Login with Azure AD	Off
Auto-shutdown	Off

Create < Previous Next > Download a template for automation Give feedback

Microsoft Azure Search resources, services, and docs (G+/)

Home > T6-2016630495-image-20230508195913 > Create a virtual machine ...

Validation passed

Management

Microsoft Defender for Cloud	Basic (free)
System assigned managed identity	Off
Login with Azure AD	Off
Auto-shutdown	Off
Enable hotpatch	Off
Patch orchestration options	Image Default

Monitoring

Alerts	Off
Boot diagnostics	On
Enable OS guest diagnostics	Off

Advanced

Extensions	None
VM applications	None
Cloud init	No
User data	No
Disk controller type	SCSI
Proximity placement group	None
Capacity reservation group	None

Create < Previous Next > Download a template for automation Give feedback

Esperaremos un poco a que se despliegue nuestra máquina virtual correctamente y vemos la siguiente pantalla de confirmación:

The screenshot shows the Microsoft Azure 'CreateVm-T6-2016630495-image-20230508195913-20230526184755 | Overview' page. It displays a green checkmark indicating the deployment is complete. Deployment details are listed, including the name 'CreateVm-T6-2016630495-image-202305081...', start time '5/26/2023, 6:55:37 PM', subscription 'Azure for Students', and resource group 'SistemasDistribuidosResourceGroup'. A table shows the status of resources created: T7-2016630495 (Virtual machine, OK), t7-2016630495.285 (Network interface, Created), T7-2016630495-vnet (Virtual network, OK), T7-2016630495-ip (Public IP address, OK), and T7-2016630495-nsg (Network security group, OK). A sidebar on the right provides links to Cost management, Microsoft Defender for Cloud, Free Microsoft tutorials, and Work with an expert.

La máquina virtual se observa de la siguiente manera:

The screenshot shows the Microsoft Azure 'T7-2016630495 | Overview' page for a Virtual machine. The 'Overview' tab is selected, displaying basic information: Resource group 'SistemasDistribuidosResourceGroup', Status 'Running', Location 'East US', Subscription 'Azure for Students', and Subscription ID 'a5d9d675-3ae1-4e2b-95b5-02150d67b8d2'. The 'Essentials' section shows the operating system as Linux (ubuntu 20.04) and the public IP address as 172.173.252.180. The 'Properties' tab is selected, showing detailed configuration: Computer name 'T7-2016630495', Operating system 'Linux (ubuntu 20.04)', Publisher '(', Offer '(', Plan '(', VM generation 'V2', VM architecture 'x64', Agent status 'Ready', Agent version '2.9.0.4', Host group 'None', and Host '('. The 'Networking' section shows the public IP address as 172.173.252.180 and the private IP address as 10.0.0.4. The 'Size' section shows the size as Standard B2s, vCPUs as 2, and RAM as 4 GiB.

Prueba de Conexión

Para probar la conexión trataremos de realizar una sesión de SSH a nuestra máquina virtual.

```
> pwd
/Users/humbertowoodly/Proyectos/IPN/sistemas-distribuidos-escom/azure
> ls -lha
Permissions Size User          Date Modified Name
.rw-----@ 2.5k humbertowoodly 24 Mar 15:19  humbertowoodly-azure.pem
> ssh -i humbertowoodly-azure.pem 172.173.252.180
The authenticity of host '172.173.252.180 (172.173.252.180)' can't be established.
ED25519 key fingerprint is SHA256:xbycod2ga+s6GugcuuaHg9COpP8VGXubKta7J0CmqaM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.173.252.180' (ED25519) to the list of known hosts.
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1037-azure x86_64)

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

System information as of Sat May 27 01:01:51 UTC 2023

System load:  0.53           Processes:            112
Usage of /:   8.7% of 28.89GB Users logged in:      0
Memory usage: 22%           IPv4 address for eth0: 10.0.0.4
Swap usage:   0%

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

3 updates can be applied immediately.
2 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Tue May  9 01:55:13 2023 from 148.204.56.241
humbertowoodly@T7-2016630495:~$ echo $CATALINA_HOME
/home/humbertowoodly/apache-tomcat-8.5.88
humbertowoodly@T7-2016630495:~$
```

Gracias a usar una imagen, se retuvieron los valores de variables de entorno y demás cosas para simplificar el resto de la tarea.

Apertura de puerto 8080 para Tomcat

Para abrir el puerto 8080 que requiere Tomcat, primero iremos a la sección de *Networking* de nuestra máquina virtual:

The screenshot shows the Azure portal interface for a virtual machine named "T7-2016630495". The left sidebar has "Networking" selected under "Settings". The main pane shows the "Inbound port rules" tab for the network interface "t7-2016630495285". It lists four existing rules: "SSH" (Priority 300, Port 22, TCP, Any, Any, Allow), "AllowVnetInBound" (Priority 65000, Any, Any, VirtualNetwork, VirtualNetwork, Allow), "AllowAzureLoadBalancerInBound" (Priority 65001, Any, Any, AzureLoadBalancer, Any, Allow), and "DenyAllInBound" (Priority 65500, Any, Any, Any, Any, Deny). A blue "Add inbound port rule" button is located at the top right of the table.

Haremos click en *Add inbound port rule* y llenaremos el formulario con los datos requeridos:

The screenshot shows the "Add inbound security rule" dialog box. The "Source" dropdown is set to "Any". The "Source port ranges" field contains "8080". The "Destination" dropdown is set to "Any". The "Service" dropdown is set to "Custom". The "Destination port ranges" field contains "8080". The "Protocol" section has "Any" selected. The "Action" section has "Allow" selected. The "Priority" field is set to "310". The "Name" field contains "AllowAnyCustom8080Inbound". At the bottom, there are "Add" and "Cancel" buttons, and a "Give feedback" link.

Finalmente esperaremos a que se aplique la configuración de red, y deberemos poder visualizar el puerto abierto en el listado principal:

The screenshot shows the Microsoft Azure portal interface for a virtual machine named 'T7-2016630495'. The left sidebar is open, showing the 'Networking' section under 'Settings'. The main content area displays the 'Inbound port rules' tab for the network interface 't7-2016630495285'. A table lists five rules:

Priority	Name	Port	Protocol	Source	Destination	Action
300	SSH	22	TCP	Any	Any	Allow
310	AllowAnyCustom8080Inbound	8080	Any	Any	Any	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

Below the table, there is a 'Need help?' section with links to 'Understand Azure load balancing' and 'Quickstart: Create a public load balancer to load balance Virtual Machines'.

Backend

En términos del backend, lo primero será desarrollar la estructura de datos tanto para la base de datos como para la aplicación en sí misma, una vez teniendo la aplicación, se generará la base de datos en la máquina virtual, se copiarán los archivos del servicio web a la máquina virtual, se compilarán, se ejecutará el servicio y se validará su funcionamiento.

Estructura de datos

Para los datos, solo necesitamos considerar dos entidades, Artículo y Carrito. Con ellas realizaremos las operaciones. Tendremos una entidad llamada FotoArtículo, pero dado que no será reflejada por el código de backend en sí mismo (puesto que siempre que se cargue un Artículo, éste obtendrá la foto relacionada al mismo mediante una sentencia JOIN de SQL) no se definirá.

Artículo

Para el Artículo, los campos a considerar (en función del planteamiento del profesor):

- **ID:** será el identificador único para cada artículo, un entero autoincremental.
- **Nombre:** el nombre del artículo, un campo de texto con longitud corta.
- **Descripción:** la descripción del artículo en cuestión, un campo de texto con longitud grande para poder almacenar párrafos de descripción.
- **Cantidad:** será la cantidad disponible en almacén de dicho artículo, un entero mayor o igual a cero.
- **Precio:** será el precio (en pesos Mexicanos) del producto, este será un campo de doble precisión.
- **Foto*:** será la fotografía del artículo, esta será serializada en Base64 y almacenada como un arreglo de bytes.

*Nota: La fotografía del producto se almacena en una tercer tabla pero esta relación será invisible a la aplicación pues siempre que sea requerida será referenciada mediante la relación con el ID del artículo.

Carrito

El carrito de compras es una tabla muy simple, su estructura es la siguiente:

- **ID Artículo:** el identificador único del artículo en el carrito.
- **Cantidad:** la cantidad de artículos que se desean comprar.

Desarrollo del servicio web

Para el desarrollo del servicio web, lo primero fue considerar las rutas que íbamos a necesitar para lograr los objetivos de la tarea, las rutas resultantes fueron:

- `POST /articulos`: para crear un artículo nuevo.
- `GET /articulos`: para obtener un listado de artículos.
- `GET /carrito_articulos`: para obtener un listado de los artículos en el carrito.
- `POST /carrito_articulos`: para añadir un artículo al carrito.
- `PUT /carrito_articulos`: para modificar la cantidad de un artículo en el carrito.
- `DELETE /carrito_articulos`: para eliminar un artículo del carrito.

En la definición de la tarea, se especificaron detalles de cómo realizar las consultas a modo de usar transacciones y agrupar ciertas operaciones dado que se retenga un estado válido de los datos en todo momento (integridad). A continuación, se visualizará el código relevante relacionado con cada ruta.

Crear artículo

En esta ruta, se recibirá un objeto JSON que cumpla con la siguiente estructura:

```
J ParamAltaArticulo.java 2, U X J Articulo.java 1, U X
tareas > tarea-7 > backend_t7 > J ParamAltaArticulo.java > ...
tareas > tarea-7 > backend_t7 > J Articulo.java > ...

ParamAltaArticulo.java:
1 package backend_t7;
2
3 public class ParamAltaArticulo {
4     Artículo articulo;
5 }
6

Articulo.java:
1 package backend_t7;
2
3 public class Artículo {
4     int id;
5     String nombre;
6     String descripción;
7     int cantidad;
8     double precio;
9     byte[] foto;
10 }
```

El método que recibe la información y la procesa tiene la siguiente funcionalidad:

1. Validar que los campos tengan datos válidos.
2. Insertar los datos en la tabla de Artículos.
3. Si existe la fotografía, insertarla en la tabla de fotografías.
4. Retornar un 200 OK.

Para lograr esto, este es el método:

```
@POST
@Path("articulos")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public Response altaArticulo(String json) throws Exception {
    ParamAltaArticulo p = (ParamAltaArticulo) j.fromJson(json, ParamAltaArticulo.class);
    Articulo articulo = p.articulo;

    Connection conexion = pool.getConnection();

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

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

    if (articulo.cantidad < 0)
        return Response.status(400).entity(j.toJson(new Error("La cantidad debe ser mayor a cero"))).build();

    if (articulo.precio < 0)
        return Response.status(400).entity(j.toJson(new Error("El precio debe ser mayor a cero"))).build();

    try {
        conexion.setAutoCommit(false);

        PreparedStatement stmt_1 = conexion.prepareStatement(
            "INSERT INTO articulos(id_articulo,nombre,descripcion,cantidad,precio) VALUES (0,?, ?, ?, ?)");

        try {
            stmt_1.setString(1, articulo.nombre);
            stmt_1.setString(2, articulo.descripcion);
            stmt_1.setInt(3, articulo.cantidad);
            stmt_1.setDouble(4, articulo.precio);
            stmt_1.executeUpdate();
        } finally {
            stmt_1.close();
        }

        if (articulo.foto != null) {
            PreparedStatement stmt_2 = conexion.prepareStatement(
                "INSERT INTO fotos_articulos(id_articulo,foto) VALUES ((SELECT id_articulo FROM articulos WHERE nombre=?),?)");
            try {
                stmt_2.setString(0, articulo.nombre);
                stmt_2.setBytes(1, articulo.foto);
                stmt_2.executeUpdate();
            } finally {
                stmt_2.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();
}
```

Obtener listado de artículos

Para obtener el listado de artículos, no requerimos ningún parámetro, el código que obtiene los artículos (y sus respectivas fotos), es el siguiente:

```
@GET
@Path("articulos")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public Response consultaArticulo(String json) throws Exception {

    Connection conexion = pool.getConnection();

    try {
        PreparedStatement stmt_1 = conexion.prepareStatement(
            "SELECT a.id_articulo,a.nombre,a.descripcion,a.cantidad,a.precio,b.foto FROM articulos a LEFT OUTER JOIN
            fotos_articulos b ON a.id_articulo=b.id_articulo");

        try {
            ResultSet rs = stmt_1.executeQuery();
            try {
                ArrayList<Articulo> r = new ArrayList<Articulo>();
                if (rs.next()) {
                    Articulo a = new Articulo();
                    a.id_articulo = rs.getInt("id_articulo");
                    a.nombre = rs.getString("nombre");
                    a.descripcion = rs.getString("descripcion");
                    a.cantidad = rs.getInt("cantidad");
                    a.precio = rs.getDouble("precio");
                    a.foto = rs.getBytes("foto");
                    r.add(a);
                }
                return Response.ok(j.toJson(r)).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();
    }
}
```

Obtener artículos en el carrito

Para obtener los artículos en el carrito de compras, no es necesario ningún parámetro, sólamente se hará una operación *JOIN* para obtener los artículos a que existen en el carrito y serializarlos. La clase con la respuesta a serializar es la siguiente:

```
package backend_t7;

import backend_t7.Articulo;

public class RespuestaCarritoArticulo {
    Articulo articulo;
    int cantidad;
}
```

La implementación del método que obtiene los datos para retornar un arreglo de estos elementos es:

```
@GET
@Path("carrito_articulos")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public Response consultaArticulo(String json) throws Exception {
    Connection conexion = pool.getConnection();

    try {
        PreparedStatement stmt_1 = conexion.prepareStatement(
            "SELECT a.cantidad,b.id_articulo,b.nombre,b.descripcion,b.cantidad,b.precio,c.foto FROM carrito a, LEFT OUTER JOIN articulos b ON a.id_articulo=b.id_articulo, LEFT OUTER JOIN fotos_articulos c ON a.id_articulo=b.id_articulo");

        try {
            ResultSet rs = stmt_1.executeQuery();
            try {
                ArrayList<RespuestaCarritoArticulo> r = new ArrayList<RespuestaCarritoArticulo>();
                if (rs.next()) {
                    RespuestaCarritoArticulo a = new RespuestaCarritoArticulo();
                    a.articulo= new Articulo();
                    a.cantidad = rs.getInt(0);
                    a.articulo.id_articulo = rs.getInt(1);
                    a.articulo.nombre = rs.getString(2);
                    a.articulo.descripcion = rs.getString(3);
                    a.articulo.cantidad = rs.getInt(4);
                    a.articulo.precio = rs.getDouble(5);
                    a.articulo.foto = rs.getBytes(6);
                    r.add(a);
                }
                return Response.ok(j.toJson(r)).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();
    }
}
```

Añadir un artículo al carrito

Para añadir un artículo al carrito, usaremos el siguiente parámetro en formato JSON:

```
package backend_t7;

public class ParamAltaCarritoArticulo {
    int id_articulo;
    int cantidad;
}
```

Y el método que crea el artículo, tiene el siguiente orden de ejecución:

1. Validar los datos que se reciben.
 - a. Que el artículo exista.
 - b. Que la cantidad esté disponible en almacén.
2. Restar la cantidad de artículos solicitada del artículo en cuestión.
3. Crear la entrada en la tabla del carrito con la cantidad y el identificador del carrito en cuestión.

