



Instituto Politécnico Nacional

Escuela Superior de Cómputo

Sistemas Distribuidos

Pineda Guerrero Carlos

Tarea 9

4CV13

Humberto Alejandro Ortega Alcocer (2016630495)

12 de Junio del 2023

Índice

Índice	1
1. Desarrollo.	4
1.1. Creación de infraestructura.	4
1.1.1. Creación de servidor de base de datos.	4
1.1.1.1. Nombre de instancia de base de datos.	5
1.1.1.5. Prueba de conexión	10
1.1.2. Creación de proyecto de Azure Functions.	13
1.1.2.1. Creación de espacio de almacenamiento.	14
1.1.2.2. Configuración de CORS	18
1.1.3. Creación de Máquina Virtual para frontend.	20
1.1.3.1. Prueba de conectividad.	25
1.2. Desarrollo de software.	26
1.2.1. Backend.	26
1.2.1.1. Instalación de herramientas de línea de comandos de Azure.	26
1.2.1.2. Inicialización de proyecto.	27
1.2.1.3. Definición de estructura de base de datos.	29
1.2.1.4. Creación de funciones.	30
1.2.1.4.1. CrearArticulo.	31
1.2.1.4.2. ObtenerArticulos.	35
1.2.1.4.3. AgregarCarrito.	38
1.2.1.4.4. EliminarCarrito.	42
1.2.1.4.5. EditarCarrito.	46
1.2.1.5. Ejecutar funciones localmente	52
1.2.2. Frontend.	53
1.2.2.1. Agregar Artículo	55
1.2.2.2. Rutas, encabezado y pié de página..	64
1.2.2.3. Página inicial.	67
1.2.2.4. Catálogo de artículos.	70
1.2.2.5. Carrito de compras.	77
1.2.2.6. Constantes y variables de entorno.	82
1.3. Despliegue de aplicación.	83
1.3.1. Despliegue de base de datos.	83
1.3.2. Despliegue de backend.	89
1.3.3. Despliegue de frontend.	93
1.3.3.1. Instalando NGINX	95
1.3.3.2. Configurando NGINX.	96
2. Pruebas de Funcionamiento	98
2.1. Página principal.	98
2.2. Captura de artículo.	99

2.2.1. Formulario sin datos.	99
2.2.2. Formulario con datos inválidos.	100
2.2.3. Formulario con datos válidos.	101
2.2.4. Respuesta exitosa de backend.	102
2.2.5. Estado de la base de datos.	103
2.3. Compra de artículos.	103
2.3.1. Pantalla principal.	103
2.3.2. Filtrando por texto.	104
2.3.3. Removiendo filtro.	105
2.4. Búsqueda de artículos.	106
2.4.1. Filtrando por texto.	106
2.3.3. Removiendo filtro.	107
2.5. Descripción de artículos.	108
2.5.1. Pantalla principal de artículos.	108
2.5.2. Presionando botón de “Ver más información”.	109
2.5.3. Ventana de “más información”.	110
2.6. Agregar al carrito con validación.	111
2.6.1. Validación de cantidad máxima de artículos.	111
2.6.2. Agregando al carrito correctamente.	112
2.6.3. Estado de la base de datos.	113
2.6.4. Operaciones sucediendo en la misma transacción.	114
2.7. Carrito de compra.	115
2.7.1. Botón de carrito de compra.	115
2.7.2. Página principal de carrito de compras.	116
2.7.3. Elementos mostrados por artículo en el carrito.	117
2.7.4. Total de la compra.	118
2.8. Eliminar artículo de carrito de compra.	119
2.8.1. Botón de “Eliminar artículo”.	119
2.8.2. Ventana de confirmación para eliminar artículo.	120
2.8.3. Carrito sin elemento eliminado.	121
2.8.3. Parte del código que usa transacciones.	122
2.8.4. Estado de la base de datos.	123
2.8.4.1. Antes de la operación.	123
2.8.4.1. Despues de la operación.	123
2.9. Actualizar cantidad de artículo en carrito de compra.	124
2.9.1. Campo para introducir la cantidad a modificar.	124
2.9.2. Introduciendo una cantidad inválida.	125
2.9.3. Ventana de validación incorrecta de cantidad de artículos.	126
2.9.5. Cambiar cantidad a un valor válido.	127
2.9.5. Código relevante a las transacciones.	128
2.9.5.1. En caso de que sea aumentar la cantidad de elementos en el carrito.	128

2.9.5.2. En caso de que sea disminuir la cantidad de elementos en el carrito.	129
2.9.6. Estado de la base de datos.	130
2.9.6.1. Estado de la base de datos antes de la operación.	130
2.9.6.2. Estado de la base de datos después de la operación.	130
2.10. Eliminar todos los artículos en el carrito de compra.	131
2.10.1. Botón de “eliminar carrito de compra”.	131
2.10.2. Ventana de confirmación de la eliminación del carrito de compra.	132
2.10.3. Carrito de compras después de la operación.	133
2.10.3. Código relacionado a las transacciones.	134
2.10.4. Estado de la base datos.	135
2.10.4.1. Estado de la base de datos antes de la operación.	135
2.10.4.2. Estado de la base de datos después de la operación.	135
2.11. Botón para volver a la vista de artículos.	136
2.11.1. Botón en página de carrito.	136
2.11.2. Prueba de URL de redirecciónamiento.	137
3. Conclusión	138

1. Desarrollo.

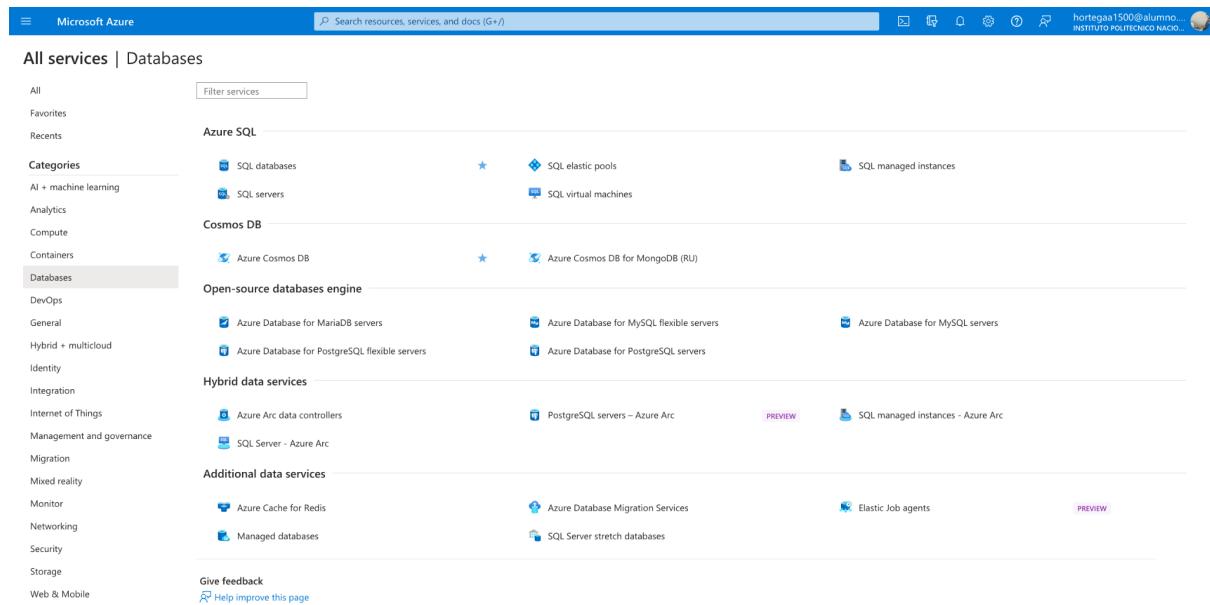
Para desarrollar esta práctica se deben realizar varios pasos puesto que tenemos que diseñar un frontend, un backend, crear y configurar la infraestructura correspondiente y desplegar la aplicación final en la misma. Lo primero que haremos será levantar la infraestructura, que consiste en crear el proyecto de funciones de Azure y configurar la cuenta de almacenamiento. Para el frontend y backend se realizará un desarrollo asistido por ChatGPT y se anexarán las conversaciones sostenidas con el mismo. Finalmente, se llevará a cabo el despliegue de la aplicación en Azure.

1.1. Creación de infraestructura.

La infraestructura requerida en esta tarea es simple. Primero crearemos la instancia de base de datos en Azure con MySQL. Luego, necesitaremos crear un proyecto de funciones de Azure, ahí es dónde se ejecutarán nuestras funciones de forma *serverless*. Lo siguiente será crear la cuenta de almacenamiento que usaremos para almacenar el código fuente de las funciones mencionadas.

1.1.1. Creación de servidor de base de datos.

Para crear un nuevo servidor lo primero será ir a la sección de todos los servicios disponibles en Azure y buscaremos por *Azure Database for MySQL flexible servers*:



The screenshot shows the Microsoft Azure portal interface. At the top, there is a search bar labeled "Search resources, services, and docs (G+)" and a user profile icon. The left sidebar has a "Categories" section with various service categories like All, Favorites, Recents, and Databases (which is currently selected). The main content area displays a grid of database-related services under the "Databases" category. These include:

- Azure SQL:** SQL databases, SQL elastic pools, SQL managed instances, SQL servers, SQL virtual machines.
- Cosmos DB:** Azure Cosmos DB, Azure Cosmos DB for MongoDB (RU).
- Open-source databases engine:** Azure Database for MariaDB servers, Azure Database for MySQL flexible servers, Azure Database for MySQL servers, Azure Database for PostgreSQL flexible servers, Azure Database for PostgreSQL servers.
- Hybrid data services:** Azure Arc data controllers, PostgreSQL servers – Azure Arc, SQL managed instances - Azure Arc, SQL Server - Azure Arc.
- Additional data services:** Azure Cache for Redis, Azure Database Migration Services, Elastic Job agents, Managed databases, SQL Server stretch databases.

At the bottom of the page, there are links for "Give feedback" and "Help improve this page".

Daremos click en *Create* seleccionando la opción *Flexible server*:

Ahora se nos presentará el formulario de creación de instancia, donde seleccionaremos nuestro grupo de recursos que hemos usado para todas las tareas, la versión de MySQL entre otras configuraciones. Lo primero es colocar el nombre pero para esto hay que realizar una aclaración.

1.1.1. Nombre de instancia de base de datos.

Al intentar de realizar el nombrado de la instancia de base de datos acorde a las especificaciones del profesor se encuentra el siguiente error pues Azure sólamente permite letras minúsculas en el nombre:

Al cambiar “T9-BD-2016630495” por “t9-bd-2016630495” se suprime el error y permite continuar con la creación de la instancia:

Basics Networking Security Tags Review + create

Create an Azure Database for MySQL flexible server. [Learn more](#)

Did you know? Did you know? new users in Azure can use MySQL - Flexible Server free for up to 750 hours using Azure free account? [Learn more](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * [Azure for Students](#) Resource group * [SistemasDistribuidosResourceGroup](#) Create new

Server details

Enter required settings for this server, including picking a location and configuring the compute and storage resources.

Server name * [t9-bd-2016630495](#) Region * [East US](#) MySQL version * [8.0](#) Workload type [For small or medium size databases](#)

Estimated costs

Compute SKU USD 6.21/month

Free upto 750 hours Standard_B1s (1 vCore) 6.21

Storage USD 2.30/month

Free upto 32 GB Storage selected 20 GiB (USD 0.12 per GiB) 20 x 0.12

Backup Retention

Backup retention is billed based on additional storage used for retaining backups. [Learn more](#)

Bandwidth

For outbound data transfer across

Server name must be at least 3 characters and at most 63 characters.
Server name must only contain lowercase letters, numbers, and hyphens.
Server name must not start or end in a hyphen.
Server name must be available.

Charge will apply if you use above the free monthly limits. Please check [your usage](#) of free services. Final charges will appear in your local currency.

[Review + create](#) [Next : Networking >](#)

Como se puede observar, se ha elegido el grupo de recursos apropiado, así como la versión de MySQL 8, la más reciente. Lo siguiente será seleccionar que queremos nuestra instancia *For development or hobby projects*, es decir, lo más barata posible, de la siguiente forma:

Basics Networking Security Tags Review + create

All services > Azure Database for MySQL flexible servers >

Flexible server

Server name * [t9-bd-2016630495](#) Region * [East US](#) MySQL version * [8.0](#) Workload type [For development or hobby projects](#)

Compute + storage

Burstable_B1s 1 vCores, 1 GiB RAM, 20 GiB storage, 360 IOPS Geo-redundancy : Disabled Configure server

Availability zone * [No preference](#)

High availability

Same zone and zone redundant high availability provide additional server resilience in the event of a failure. You can also specify high availability options in 'Compute + storage'.

Enable high availability

Authentication

Select the authentication methods you would like to support for accessing this MySQL server. MySQL password

Estimated costs

Compute SKU USD 6.21/month

Free upto 750 hours Standard_B1s (1 vCore) 6.21

Storage USD 2.30/month

Free upto 32 GB Storage selected 20 GiB (USD 0.12 per GiB) 20 x 0.12

Backup Retention

Backup retention is billed based on additional storage used for retaining backups. [Learn more](#)

Bandwidth

For outbound data transfer across services in different regions will incur additional charges. Any inbound data transfer is free. [Learn more](#)

Estimated total USD 8.50/month

Charges will apply if you use above the free monthly limits. Please check [your usage](#) of free services. Final charges will appear in your local currency.

[Review + create](#) [Next : Networking >](#)

Lo siguiente será configurar la autenticación, para la cual se usarán estos valores:

- **Usuario:** humberto
- **Contraseña:** Tacos12345

Estos valores se eligieron porque son fáciles de recordar y son únicamente para fines exclusivos de esta tarea (toda la infraestructura asociada será eliminada antes de publicar este documento):

The screenshot shows the 'Create a MySQL flexible server' wizard in Microsoft Azure. The current step is 'Set authentication'. The configuration includes an Admin username ('humberto') and a password ('Tacos12345'). On the right, there is a summary of estimated costs:

Category	Cost
Compute Sku	USD 6.21/month
Storage	USD 2.30/month
Total	USD 8.50/month

Ahora iremos a *Next: Networking* dónde realizaremos la configuración de red de la instancia permitiendo el tráfico interno de Azure en la misma (por el backend) y también desde nuestra IP pública local para realizar pruebas remotas:

The screenshot shows the 'Create a MySQL flexible server' wizard in Microsoft Azure. The current step is 'Networking'. The configuration includes 'Public access (allowed IP addresses)' and a note about Firewall rules. On the right, there is a summary of estimated costs:

Category	Cost
Compute Sku	USD 6.21/month
Storage	USD 2.30/month
Total	USD 8.50/month

Ahora veremos un resumen de las opciones seleccionadas:

The screenshot shows the 'Review + create' step for a MySQL flexible server. It includes sections for Estimated costs, Product details, Basics (Change), Networking (Change), and Security (Change). The estimated cost is USD 8.50/month.

Category	Value
Compute Sku	USD 6.21/month
Storage	USD 2.30/month
Backup Retention	7 days(s)
Bandwidth	For outbound data transfer across services in different regions will incur additional charges. Any inbound data transfer is free.
Estimated total	USD 8.50/month

Basics (Change)

Setting	Value
Description	Azure for Students
Resource group	SistemasDistribuidosResourceGroup
Server name	t9-bd-2016630495
Server admin login name	humberto
Location	East US
Availability zone	No preference
High availability	Not Enabled
MySQL version	8.0
Compute + storage	Burstable, B1s, 1 vCores, 1 GiB RAM, 20 storage, 360 IOPS
Backup retention period (in days)	7 day(s)
Storage Auto-growth	Enabled
Geo-redundancy	Not Enabled

Networking (Change)

Setting	Value
Connectivity method	Public access (allowed IP addresses)
Allow public access from any Azure service within Azure to this server	Yes
Firewall rules	1
SSL/TLS	SSL is enforced and TLS version is 1.2. This can be changed after server is created. Learn more

Security (Change)

Buttons: Create, < Previous, Download a template for automation

Iniciaremos entonces el despliegue de nuestra base de datos lo cual tomará, según las propias estimaciones de Azure presentes en la pantalla anterior, alrededor de 5 minutos:

The screenshot shows the 'Overview' page for the MySQLFlexibleServer. The status is 'Deployment is in progress'. Deployment details show the resource is accepted.

Resource	Type	Status	Operation details
t9-bd-2016630495	Microsoft.DBforMySQL/flexible...	Accepted	Operation details

Get started with MySQL Flexible Server
Free Microsoft tutorials
Create PHP + MySQL App
Work with an expert
Azure experts are service provider partners who can help manage your assets on Azure and be your first line of support.
Find an Azure expert >

Una vez creada la base de datos se crearán las reglas de cortafuegos (*firewall*) con las que se permitirá el acceso desde la red interna de Azure y mi IP pública por lo que se mostrará la siguiente pantalla:

The screenshot shows the Microsoft Azure portal. The URL is [https://portal.azure.com/#blade/Microsoft_Azure_DatabaseForMySQL_FlexibleServer/OverviewBlade/resourceId%3D8be5e36201eb49558f277f214fb7c785](#). The page title is "MySQLFlexibleServer_8be5e36201eb49558f277f214fb7c785 | Overview". The main content area displays a green checkmark icon and the message "Your deployment is complete". It shows deployment details: Deployment name: MySQLFlexibleServer_8be5e36201eb49558f277f214fb7c785, Start time: 6/9/2023, 9:02:35 PM, Subscription: Azure for Students, Correlation ID: cedeb261-7bfe-464a-b9d2-d3204ab2daea. Below this, there's a table titled "Deployment details" listing three resources: "firewallRules-642e149dc9c04ccaa02048324" (Type: Microsoft.Resources/deployments, Status: OK), "firewallRules-642e149dc9c04ccaa02048324" (Type: Microsoft.Resources/deployments, Status: OK), and "19-bd-2016630495" (Type: Microsoft.DBforMySQL/flexibleServers, Status: Created). To the right, there are sections for "Cost Management", "Get started with MySQL Flexible Server", "Free Microsoft tutorials", "Work with an expert", and "Give feedback". A sidebar on the left shows "All services > MySQLFlexibleServer_8be5e36201eb49558f277f214fb7c785 | Overview".

Y podremos verificar que la infraestructura se ha creado correctamente visitando la pantalla principal de nuestro servidor MySQL:

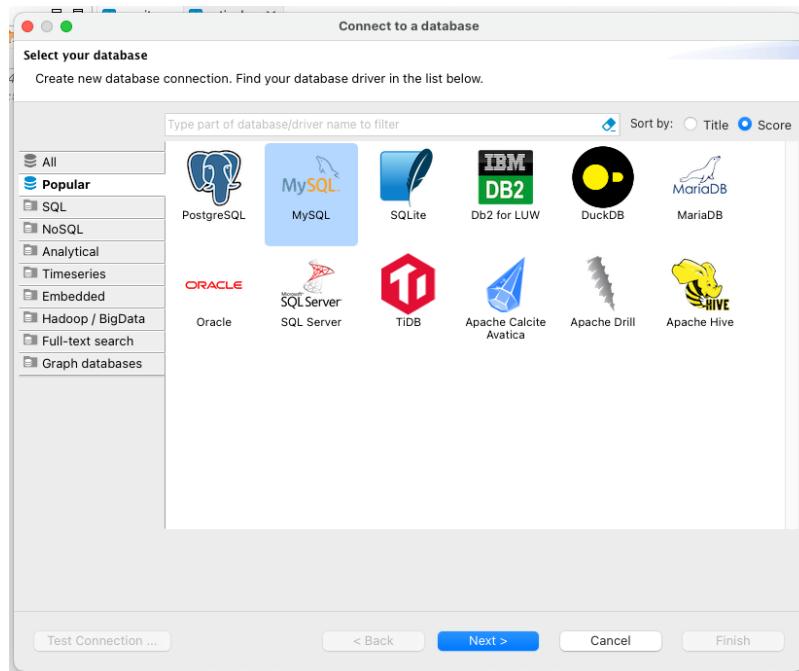
The screenshot shows the Microsoft Azure portal. The URL is [https://portal.azure.com/#blade/Microsoft_Azure_DatabaseForMySQL_FlexibleServer/OverviewBlade/resourceId%3Dt9-bd-2016630495](#). The page title is "t9-bd-2016630495 | Overview". The main content area displays a green checkmark icon and the message "Azure Database for MySQL – Live Webinar series: Learn about the latest updates (with demos) and interact directly with product group on the 2nd Wednesday of every month! [Subscribe to our YouTube channel](#) [Today]". It shows essential information: Subscription: Azure for Students, Subscription ID: a5d9d675-3aef-4e2b-95b5-02150a67b8d2, Resource group: SistemasDistribuidosResourceGroup, Status: Available, Location: East US. Below this, there are sections for "Getting started", "Start your project", "Configure your database", "Get ready for production use", and "Samples". A sidebar on the left shows "All services > t9-bd-2016630495 | Overview".

1.1.5. Prueba de conexión

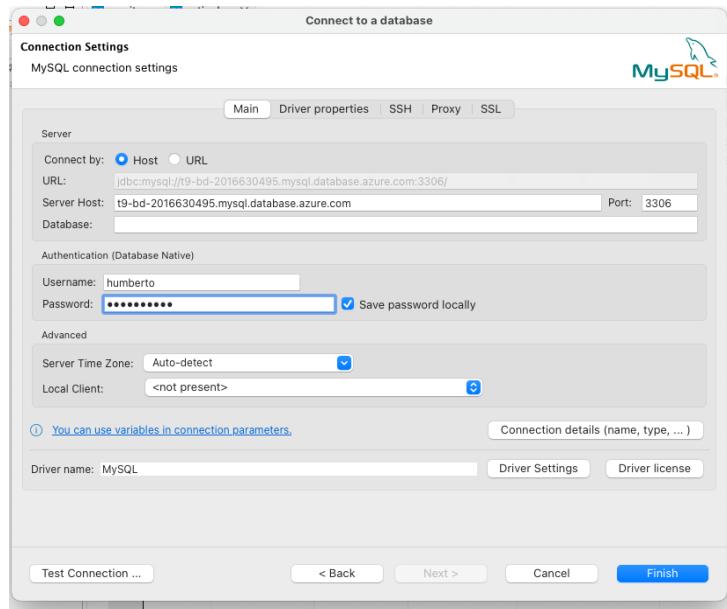
Para probar que la instancia fue configurada correctamente, usaremos nuestro cliente de SQL local *DBeaver* para iniciar la conexión al servidor de base de datos usando las credenciales definidas. Para esto, primero visualizaremos la información de conexión en el apartado *Connect* de nuestra instancia:

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with various service icons like Compute + storage, Networking, Databases, and Power Platform. The main area is titled 't9-bd-2016630495 | Connect'. It displays a 'Pre-requisites check' section with three green checkmarks: 'Firewall rules are enabled on this server.', 'SSL is enforced and TLS version is 1.2.', and 'Server is in Ready state.' Below this is a 'Connection details' section where the connection string is shown: 'hostname=t9-bd-2016630495.mysql.database.azure.com', 'username=humberto', 'password=[your password]', and 'ssl-mode=require'. There are also sections for 'View all databases', 'Connect from browser or locally' (which includes MySQL Workbench, Import and export data, and Connect from your app), and a 'Connect from your app' section.