El método es el siguiente:

```
@POST
@Path("carrito_articulos")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)

public Response altaCarritoArticulo(String json) throws Exception {
    ParamAltaCarritoArticulo p = (ParamAltaCarritoArticulo) j.fromJson(json, ParamAltaCarritoArticulo.class);
    CarritoArticulo carrito_articulo = p.carrito_articulo;

    Connection conexion = pool.getConnection();

    if (carrito_articulo.id_articulo == 0)
        return Response.status(400).entity(j.toJson(new Error("Se debe ingresar el id del articulo"))).build();

    if (carrito_articulo.cantidad <= 0)
        return Response.status(400).entity(j.toJson(new Error("La cantidad debe ser mayor a cero"))).build();

    // Obtener la cantidad de artículos disponibles desde la base de datos y validar
    // si hay stock
    int cantidad_disponible = 0;
    try {
        PreparedStatement stmt_1 = conexion.prepareStatement(
            "SELECT cantidad FROM articulos WHERE id_articulo=?");
        try {
            stmt_1.setInt(1, carrito_articulo.id_articulo);
            ResultSet rs = stmt_1.executeQuery();
            try {
                if (rs.next()) {
                    cantidad_disponible = rs.getInt("cantidad");
                }
            } finally {
                rs.close();
            }
        } finally {
            stmt_1.close();
        }
    } catch (Exception e) {
        return Response.status(400).entity(j.toJson(new Error(e.getMessage()))).build();
    }

    if (cantidad_disponible == 0)
        return Response.status(400).entity(j.toJson(new Error("No hay stock disponible"))).build();

    // Restamos la cantidad de artículos disponibles con la cantidad solicitada
    cantidad_disponible -= carrito_articulo.cantidad;

    // Actualizamos la cantidad de artículos disponibles en la base de datos
    try {
        PreparedStatement stmt_1 = conexion.prepareStatement(
            "UPDATE articulos SET cantidad=? WHERE id_articulo=?");
        try {
            stmt_1.setInt(1, cantidad_disponible);
            stmt_1.setInt(2, carrito_articulo.id_articulo);
        } finally {
            stmt_1.close();
        }
    } catch (Exception e) {
        return Response.status(400).entity(j.toJson(new Error(e.getMessage()))).build();
    }
}
```

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

try {
    conexion.setAutoCommit(false);

    PreparedStatement stmt_1 = conexion.prepareStatement(
        "INSERT INTO carrito(id_articulo,cantidad) VALUES (?,?)");

    try {
        stmt_1.setInt(2, carrito_articulo.id_articulo);
        stmt_1.setInt(3, carrito_articulo.cantidad);
        stmt_1.executeUpdate();
    } finally {
        stmt_1.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();
}
```

Editar la cantidad de un artículo en el carrito

Para este método usaremos los siguientes parámetros:

```
package backend_t7;

public class ParamModificaCarritoArticulo {
    int id_articulo;
    int cantidad;
}
```

El método realiza las siguientes tareas:

1. Validarlos datos que se reciben.
 - a. Que el artículo exista.
 - b. Que el artículo exista en el carrito.
 - c. Que la cantidad solicitada cumpla con lo disponible en almacén.
2. Restar o sumar la cantidad de artículos restantes (según sea el caso) a nuestro artículo original en almacén.
3. Modificar la cantidad en la tabla de carrito.

La definición del método es la siguiente:

```
@PUT
@Path("carrito_articulos")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public Response modificaCarritoArticulo(String json) throws Exception {
    ParamModificaCarritoArticulo p = (ParamModificaCarritoArticulo) j.fromJson(json,
        ParamModificaCarritoArticulo.class);
    Carrito carrito_articulo = p.carrito_articulo;

    Connection conexion = pool.getConnection();

    if (carrito_articulo.id_articulo == 0)
        return Response.status(400).entity(j.toJson(new Error("Se debe ingresar el id del articulo"))).build();

    if (carrito_articulo.cantidad == 0)
        return Response.status(400).entity(j.toJson(new Error("Se debe ingresar la cantidad"))).build();

    // Obtener la cantidad de artículos disponibles desde la base de datos
    int cantidad_disponible = 0;
    try {
        PreparedStatement stmt_1 = conexion.prepareStatement(
            "SELECT cantidad FROM articulos WHERE id_articulo=?");
        try {
            stmt_1.setInt(1, carrito_articulo.id_articulo);
            ResultSet rs = stmt_1.executeQuery();
            try {
                if (rs.next()) {
                    cantidad_disponible = rs.getInt("cantidad");
                }
            } finally {
                rs.close();
            }
        } finally {
            stmt_1.close();
        }
    } catch (Exception e) {
        return Response.status(400).entity(j.toJson(new Error(e.getMessage()))).build();
    }

    // Sumamos la cantidad de artículos disponibles con la cantidad solicitada
    cantidad_disponible += carrito_articulo.cantidad;

    // Actualizamos la cantidad de artículos disponibles en la base de datos
    try {
        PreparedStatement stmt_1 = conexion.prepareStatement(
            "UPDATE articulos SET cantidad=? WHERE id_articulo=?");
        try {
            stmt_1.setInt(1, cantidad_disponible);
            stmt_1.setInt(2, carrito_articulo.id_articulo);
            stmt_1.executeUpdate();
        } finally {
            stmt_1.close();
        }
    } catch (Exception e) {
        return Response.status(400).entity(j.toJson(new Error(e.getMessage()))).build();
    }

    try {
        conexion.setAutoCommit(false);

        PreparedStatement stmt_1 = conexion.prepareStatement(
            "UPDATE carrito SET cantidad=? WHERE id_articulo=?");
        try {
            stmt_1.setInt(1, carrito_articulo.cantidad);
            stmt_1.setInt(2, carrito_articulo.id_articulo);
            stmt_1.executeUpdate();
        } finally {
            stmt_1.close();
        }
    } finally {
        conexion.setAutoCommit(true);
    }
}
```

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

Eliminar un artículo en el carrito

Para eliminar un artículo del carrito, este será el cuerpo del request que será serializado en formato JSON por el cliente:

```
package backend_t7;

public class ParamBorraCarritoArticulo {
    int id_articulo;
}
```

Lo que deberá realizar el método es lo siguiente:

1. Sumar la cantidad del artículo en cuestión de vuelta al artículo original en almacén.
2. Eliminar la entrada en la tabla del carrito.

La implementación del método es la siguiente:

```
@DELETE
@Path("carrito_articulos")
@Produces(MediaType.APPLICATION_JSON)
public Response borraCarrito(String json) throws Exception {
    ParamBorraCarritoArticulo p = (ParamBorraCarritoArticulo) j.fromJson(json,
        ParamBorraCarritoArticulo.class);

    Connection conexion = pool.getConnection();

    // Obtenemos el artículo en cuestión desde la base de datos
    Carrito carrito_articulo = null;
    try {
        PreparedStatement stmt_1 = conexion.prepareStatement(
            "SELECT * FROM carrito WHERE id_articulo=?");
        try {
            stmt_1.setInt(1, p.id_articulo);
            ResultSet rs = stmt_1.executeQuery();
            try {
                if (rs.next()) {
                    carrito_articulo = new Carrito();
                    carrito_articulo.id_articulo = rs.getInt("id_articulo");
                    carrito_articulo.cantidad = rs.getInt("cantidad");
                }
            } finally {
                rs.close();
            }
        } finally {
            stmt_1.close();
        }
    } catch (Exception e) {
        return Response.status(400).entity(j.toJson(new Error(e.getMessage()))).build();
    }
}
```

```
try {
    conexion.setAutoCommit(false);

    PreparedStatement stmt_1 = conexion.prepareStatement(
        "DELETE FROM carrito where id_articulo=?");
    PreparedStatement stmt_2 = conexion.prepareStatement(
        "UPDATE articulos SET cantidad=cantidad+? WHERE id_articulo=?");

    try {
        stmt_2.setInt(0, carrito_articulo.cantidad);
        stmt_2.setInt(1, carrito_articulo.id_articulo);
        stmt_2.executeUpdate();

        stmt_1.setInt(0, carrito_articulo.id_articulo);
        stmt_1.executeUpdate();
    } finally {
        stmt_1.close();
    }
} 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();
}
```

Base de datos

La definición de la base de datos se incluye en el siguiente script de SQL, así puede verse la estructura general de nuestra base de datos, así como las llaves foráneas y los índices.

```
-- Base de datos para la tarea 7 (la borra si ya existe)
DROP DATABASE IF EXISTS tarea7;
CREATE DATABASE tarea7;
USE tarea7;

-- Tabla de articulos
CREATE TABLE articulos (
    id_articulo INT NOT NULL AUTO_INCREMENT,
    nombre VARCHAR(100) NOT NULL,
    descripcion VARCHAR(1000) NOT NULL,
    cantidad INT NOT NULL,
    precio DECIMAL(10,2) NOT NULL,
    PRIMARY KEY (id_articulo)
);

-- Tabla de fotos
CREATE TABLE fotos_articulos (
    foto LONGBLOB NOT NULL,
    id_articulo INT NOT NULL
);

-- Tabla de carrito
CREATE TABLE carrito (
    id_articulo INT NOT NULL,
    cantidad INT NOT NULL
);

-- Crear la llave foránea para las fotos.
ALTER TABLE fotos_articulos
ADD FOREIGN KEY (id_articulo) REFERENCES articulos(id_articulo);
```

```
-- Crear la llave foránea para el carrito.  
ALTER TABLE carrito  
ADD FOREIGN KEY (id_articulo) REFERENCES articulos(id_articulo);  
  
-- Crear indice único para el nombre del artículo  
CREATE UNIQUE INDEX nombre_articulo ON articulos(nombre);
```

Creación de la base de datos en la máquina virtual

Para crear la base de datos en la máquina virtual, usaremos el comando de MySQL de administrador, con las credenciales creadas en la clase pasada, y redirigiremos la entrada a nuestro script creado.

```
humbertowoody@T7-2016630495:~$ mysql -u hugo -p < base_de_datos_tarea7.sql  
Enter password:  
humbertowoody@T7-2016630495:~$ mysql -u hugo -p
```

En la siguiente página se muestra la verificación de que las tablas existan. Dado que el nombre de la base de datos es el mismo que se utilizó en la tarea anterior, no es necesario modificar la autenticación y solo restaría desplegar el servicio web.

```
humberto@T7-2016630495:~$ mysql -u hugo -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.33-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2023, 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;
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 |
+-----+
| articulos
| carrito
| fotos_articulos
+-----+
3 rows in set (0.00 sec)

mysql> show columns from articulos;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_articulo | int | NO | PRI | NULL | auto_increment |
| nombre | varchar(100) | NO | UNI | NULL | |
| descripcion | varchar(1000) | NO | | NULL | |
| cantidad | int | NO | | NULL | |
| precio | decimal(10,2) | NO | | NULL | |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> show columns from carrito;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id_articulo | int | NO | MUL | NULL | |
| cantidad | int | NO | | NULL | |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> show columns from fotos_articulos;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| foto | longblob | NO | | NULL | |
| id_articulo | int | NO | MUL | NULL | |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> exit
Bye
humberto@T7-2016630495:~$
```

Copia de archivos de servicio web a la máquina virtual

Para copiar los archivos a la máquina virtual, usaremos el programa SCP, con el cual copiaremos todos los archivos en nuestra carpeta actual en la carpeta "backend_t7" que crearemos previamente usando mkdir.

```
> scp -i ../../azure/humbertowoodys-azure.pem ./ 172.173.252.180:/home/humbertowoodys/backend_t7/
AdaptadorGsonBase64.java
Articulo.java
Carrito.java
Error.java
ParamAltaArticulo.java
ParamAltaCarritoArticulo.java
ParamBorraCarritoArticulo.java
ParamModificaCarritoArticulo.java
RespuestaCarritoArticulo.java
Servicio.java
~/Proyectos/IPN/sistemas-distribuidos-escom/tareas/tarea-7/backend_t7 main
>
```

Los archivos en la máquina virtual se ven así:

```
humbertowoodys@T7-2016630495:~/backend_t7$ pwd
/home/humbertowoodys/backend_t7
humbertowoodys@T7-2016630495:~/backend_t7$ ls -lha
total 60K
drwxrwxr-x 2 humbertowoodys humbertowoodys 4.0K May 27 03:44 .
drwxr-xr-x 12 humbertowoodys humbertowoodys 4.0K May 27 03:42 ..
-rw-r--r-- 1 humbertowoodys humbertowoodys 669 May 27 03:44 AdaptadorGsonBase64.java
-rw-r--r-- 1 humbertowoodys humbertowoodys 144 May 27 03:44 Articulo.java
-rw-r--r-- 1 humbertowoodys humbertowoodys 81 May 27 03:44 Carrito.java
-rw-r--r-- 1 humbertowoodys humbertowoodys 121 May 27 03:44 Error.java
-rw-r--r-- 1 humbertowoodys humbertowoodys 77 May 27 03:44 ParamAltaArticulo.java
-rw-r--r-- 1 humbertowoodys humbertowoodys 98 May 27 03:44 ParamAltaCarritoArticulo.java
-rw-r--r-- 1 humbertowoodys humbertowoodys 83 May 27 03:44 ParamBorraCarritoArticulo.java
-rw-r--r-- 1 humbertowoodys humbertowoodys 102 May 27 03:44 ParamModificaCarritoArticulo.java
-rw-r--r-- 1 humbertowoodys humbertowoodys 129 May 27 03:44 RespuestaCarritoArticulo.java
-rw-r--r-- 1 humbertowoodys humbertowoodys 15K May 27 03:44 Servicio.java
humbertowoodys@T7-2016630495:~/backend_t7$
```

Compilación de servicio web en la máquina virtual

Para compilar el servicio web, usaremos el comando proporcionado por el profesor en la tarea anterior, únicamente modificando la ruta para hacer referencia a nuestro servicio actual. La compilación es la siguiente:

```
humbertowoodys@T7-2016630495:~$ ls
META-INF   Servicio.zip apache-tomcat-8.5.88      backend_t7          compila_json.sh jaxrs-ri           mysql-connector-j-8.0.33      servicio_json
Servicio.war WEB-INF    apache-tomcat-8.5.88.zip base_de_datos_tarea7.sql compil_url.sh jaxrs-ri-2.24.zip mysql-connector-j-8.0.33.zip servicio_url
humbertowoodys@T7-2016630495:~$ javac -cp $CATALINA_HOME/lib/javax.ws.rs-api-2.0.1.jar:$CATALINA_HOME/lib/gson-2.3.1.jar:. backend_t7/Servicio.java
humbertowoodys@T7-2016630495:~$
```

Ahora crearemos la carpeta para almacenar las clases de nuestro servicio, y copiaremos las clases a la misma:

```
humbertowoodys@T7-2016630495:~$ mkdir WEB-INF/classes/backend_t7
humbertowoodys@T7-2016630495:~$ cp backend_t7/*.class WEB-INF/classes/backend_t7/.
```

Luego, crearemos el .war correspondiente a nuestro servicio web:

```
humberto@T7-2016630495:~$ 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 = 656) (out= 294)(deflated 55%)
adding: WEB-INF/classes/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/servicio_url/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/backend_t7/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/backend_t7/Articulo.class(in = 144) (out= 112)(deflated 22%)
adding: WEB-INF/classes/backend_t7/AdaptadorGsonBase64.class(in = 669) (out= 359)(deflated 46%)
adding: WEB-INF/classes/backend_t7/ParamModificaCarritoArticulo.class(in = 102) (out= 87)(deflated 14%)
adding: WEB-INF/classes/backend_t7/RespuestaCarritoArticulo.class(in = 129) (out= 96)(deflated 25%)
adding: WEB-INF/classes/backend_t7/ParamAltaCarritoArticulo.class(in = 98) (out= 84)(deflated 14%)
adding: WEB-INF/classes/backend_t7/Carrito.class(in = 81) (out= 74)(deflated 8%)
adding: WEB-INF/classes/backend_t7/Servicio.class(in = 15104) (out= 2149)(deflated 85%)
adding: WEB-INF/classes/backend_t7/Error.class(in = 121) (out= 87)(deflated 28%)
adding: WEB-INF/classes/backend_t7/ParamAltaArticulo.class(in = 77) (out= 65)(deflated 15%)
adding: WEB-INF/classes/backend_t7/ParamBorraCarritoArticulo.class(in = 83) (out= 78)(deflated 6%)
adding: WEB-INF/classes/servicio_json/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/servicio_json/Usuario.class(in = 435) (out= 295)(deflated 32%)
adding: WEB-INF/classes/servicio_json/AdaptadorGsonBase64.class(in = 1805) (out= 741)(deflated 58%)
adding: WEB-INF/classes/servicio_json/ParamBorraUsuario.class(in = 259) (out= 205)(deflated 20%)
adding: WEB-INF/classes/servicio_json/ParamConsultaUsuario.class(in = 265) (out= 209)(deflated 21%)
adding: WEB-INF/classes/servicio_json/ParamAltaUsuario.class(in = 264) (out= 198)(deflated 25%)
adding: WEB-INF/classes/servicio_json/Servicio.class(in = 8552) (out= 3930)(deflated 54%)
adding: WEB-INF/classes/servicio_json/ParamModificaUsuario.class(in = 272) (out= 204)(deflated 25%)
adding: WEB-INF/classes/servicio_json/Error.class(in = 284) (out= 220)(deflated 22%)
ignoring entry META-INF/
adding: META-INF/context.xml(in = 303) (out= 217)(deflated 28%)
```

Ahora solo resta reiniciar catalina y tendremos nuestro backend funcional en la máquina virtual.

Frontend

Para desarrollar el frontend, se utilizó ChatGPT para generar cada una de las vistas, así como para generar los estilos y el código en JavaScript, para cada una de las pantallas, se incluirá la conversación sostenida con ChatGPT así como una visualización de su contenido. Las vistas que se definieron para cumplir con los requerimientos planteados son:

- **Página Principal:** esta será una simple página principal donde se muestren botones para las dos acciones principales del sistema: crear un artículo y visualizar artículos.
- **Página de Creación de Artículo:** esta página será un formulario simple donde se pueda crear un artículo, deberá serializar las fotos en formato Base64 y deberá enviar la información en formato JSON al endpoint correspondiente del servicio web.
- **Página de visualización de artículos:** esta página será una lista de los artículos disponibles en almacén, para esto se hará un request al cargar la página que obtendrá el catálogo completo y a su vez mostrará una barra de búsqueda que permitirá filtrar los artículos en función del nombre.
- **Página de carrito:** en esta página se mostrará el contenido del carrito, es decir, un listado de los artículos en el carrito así como la cantidad de cada

uno y un precio por artículo (precio individual x cantidad individual) así como un precio total de los contenidos del carrito. Permitirá borrar artículos del carrito individualmente y permitirá borrar todo el carrito. También, permitirá editar la cantidad de artículos para cada uno. Todo esto llamando a las rutas pertinentes.

Creación de páginas con ChatGPT

A continuación se muestra cada una de las conversaciones obtenidas con ChatGPT para la creación de las mismas así como una visualización rápida del diseño final de cada una.

Página principal

La página principal en realidad fue la última en realizarse (esto porque comencé con la de carrito porque ya llevaba un rato y no quedaba y luego me regresé a re-hacer todas puesto que resultó ser más fácil pedírselo al chat), pero la conversación correspondiente es la siguiente:

HU perfecto! ahora, necesito una página principal, título "Tarea 7", mi nombre "Humberto Alejandro Ortega Alcocer", mi boleta "2016630495" y dos links/botones (lo más simple) para ir a las páginas: crear_articulo.html y lista_articulos.html, no tanto diseño pero pues que esté bonita:) que funcione en móvil de preferencia, no importa escritorio, gracias!

iPor supuesto! A continuación, te proporciono un código HTML básico que crea una página de inicio móvil compatible. Este código incluirá tu nombre, boleta y enlaces a las páginas que has mencionado.

```
html
Copy code
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Tarea 7</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            height: 100vh;
            margin: 0;
            background-color: #f8f9fa;
        }

        h1, h2 {
            text-align: center;
        }

        button {
            width: 80px;
            padding: 15px;
            margin: 10px 0;
            font-size: 18px;
            border: none;
            border-radius: 5px;
            background-color: #4CAF50;
            color: white;
        }

        button:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>
    <h1>Tarea 7</h1>
    <h2>Humberto Alejandro Ortega Alcocer</h2>
    <h2>Boleta: 2016630495</h2>

    <button onclick="location.href='crear_articulo.html'">Crear Artículo</button>
    <button onclick="location.href='lista_articulos.html'">Lista de Artículos</button>
</body>
</html>
```

Este código crea una página simple con tu nombre, boleta y dos botones. El primer botón redirige a 'crear_articulo.html' y el segundo redirige a 'lista_articulos.html'. La página es compatible con dispositivos móviles gracias a la metaetiqueta "viewport" y a que utiliza Flexbox para centrar todo el contenido.

Si tienes alguna otra solicitud, no dudes en preguntar. ¡Estoy aquí para ayudarte!

El código correspondiente a la página es el siguiente:

```
<!DOCTYPE html>

<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Tarea 7</title>
    <style>
      body {
        font-family: Arial, sans-serif;
        display: flex;
        flex-direction: column;
        align-items: center;
        justify-content: center;
        height: 100vh;
        margin: 0;
        background-color: #f8f9fa;
      }

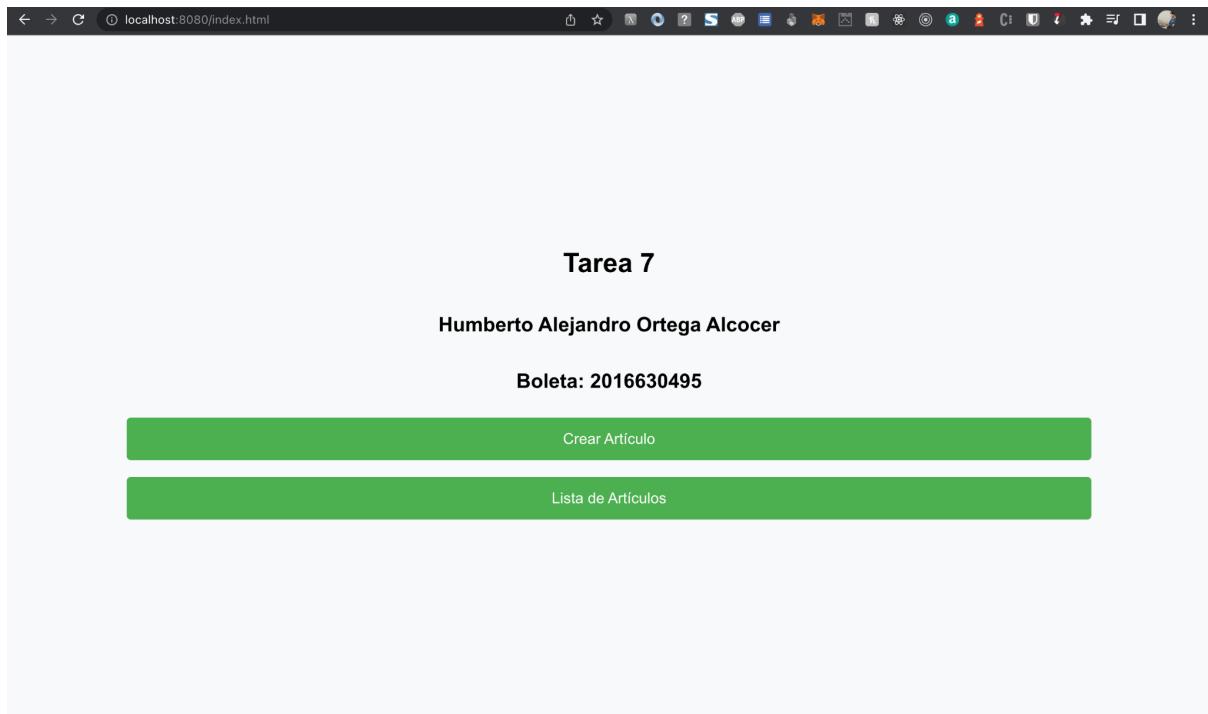
      h1,
      h2 {
        text-align: center;
      }

      button {
        width: 80px;
        padding: 15px;
        margin: 10px 0;
        font-size: 18px;
        border: none;
        border-radius: 5px;
        background-color: #4caf50;
        color: white;
      }

      button:hover {
        background-color: #45a049;
      }
    </style>
  </head>
  <body>
    <h1>Tarea 7</h1>
    <h2>Humberto Alejandro Ortega Alcocer</h2>
    <h2>Boleta: 2016630495</h2>

    <button onclick="location.href='crear_articulo.html'">
      Crear Artículo
    </button>
    <button onclick="location.href='lista_articulos.html'">
      Lista de Artículos
    </button>
  </body>
</html>
```

Y se visualiza de la siguiente manera (en computadora, las pruebas finales serán realizadas en un dispositivo móvil):



Página de creación de artículo

La conversación sostenida con ChatGPT para la creación de esta página fue:

Hola necesito que me des el código en html de un formulario simple para crear un artículo, los datos son: nombre, descripción, precio, cantidad, añade validación (simple) y estilos básicos para que no se vea tan feo, un título que diga: Tarea 7 - Crear Artículo

iHola! Por supuesto, aquí tienes un ejemplo de un formulario en HTML5 que incluye algunas validaciones básicas y estilos básicos:

```
html
<!DOCTYPE html>
<html>
<head>
    <title>Tarea 7 - Crear Artículo</title>
    <style>
        body {
            font-family: Arial, sans-serif;
        }

        form {
            width: 300px;
            margin: auto;
        }

        input, textarea {
            width: 100%;
            margin-bottom: 10px;
            padding: 10px;
        }

        button {
            padding: 10px;
            width: 100%;
            background-color: #4CAF50;
            color: white;
            border: none;
        }

        button:hover {
            background-color: #45a049;
        }

        h1 {
            text-align: center;
        }
    </style>
</head>
<body>
    <h1>Tarea 7 - Crear Artículo</h1>
    <form action="/ruta_a_tu_servidor" method="post">
        <input type="text" name="nombre" id="nombre" required placeholder="Nombre" />
        <textarea name="descripcion" id="descripcion" required placeholder="Descripción" />
        <input type="number" name="precio" id="precio" min="0" step="0.01" required placeholder="Precio" />
        <input type="number" name="cantidad" id="cantidad" min="0" step="1" required placeholder="Cantidad" />
        <button type="submit">Crear artículo</button>
    </form>
</body>
</html>
```

Este formulario incluye campos para el nombre, descripción, precio y cantidad. Todos los campos son obligatorios (`required`) y los campos de precio y cantidad tienen restricciones sobre los valores que pueden aceptar (`min="0"`, `step="0.01"` para precio y `step="1"` para cantidad).

La etiqueta `form` tiene un atributo `action` que deberías modificar con la ruta correcta a tu servidor que maneja la petición POST del formulario.

El código incluye también algunos estilos CSS básicos para hacer que el formulario se vea más atractivo.

Send a message...

ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT May 24 Version

The screenshot shows a ChatGPT interface. At the top, there is a message from the user asking for a more attractive form. Below it, ChatGPT provides a solution using JSON and JavaScript. It includes a code editor window with the following HTML and CSS/JavaScript code:

```
<!DOCTYPE html>
<html>
<head>
    <title>Tarea 7 - Crear Artículo</title>
    <style>
        body {
            font-family: Arial, sans-serif;
        }

        form {
            width: 300px;
            margin: auto;
        }

        input, textarea {
            width: 100%;
            margin-bottom: 10px;
            padding: 10px;
        }

        button {
            padding: 10px;
            width: 100%;
            background-color: #4CAF50;
            color: white;
            border: none;
        }

        button:hover {
            background-color: #45a049;
        }

        h1 {
            text-align: center;
        }
    </style>
</head>
<body>
    <h1>Tarea 7 - Crear Artículo</h1>
    <form id="crear-articulo-form">
        <input type="text" name="nombre" id="nombre" required placeholder="Nombre del artículo" />
        <textarea name="descripcion" id="descripcion" required placeholder="Descripción del artículo" />
        <input type="number" name="precio" id="precio" min="0" step="0.01" required placeholder="Precio" />
        <input type="number" name="cantidad" id="cantidad" min="0" step="1" required placeholder="Cantidad" />
        <button type="submit">Crear artículo</button>
    </form>

    <script>
        document.getElementById('crear-articulo-form').addEventListener('submit', function(event) {
            event.preventDefault();

            var articulo = {
                nombre: document.getElementById('nombre').value,
                descripcion: document.getElementById('descripcion').value,
                precio: document.getElementById('precio').value,
                cantidad: document.getElementById('cantidad').value
            };
        });
    </script>

```

At the bottom of the code editor, there is a message input field with the placeholder "Send a message..." and a "Regenerate response" button.

```
<input type="number" name="precio" id="precio" min="0" step="0.01" required>
<input type="number" name="cantidad" id="cantidad" min="0" step="1" required>
<button type="submit">Crear articulo</button>
</form>

<script>
document.getElementById('crear-articulo-form').addEventListener('submit', event => {
    event.preventDefault();

    var articulo = {
        nombre: document.getElementById('nombre').value,
        descripcion: document.getElementById('descripcion').value,
        precio: document.getElementById('precio').value,
        cantidad: document.getElementById('cantidad').value
    };

    fetch('http://10.10.10.10:8080/crear_articulo', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(articulo)
    }).then(response => {
        if (!response.ok) {
            throw new Error(`HTTP error! status: ${response.status}`)
        }
        return response.json();
    }).then(json => {
        console.log('Articulo creado:', json);
    }).catch(error => {
        console.log('Error:', error);
    });
});
</script>
</body>
</html>
```

Este script JavaScript escucha el evento de envío del formulario, previene el comportamiento por defecto (que es enviar el formulario), luego recoge los datos del formulario, los convierte a JSON y los envía al servidor utilizando 'fetch'. Además, maneja las respuestas y errores de la petición.

HU perfecto! solo una cosa, falta un campo llamado "foto", este debe ser serializado y enviado en base64

¡Claro, puedo agregar ese campo al formulario! El campo será un input de tipo "file" que permitirá al usuario seleccionar un archivo de imagen desde su dispositivo. Luego, con JavaScript, leeremos este archivo como un dato binario y lo codificaremos en base64. Aquí está el código modificado:

The screenshot shows the ChatGPT interface with the following code in the code editor:

```
<!DOCTYPE html>
<html>
<head>
<title>Tarea 7 - Crear Articulo</title>
<style>
body {
    font-family: Arial, sans-serif;
}

form {
    width: 300px;
    margin: auto;
}

input, textarea {
    width: 100%;
    margin-bottom: 10px;
}
```

Below the code editor, there is a message input field with the placeholder "Send a message..." and a "Regenerate response" button. At the bottom of the screen, a small footer reads "ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT May 24 Version".