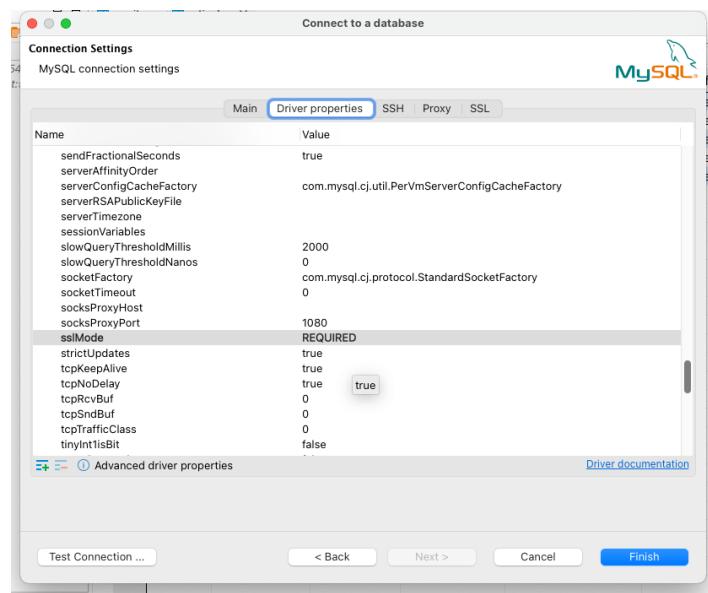
En nuestro cliente de bases de datos, daremos click en crear nueva conexión y seleccionaremos MySQL:



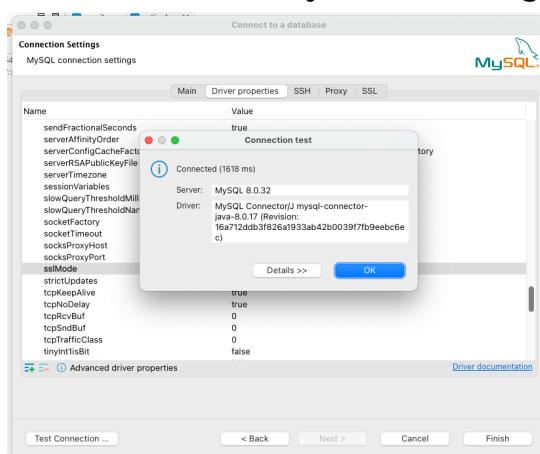
Colocaremos los datos disponibles en Azure:



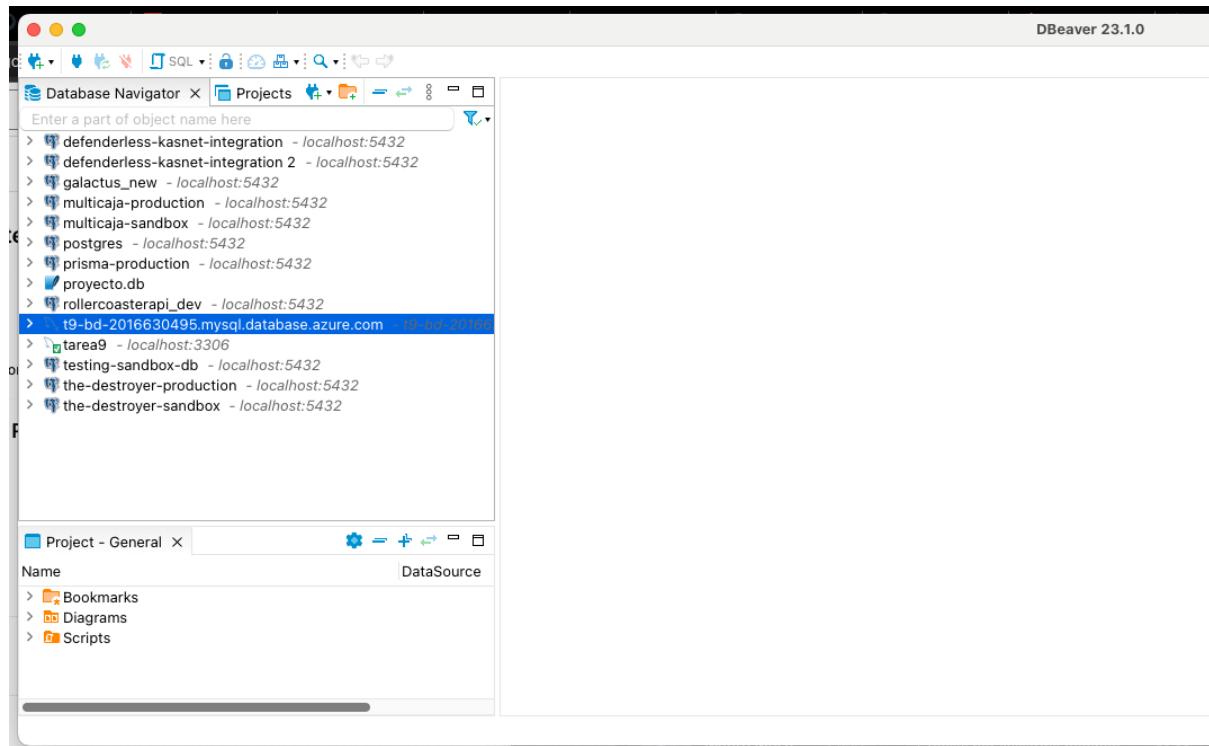
Activaremos el modo SSL requerido para el driver:



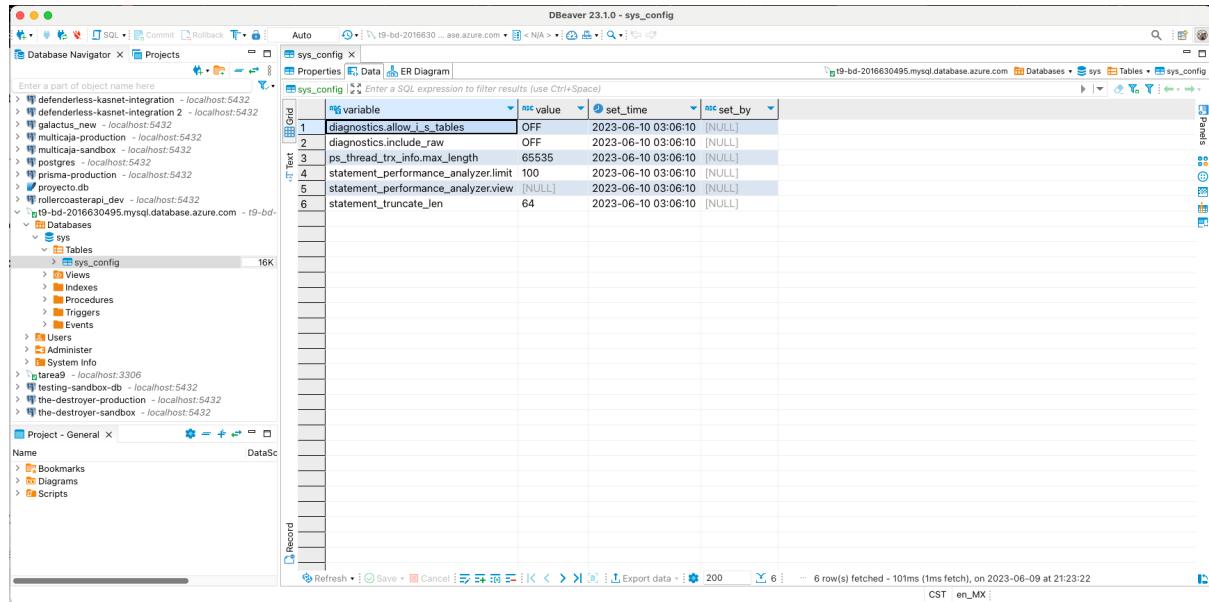
Haremos click en el botón *Test connection* y veremos el siguiente resultado:



Haremos click en *Finish* y observaremos que se añade al panel izquierdo:



Finalmente listaremos las tablas para verificar que funcione la conexión:



1.1.2. Creación de proyecto de Azure Functions.

Para crear el proyecto de funciones que necesitamos para nuestro backend, lo primero será ir a la sección de *Function App* de Azure:

The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes 'Microsoft Azure', a search bar, and user information ('hortega1500@alumno.ipn.mx INSTITUTO POLITÉCNICO NACIONAL'). Below the navigation is a toolbar with icons for Home, Refresh, Export to CSV, Open query, Assign tags, Start, Restart, Stop, and Delete. A filter bar at the top allows filtering by 'Subscription equals all', 'Resource group equals all', and 'Location equals all'. The main content area displays a message: 'No function apps match your filters' with a lightning bolt icon. It includes a link to 'Create Function App' and a 'Clear filters' button. At the bottom right is a 'Give feedback' link.

Ahora introduciremos los datos que usaremos en el formulario, para esto usaremos el nombre en el formato especificado por el profesor, el grupo de recursos que hemos usado a lo largo del semestre, el lenguaje Node.js v18 en la región del este de Estados Unidos y ejecutándose en un servidor Linux:

The screenshot shows the 'Create Function App' wizard in the 'Basics' step. The top navigation bar includes 'Microsoft Azure', a search bar, and user information ('hortega1500@alumno.ipn.mx INSTITUTO POLITÉCNICO NACIONAL'). The page title is 'Create Function App'. The 'Basics' tab is selected. The 'Project Details' section shows 'Subscription' set to 'Azure for Students' and 'Resource Group' set to 'SistemasDistribuidosResourceGroup'. The 'Instance Details' section shows 'Function App name' as 'T9-AF-2016630495'. Under 'Do you want to deploy code or container image?', the 'Code' option is selected. The 'Runtime stack' is set to 'Node.js' and 'Version' to '18 LTS'. The 'Region' is set to 'East US'. The 'Operating system' section shows 'Operating System' set to 'Linux'. At the bottom are buttons for 'Review + create', '< Previous', and 'Next : Storage >'.

Seleccionaremos la opción de cobro en función del consumo (*Serverless*):

Create new

Instance Details

Function App name * T9-AF-2016630495 .azurewebsites.net

Do you want to deploy code or container image? Code Container Image

Runtime stack * Node.js

Version * 18 LTS

Region * East US

Operating system

The Operating System has been recommended for you based on your selection of runtime stack.

Operating System * Linux Windows

Hosting

The plan you choose dictates how your app scales, what features are enabled, and how it is priced. [Learn more](#)

Hosting options and plans * Consumption (Serverless) Optimized for serverless and event-driven workloads. Functions Premium Event based scaling and network isolation, ideal for workloads running continuously. App service plan Fully isolated and dedicated environment suitable for workloads that need large SKUs or need to co-locate Web Apps and Functions.

[Review + create](#) [< Previous](#) [Next : Storage >](#)

1.1.2.1. Creación de espacio de almacenamiento.

Lo siguiente será crear una cuenta de almacenamiento para guardar ahí el código correspondiente a nuestra aplicación de funciones, usaremos el nombre: "t9al2016630495" el cual se parece al esquema de nombres propuesto por el profesor:

Basics Storage Networking Monitoring Deployment Tags Review + create

Storage

When creating a function app, you must create or link to a general-purpose Azure Storage account that supports Blobs, Queue, and Table storage.

Storage account * [\(New\) sistemasdistribuido9df1](#) Create new

New Storage Account

Name * t9al2016630495

OK Cancel

[Review + create](#) [< Previous](#) [Next : Networking >](#)

Luego de dar click en *Ok* veremos cómo el nuevo nombre aparece en la lista desplegable para la cuenta de almacenamiento y la seleccionaremos:

The screenshot shows the 'Storage' tab of the 'Create Function App' wizard. It displays a dropdown menu where '(New) t9al2016630495' is selected. Below the dropdown is a 'Create new' button. At the bottom of the page are navigation buttons: 'Review + create' (highlighted in blue), '< Previous', and 'Next : Networking >'.

En la sección de configuración de red, lo único es dejar en *On* la opción para permitir el acceso público a nuestras funciones:

The screenshot shows the 'Networking' tab of the 'Create Function App' wizard. It features a 'Enable public access' section with a radio button set to 'On'. A note below states: '⚠ Network injection is only available in Functions Premium and Basic, Standard, Premium, Premium V2, Premium V3 Dedicated App Service plans.' Below this is another section for 'Enable network injection' with a radio button set to 'Off'. At the bottom of the page are navigation buttons: 'Review + create' (highlighted in blue), '< Previous', and 'Next : Monitoring >'.

Lo siguiente será habilitada esta opción para tener métricas más completas en caso de que se requiera diagnosticar cualquier anomalía con la aplicación:

Basics Storage Networking Monitoring Deployment Tags Review + create

Azure Monitor application insights is an Application Performance Management (APM) service for developers and DevOps professionals. Enable it below to automatically monitor your application. It will detect performance anomalies, and includes powerful analytics tools to help you diagnose issues and to understand what users actually do with your app. Your bill is based on amount of data used by Application Insights and your data retention settings. [Learn more](#)

App Insights pricing

Application Insights

Enable Application Insights *

No Yes

Application Insights *

(New) T9-AF-2016630495 (East US)

Create new

Region

East US

Review + create < Previous Next : Deployment >

Finalmente observaremos el resumen de nuestra aplicación de funciones de Azure:

Basics Storage Networking Monitoring Deployment Tags Review + create

Summary

Function App by Microsoft

Details

Subscription: a5d9d675-3ae1-4e2b-95b5-02150d67b8d2
Resource Group: SistemasDistribuidosResourceGroup
Name: T9-AF-2016630495
Runtime stack: Node.js 18 LTS

Hosting

Storage (New)

Storage account: t9ai2016630495

Plan (New)

Hosting options and plans

Name: ASP-SistemasDistribuidosResourceGro-83ae
Operating System: Linux
Region: East US
SKU: Dynamic

Monitoring (New)

Application Insights: Enabled
Name: T9-AF-2016630495
Region: East US

Deployment

Continuous deployment: Not enabled / Set up after app creation

Create < Previous Next > Download a template for automation

Daremos click en *Create* y esperaremos al despliegue:

The screenshot shows the Microsoft Azure portal interface for a Web Function App named "Microsoft.Web-FunctionApp-Portal-f1139619-a7fc". The "Deployment" tab is selected. A message at the top states "... Deployment is in progress". Below this, deployment details are listed in a table:

Resource	Type	Status	Operation details
ASP-SistemasDistribuidosResourceGro-83a	Microsoft.Web/serverfarms	OK	Operation details
t9al2016630495	Microsoft.Storage/storageAccounts	Accepted	Operation details
newWorkspaceTemplate	Microsoft.Resources/deployments	Created	Operation details

On the right side of the screen, there are promotional cards for Microsoft Defender for Cloud, Free Microsoft tutorials, and Work with an expert.

Una vez finalizado observaremos la siguiente pantalla:

The screenshot shows the Microsoft Azure portal interface for the same Web Function App. The "Deployment" tab is selected. A message at the top states "Your deployment is complete". Below this, deployment details are listed in a table:

Resource	Type	Status	Operation details
T9-AF-2016630495	Microsoft.Web/sites	OK	Operation details
T9-AF-2016630495	microsoft.insights/components	OK	Operation details
T9-AF-2016630495	microsoft.insights/components	OK	Operation details
t9al2016630495	Microsoft.Storage/storageAccounts	OK	Operation details
ASP-SistemasDistribuidosResourceGro-83a	Microsoft.Web/serverfarms	OK	Operation details
t9al2016630495	Microsoft.Storage/storageAccounts	OK	Operation details
newWorkspaceTemplate	Microsoft.Resources/deployments	OK	Operation details

On the right side of the screen, there are promotional cards for Cost Management, Microsoft Defender for Cloud, Free Microsoft tutorials, and Work with an expert.

Ahora si visitamos la página de nuestra aplicación de Azure observaremos la siguiente información:

The screenshot shows the Microsoft Azure portal's Overview page for a Function App named "T9-AF-2016630495". The left sidebar contains navigation links for Home, Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), Functions, App files, Proxies, Deployment, Deployment slots, Deployment Center, Settings, Configuration, Authentication, Application Insights, Identity, Backups, Custom domain, Certificates, Networking, Scale up (App Service plan), Scale out, Locks, and Announcements.

Essentials section details:

- Resource group: SistemasDistribuidosResourceGroup
- Status: Running
- Location: East US
- Subscription ID: a5d9d675-3ae1-4e2b-95b5-02150d67b8d2
- Tags: Click here to add tags
- URL: https://t9-af-2016630495.azurewebsites.net
- Operating System: Linux
- App Service Plan: ASP-SistemasDistribuidosResourceGro-83ae (Y1.0)
- Properties: See More
- Runtime version: 4.21.3.3

Metrics section:

- Memory working set**: Shows values from 0B to 100B over UTC-06:00.
- Function Execution Count**: Shows values from 0 to 100 over UTC-06:00.
- MB Milliseconds**: Shows values from 0 to 100 over UTC-06:00.

1.1.2.2. Configuración de CORS

Una de las configuraciones relevantes a realizar es la de CORS a modo de que nuestros clientes web puedan interactuar correctamente con nuestra API basada en funciones sin mayor problema. Para esto iremos al módulo de CORS en el menú lateral de la función:

The screenshot shows the Microsoft Azure portal's CORS configuration page for the Function App "T9-AF-2016630495". The left sidebar includes links for Certificates, Networking, Scale up (App Service plan), Scale out, Locks, App Service plan, Quotas, Change App Service plan, API, API Management, CORS (selected), Monitoring, Alerts, Metrics, Advisor recommendations, Health check, Logs, Diagnostic settings, App Service logs, Log stream, Automation, Tasks (preview), Export template, Support + troubleshooting, Resource health, and New Support Request.

CORS configuration:

- Request Credentials: Enable Access-Control-Allow-Credentials
- Allowed Origins: https://portal.azure.com

Dado que queremos permitir el tráfico de cualquier origen (por simpleza y únicamente por fines prácticos de la tarea actual), eliminaremos el registro para <https://portal.azure.com>. Finalmente, añadiremos un '*' con lo cual estaremos permitiendo el tráfico de cualquier origen:

The screenshot shows the Microsoft Azure portal interface. The left sidebar navigation bar includes 'Certificates', 'Networking', 'Scale up (App Service plan)', 'Locks', 'App Service plan' (selected), 'Quotas', 'Change App Service plan', 'API', 'API Management' (selected), 'CORS' (selected), 'Monitoring' (with 'Logs' and 'Diagnostic settings' sub-options), 'Automation' (with 'Tasks (preview)' and 'Export template' sub-options), 'Support + troubleshooting' (with 'Resource health' and 'New Support Request' sub-options). The main content area is titled 'T9-AF-2016630495 | CORS'. It contains sections for 'Request Credentials' (checkbox for 'Enable Access-Control-Allow-Credentials') and 'Allowed Origins' (text input field containing '*'). A success message at the top right says 'Updated API CORS Settings successfully'. The top navigation bar shows 'Search resources, services, and docs (G+)', user information 'hortega15006@alumno.ipn.mx', and other account details.

Haremos click en Save y ¡listo!

1.1.3. Creación de Máquina Virtual para frontend.

Para el despliegue del frontend se decidió por usar una máquina virtual ejecutando un software nginx (servidor web similar a Apache) que mostrará el *build* minificado de la aplicación en React.

La máquina virtual la crearemos yendo a la sección de máquinas virtuales en Azure:

The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes 'Microsoft Azure', a search bar, and user information. Below it, the 'Virtual machines' section is displayed under 'Instituto Politécnico Nacinal (correo.ipn.mx)'. The main content area shows a message: 'No virtual machines to display' with a sub-instruction: 'Create a virtual machine that runs Linux or Windows. Select an image from the marketplace or use your own customized image.' A prominent blue 'Create' button is visible. The bottom right corner features a 'Give feedback' link.

Haremos click en *Create* y comenzaremos a llenar el formulario:

The screenshot shows the 'Create a virtual machine' wizard in the 'Basics' step. The top navigation bar includes 'Microsoft Azure', a search bar, and user information. The page title is 'Create a virtual machine'. The 'Basics' tab is selected. The form fields include:
- Subscription: Azure for Students
- Resource group: SistemasDistribuidosResourceGroup
- Virtual machine name: T9-FE-2016630495
- Region: (US) East US
- Availability options: No infrastructure redundancy required
- Security type: Standard
- Image: Ubuntu Server 20.04 LTS - x64 Gen2 (selected)
- VM architecture: x64 (selected)
At the bottom, there are buttons for 'Review + create' and 'Next : Disks >'. The bottom right corner features a 'Give feedback' link.

Usaremos un nombre parecido a la nomenclatura usada por el profesor, Ubuntu 20.04 y arquitectura x64.

Usaremos 2 vCPUs y 4GiB de RAM, así como la llave que he estado usando durante todo el semestre y dejaremos abiertos los puertos 22 (SSH), 80 (HTTP) y 443 (HTTPS) para poder acceder a nuestra aplicación web.

Microsoft Azure

Search resources, services, and docs (G+)

hortegaa1500@alumno... INSTITUTO POLITÉCNICO NACIONAL

Create a virtual machine

Size * Standard_B2s - 2 vcpus, 4 GiB memory (\$30.37/month)

Administrator account Authentication type SSH public key

Username * humbertowoody

SSH public key source Use existing key stored in Azure

Stored Keys humbertowoody-azure

Inbound port rules Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

Public inbound ports * Allow selected ports

Select inbound ports * HTTP (80), HTTPS (443), SSH (22)

All traffic from the internet will be blocked by default. You will be able to change inbound port rules in the VM > Networking page.

Review + create < Previous Next : Disks > Give feedback

Usaremos un HDD para bajar costos:

Microsoft Azure

Search resources, services, and docs (G+)

hortegaa1500@alumno... INSTITUTO POLITÉCNICO NACIONAL

Create a virtual machine

Basics Disks Networking Management Monitoring Advanced Tags Review + create

Azure VMs have one operating system disk and a temporary disk for short-term storage. You can attach additional data disks. The size of the VM determines the type of storage you can use and the number of data disks allowed. [Learn more](#)

VM disk encryption Azure disk storage encryption automatically encrypts your data stored on Azure managed disks (OS and data disks) at rest by default when persisting it to the cloud.

Encryption at host Encryption at host is not registered for the selected subscription. [Learn more about enabling this feature](#)

OS disk OS disk type * Standard HDD (locally-redundant storage)

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.

Delete with VM

Key management Platform-managed key

Enable Ultra Disk compatibility Ultra disk is supported in Availability Zone(s) 1,2,3 for the selected VM size Standard_B2s.

Data disks for T9-FE-2016630495 You can add and configure additional data disks for your virtual machine or attach existing disks. This VM also comes with a temporary disk.

LUN	Name	Size (GiB)	Disk type	Host caching	Delete with VM
					<input type="checkbox"/>

Create and attach a new disk Attach an existing disk

Review + create < Previous Next : Networking > Give feedback

En *Networking* debemos asegurarnos de que tengamos una IP pública nueva seleccionada, así como los puertos que solicitamos en el paso anterior:

The screenshot shows the 'Networking' tab of the Azure VM creation wizard. Key configurations include:

- Virtual network:** (new) T9-FE-2016630495-vnet
- Subnet:** (new) default (10.0.0.0/24)
- Public IP:** (new) T9-FE-2016630495-ip
- NIC network security group:** Basic (selected)
- Public inbound ports:** Allow selected ports (selected), specifically HTTP (80), HTTPS (443), and SSH (22).
- Warning:** A note 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."
- Buttons:** Review + create, < Previous, Next : Management >, Give feedback.

Haremos click en *Review + create* y observaremos el resumen de la máquina virtual que vamos a crear:

The screenshot shows the 'Review + create' page with the following details:

- Validation:** Validation passed
- Price:** 0.0416 USD/hr (Pricing for other VM sizes)
- TERMS:** By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with the offering(s), with the same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. See the [Azure Marketplace Terms](#) for additional details.
- Basics:**
 - Subscription: Azure for Students
 - Resource group: SistemasDistribuidosResourceGroup
 - Virtual machine name: T9-FE-2016630495
 - Region: East US
 - Availability options: No infrastructure redundancy required
 - Security type: Standard
 - Image: Ubuntu Server 20.04 LTS - Gen2
 - VM architecture: x64
 - Size: Standard B2s (2 vcpus, 4 GB memory)
 - Authentication type: SSH public key
 - Username: humbertoaloccer
 - Key pair name: humbertoaloccer-azure
 - Public inbound ports: SSH, HTTP, HTTPS
 - Azure Spot: No
- Disks:**
 - OS disk type: Standard HDD LRS
 - Use managed disks: Yes
 - Delete OS disk with VM: Enabled
 - Ephemeral OS disk: No
- Networking:**
 - Virtual network: (new) T9-FE-2016630495-vnet
 - Subnet: (new) default (10.0.0.0/24)
 - Public IP: (new) T9-FE-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

At the bottom, there are buttons for Create, < Previous, Next >, and Download a template for automation.