```
padding: 10px;
width: 100%;
background-color: #4CAF50;
color: white;
border: none;
}

button:hover {
background-color: #45a049;
}

h1 {
text-align: center;
}
</style>
</head>
<body>
<h1>Tarea 7 - Crear Artículo</h1>
<form id="crear-articulo-form">
<input type="text" name="nombre" id="nombre" required placeholder="Nombre del artículo" />
<textarea name="descripcion" id="descripcion" required placeholder="Descripción del artículo" />
<input type="number" name="precio" id="precio" min="0" step="0.01" />
<input type="number" name="cantidad" id="cantidad" min="0" step="1" />
<input type="file" id="foto" required>
<button type="submit">Crear artículo</button>
</form>

<script>
document.getElementById('crear-articulo-form').addEventListener('submit', event => {
event.preventDefault();

var reader = new FileReader();
reader.onloadend = function() {
var base64String = reader.result.replace("data:", "")
.replace(/^.+/, "");

var articulo = {
nombre: document.getElementById('nombre').value,
descripcion: document.getElementById('descripcion').value,
precio: document.getElementById('precio').value,
cantidad: document.getElementById('cantidad').value,
foto: base64String
};

fetch('http://10.10.10.10:8080/crear_articulo', {
method: 'POST',
headers: {
'Content-Type': 'application/json'
},
body: JSON.stringify(articulo)
}).then(response => {
if (!response.ok) {
throw new Error(`HTTP error! status: ${response.status}`);
}
return response.json();
}).then(json => {
console.log('Artículo creado:', json);
}).catch(error => {
console.log('Error:', error);
});
}
reader.readAsDataURL(document.getElementById('foto').files[0]);
});
</script>
</body>
</html>
```

Ahora, cuando se selecciona un archivo de imagen, se lee como un dato binario con `'FileReader.readAsDataURL()'`, se elimina la parte inicial del resultado que incluye el tipo de MIME, y se guarda el restante en la propiedad "foto" del objeto "articulo" antes de enviarlo al servidor. Recuerda que este código asume que siempre se selecciona una imagen antes de enviar el formulario, y no se hace [Regenerate response](#).

Send a message...



ChatGPT may produce inaccurate information about people, places, or facts. [ChatGPT May 24 Version](#)

El código correspondiente a la página solicitada es el siguiente:

```
<!DOCTYPE html>
<html>
<head>
    <title>Tarea 7 - Crear Artículo</title>
    <style>
        body {
            font-family: Arial, sans-serif;
        }

        form {
            width: 300px;
            margin: auto;
        }

        input,
        textarea {
            width: 100%;
            margin-bottom: 10px;
            padding: 10px;
        }

        button {
            padding: 10px;
            width: 100%;
            background-color: #4CAF50;
            color: white;
            border: none;
        }

        button:hover {
            background-color: #45A049;
        }

        h1 {
            text-align: center;
        }
    </style>
</head>
<body>
    <h1>Tarea 7 - Crear Artículo</h1>
    <form id="crear-articulo-form">
        <input
            type="text"
            name="nombre"
            id="nombre"
            required
            placeholder="Nombre"
        />
        <textarea
            name="descripcion"
            id="descripcion"
            required
            placeholder="Descripción"
        ></textarea>
        <input
            type="number"
            name="precio"
            id="precio"
            min="0"
            step="0.01"
            required
            placeholder="Precio"
        />
        <input
            type="number"
            name="cantidad"
            id="cantidad"
        />
    </form>
</body>
```

```
min="0"
step="1"
required
placeholder="Cantidad"
/>
<input type="file" id="foto" required />
<button type="submit">Crear articulo</button>
<hr />
<a href="index.html">Haz click aqui para volver al menú principal.</a>
</form>
<hr />
<p>Humberto Alejandro Ortega Alcocer - 2016630495</p>
<script>
document
.getElementById("crear-articulo-form")
.addEventListener("submit", function (event) {
    event.preventDefault();

    var reader = new FileReader();
    reader.onloadend = function () {
        var base64String = reader.result
            .replace("data:", "")
            .replace(/^.+,/, "");

        var articulo = {
            nombre: document.getElementById("nombre").value,
            descripcion: document.getElementById("descripcion").value,
            precio: document.getElementById("precio").value,
            cantidad: document.getElementById("cantidad").value,
            foto: base64String,
        };

        fetch("http://10.10.10.10:8080/crear_articulo", {
            method: "POST",
            headers: {
                "Content-Type": "application/json",
            },
            body: JSON.stringify(articulo),
        })
            .then((response) => {
                if (!response.ok) {
                    throw new Error(`HTTP error! status: ${response.status}`);
                }
                return response.json();
            })
            .then((json) => {
                console.log("Articulo creado:", json);
            })
            .catch((error) => {
                console.log("Error:", error);
            });
    };
    reader.readAsDataURL(document.getElementById("foto").files[0]);
});
</script>
</body>
</html>
```

La visualización de dicha página (en mi computadora local para probar):

The screenshot shows a web page titled "Tarea 7 - Crear Artículo". It contains four input fields: "Nombre", "Descripción", "Precio", and "Cantidad". Below these is a file input field with the placeholder "Choose File" and the message "No file chosen". At the bottom is a green button labeled "Crear artículo". Below the form is a link: "Haz click aquí para volver al menú principal".

Humberto Alejandro Ortega Alcocer - 2016630495

Página de visualización de artículos

Al igual que con las páginas anteriores, se solicitó a ChatGPT generar la página, esta es la conversación sostenida con ChatGPT:

The screenshot shows a ChatGPT conversation. The user asks for a page to list articles, specifying JSON requests and responses. The AI responds with sample code for an HTML file. The code includes a title, styles for the body and header, and a container for the articles.

```
<!DOCTYPE html>
<html>
<head>
    <title>Lista de Articulos</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
        }

        .header {
            background-color: #4CAF50;
            padding: 10px;
            color: white;
            text-align: center;
        }

        .container {
            display: flex;
            flex-wrap: wrap;
            justify-content: space-around;
            padding: 20px;
        }
    </style>
</head>
<body>
```

```
.container {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-around;
    padding: 20px;
}

.article {
    border: 1px solid #ccc;
    border-radius: 5px;
    margin: 10px;
    padding: 10px;
    width: calc(100% / 4 - 40px);
}

.article button {
    width: 100%;
    padding: 10px;
    margin-top: 10px;
    background-color: #4CAF50;
    color: white;
    border: none;
}

.article button:hover {
    background-color: #45a049;
}

.search {
    margin: 10px;
}

```

</style>

```
</head>
<body>
    <div class="header">
        <h1>Lista de Articulos</h1>
        <input type="text" id="search" class="search" placeholder="Buscar articulo...">
        <button id="carrito-btn">Carrito</button>
    </div>
    <div id="articles-container" class="container"></div>

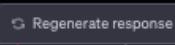
    <script>
        document.getElementById('carrito-btn').addEventListener('click', function() {
            window.location.href = '/carrito';
        });

        fetch('http://10.10.10.10:8080/articulos')
            .then(response => response.json())
            .then(data => {
                var container = document.getElementById('articles-container');
                data.forEach(articulo => {
                    var articleDiv = document.createElement('div');
                    articleDiv.className = 'article';

                    articleDiv.innerHTML = `
                        <h2>${articulo.nombre}</h2>
                        <p>Precio: ${articulo.precio}</p>
                        <p>Cantidad Disponible: ${articulo.cantidad}</p>
                        <button class="ver-mas-btn" data-id="${articulo.id}">Ver más</button>
                        <input type="number" min="1" max="${articulo.cantidad}" data-id="${articulo.id}">
                        <button class="add-to-cart-btn" data-id="${articulo.id}">Añadir al carrito</button>
                    `;

                    container.appendChild(articleDiv);
                });
            });
    </script>

```

Send a message...  

ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT May 24 Version

```
>
<div class="header">
    <h1>Lista de Artículos</h1>
    <input type="text" id="search" class="search" placeholder="Buscar artículo..." />
    <button id="carrito-btn">Carrito</button>
</div>
<div id="articles-container" class="container"></div>

<script>
    document.getElementById('carrito-btn').addEventListener('click', function() {
        window.location.href = '/carrito';
    });

    fetch('http://10.10.10.10:8080/articulos')
        .then(response => response.json())
        .then(data => {
            var container = document.getElementById('articles-container');
            data.forEach(articulo => {
                var articleDiv = document.createElement('div');
                articleDiv.className = 'article';

                articleDiv.innerHTML =
                    <h2>${articulo.nombre}</h2>
                    <p>Precio: ${articulo.precio}</p>
                    <p>Cantidad Disponible: ${articulo.cantidad}</p>
                    <button class="ver-mas-btn" data-id="${articulo.id}">Ver más</button>
                    <input type="number" min="1" max="${articulo.cantidad}" value="1" />
                    <button class="add-to-cart-btn" data-id="${articulo.id}">Añadir al carrito</button>
            );
            container.appendChild(articleDiv);
        });

        document.querySelectorAll('.ver-mas-btn').forEach(btn => {
            btn.addEventListener('click', function() {
                var id = this.getAttribute('data-id');
                var articulo = data.find(articulo => articulo.id == id);
                alert(`Descripción: ${articulo.descripcion}`);
            });
        });

        document.querySelectorAll('.add-to-cart-btn').forEach(btn => {
            btn.addEventListener('click', function() {
                var id = this.getAttribute('data-id');
                var cantidad = this.previousElementSibling.value;
                fetch('http://10.10.10.10:8080/alta_carrito', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json'
                    },
                    body: JSON.stringify({ id, cantidad })
                })
                .then(response => response.json())
                .then(json => {
                    console.log('Artículo añadido:', json);
                });
            });
        });
    });
</script>
</body>
</html>
```

Este código creará una lista de artículos basada en los datos recibidos del servidor. Cada artículo incluirá un botón "Ver más" que mostrará una alerta con la descripción del artículo, y un botón "Añadir al carrito" que enviará una solicitud POST al servidor con el id del artículo y la cantidad seleccionada.

El botón "Carrito" en la parte superior de la página redirigirá al usuario a la página del carrito.

Por último, el campo de búsqueda es solo una entrada de texto sin funcionalidad en este momento. Necesitarías añadir código JavaScript adicional para filtrar los artículos basándote en lo que el usuario escribe. [Regenerate response](#)

Send a message...



ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT May 24 Version

El código de la página resultante es el siguiente:

```
<!DOCTYPE html>
<html>
<head>
    <title>Lista de Artículos</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
        }

        .header {
            background-color: #4CAF50;
            padding: 10px;
            color: white;
            text-align: center;
        }

        .container {
            display: flex;
            flex-wrap: wrap;
            justify-content: space-around;
            padding: 20px;
        }

        .article {
            border: 1px solid #ccc;
            border-radius: 5px;
            margin: 10px;
            padding: 10px;
            width: calc(100% / 4 - 40px);
        }

        .article button {
            width: 100%;
            padding: 10px;
            margin-top: 10px;
            background-color: #4CAF50;
            color: white;
            border: none;
        }

        .article button:hover {
            background-color: #45A049;
        }

        .search {
            margin: 10px;
        }
    </style>
</head>
<body>
    <div class="header">
        <h1>Lista de Artículos</h1>
        <input
            type="text"
            id="search"
            class="search"
            placeholder="Buscar artículos...">
    </div>
    <div id="articles-container" class="container"></div>

    <hr />
    <a href="#">Volver al menú principal</a>
    <p>Humberto Alejandro Ortega Alcocer - 2016630495</p>
</body>
```

```
<script>
  document
    .getElementById("carrito-btn")
    .addEventListener("click", function () {
      window.location.href = "/carrito.html";
    });

  fetch("http://10.10.10.10:8080/articulos")
    .then((response) => response.json())
    .then((data) => {
      var container = document.getElementById("articles-container");
      data.forEach((articulo) => {
        var articleDiv = document.createElement("div");
        articleDiv.className = "article";

        articleDiv.innerHTML = `
          <h2>${articulo.nombre}</h2>
          <p>Precio: ${articulo.precio}</p>
          <p>Cantidad Disponible: ${articulo.cantidad}</p>
          <button class="ver-mas-btn" data-id="${articulo.id}">Ver más</button>
          <input type="number" min="1" max="${articulo.cantidad}" value="1">
          <button class="add-to-cart-btn" data-id="${articulo.id}">Añadir al carrito</button>
        `;

        container.appendChild(articleDiv);
      });
    });

  document.querySelectorAll(".ver-mas-btn").forEach((btn) => {
    btn.addEventListener("click", function () {
      var id = this.getAttribute("data-id");
      var articulo = data.find((articulo) => articulo.id == id);
      alert(`Descripción: ${articulo.descripcion}`);
    });
  });

  document.querySelectorAll(".add-to-cart-btn").forEach((btn) => {
    btn.addEventListener("click", function () {
      var id = this.getAttribute("data-id");
      var cantidad = this.previousElementSibling.value;
      fetch("http://10.10.10.10:8080/alta_carrito", {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify({ id, cantidad }),
      })
        .then((response) => response.json())
        .then((json) => {
          console.log("Artículo añadido:", json);
        });
    });
  });
</script>
</body>
</html>
```

Y la visualización de la página en cuestión (en mi computadora, posteriormente se harán pruebas en un dispositivo móvil) es la siguiente:

The screenshot shows a web browser window with the URL `localhost:8080/lista_articulos.html`. The page title is "Lista de Artículos". There is a search bar labeled "Buscar artículos..." and a "Carrito" button. The main content area displays three articles:

- Laptop**: Precio: \$25000, Cantidad Disponible: 15. Buttons: Ver más, Añadir al carrito.
- Jabón Zote**: Precio: \$20, Cantidad Disponible: 5. Buttons: Ver más, Añadir al carrito.
- Shampoo**: Precio: \$30, Cantidad Disponible: 10. Buttons: Ver más, Añadir al carrito.

[Volver al menú principal](#)

Humberto Alejandro Ortega Alcocer - 2016630495

Y, al hacer click en ver más:

The screenshot shows the same web application interface. A modal dialog box has appeared in the center of the screen with the text "localhost:8080 says" and "Descripción: Un jabón enorme, grandote, un jabonzote. Estoy cansado profe." followed by an "OK" button. The main content area below the dialog box is identical to the previous screenshot, showing the three articles: Laptop, Jabón Zote, and Shampoo.

[Volver al menú principal](#)

Humberto Alejandro Ortega Alcocer - 2016630495

Página de carrito

Para generar la página de carrito, se sostuvo la siguiente conversación con ChatGPT:

The screenshot shows a ChatGPT conversation. The user (HU) asks for the HTML code for a shopping cart. ChatGPT responds with a JSON schema for the cart items and a detailed description of the requirements. It then provides a sample HTML and CSS code for a shopping cart page.

genial! ahora podrías darme el html para el carrito? debe hacer un request a /consulta_carrito y el json tendrá algo así

```
[  
  {  
    articulo {objeto del articulo},  
    cantidad: int,  
  },  
  ...  
]
```

la idea es que el usuario pueda visualizar el carrito, dónde vea el producto, el costo unitario, la cantidad y el total (costo * cantidad), una opción de eliminar del carrito para cada artículo (con confirmación) y otra opción para cambiar la cantidad de un artículo (checando que haya disponibles aún (para esto, el artículo que viene en el json incluye la cantidad disponible)) si la cantidad es más de la disponible, un mensaje de error, al final una suma del total de todos los precios y un botón para borrar carrito completo (con confirmación) que llame a DELETE /borrar_carrito

Por supuesto! A continuación se muestra un código HTML con JavaScript incorporado que cumple con tus requisitos. Este código es solo un ejemplo y podría mejorarse para cumplir con mejores prácticas de desarrollo.

```
html  
<!DOCTYPE html>  
<html>  
<head>  
  <title>Carrito de Compras</title>  
  <style>  
    body {  
      font-family: Arial, sans-serif;  
      margin: 0;  
      padding: 0;  
    }  
  
    .header {  
      background-color: #4CAF50;  
      padding: 10px;  
      color: white;  
      text-align: center;  
    }  
  
    .container {  
      padding: 20px;  
    }  
  </style>  
</head>  
<body>  
  <div class="header">Carrito de Compras</div>  
  <div class="container">  
    <p>Este es tu carrito de compras. Puedes eliminar un artículo o cambiar su cantidad.</p>  
    <table border="1">  
      <thead>  
        <tr>  
          <th>Artículo</th>  
          <th>Cantidad</th>  
          <th>Acciones</th>  
        </tr>  
      </thead>  
      <tbody>  
        <tr>  
          <td>Artículo 1</td>  
          <td>2</td>  
          <td>  
            <a href="#">Eliminar</a>  
            <a href="#">Cambiar cantidad</a>  
          </td>  
        </tr>  
        <tr>  
          <td>Artículo 2</td>  
          <td>1</td>  
          <td>  
            <a href="#">Eliminar</a>  
            <a href="#">Cambiar cantidad</a>  
          </td>  
        </tr>  
      </tbody>  
    </table>  
    <div>Total: $100</div>  
    <button>Borrar carrito</button>  
  </div>  
</body>
```

Send a message...

ChatGPT may produce inaccurate information about people, places, or facts. [ChatGPT May 24 Version](#)

```
}

.container {
    padding: 20px;
}

.article {
    border: 1px solid #ccc;
    border-radius: 5px;
    margin: 10px;
    padding: 10px;
}

.article button {
    width: 100%;
    padding: 10px;
    margin-top: 10px;
    background-color: #4CAF50;
    color: white;
    border: none;
}

.article button:hover {
    background-color: #45a049;
}

</style>
</head>
<body>
    <div class="header">
        <h1>Carrito de Compras</h1>
    </div>
    <div id="cart-container" class="container"></div>
    <button id="clear-cart-btn">Borrar Carrito</button>

    <script>
        document.getElementById('clear-cart-btn').addEventListener('click',
            if (confirm('¿Seguro que quieres borrar el carrito?')) {
                fetch('http://10.10.10.8080/borrar_carrito', {
                    method: 'DELETE'
                })
                .then(response => {
                    if (response.ok) {
                        location.reload();
                    } else {
                        alert('Error al borrar el carrito');
                    }
                });
            }
        );
    </script>
    <div style="text-align: right; margin-top: 10px;>
        <a href="#">Regenerate response</a>
    </div>

```

Send a message... 

ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT May 24 Version