Management

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

Haremos click en Create y nos mostrará esta pantalla Azure dónde podremos observar el proceso de nuestro despliegue:

Deployment is in progress

Resource	Type	Status	Operation details
T9-FE-2016630495	Microsoft.Compute/virtualMachines	Created	Operation details
T9-FE-2016630495-152	Microsoft.Network/networkInterfaces	Created	Operation details
T9-FE-2016630495-vnet	Microsoft.Network/virtualNetworks	OK	Operation details
T9-FE-2016630495-ip	Microsoft.Network/publicIPAddresses	OK	Operation details
T9-FE-2016630495-mng	Microsoft.Network/networkSecurityGroups	OK	Operation details

Una vez finalizado el despliegue observaremos la siguiente pantalla:

Haremos click en Go to resource y veremos la máquina virtual creada correctamente:

1.1.3.1. Prueba de conectividad.

Para probar que tenemos conectividad a la máquina virtual, trataremos de establecer una conexión por túnel de SSH usando las llaves generadas al inicio de semestre:

```
> ls
humbertowoody-azure.pem
> ssh -i humbertowoody-azure.pem 20.231.41.187
The authenticity of host '20.231.41.187 (20.231.41.187)' can't be established.
ED25519 key fingerprint is SHA256:pLOJ3kpGI1AUudQS9eCuOkd6TtHDYlK/kh23ZK8bWbs.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '20.231.41.187' (ED25519) to the list of known hosts.
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1038-azure x86_64)

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

System information as of Sun Jun 11 01:35:55 UTC 2023

System load:  0.05           Processes:          120
Usage of /:   5.2% of 28.89GB  Users logged in:    0
Memory usage: 6%
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

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

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

humbertowoody@T9-FE-2016630495:~$ echo "wuuu"
wuuu
humbertowoody@T9-FE-2016630495:~$
```

Como podemos observar la máquina es alcanzable de forma pública.

1.2. Desarrollo de software.

Para desarrollar el software de esta aplicación, se comenzó con la intención de usar herramientas que simplificaran la construcción de ambas partes, frontend y backend. Para ello se eligió utilizar el mismo lenguaje en ambos proyectos: JavaScript, esto mediante el uso de Node.js 18 con TypeScript como sistema de tipado.

En el caso del backend, se utilizaron las herramientas de Azure para crear el *boilerplate* de código de las funciones y se desarrolló a partir de ello.

En el frontend se utilizó React mediante un *bootstrapper* llamado Create React App y se añadió la librería MaterialUI a fin de simplificar el desarrollo utilizando el sistema de diseño provisto en la misma.

1.2.1. Backend.

Para el backend estamos utilizando Node 18 (mediante *Node Version Manager* o nvm) y el proyecto se genera mediante la interacción con las herramientas de Azure de línea de comandos que permiten generar el template de código y subsecuentes iteraciones.

1.2.1.1. Instalación de herramientas de línea de comandos de Azure.

Lo primero será instalar las herramientas de línea de comandos para Azure Functions. La documentación correspondiente:

Install the Azure Functions Core Tools

Azure Functions Core Tools [includes](#) a version of the same runtime that powers Azure Functions runtime that you can run on your local development computer. It also provides commands to create functions, connect to Azure, and deploy function projects.

Starting with version 2.x, Core Tools runs on [Windows](#), [macOS](#), and [Linux](#).

The screenshot shows the "macOS" tab selected in the navigation bar. A callout box highlights an "Important" note: "Functions doesn't currently support Python function development on ARM64 devices. To develop Python functions on a Mac with an M1 chip, you must run in an emulated x86 environment. To learn more, see [x86 emulation on ARM64](#)". Below this, instructions for macOS use Homebrew to install the Core Tools. Step 1: "Install Homebrew", Step 2: "Install the Core Tools package". A code block shows the terminal command:

```
Bash
brew tap azure/functions
brew install azure-functions-core-tools@4
# if upgrading on a machine that has 2.x or 3.x installed:
brew link --overwrite azure-functions-core-tools@4
```

<https://learn.microsoft.com/en-us/azure/azure-functions/functions-run-local?tabs=v4%2Cmacos%2Cts%2Cportal%2Cbash#install-the-azure-functions-core-tools>

Así que iremos a nuestra terminal y ejecutaremos los comandos solicitados:

```
> brew tap azure/functions
=> Tapping azure/functions
Cloning into '/opt/homebrew/Library/Taps/azure/homebrew-functions' ...
remote: Enumerating objects: 679, done.
remote: Counting objects: 100% (277/277), done.
remote: Compressing objects: 100% (110/110), done.
remote: Total 679 (delta 203), reused 206 (delta 167), pack-reused 402
Receiving objects: 100% (679/679), 105.22 KiB | 997.00 KiB/s, done.
Resolving deltas: 100% (439/439), done.
Tapped 5 formulae (23 files, 149.0KB).
> brew install azure-functions-core-tools@4
=> Fetching azure/functions/azure-functions-core-tools@4
=> Downloading https://functionscdn.azureedge.net/public/4.0.5198/Azure.Functions.Cli.osx-arm64.4.0.5198.zip
Already downloaded: /Users/humbertowoodly/Library/Caches/Homebrew/downloads/baebf2358f6e68ca5a027459d842e8aa0ea24f9d27289ee1a24bbdee184f252f--Azure.Functions.Cli.osx-arm64.4.0.5198.zip
=> Installing azure-functions-core-tools@4 from azure/functions

Telemetry
_____
The Azure Functions Core tools collect usage data in order to help us improve your experience.
The data is anonymous and doesn't include any user specific or personal information. The data is collected by Microsoft.

You can opt-out of telemetry by setting the FUNCTIONS_CORE_TOOLS_TELEMETRY_OPTOUT environment variable to '1' or 'true' using your favorite shell.
> /opt/homebrew/Cellar/azure-functions-core-tools@4/4.0.5198: 4,186 files, 467.1MB, built in 13 seconds
=> Running 'brew cleanup azure-functions-core-tools@4' ...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
>
```

Probaremos que el comando fue instalado correctamente:

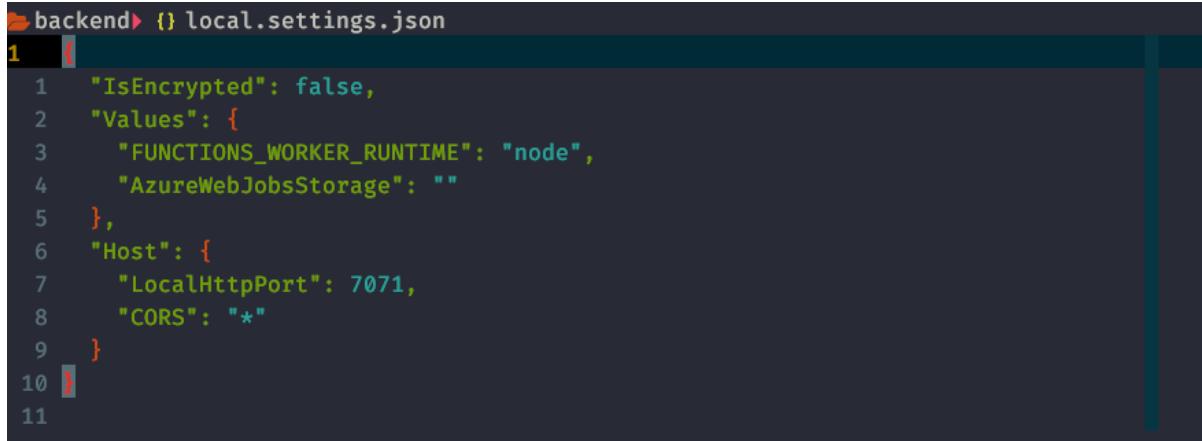
```
> func --version
4.0.5198
~
>
```

1.2.1.2. Inicialización de proyecto.

Ahora inicializamos el proyecto creando la carpeta dónde almacenaremos el código y usando func init:

```
> mkdir backend
> cd backend
> func init
Select a number for worker runtime:
1. dotnet
2. dotnet (isolated process)
3. node
4. python
5. powershell
6. custom
Choose option: 3
node
Select a number for language:
1. javascript
2. typescript
Choose option: 2
typescript
Did you know? There is a new Node.js programming model in public preview. Learn how you can try it out today at https://aka.ms/AzFuncNodeV4
Writing package.json
Writing .funcignore
Writing tsconfig.json
Writing .gitignore
Writing host.json
Writing local.settings.json
Writing /Users/humbertowoodly/Proyectos/backend/.vscode/extensions.json
> ls
host.json local.settings.json package.json tsconfig.json
```

Ahora abriremos el archivo **local.settings.json** y lo editaremos para agregar el bloque *Host* en el cual podemos especificar el puerto de ejecución y la política de CORS (para ejecución local), es importante que coloquemos ésta última pues nuestro frontend en React requiere de las políticas de CORS funcionando correctamente:



```
backend> {} local.settings.json
1 {
2   "IsEncrypted": false,
3   "Values": {
4     "FUNCTIONS_WORKER_RUNTIME": "node",
5     "AzureWebJobsStorage": ""
6   },
7   "Host": {
8     "LocalHttpPort": 7071,
9     "CORS": "*"
10 }
11 }
```

Ahora editaremos nuestro **package.json** de la siguiente manera para añadir un autor, un nombre y descripción:



```
backend> {} package.json
21 {
20   "name": "Tarea9-Distribuidos-ESCOM",
19   "version": "1.0.0",
18   "description": "Backend para la Tarea 9 de la materia de Sistemas Distribuidos",
17   "author": "Humberto Alejandro Ortega Alcocer",
16   "scripts": {
15     "build": "tsc",
14     "watch": "tsc -w",
13     "prestart": "npm run build",
12     "start": "func start",
11     "test": "echo \\\"No tests yet ... \\\""
10   },
9   "dependencies": {
8   },
7   "devDependencies": {
6     "@azure/functions": "^3.0.0",
5     "@types/node": "18.x",
4     "azure-functions-core-tools": "^4.x",
3     "typescript": "^4.0.0"
2   }
1 }
22 ````
```

Si observamos la parte de *scripts*, observaremos que contamos con dos comandos útiles: *prestart* y *start*. El primero llama a *build* que a su vez realiza la transpilación de TypeScript a JavaScript, el segundo inicia la ejecución de funciones usando el código fuente transpilado en JavaScript para ejecutar las funciones.

Ahora instalaremos las dependencias iniciales necesarias para el proyecto usando el manejador de paquetes de node (*Node Package Manager*) **npm** de la siguiente manera:

```
> nvm use 10.66 node --version 66 npm --version 66 npm install
  using node v10.66.0 (npm v9.5.2)
v10.66.0
9.5.1
npm WARN deprecated debug@4.1.1: Debug versions >=3.2.0 <>3.2.7 || >=4 <>4.3.1 have a low-severity ReDoS regression when used in a Node.js environment. It is recommended you upgrade to 3.2.7 or 4.3.1. (https://github.com/visionmedia/debug/issues/97)
added 42 packages, and audited 43 packages in 3s

3 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Ahora instalaremos dos paquetes que requerimos para el desarrollo de nuestra tarea:

1. **sequelize**: Esta librería nos permite interactuar con distintos gestores de base de datos realizando una abstracción a modo de ORM para nuestra aplicación.
2. **mysql2**: Esta librería es el *driver* de base de datos que usará Sequelize para realizar la conexión con nuestra base de datos.

Para instalarlas usaremos el manejador de paquetes de node (*Node Package Manager*) **npm** con el siguiente comando:

```
> npm install sequelize mysql2

added 30 packages, and audited 73 packages in 2s

4 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Con esto estaremos listos para comenzar el desarrollo de nuestras funciones.

1.2.1.3. Definición de estructura de base de datos.

Antes de comenzar a programar debemos tener muy clara la estructura de nuestra base de datos, a continuación se comparte el código presente en **base_de_datos.sql** el cual genera la base de datos que usaremos para la tarea:

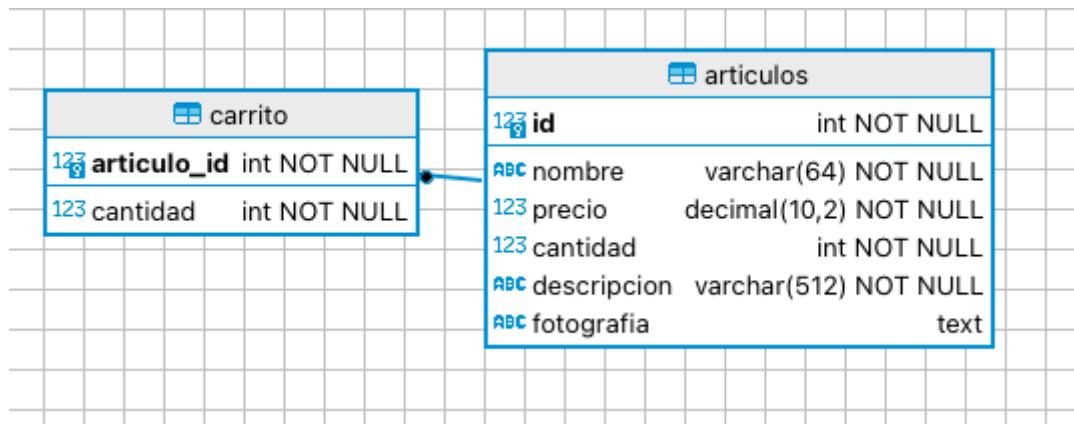
```
-- Sistemas Distribuidos - Tarea 9
-- Script para generar la base de datos requerida por la tarea.
-- Humberto Alejandro Ortega Alcocer
-- 2016630495

-- Eliminamos la base de datos si existe.
DROP DATABASE IF EXISTS tarea9;

-- Creamos la base de datos y la seleccionamos.
CREATE DATABASE tarea9;
USE tarea9;
```

```
-- Creamos la tabla de Articulos.  
CREATE TABLE articulos(  
    id INT NOT NULL AUTO_INCREMENT,  
    nombre VARCHAR(64) NOT NULL,  
    precio DECIMAL(10,2) NOT NULL,  
    cantidad INT NOT NULL,  
    descripcion VARCHAR(512) NOT NULL,  
    fotografia LONGBLOB,  
    PRIMARY KEY (id)  
);  
  
-- Creamos la tabla de Carrito.  
CREATE TABLE carrito (  
    articulo_id INT NOT NULL,  
    cantidad INT NOT NULL,  
    PRIMARY KEY (articulo_id),  
    FOREIGN KEY (articulo_id) REFERENCES articulos(id)  
);
```

La estructura de la base de datos final observada desde DBeaver es la siguiente:



Ahora podremos comenzar a realizar la programación correspondiente a cada una de nuestras funciones.

1.2.1.4. Creación de funciones.

Para crear cada función se enalizará lo que debe realizar, las consideraciones a tener en cuenta, la inicialización, edición de métodos disponibles y la interacción correspondiente con ChatGPT (de haberla) para solventar cualquier eventualidad.

Es importante mencionar que, para el caso de Node.js, al usar `func start`, si realizáramos alguna modificación en nuestro código estos cambios no se reflejarán en nuestra aplicación en ejecución hasta que reiniciemos el proceso.

Otro dato importante es que, en los códigos aquí presentados se usará el *string* de conexión para mi base de datos local de pruebas, cuando se llegue a la sección de despliegue se hablará sobre las modificaciones pertinentes para que trabaje con el servidor de MySQL en Azure.

1.2.1.4.1. CrearArticulo.

La primer función que crearemos será la de CrearArticulo con la que pretendemos que el usuario pueda ingresar los datos de un nuevo artículo en el frontend y ésta función lo reciba, valide y almacene en la base de datos.

Así que comenzaremos inicializando la función:

```
> func new
Select a number for template:
1. Azure Blob Storage trigger
2. Azure Cosmos DB trigger
3. Durable Functions activity
4. Durable Functions entity
5. Durable Functions Entity HTTP starter
6. Durable Functions HTTP starter
7. Durable Functions orchestrator
8. Azure Event Grid trigger
9. Azure Event Hub trigger
10. HTTP trigger
11. IoT Hub (Event Hub)
12. Kafka output
13. Kafka trigger
14. Azure Queue Storage trigger
15. RabbitMQ trigger
16. SendGrid
17. Azure Service Bus Queue trigger
18. Azure Service Bus Topic trigger
19. SignalR negotiate HTTP trigger
20. Timer trigger
Choose option: 10
HTTP trigger
Function name: [HttpTrigger] CrearArticulo
Writing /Users/humbertowoodly/Proyectos/backend/CrearArticulo/index.ts
Writing /Users/humbertowoodly/Proyectos/backend/CrearArticulo/function.json
The function "CrearArticulo" was created successfully from the "HTTP trigger" template.
Did you know? There is a new Node.js programming model in public preview. Learn how you can try it out today at https://aka.ms/AzFuncNodeV4
```

Comenzaremos editando nuestro archivo **function.json** en el cual permitiremos únicamente peticiones de tipo POST:

```
CrearArticulo > {} function.json
1  {
2    "bindings": [
3      {
4        "authLevel": "function",
5        "type": "httpTrigger",
6        "direction": "in",
7        "name": "req",
8        "methods": ["post"]
9      },
10     {
11       "type": "http",
12       "direction": "out",
13       "name": "res"
14     }
15   ],
16   "scriptFile": "../dist/CrearArticulo/index.js"
17 }
```

El código generado para esta función no requirió de ChatGPT, sino que fue realizado con GitHub Copilot directamente en mi editor de código (Visual Studio Code).

Lo primero fue definir las interfaces requeridas para modelar nuestros datos:

```
// Interface para el DTO de crear artículo.  
  
interface CrearArticuloDTO {  
    nombre: string;  
    descripcion?: string;  
    precio: number;  
    cantidad: number;  
    fotografía?: string;  
}  
  
// Interface para representar un artículo.  
  
interface Artículo {  
    id: number;  
    nombre: string;  
    descripción?: string;  
    precio: number;  
    cantidad: number;  
    fotografía?: string;  
}
```

Lo siguiente fue desarrollar la función, a grandes rasgos lo que se hace es:

1. Recibir el payload.
2. Validar los campos.
3. Iniciar la conexión a la base de datos.
4. Insertar los datos válidos.
5. Obtener el artículo creado.
6. Cerrar la conexión con la base de datos.
7. Responder con el artículo creado.

En caso de que falle en algún punto se responde con un código acorde, por ejemplo, para validaciones será un 400 y para errores de conexión o ejecución en la base de datos será de 500. El código en cuestión es el siguiente:

```
const httpTrigger: AzureFunction = async function (  
    context: Context,  
    req: HttpRequest  
) : Promise<void> {  
    context.log("Se ha recibido una petición HTTP para Crear Artículo.");  
  
    // Validamos que hayamos recibido un body.  
    if (req.body) {  
        // Creamos el objeto a partir del body.  
        const artículoDto: CrearArticuloDTO = req.body;
```

```
// Validamos los campos obligatorios.
if (articuloDto.nombre && articuloDto.precio && articuloDto.cantidad) {
// Validar que los datos sean correctos
let articuloValido: boolean = true;

if (articuloDto.nombre.length < 3 || articuloDto.nombre.length > 50) {
articuloValido = false;
} else if (articuloDto.precio < 0) {
articuloValido = false;
} else if (articuloDto.cantidad < 0) {
articuloValido = false;
}

// Validar campos opcionales.
if (articuloDto.descripcion && articuloDto.descripcion.length > 200) {
articuloValido = false;
}

// Si el artículo es válido, insertar en la base de datos.
if (articuloValido) {
// A partir de aquí todo dentro de un try/catch.
try {
// Creamos objeto del ORM.
const sequelize = new Sequelize(
"mysql://root:root@localhost:3306/tarea9",
{
dialect: "mysql",
}
);

// Insertamos usando un query raw en mysql
await sequelize.query(
`INSERT INTO articulos (nombre, descripcion, precio, cantidad, fotografia) VALUES (?, ?, ?, ?, ?)`,
{
type: QueryTypes.INSERT,
replacements: [
articuloDto.nombre,
articuloDto.descripcion || null,
articuloDto.precio,
articuloDto.cantidad,
articuloDto.fotografia || null,
],
}
);

// Obtenemos el artículo que acabamos de insertar.
const articulos: Articulo[] = await sequelize.query(
`SELECT * FROM articulos WHERE nombre = ?;`,
{

```

```
type: QueryTypes.SELECT,
replacements: [articuloDto.nombre],
}

);

// Cerramos la conexión.
await sequelize.close();

// Retornamos el código 201 (Created) y el articulo creado.
context.res = {
status: 201,
body: articulos[0],
};

} catch (error) {
context.log(error);
// Retornamos el código 500 (Internal Server Error) y un mensaje de error.
context.res = {
status: 500,
body: {
message: "Error al crear el artículo",
error: error,
},
};
}

} else {
// Retornamos el código 400 (Bad Request) y un mensaje de error.
context.res = {
status: 400,
body: "Los datos del artículo no son válidos",
};
}

} else {
// Retornamos el código 400 (Bad Request) y un mensaje de error.
context.res = {
status: 400,
body: "No se recibieron todos los datos necesarios para crear el artículo",
};
}

} else {
// Retornamos el código 400 (Bad Request) y un mensaje de error.
context.res = {
status: 400,
body: "No se recibieron datos",
};
}
};
```

1.2.1.4.2. ObtenerArticulos.

La función de Obtener Artículos es muy simple aunque debe considerar dos flujos de ejecución:

1. El usuario solo quiere obtener un listado completo de los artículos presentes en la base de datos.
2. El usuario desea obtener los artículos que hagan *match* con algún texto de búsqueda particular.

Comenzaremos creando la función en cuestión:

```
> func new
Select a number for template:
1. Azure Blob Storage trigger
2. Azure Cosmos DB trigger
3. Durable Functions activity
4. Durable Functions entity
5. Durable Functions Entity HTTP starter
6. Durable Functions HTTP starter
7. Durable Functions orchestrator
8. Azure Event Grid trigger
9. Azure Event Hub trigger
10. HTTP trigger
11. IoT Hub (Event Hub)
12. Kafka output
13. Kafka trigger
14. Azure Queue Storage trigger
15. RabbitMQ trigger
16. SendGrid
17. Azure Service Bus Queue trigger
18. Azure Service Bus Topic trigger
19. SignalR negotiate HTTP trigger
20. Timer trigger
Choose option: 10
HTTP trigger
Function name: [HttpTrigger] ObtenerArticulos
Writing /Users/humbertowoodsy/Proyectos/backend/ObtenerArticulos/index.ts
Writing /Users/humbertowoodsy/Proyectos/backend/ObtenerArticulos/function.json
The function "ObtenerArticulos" was created successfully from the "HTTP trigger" template.
Did you know? There is a new Node.js programming model in public preview. Learn how you can try it out today at https://aka.ms/AzFuncNodeV4
```

Y editaremos el archivo **function.json** de la siguiente manera para permitir únicamente solicitudes de tipo GET:

```
ObtenerArticulos > {} function.json
1  {
2    "bindings": [
3      {
4        "authLevel": "function",
5        "type": "httpTrigger",
6        "direction": "in",
7        "name": "req",
8        "methods": ["get"]
9      },
10     {
11       "type": "http",
12       "direction": "out",
13       "name": "res"
14     }
15   ],
16   "scriptFile": "../dist/ObtenerArticulos/index.js"
```

Comenzaremos nuestro desarrollo definiendo las interfaces que necesitaremos para la ejecución:

```
// Interface para representar un artículo.  
interface Artículo {  
    id: number;  
    nombre: string;  
    descripción?: string;  
    precio: number;  
    cantidad: number;  
    fotografía?: string;  
}
```

A fin de cumplir con los dos flujos de ejecución previamente planteados, se propone el siguiente flujo dentro de la función a desarrollar:

1. Obtener el parámetro de URL *nombre*, si es que se encuentra.
2. Abrir la conexión con la base de datos.
3. Verificar si recibimos un *nombre*.
 - a. Si lo recibimos, realizamos un SELECT filtrando los resultados con un LIKE tanto en el nombre del artículo como en la descripción.
 - b. Si no lo recibimos, realizamos un SELECT sin filtros.
4. Cerramos la conexión con la base de datos.
5. Respondemos con el listado de artículos obtenido.

En caso de que hubiera algún error de validación o de ejecución se responderá con el código 400 o 500 según corresponda y un mensaje que permita identificar el error de mejor forma.