```
        '',
    });

fetch('http://10.10.10.10:8080/consulta_carrito')
.then(response => response.json())
.then(data => {
    var container = document.getElementById('cart-container');
    var total = 0;
    data.forEach(item => {
        var articleDiv = document.createElement('div');
        articleDiv.className = 'article';

        var subtotal = item.articulo.precio * item.cantidad;
        total += subtotal;

        articleDiv.innerHTML =
            <h2>${item.articulo.nombre}</h2>
            <p>Precio: ${item.articulo.precio}</p>
            <p>Cantidad: ${item.cantidad}</p>
            <p>Total: ${subtotal}</p>
            <input type="number" min="1" max="${item.articulo.ca
            <button class="update-quantity-btn" data-id="${item.
            <button class="remove-from-cart-btn" data-id="${item
        ';

        container.appendChild(articleDiv);
    });
}

container.innerHTML += `<p>Total: ${total}</p>`;

document.querySelectorAll('.update-quantity-btn').forEach(bt
btn.addEventListener('click', function() {
    var id = this.getAttribute('data-id');
    var cantidad = this.previousElementSibling.value;
    fetch('http://10.10.10.10:8080/actualizar_carrito',
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({ id, cantidad })
    )
    .then(response => {
        if (response.ok) {
            location.reload();
        } else {
            alert('Error al actualizar la cantidad');
        }
    });
});
```

Regenerate response

Send a message... 

ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT May 24 Version

```
        if (response.ok) {
            location.reload();
        } else {
            alert('Error al actualizar la cantidad');
        }
    });
});
});

document.querySelectorAll('.remove-from-cart-btn').forEach(btn => {
    btn.addEventListener('click', function() {
        var id = this.getAttribute('data-id');
        if (confirm(`¿Seguro que quieres eliminar este artículo?`)) {
            fetch(`http://10.10.10.10:8080/eliminar_del_carrito/${id}`)
                .method: 'DELETE'
        })
        .then(response => {
            if (response.ok) {
                location.reload();
            } else {
                alert('Error al eliminar el artículo del carrito');
            }
        });
    });
});
</script>
</body>
</html>
```

Este código mostrará cada artículo en el carrito de compras, incluyendo su nombre, precio, cantidad y total. Cada artículo incluirá un botón "Eliminar del Carrito" que eliminará el artículo del carrito después de una confirmación, y un botón "Actualizar Cantidad" que actualizará la cantidad del artículo en el carrito.

El botón "Borrar Carrito" eliminará todos los artículos del carrito después de una confirmación.

Ten en cuenta que este código asume que los endpoints para actualizar la cantidad de un artículo en el carrito y eliminar un artículo del carrito son 'http://10.10.10.10:8080/actualizar_carrito' y 'http://10.10.10.10:8080/eliminar_del_carrito/{id}', respectivamente. Si tus endpoints son diferentes, deberías actualizar el código en consecuencia.

 Regenerate response

| Send a message...



ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT May 24 Version

El código generado es el siguiente:

```
<!DOCTYPE html>
<html>
<head>
    <title>Carrito de Compras</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
        }

        .header {
            background-color: #4CAF50;
            padding: 10px;
            color: white;
            text-align: center;
        }

        .container {
            padding: 20px;
        }

        .article {
            border: 1px solid #ccc;
            border-radius: 5px;
            margin: 10px;
            padding: 10px;
        }

        .article button {
            width: 100%;
            padding: 10px;
            margin-top: 10px;
            background-color: #4CAF50;
            color: white;
            border: none;
        }

        .article button:hover {
            background-color: #45A049;
        }
    </style>
</head>
<body>
    <div class="header">
        <h1>Carrito de Compras</h1>
    </div>
    <div id="cart-container" class="container"></div>
    <button id="clear-cart-btn">Borrar Carrito</button>
    <hr />
    <a href="index.html">Volver al menú principal</a>
    <a href="lista_articulos.html">Volver al listado de artículos</a>
    <p>Humberto Alejandro Ortega Alcocer - 2016630495</p>
    <script>
        document
            .getElementById("clear-cart-btn")
            .addEventListener("click", function () {
                if (confirm("¿Seguro que quieres borrar el carrito?")) {
                    fetch("http://10.10.10.10:8080/borrar_carrito", {
                        method: "DELETE",
                    }).then((response) => {
                        if (response.ok) {
                            location.reload();
                        } else {
                            alert("Error al borrar el carrito");
                        }
                    });
                }
            });
    </script>
</body>
</html>
```

```
}

});

fetch("http://10.10.10.10:8080/consulta_carrito")
.then((response) => response.json())
.then((data) => {
  var container = document.getElementById("cart-container");
  var total = 0;
  data.forEach((item) => {
    var articleDiv = document.createElement("div");
    articleDiv.className = "article";

    var subtotal = item.articulo.precio * item.cantidad;
    total += subtotal;

    articleDiv.innerHTML = `
      <h2>${item.articulo.nombre}</h2>
      <p>Precio: ${item.articulo.precio}</p>
      <p>Cantidad: ${item.cantidad}</p>
      <p>Total: ${subtotal}</p>
      <input type="number" min="1" max="${item.articulo.cantidad}" value="${item.cantidad}">
      <button class="update-quantity-btn" data-id="${item.articulo.id}">Actualizar Cantidad</button>
      <button class="remove-from-cart-btn" data-id="${item.articulo.id}">Eliminar del Carrito</button>
    `;
    container.appendChild(articleDiv);
  });
  container.innerHTML += `<p>Total: ${total}</p>`;

  document.querySelectorAll(".update-quantity-btn").forEach((btn) => {
    btn.addEventListener("click", function () {
      var id = this.getAttribute("data-id");
      var cantidad = this.previousElementSibling.value;
      fetch("http://10.10.10.10:8080/actualizar_carrito", {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify({ id, cantidad }),
      }).then((response) => {
        if (response.ok) {
          location.reload();
        } else {
          alert("Error al actualizar la cantidad");
        }
      });
    });
  });
}

document.querySelectorAll(".remove-from-cart-btn").forEach((btn) => {
  btn.addEventListener("click", function () {
    var id = this.getAttribute("data-id");
    if (
      confirm(
        "¿Seguro que quieres eliminar este articulo del carrito?"
      )
    ) {
      fetch(`http://10.10.10.10:8080/eliminar_del_carrito/${id}`, {
        method: "DELETE",
      }).then((response) => {
        if (response.ok) {
          location.reload();
        } else {
          alert("Error al eliminar el articulo del carrito");
        }
      });
    }
  });
})
```

```
        });
    });
});
</script>
</body>
</html>
```

Y la visualización será la siguiente (en mi local, para las pruebas se usará un dispositivo móvil):

The screenshot shows a mobile browser window with the URL `localhost:8080/carrito.html`. The title bar says "Carrito de Compras". The main content area displays two items: "Shampoo" and "Jabón Zote".

Shampoo
Precio: \$30
Cantidad: 2
Total: \$60
[2]
Actualizar Cantidad
Eliminar del Carrito

Jabón Zote
Precio: \$20
Cantidad: 4
Total: \$80
[4]
Actualizar Cantidad
Eliminar del Carrito

Total: \$140
Borrar Carrito
[Volver al menú principal](#) [Volver al listado de artículos](#)
Humberto Alejandro Ortega Alcocer - 2016630495

Al modificar una cantidad del carrito:

The screenshot shows a mobile browser window with the URL `localhost:8080/carrito.html`. A black confirmation dialog box is displayed in the center, containing the text "localhost:8080 says" and "Cantidad actualizada correctamente", with an "OK" button.

The main content area shows the same shopping cart items as before, but the quantity for Shampoo has been updated to 3.

Shampoo
Precio: \$30
Cantidad: 2
Total: \$60
[3]
Actualizar Cantidad
Eliminar del Carrito

Jabón Zote
Precio: \$20
Cantidad: 4
Total: \$80
[4]
Actualizar Cantidad
Eliminar del Carrito

Total: \$140
Borrar Carrito
[Volver al menú principal](#) [Volver al listado de artículos](#)
Humberto Alejandro Ortega Alcocer - 2016630495

Al eliminar un elemento del carrito:

A screenshot of a web browser window titled "localhost:8080/carrito.html". The main content shows two items: "Shampoo" and "Jabón Zote". The "Shampoo" section has a quantity of 3, a price of \$30, and a total of \$60. The "Jabón Zote" section has a quantity of 4, a price of \$20, and a total of \$80. Below the items is a "Borrar Carrito" button. A black confirmation dialog box is centered over the page, containing the text "localhost:8080 says" and "¿Seguro que quieres eliminar este artículo del carrito?". It has "Cancel" and "OK" buttons.

Y después de confirmar:

A screenshot of a web browser window titled "localhost:8080/carrito.html". The title bar now reads "Carrito de Compras". The content shows only the "Jabón Zote" item, which has a quantity of 4, a price of \$20, and a total of \$80. Below the item is a "Borrar Carrito" button. A small message at the bottom left indicates "Total: \$80". The footer includes links to "Volver al menú principal" and "Volver al listado de artículos", and the text "Humberto Alejandro Ortega Alcocer - 2016630495".

Borrando todo el carrito:

A screenshot of a web browser window titled "localhost:8080/carrito.html". A black confirmation dialog box is centered over the page, containing the text "localhost:8080 says" and "¿Seguro que quieres borrar el carrito?". It has "Cancel" and "OK" buttons. The main content area is mostly empty, showing only the "Jabón Zote" item and its details (quantity 4, total \$80) and a "Borrar Carrito" button. The footer includes links to "Volver al menú principal" and "Volver al listado de artículos", and the text "Humberto Alejandro Ortega Alcocer - 2016630495".

Copia de archivos a la máquina virtual

Para copiar los archivos a la máquina virtual, usaremos el comando de scp, de la siguiente manera:

```
> pwd  
/Users/humbertowoody/Proyectos/IPN/sistemas-distribuidos-escom/tareas/tarea-7/frontend_t7  
> scp -i ../../../../azure/humbertowood-azure.pem ./html 172.173.252.180:/home/humbertowoody/frontend_t7/  
carrito.html  
crear_articulo.html  
index.html  
lista_articulos.html  
~/Proyectos/IPN/sistemas-distribuidos-escom/tareas/tarea-7/frontend_t7 main*  
>
```

Confirmaremos que los archivos se hayan copiado exitosamente:

```
humbertowoody@T7-2016630495:~/frontend_t7$ pwd  
/home/humbertowoody/frontend_t7  
humbertowoody@T7-2016630495:~/frontend_t7$ ls -lha  
total 28K  
drwxrwxr-x 2 humbertowoody humbertowoody 4.0K May 27 04:38 .  
drwxr-xr-x 13 humbertowoody humbertowoody 4.0K May 27 04:37 ..  
-rw-r--r-- 1 humbertowoody humbertowoody 4.0K May 27 04:37 carrito.html  
-rw-r--r-- 1 humbertowoody humbertowoody 3.1K May 27 04:37 crear_articulo.html  
-rw-r--r-- 1 humbertowoody humbertowoody 1.1K May 27 04:37 index.html  
-rw-r--r-- 1 humbertowoody humbertowoody 4.2K May 27 04:37 lista_articulos.html  
humbertowoody@T7-2016630495:~/frontend_t7$
```

Copia de archivos a directorio de Tomcat

Para que los archivos estén disponibles, los copiaremos al directorio de Tomcat correspondiente, de la siguiente manera:

```
humbertowoody@T7-2016630495:~$ pwd  
/home/humbertowoody  
humbertowoody@T7-2016630495:~$ cp frontend_t7/*.html apache-tomcat-8.5.88/webapps/ROOT/  
humbertowoody@T7-2016630495:~$ ls -lha apache-tomcat-8.5.88/webapps/ROOT/  
total 60K  
drwxrwxr-x 2 humbertowoody humbertowoody 4.0K May 27 04:40 .  
drwxrwxr-x 4 humbertowoody humbertowoody 4.0K May 9 01:39 ..  
-rw-r--r-- 1 humbertowoody humbertowoody 3.6K May 9 01:09 WSClient.js  
-rw-r--r-- 1 humbertowoody humbertowoody 4.0K May 27 04:40 carrito.html  
-rw-r--r-- 1 humbertowoody humbertowoody 3.1K May 27 04:40 crear_articulo.html  
-rw-r--r-- 1 humbertowoody humbertowoody 1.1K May 27 04:40 index.html  
-rw-r--r-- 1 humbertowoody humbertowoody 4.2K May 27 04:40 lista_articulos.html  
-rw-r--r-- 1 humbertowoody humbertowoody 10K May 9 01:09 prueba_json.html  
-rw-r--r-- 1 humbertowoody humbertowoody 10K May 9 01:09 prueba_url.html  
-rw-r--r-- 1 humbertowoody humbertowoody 1.7K May 9 01:09 usuario_sin_foto.png  
humbertowoody@T7-2016630495:~$
```

Prueba de funcionamiento

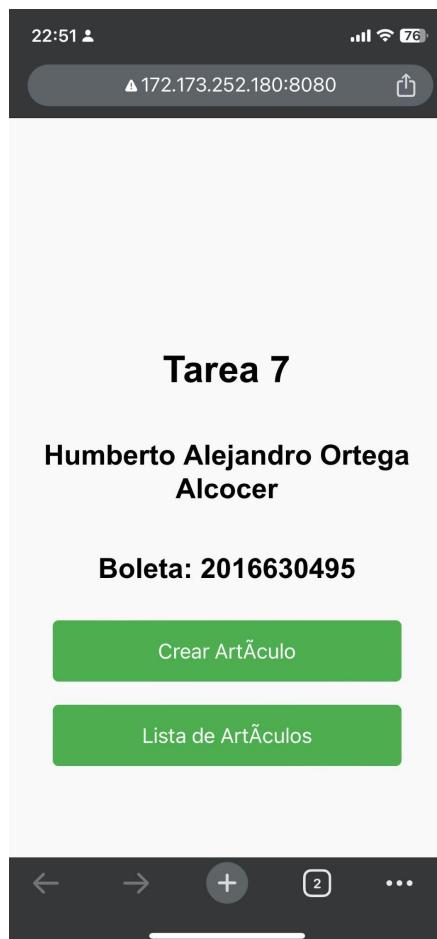
En esta sección se realizarán las pruebas de funcionamiento general del sistema siguiendo un flujo que permita mostrar cada una de las especificaciones solicitadas en los requerimientos funcionales. Para esto, usaré mi celular para tomar capturas de cada pantalla.

Cabe destacar que fue necesario realizar los siguientes ajustes al frontend:

- Se sustituyó la IP que usé durante las pruebas (mi IP de LAN de mi computadora local) por la IP pública de la máquina virtual: 172.173.252.180
- Dado que no se colocó que se usara UTF-8 para las páginas, la mayoría muestran problemas con acentos, por el tiempo no se pudo corregir, pero se puede observar en las capturas anteriores a esta prueba total, que funcionaban bien los acentos en local.

Página principal

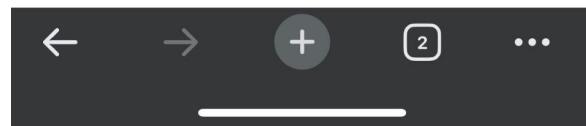
La página principal en móvil no es la *más bonita* pero se visualiza de la siguiente manera:



Creación de artículo

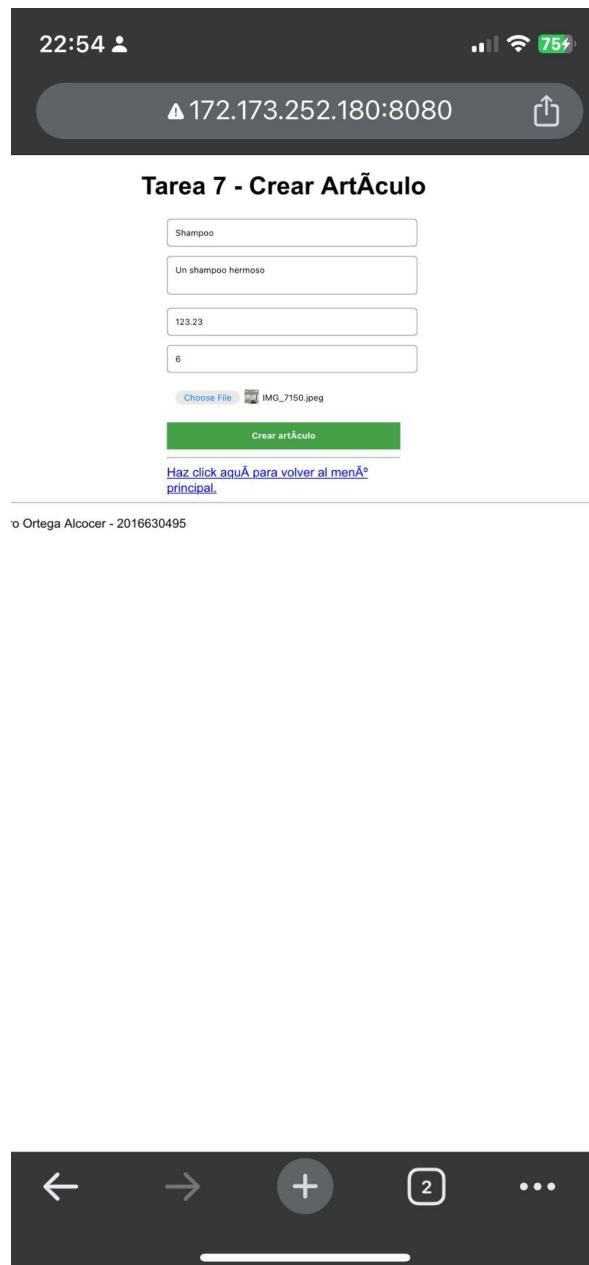
Como se puede observar, el diseño no le quedó a ChatGPT para móviles.

The screenshot shows a mobile browser interface with the address bar displaying '172.173.252.180:8080'. The main content is a form titled 'Tarea 7 - Crear Artículo'. The form fields include 'Nombre', 'Descripción', 'Precio', and 'Cantidad'. Below these is a file input field labeled 'Choose File' with the placeholder 'no file selected'. A green button labeled 'Crear artículo' is present. At the bottom, there is a link 'Haz click aquí para volver al menú principal.' and the footer 'Humberto Alejandro Ortega Alcocer - 2016630495'.