El código que refleja este flujo de ejecución es el siguiente:

```
const httpTrigger: AzureFunction = async function (  
    context: Context,  
    req: HttpRequest  
) : Promise<void> {  
    context.log("Se ha recibido una petición HTTP para Obtener Artículos");  
  
    // Se revisa si se recibió un query parameter "nombre"  
    const nombre: string = req.query.nombre || (req.body && req.body.nombre);  
  
    if (nombre) {  
        context.log("Se recibió el query parameter nombre: " + nombre);  
    }  
  
    try {  
        // Abrimos la conexión a la base de datos.  
        const sequelize = new Sequelize("mysql://root:root@localhost:3306/tarea9", {  
            dialect: "mysql",  
        });  
    }
```

```
// Variable para el resultado.  
let articulos: Articulo[] = [];  
  
// Si recibimos el nombre, usamos un query por nombre.  
if (nombre) {  
  articulos = await sequelize.query<Articulo>(  
    "SELECT id, nombre, descripcion, precio, cantidad, CONVERT(fotografia using utf8) as  
    fotografia FROM articulos WHERE nombre LIKE :nombre OR descripcion LIKE :nombre",  
    {  
      type: QueryTypes.SELECT,  
      // Mapeamos los resultados a la interface Articulo.  
      mapToModel: true,  
      // Reemplazamos el query parameter :nombre por el valor recibido.  
      replacements: { nombre: `#${nombre}` },  
    }  
  );  
}  
  
// Obtenemos los artículos de la base de datos.  
articulos = await sequelize.query<Articulo>(  
  "SELECT id, nombre, descripcion, precio, cantidad, CONVERT(fotografia using utf8) as  
  fotografia FROM articulos",  
  {  
    type: QueryTypes.SELECT,  
    // Mapeamos los resultados a la interface Articulo.  
    mapToModel: true,  
  }  
);  
  
// Cerramos la conexión  
await sequelize.close();  
  
// Enviamos la respuesta.  
context.res = {  
  status: 200,  
  // Enviamos los artículos como respuesta.  
  body: articulos,  
};  
} catch (error) {  
  context.res = {  
    status: 500,  
    body: { mensaje: "Error al obtener los artículos.", error },  
  };  
}  
};
```

1.2.1.4.3. AgregarCarrito.

En la función de Agregar al Carrito lo que se busca es que el usuario proporcione el ID de un Artículo y la cantidad del mismo, se valide que exista dicha cantidad en almacén y, en caso de ser así, que se añada el registro a una tabla de Carrito de compras y se descuento la cantidad en el carrito de aquella disponible en almacén.

Comenzaremos creando la función:

```
> func new
Select a number for template:
1. Azure Blob Storage trigger
2. Azure Cosmos DB trigger
3. Durable Functions activity
4. Durable Functions entity
5. Durable Functions Entity HTTP starter
6. Durable Functions HTTP starter
7. Durable Functions orchestrator
8. Azure Event Grid trigger
9. Azure Event Hub trigger
10. HTTP trigger
11. IoT Hub (Event Hub)
12. Kafka output
13. Kafka trigger
14. Azure Queue Storage trigger
15. RabbitMQ trigger
16. SendGrid
17. Azure Service Bus Queue trigger
18. Azure Service Bus Topic trigger
19. SignalR negotiate HTTP trigger
20. Timer trigger
Choose option: 10
HTTP trigger
Function name: [HttpTrigger] AgregarCarrito
Writing /Users/humbertowoody/Proyectos/backend/AgregarCarrito/index.ts
Writing /Users/humbertowoody/Proyectos/backend/AgregarCarrito/function.json
The function "AgregarCarrito" was created successfully from the "HTTP trigger" template.
Did you know? There is a new Node.js programming model in public preview. Learn how you can try it out today at https://aka.ms/AzFuncNodeV4
```

Una vez creada la función, editaremos el archivo **function.json** en el directorio de la función creada para que sólamente permita solicitudes de tipo POST:



```
AgregarCarrito> {} function.json
1  {
2    "bindings": [
3      {
4        "authLevel": "function",
5        "type": "httpTrigger",
6        "direction": "in",
7        "name": "req",
8        "methods": ["post"]
9      },
10     {
11       "type": "http",
12       "direction": "out",
13       "name": "res"
14     }
15   ],
16   "scriptFile": "../dist/AgregarCarrito/index.js"
```

Para el desarrollo de la función, primero debemos considerar las interfaces requeridas para modelar la información con la que vamos a interactuar. Las interfaces son las siguientes:

```
// Interfaz para los datos que necesitamos.  
interface AgregarArticuloCarritoDTO {  
    articulo_id: number;  
    cantidad: number;  
}  
  
// Interfaz para representar un artículo.  
interface Articulo {  
    id: number;  
    nombre: string;  
    descripcion?: string;  
    precio: number;  
    cantidad: number;  
    fotografia?: string;  
}  
  
// Interfaz para representar un artículo en el carrito.  
interface Carrito {  
    articulo_id: number;  
    cantidad: number;  
}
```

El flujo de ejecución que queremos lograr en esta función es el siguiente:

1. Recibir el payload con el ID del artículo y la cantidad.
2. Iniciar la conexión con la base de datos.
3. Validar que el artículo exista.
4. Validar que el artículo cuente con la cantidad solicitada.
5. Validar que el artículo no esté ya en el carrito de compras.
6. Crear una transacción de la base de datos.
7. Insertar el artículo en el carrito de compras.
8. Actualizar la cantidad disponible del artículo en almacén.
9. Ejecutar la transacción de la base de datos.
10. Cerrar la conexión con la base de datos.
11. Responder con el artículo actualizado.

Para lograr este flujo, el código es el siguiente:

```
const httpTrigger: AzureFunction = async function (  
    context: Context,  
    req: HttpRequest  
) : Promise<void> {  
    context.log(  
        "Se ha recibido una petición HTTP para Agregar un Artículo al Carrito"  
    );  
  
    // Validamos que hayamos recibido un body.
```

```
if (req.body) {
    // Validamos los datos del Body.
    const agregarArticuloCarritoDto: AgregarArticuloCarritoDTO = req.body;

    try {
        // Iniciamos la conexión con la base de datos.
        const sequelize = new Sequelize(
            "mysql://root:root@localhost:3306/tarea9",
            {
                dialect: "mysql",
                dialectOptions: {
                    multipleStatements: true,
                },
            }
        );

        // Obtenemos el artículo.
        const articulos: Articulo[] = await sequelize.query<Articulo>(
            "SELECT id, nombre, descripción, precio, cantidad, CONVERT(fotografía using utf8) as fotografía FROM artículos WHERE id = ?",
            {
                type: QueryTypes.SELECT,
                mapToModel: true,
                replacements: [agregarArticuloCarritoDto.articulo_id],
            }
        );

        // Verificamos que se haya encontrado al menos uno.
        if (articulos.length === 0) {
            context.res = {
                status: 404,
                body: "No se encontró el artículo.",
            };

            return;
        }

        const artículo: Articulo = articulos[0];

        // Validamos que haya suficientes artículos.
        if (artículo.cantidad < agregarArticuloCarritoDto.cantidad) {
            context.res = {
                status: 400,
                body: "No hay suficientes artículos.",
            };

            return;
        }

        // Validamos que el artículo no esté ya en el carrito.
        const carritos: Carrito[] = await sequelize.query<Carrito>(
    
```

```
"SELECT articulo_id, cantidad FROM carrito WHERE articulo_id = :articulo_id",
{
  type: QueryTypes.SELECT,
  mapToModel: true,
  replacements: { articulo_id: articulo.id },
}
);

// Verificamos que se haya encontrado al menos uno.
if (carritos.length === 0) {
  // Creamos la transaccion.
  const transaction = await sequelize.transaction();

  // Agregamos el articulo al carrito.
  await sequelize.query(
    "INSERT INTO carrito (articulo_id, cantidad) VALUES (:articulo_id, :cantidad)",
    {
      type: QueryTypes.INSERT,
      replacements: {
        articulo_id: articulo.id,
        cantidad: agregarArticuloCarritoDto.cantidad,
      },
      transaction,
    }
  );

  // Quitamos la cantidad de articulos del inventario.
  await sequelize.query(
    "UPDATE articulos SET cantidad = cantidad - :cantidad WHERE id = :id",
    {
      type: QueryTypes.UPDATE,
      replacements: {
        id: articulo.id,
        cantidad: agregarArticuloCarritoDto.cantidad,
      },
      transaction,
    }
  );

  // Restamos la cantidad de articulos del inventario.
  articulo.cantidad -= agregarArticuloCarritoDto.cantidad;

  // Hacemos commit de la transacción.
  await transaction.commit();

  // Regresamos el articulo.
  context.res = {
    status: 200,
    body: articulo,
  };
} else {
}
```

```
// Informamos al usuario de que el artículo ya está en el carrito y que debe usar la función de actualizar.

context.res = {
status: 402,
body: "El artículo ya está en el carrito. Use la función de actualizar.",
};

// Cerrar la conexión con la base de datos.

await sequelize.close();
} catch (error) {
// Si hubo un error en la base de datos regresamos un error 500.

context.res = {
status: 500,
body: {
mensaje: "Ocurrió un error en la base de datos.",
error: error,
},
};
}

} else {
// Si no recibimos un body, regresamos un error.

context.res = {
status: 400,
body: "No se recibió nada en el body",
};
}
};
```

1.2.1.4.4. EliminarCarrito.

La función Eliminar Carrito permite al usuario eliminar un artículo en el carrito de compras, para esto se deberá eliminar el elemento de la tabla de carrito de compras y se deberán de regresar los elementos que estaban presentes dicho carrito a la cantidad disponible en almacén del artículo original.

Lo primero será crear la función:

```
> func new
Select a number for template:
1. Azure Blob Storage trigger
2. Azure Cosmos DB trigger
3. Durable Functions activity
4. Durable Functions entity
5. Durable Function Entity HTTP starter
6. Durable Function HTTP starter
7. Durable Functions orchestrator
8. Azure Event Grid trigger
9. Azure Event Hub trigger
10. HTTP trigger
11. IoT Hub (Event Hub)
12. Kafka output
13. Kafka trigger
14. Azure Queue Storage trigger
15. RabbitMQ trigger
16. SendGrid
17. Azure Service Bus Queue trigger
18. Azure Service Bus Topic trigger
19. SignalR negotiate HTTP trigger
20. Timer trigger
Choose option: 10
HTTP trigger
Function name: [HttpTrigger] EliminarCarrito
Writing /Users/humbertowoodly/Proyectos/backend/EliminarCarrito/index.ts
Writing /Users/humbertowoodly/Proyectos/backend/EliminarCarrito/function.json
The function "EliminarCarrito" was created successfully from the "HTTP trigger" template.
Did you know? There is a new Node.js programming model in public preview. Learn how you can try it out today at https://aka.ms/AzFuncNodeV4
```

Ahora editaremos el archivo **function.json** para que nuestra función únicamente reciba solicitudes tipo POST:

```
▶ EliminarCarrito▶ {} function.json
1 {
2   "bindings": [
3     {
4       "authLevel": "function",
5       "type": "httpTrigger",
6       "direction": "in",
7       "name": "req",
8       "methods": ["post"]
9     },
10    {
11      "type": "http",
12      "direction": "out",
13      "name": "res"
14    }
15  ],
16  "scriptFile": "../dist/EliminarCarrito/index.js"
17 }
```

Con esto realizado, ahora visualizaremos las interfaces requeridas para la función en cuestión:

```
// Interfaz para representar un carrito.
interface Carrito {
  articulo_id: number;
  cantidad: number;
}

// Interfaz para representar un artículo.
interface Articulo {
  id: number;
  nombre: string;
  precio: number;
  descripcion: string;
  cantidad: number;
  fotografia?: string;
}
```

El flujo de ejecución que debemos seguir en esta función es el siguiente:

1. Obtenemos el ID del Artículo a eliminar del carrito de compras del *body* de nuestra solicitud.
2. Abrimos la conexión con la base de datos.
3. Validamos que el artículo esté presente en el carrito de compras.
4. Creamos una transacción de la base de datos.
5. Actualizamos la cantidad de artículos en almacén devolviendo la cantidad presente en el carrito.
6. Eliminamos el artículo del carrito de compras.
7. Aplicamos la transacción de la base de datos.
8. Cerramos la conexión con la base de datos.
9. Respondemos un código 200 con un mensaje de éxito.

En caso de que hubieran problemas de validación o de ejecución se responderán con códigos 400 o 500 según sea el caso además de mensajes de error para ayudar a encontrar el error.