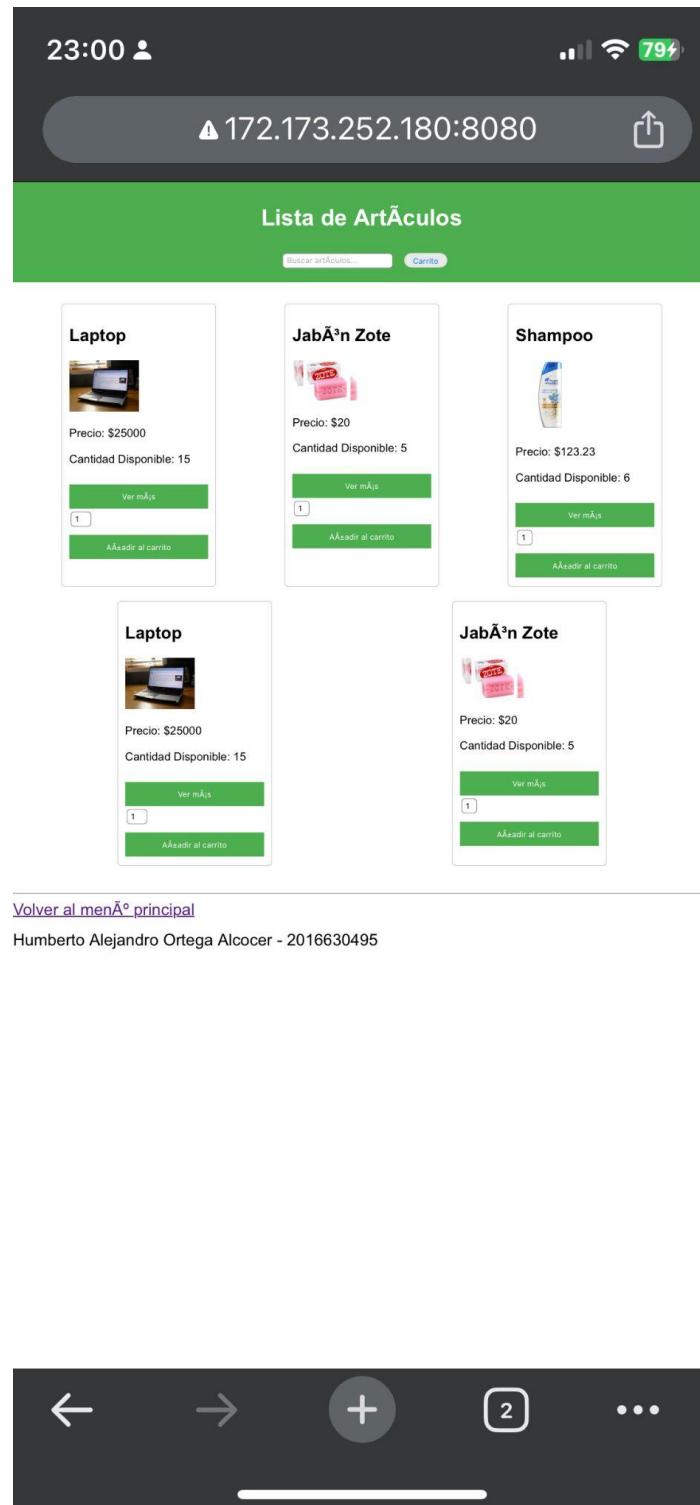
Creación de artículo exitosa

Cuando se realiza la creación de un artículo, el formulario se visualiza de la siguiente manera:



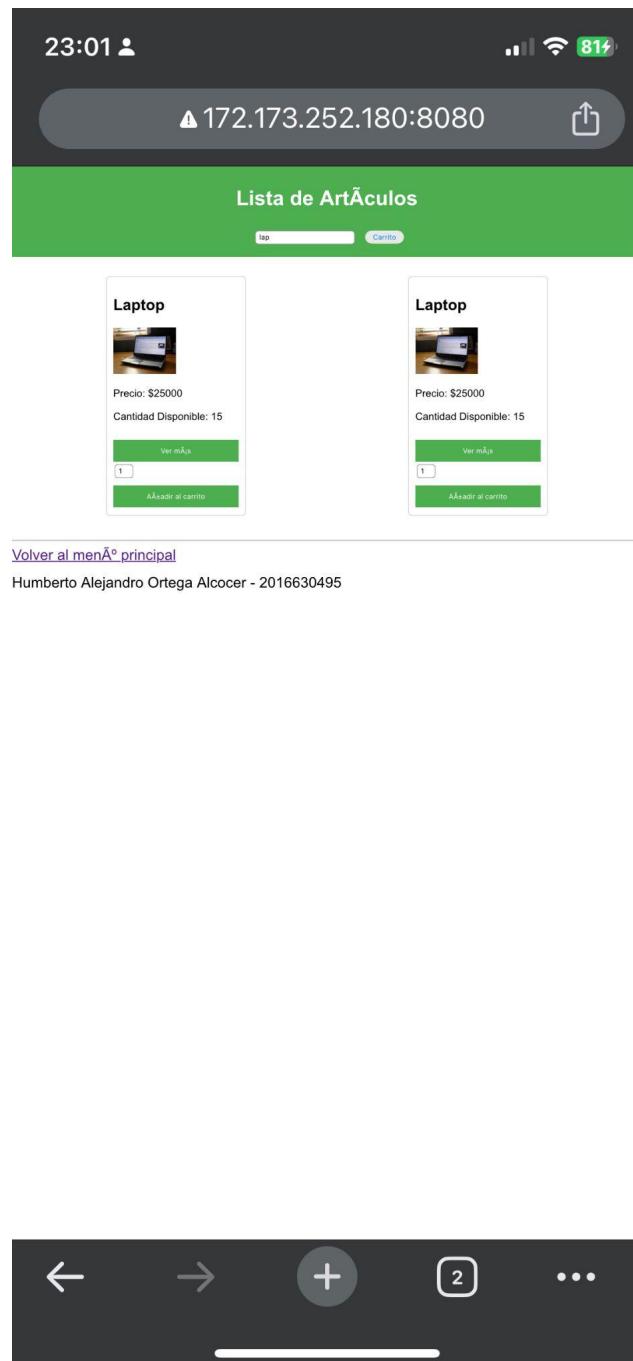
Catálogo de artículos

El catálogo de artículos se visualiza de la siguiente manera (ya había realizado varias pruebas, por eso los artículos ya existentes):



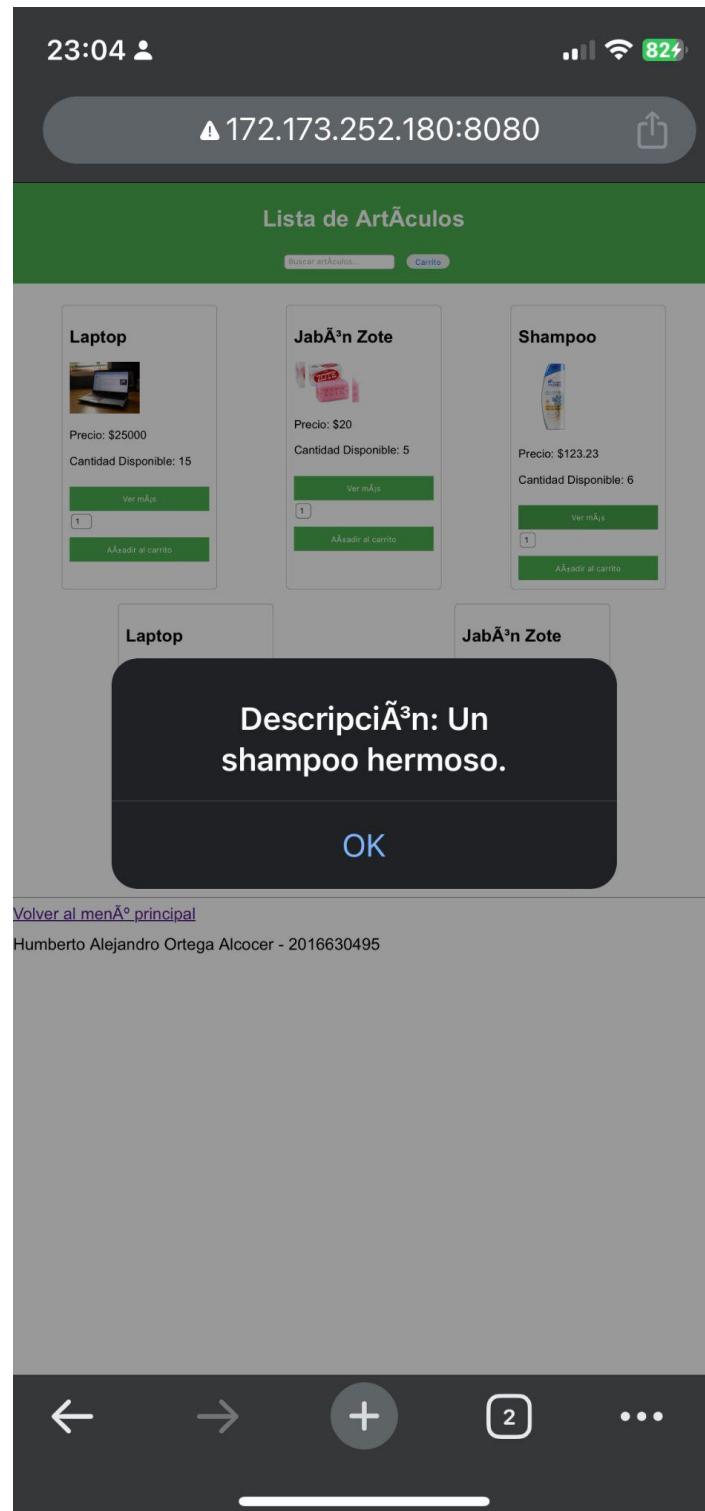
Filtrado de artículos por nombre

Aquí se observa como en el recuadro de búsqueda se coloca “lap” y solo se muestran los resultados que hacen match con la expresión. Esta búsqueda se hace con JavaScript, es decir, no se realiza ningún request al backend ni una consulta, esto fue porque me pareció más eficiente ya que contamos con los datos ya cargados.



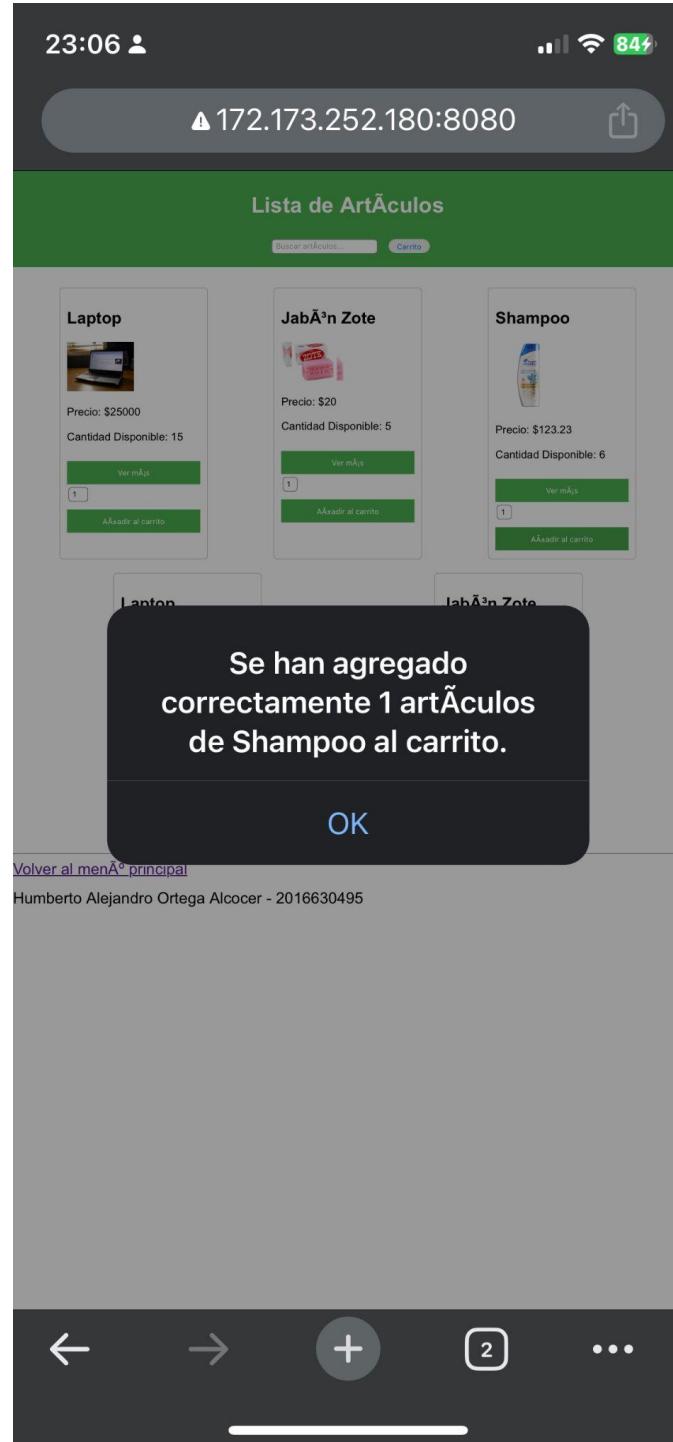
Detalles de un artículo

Al hacer click en *Ver más*, se muestra una alerta con la descripción del artículo:



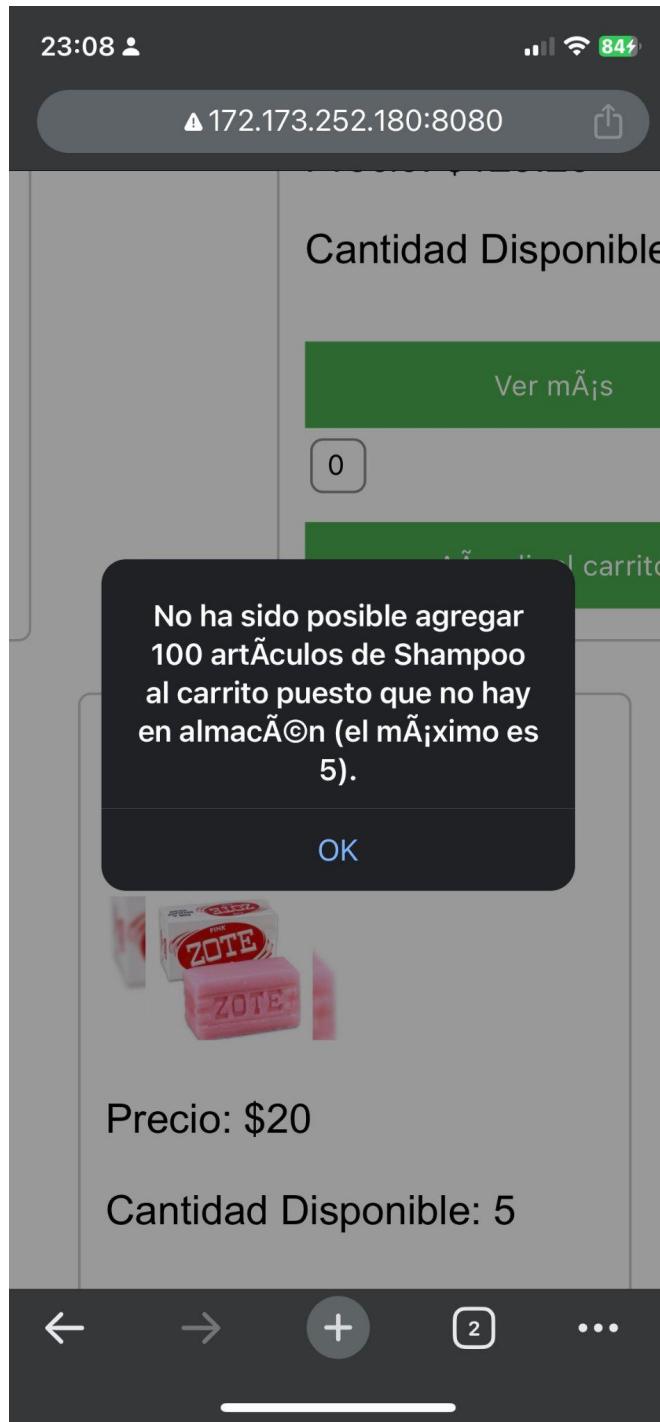
Agregar al carrito exitoso

Al hacer click en *Agregar al carrito* se mostrará una confirmación de la adición:



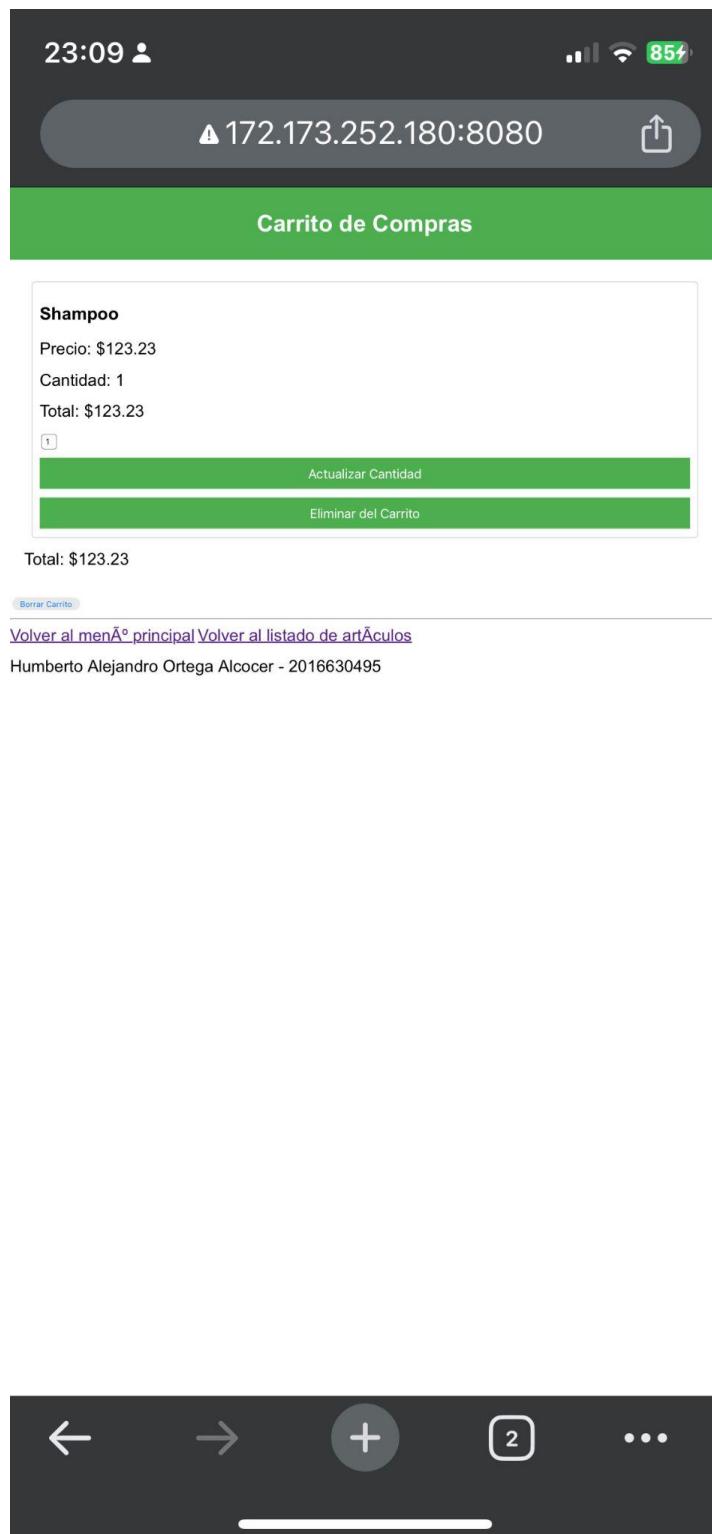
Agregar al carrito fallido

Dado que inicialmente teníamos 6 artículos disponibles, si ahora intentamos agregar 100 artículos al carrito, nos arrojará este error la aplicación:



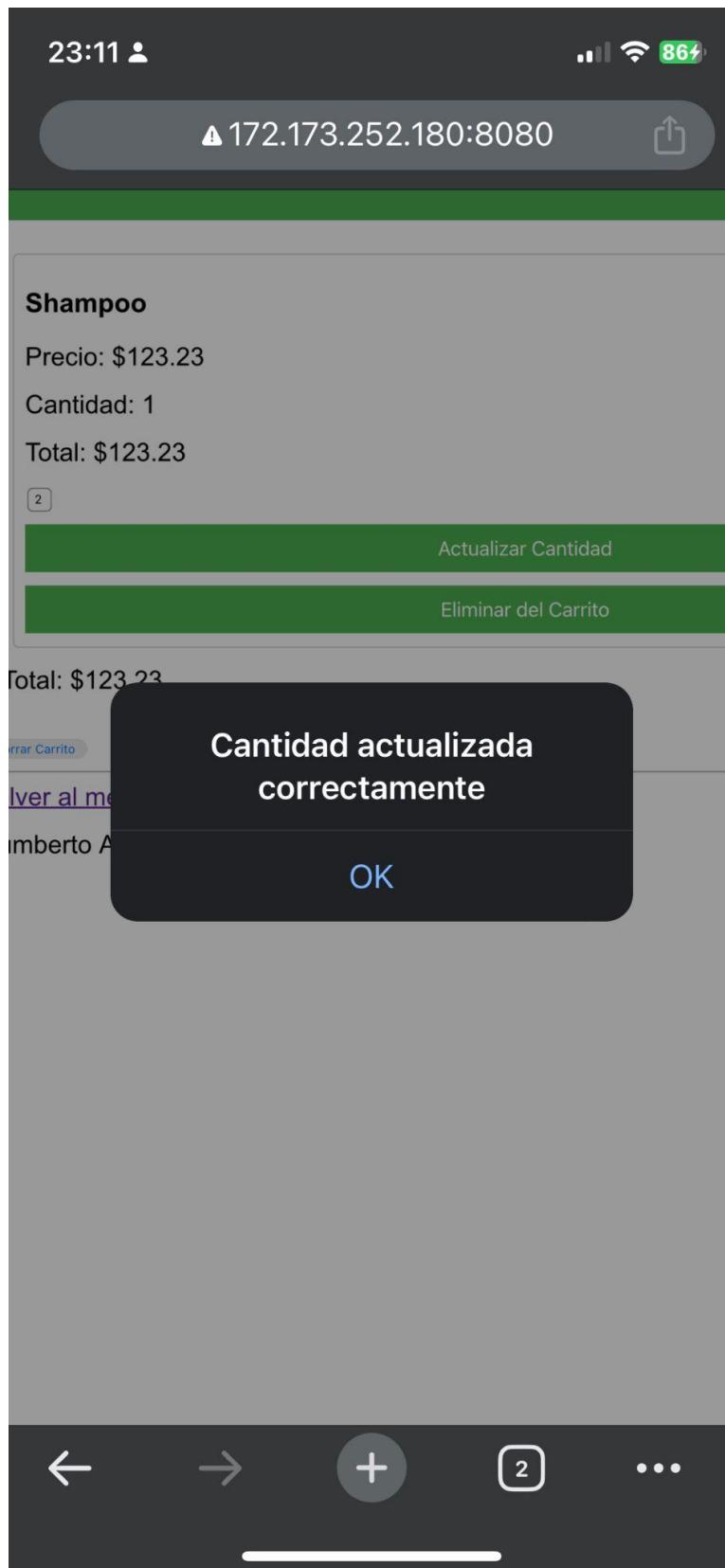
Carrito de compras

La visualización del carrito es la siguiente:



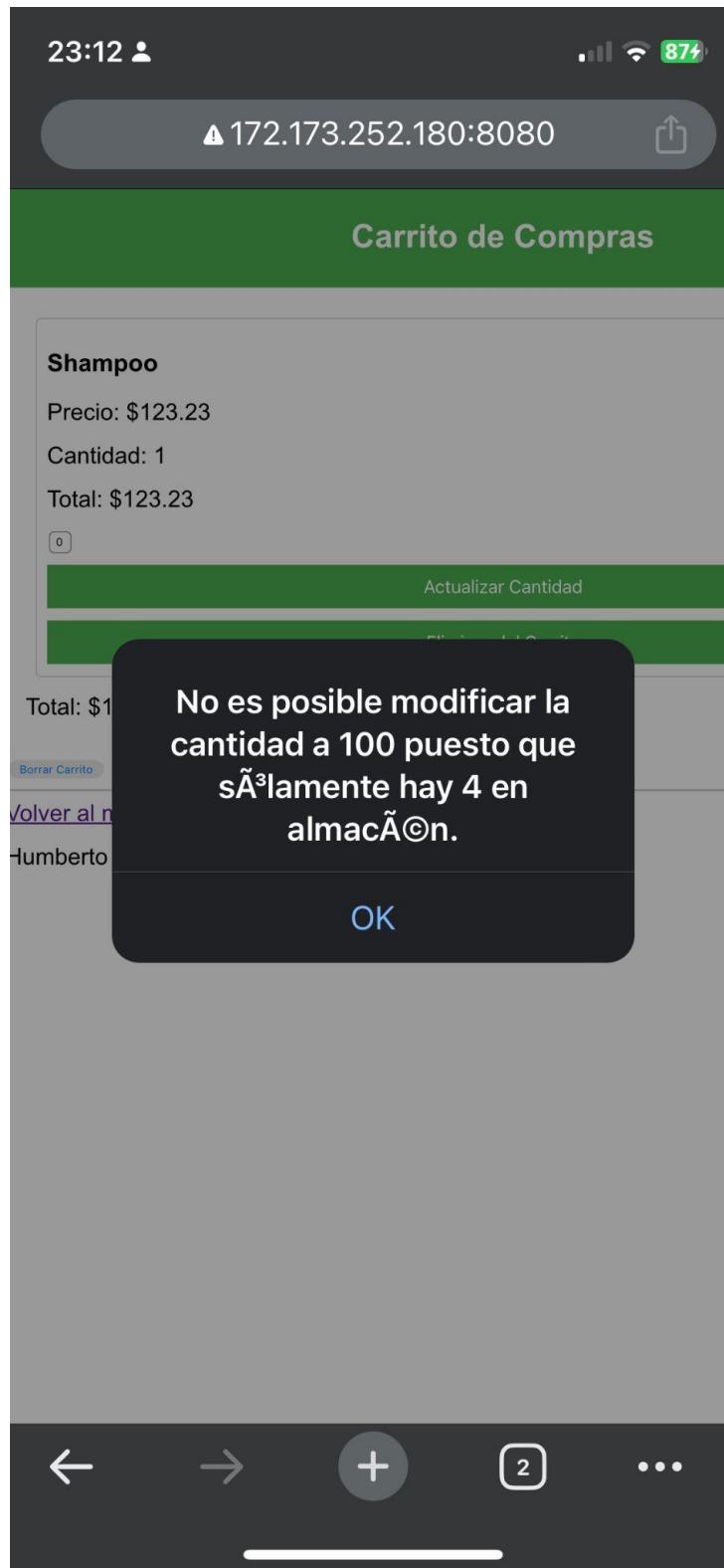
Modificar la cantidad de un artículo en el carrito exitosamente

Si intentamos aumentar la cantidad a un valor existente en almacén (2), se mostrará una confirmación con los cambios:



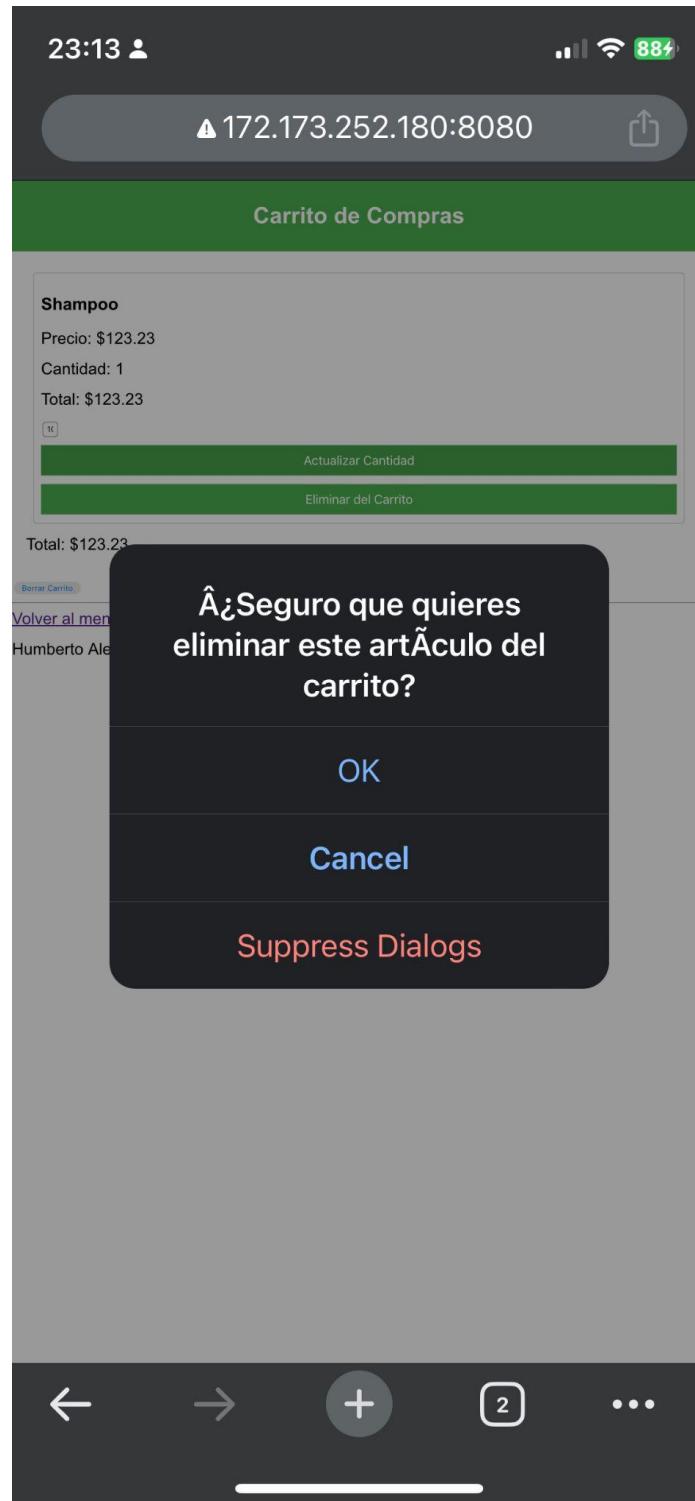
Modificar la cantidad de un artículo en el carrito fallida

Ahora, si intentamos actualizar la cantidad a 100, nos mostrará un error:

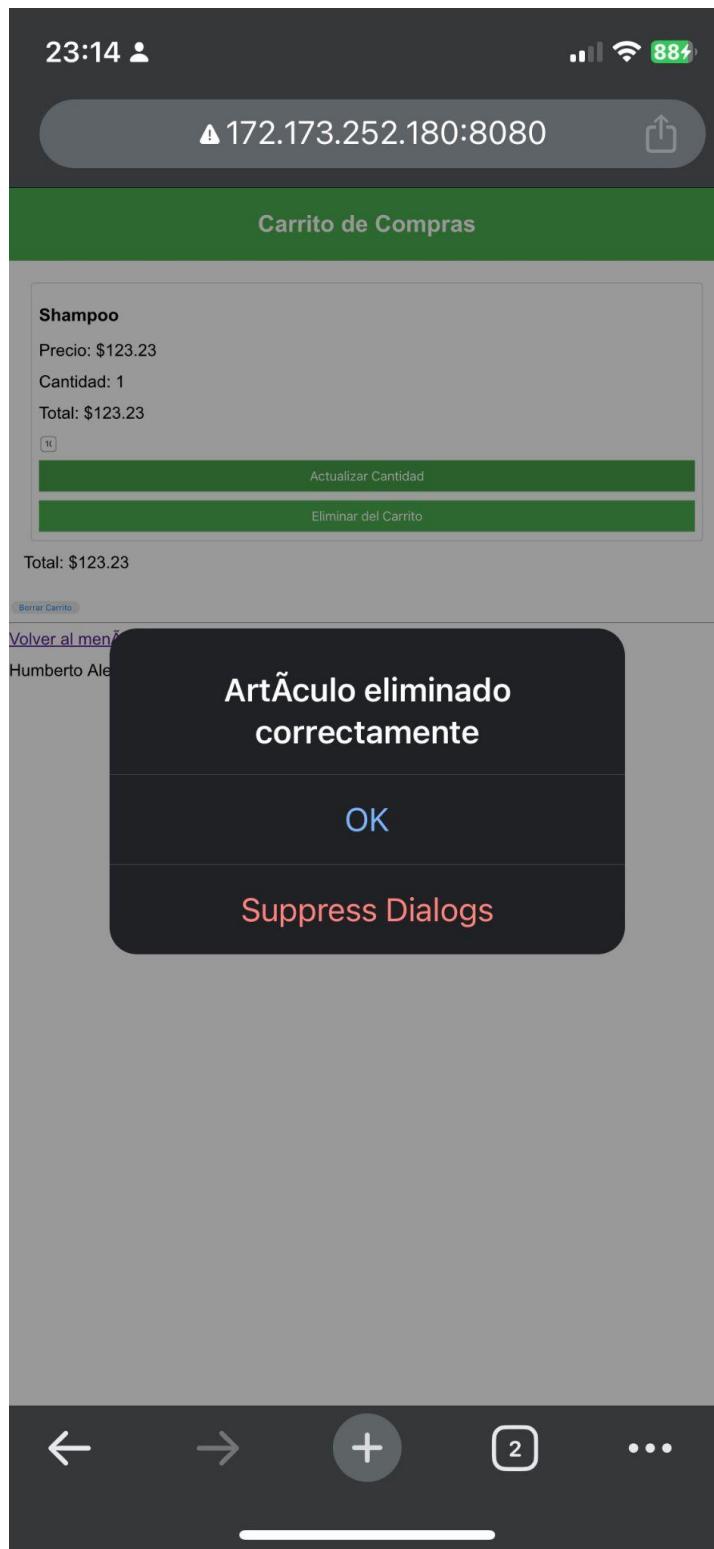


Eliminar un elemento del carrito

Al eliminar un elemento del carrito, el sistema primero nos pide confirmar si deseamos realizar la operación, de la siguiente manera:



Si hacemos click en OK, se mostrará una confirmación de la eliminación exitosa:

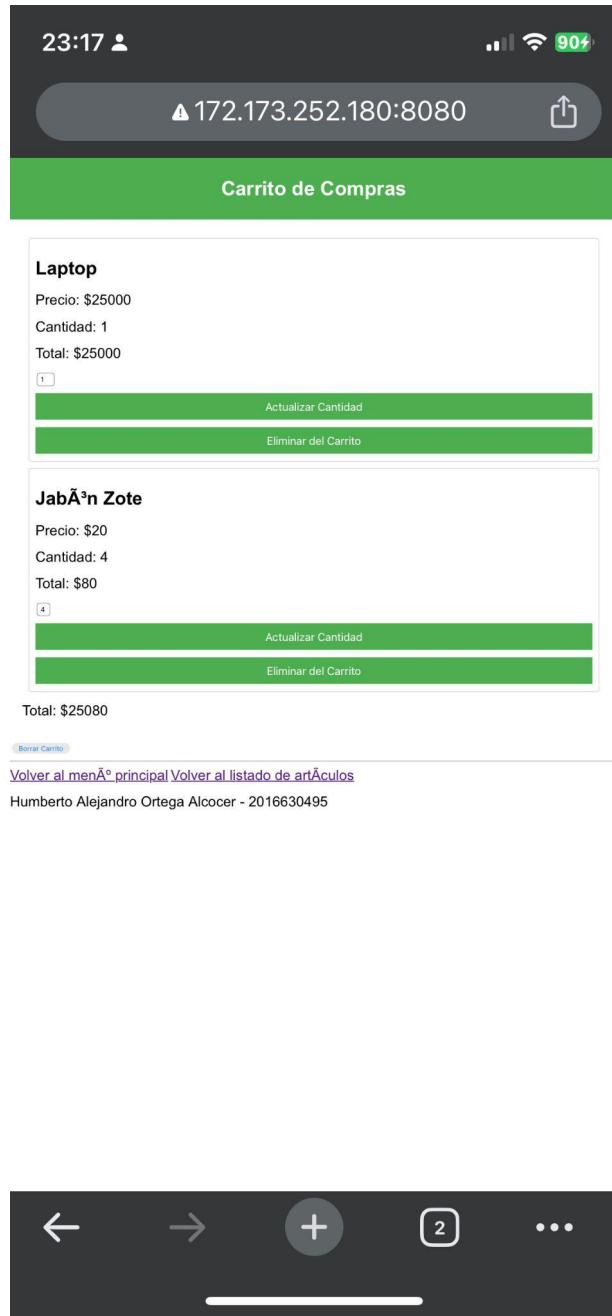


Y finalmente observaremos el carrito sin el elemento:

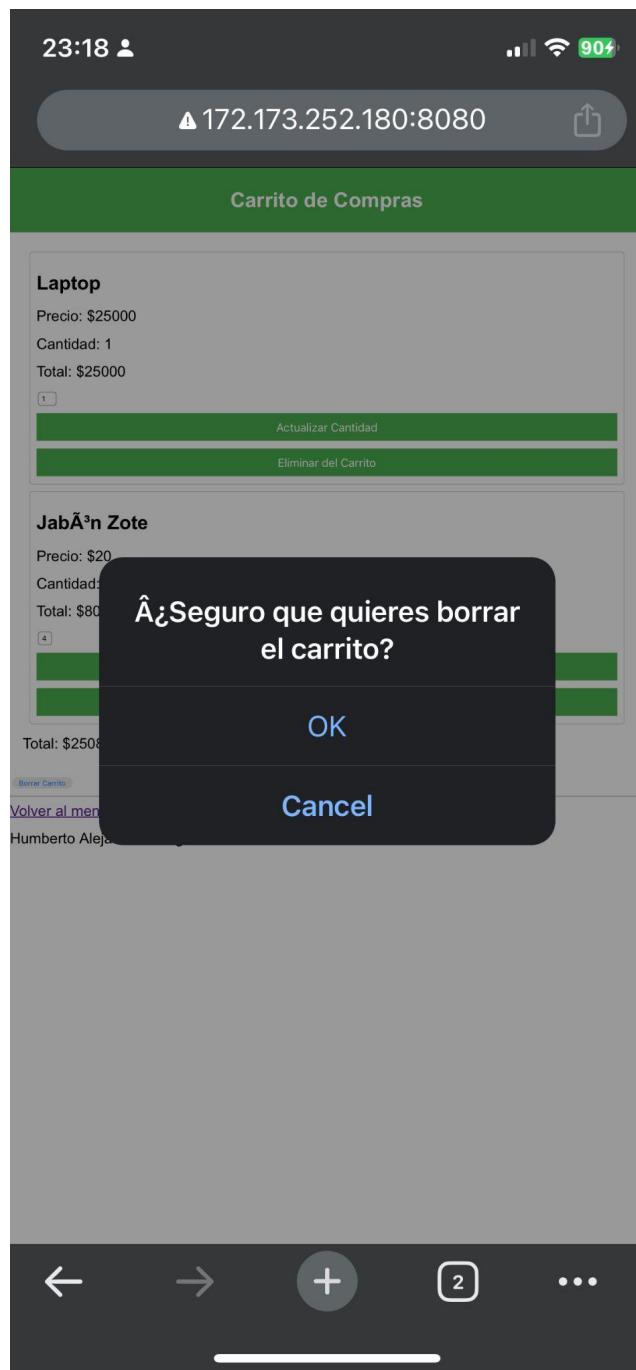


Eliminar todos los elementos del carrito

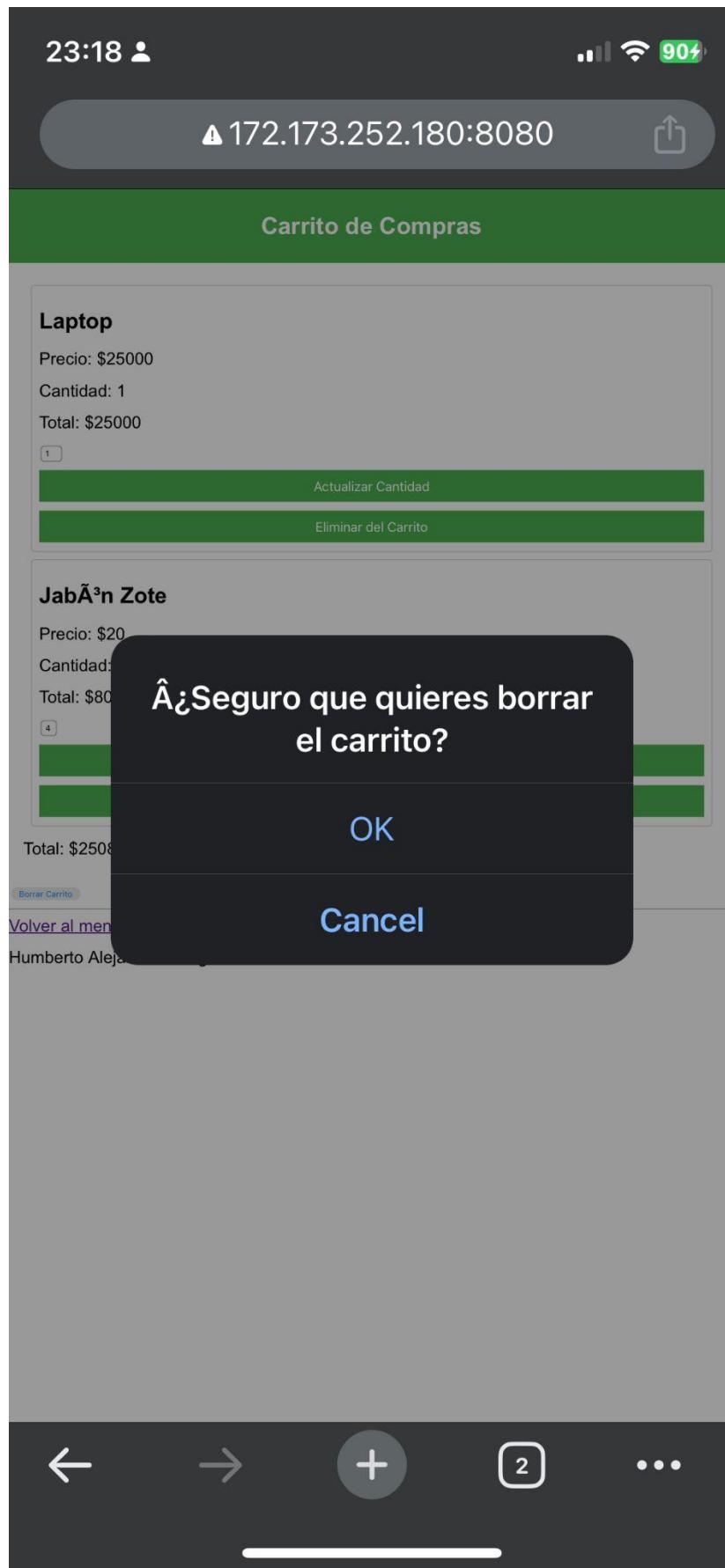
Para esta funcionalidad, hice trampa con JavaScript, lo que hago es recorrer el arreglo con cada objeto y llamar para cada uno al endpoint de eliminar elemento del carrito, esto porque me pareció lo más simple de realizar. Suponiendo que contamos con un carrito inicial como el siguiente:



Al hacer click en *Borrar Carrito* (el botón es uno chiquito abajo a la izquierda), se pedirá confirmación del usuario:



Y se informará que la acción realizada ha sido correcta:



Por lo que finalmente se mostrará el carrito vacío:



Conclusión

Realizar esta tarea fue sumamente *laborioso*, no voy a usar el término complicado porque, objetivamente, no lo fue. Simplemente el hecho de tener que programar los métodos de Java directamente en el servidor (puesto que no existen las dependencias para la arquitectura ARM (aún) y mi laptop personal es M1 de Apple), requirió una enorme paciencia porque programar por SSH implica pérdidas de conexión, VIM haciendo cosas extrañas, pérdida de avances (3 veces, mínimo, la última consideré ya no seguir adelante con la tarea) y detalles de ejecución.

Me sirvió muchísimo la parte de backend para usar Java, es un lenguaje que no uso normalmente y me gustó entender cómo puedo atar métodos a ciertos parámetros, cómo funciona la serialización y de-serialización de información, así como aprovecharme del Paradigma Orientado a Objetos para sacarle jugo al lenguaje y ahorrarme varias líneas repetidas.

En el lado de frontend, no sé qué puedo agregar, ChatGPT generó objetivamente todo el código, me parece maravilloso lo que uno puede conseguir sólamente con pedir muy específicamente lo que se necesita. Claro que fue requerido modificar grandes cantidades de código para que funcionara, pero el hecho de tener una idea base de como realizar todos los requerimientos funcionales, ayuda bastante.

Lo que más me gustó fue ver todo interactuar entre sí, mi backend consultando la base de datos (realmente algunas operaciones se realizaron sí, en transacciones, pero en transacciones separadas así que ... es lo mismo que no haber usado transacciones), mi frontend accediendo al backend, y todos los datos viajando de forma serializada a través de la red.

Estoy muy contento de que haya funcionado, fue una pesadilla que duró dos semanas, quizás esto no debería comentarse pero el estrés y ansiedad de esta tarea no ha tenido igual en todo el semestre, estoy muy tranquilo de que se logró, voy a dormir muy feliz el día de hoy.

– Humberto Alejandro Ortega Alcocer.