El código que representa el flujo planteado anteriormente es:

```
const httpTrigger: AzureFunction = async function (
  context: Context,
  req: HttpRequest
): Promise<void> {
  context.log(
    "Se recibió una petición HTTP para Eliminar un registro del Carrito de Compras"
  );

  const id = req.body.articulo_id;

  if (!id) {
    context.res = {
      status: 400,
      body: "Falta el ID del Artículo en el Carrito de Compras",
    };
  } else {
    try {
      // Conectarse a la base de datos
      const sequelize = new Sequelize(
        "mysql://root:root@localhost:3306/tarea9",
        {
          dialect: "mysql",
          dialectOptions: {
            multipleStatements: true,
          },
        }
      );

      // Obtener el carrito de compras.
      const carritos: Carrito[] = await sequelize.query<Carrito>(
```

```
"SELECT * FROM carrito WHERE articulo_id = :articulo_id",
{
replacements: { articulo_id: id },
type: QueryTypes.SELECT,
};

// Validamos que el articulo exista.
if (carritos.length <= 0) {
context.res = {
status: 404,
body: "No existe el articulo en el carrito de compras",
};
} else {
// Crear una transacción.
const transaction: Transaction = await sequelize.transaction();

// Agregar de vuelta las unidades del carrito al artículo.
await sequelize.query(
"UPDATE articulos SET cantidad = cantidad + :cantidad WHERE id = :id",
{
replacements: {
id: carritos[0].articulo_id,
cantidad: carritos[0].cantidad,
},
type: QueryTypes.UPDATE,
transaction,
}
);

// Eliminar el articulo del carrito de compras.
await sequelize.query(
"DELETE FROM carrito WHERE articulo_id = :articulo_id",
{
replacements: { articulo_id: id },
type: QueryTypes.DELETE,
transaction,
}
);

// Ejecutar la transacción.
await transaction.commit();

// Responder con un 200.
context.res = {
status: 200,
body: "Artículo eliminado del carrito de compras",
};
}

// Cerrar la conexión a la base de datos.
await sequelize.close();
```

```
} catch (error) {
context.res = {
status: 500,
body: {
mensaje: "Error al eliminar el artículo del carrito de compras",
error,
},
};
}
}
};

}
```

1.2.1.4.5. EditarCarrito.

La función de Editar Carrito será la encargada de manejar el caso dónde el usuario desea actualizar la cantidad de artículos presentes en el carrito de compras para un artículo particular. Esta operación es la más extensa del proyecto ya que debe considerar varias validaciones y casos de uso.

Lo primero será crear la función:

```
> func new
Select a number for template:
1. Azure Blob Storage trigger
2. Azure Cosmos DB trigger
3. Durable Functions activity
4. Durable Functions entity
5. Durable Functions Entity HTTP starter
6. Durable Functions HTTP starter
7. Durable Functions orchestrator
8. Azure Event Grid trigger
9. Azure Event Hub trigger
10. HTTP trigger
11. IoT Hub (Event Hub)
12. Kafka output
13. Kafka trigger
14. Azure Queue Storage trigger
15. RabbitMQ trigger
16. SendGrid
17. Azure Service Bus Queue trigger
18. Azure Service Bus Topic trigger
19. SignalR negotiate HTTP trigger
20. Timer trigger
Choose option: 10
HTTP trigger
Function name: [HttpTrigger] EditarCarrito
Writing /Users/humbertowoodly/Proyectos/backend/EditarCarrito/index.ts
Writing /Users/humbertowoodly/Proyectos/backend/EditarCarrito/function.json
The function "EditarCarrito" was created successfully from the "HTTP trigger" template.
Did you know? There is a new Node.js programming model in public preview. Learn how you can try it out today at https://aka.ms/AzFuncNodeV4
```

Ahora editaremos el archivo **function.json** para que sólamente reciba solicitudes POST:

```
EditarCarrito▶ {} function.json
1
1   "bindings": [
2     {
3       "authLevel": "function",
4       "type": "httpTrigger",
5       "direction": "in",
6       "name": "req",
7       "methods": ["post"]
8     },
9     {
10       "type": "http",
11       "direction": "out",
12       "name": "res"
13     }
14   ],
15   "scriptFile": "../dist/EditarCarrito/index.js"
16 }
```

Ahora definiremos las interfaces que requerimos para la función que acabamos de crear a fin de modelar los datos con los que vamos a trabajar:

```
// Interfaz para la solicitud.
interface Solicitud {
  articulo_id: number;
  cantidad: number;
}

// Interfaz para representar un articulo.
interface Articulo {
  id: number;
  nombre: string;
  precio: number;
  descripcion: string;
  cantidad: number;
  fotografia?: string;
}

// Interfaz para representar un elemento en el carrito.
interface Carrito {
  articulo_id: number;
  cantidad: number;
  articulo?: Articulo;
}
```

El flujo de ejecución que queremos crear para esta función es el siguiente:

1. Extraer el payload de la solicitud.
2. Abrir la conexión con la base de datos.
3. Validar que el artículo exista en el carrito de compras.
4. Obtener el artículo en almacén.
5. Existen dos casos, que el usuario deseé añadir más elementos al carrito o restar elementos del carrito.
 - a. Si desea añadir elementos al carrito.
 - i. Validar que existan suficientes artículos en almacén.
 - ii. Crear una transacción de base de datos.
 - iii. Actualizar la cantidad de artículos en carrito.
 - iv. Actualizar la cantidad de artículos en almacén.
 - v. Ejecutar la transacción de base de datos.
 - b. Si se desea restar elementos del carrito.
 - i. Creamos una transacción de base de datos.
 - ii. Actualizamos la cantidad de artículos en el carrito de compras.
 - iii. Actualizamos la cantidad de artículos en almacén.
 - iv. Ejecutamos la transacción de la base de datos.
6. Cerramos la conexión con la base de datos.
7. Respondemos con la entrada en el carrito de compras actualizada para el artículo en cuestión.

En caso de que haya errores de validación o de ejecución se responderá con un código 400 o 500 según sea el caso además de un mensaje para identificar más fácil el error en cuestión.

El código que representa el flujo de ejecución descrito es el siguiente:

```
const httpTrigger: AzureFunction = async function (context, req): Promise<void> {
    context.log("Se ha recibido una petición HTTP para Modificar al Carrito");

    // Extraemos el cuerpo de la solicitud.
    const solicitud: Solicitud = req.body;

    if (solicitud) {
        try {
            // Abrimos la conexión a la base de datos.
            const sequelize = new Sequelize(
                "mysql://root:root@localhost:3306/tarea9",
                {
                    dialect: "mysql",
                    dialectOptions: {
                        multipleStatements: true,
                
```

```
},  
}  
);  
  
// Obtenemos el carrito de compras.  
const carritos: Carrito[] = await sequelize.query<Carrito>(  
"SELECT * FROM carrito WHERE articulo_id = :articulo_id",  
{  
replacements: { articulo_id: solicitud.articulo_id },  
type: QueryTypes.SELECT,  
}  
);  
  
// Validamos que el artículo exista.  
if (carritos.length <= 0) {  
context.res = {  
status: 404,  
body: "El artículo no existe en el carrito de compras.",  
};  
} else {  
// Obtenemos el carrito.  
const carrito: Carrito = carritos[0];  
  
// Obtenemos el artículo.  
const articulos: Articulo[] = await sequelize.query<Articulo>(  
"SELECT * FROM articulos WHERE id = :id",  
{  
replacements: { id: solicitud.articulo_id },  
type: QueryTypes.SELECT,  
}  
);  
  
// Validamos que el artículo exista.  
if (articulos.length <= 0) {  
context.res = {  
status: 404,  
body: "El artículo no existe.",  
};  
} else {  
// Obtenemos el artículo.  
const articulo: Articulo = articulos[0];  
  
if (carrito.cantidad < solicitud.cantidad) {  
// El usuario agregó más artículos al carrito.  
// Validamos que haya suficientes artículos en existencia.  
if (articulo.cantidad < solicitud.cantidad) {  
context.res = {  
status: 400,  
body: "No hay suficientes artículos en existencia.",  
};  
} else {
```

```
// Creamos una transacción.  
const transaction = await sequelize.transaction();  
  
// Actualizamos la cantidad de artículos en el carrito.  
await sequelize.query(  
"UPDATE carrito SET cantidad = :cantidad WHERE articulo_id = :articulo_id",  
{  
replacements: {  
cantidad: solicitud.cantidad,  
articulo_id: solicitud.articulo_id,  
},  
type: QueryTypes.UPDATE,  
transaction,  
}  
);  
  
// Actualizamos la cantidad de artículos en existencia.  
await sequelize.query(  
"UPDATE articulos SET cantidad = :cantidad WHERE id = :id",  
{  
replacements: {  
cantidad:  
articulo.cantidad -  
(solicitud.cantidad - carrito.cantidad),  
id: solicitud.articulo_id,  
},  
type: QueryTypes.UPDATE,  
transaction,  
}  
);  
  
// Confirmamos la transacción.  
await transaction.commit();  
  
// Obtenemos el carrito de compras.  
const carritoRespuesta: Carrito = {  
articulo_id: solicitud.articulo_id,  
cantidad: solicitud.cantidad,  
articulo: articulo,  
};  
  
// Regresamos el carrito de compras.  
context.res = {  
status: 200,  
body: carritoRespuesta,  
};  
}  
} else {  
// El usuario quitó artículos del carrito.  
// Creamos una transacción.  
const transaction = await sequelize.transaction();
```

```
// Actualizamos la cantidad de artículos en el carrito.
await sequelize.query(
"UPDATE carrito SET cantidad = :cantidad WHERE articulo_id = :articulo_id",
{
replacements: {
cantidad: solicitud.cantidad,
articulo_id: solicitud.articulo_id,
},
type: QueryTypes.UPDATE,
transaction,
}
);

// Actualizamos la cantidad de artículos en existencia.
await sequelize.query(
"UPDATE articulos SET cantidad = :cantidad WHERE id = :id",
{
replacements: {
cantidad:
articulo.cantidad + (carrito.cantidad - solicitud.cantidad),
id: solicitud.articulo_id,
},
type: QueryTypes.UPDATE,
transaction,
}
);

// Confirmamos la transacción.
await transaction.commit();

// Obtenemos el carrito de compras.
const carritoRespuesta: Carrito = {
articulo_id: solicitud.articulo_id,
cantidad: solicitud.cantidad,
articulo: articulo,
};

// Regresamos el carrito de compras.
context.res = {
status: 200,
body: carritoRespuesta,
};
}

}

}

// Cerramos la conexión a la base de datos.
await sequelize.close();
} catch (error) {
context.res = {
status: 500,

```

```
body: "Error al conectarse a la base de datos.",  
};  
}  
} else {  
context.res = {  
status: 400,  
body: "No se recibieron datos.",  
};  
}  
};
```

1.2.1.5. Ejecutar funciones localmente

Para ejecutar las funciones que acabamos de crear de forma local, basta con usar el comando de **npm start** que se observa de la siguiente manera:

```
> npm start  
  
> Tarea9-Distribuidos-ESCOM@1.0.0 prestart  
> npm run build  
  
> Tarea9-Distribuidos-ESCOM@1.0.0 build  
> tsc  
  
> Tarea9-Distribuidos-ESCOM@1.0.0 start  
> func start  
  
Azure Functions Core Tools  
Core Tools Version:      4.0.5198 Commit hash: N/A  (64-bit)  
Function Runtime Version: 4.21.1.20667  
  
Functions:  
  
    AgregarCarrito: [POST] http://localhost:7071/api/AgregarCarrito  
    CrearArticulo: [POST] http://localhost:7071/api/CrearArticulo  
    EditarCarrito: [POST] http://localhost:7071/api/EditarCarrito  
    EliminarCarrito: [POST] http://localhost:7071/api/EliminarCarrito  
    ObtenerArticulos: [GET] http://localhost:7071/api/ObtenerArticulos  
    ObtenerCarrito: [GET] http://localhost:7071/api/ObtenerCarrito  
  
For detailed output, run func with --verbose flag.  
[2023-06-10T21:26:12.466Z] Worker process started and initialized.  
[2023-06-10T21:26:17.169Z] Host lock lease acquired by instance ID '00000000000000000000000000000076C0736D'.
```

Ahora podremos acceder a cada una de las funciones de forma local.

1.2.2. Frontend.

Para desarrollar el frontend de esta tarea se optó por traducir el frontend desarrollado en la tarea 7 a una versión en React porque supone una mejora en el control de flujo de la aplicación dados errores que fueron arreglados con formas poco elegantes y que en React son fáciles de identificar y tratar.

Para esto, se propuso el desarrollo de los siguientes componentes:

- Página de inicio: un página principal donde los usuarios puedan seleccionar entre visitar el formulario para agregar artículos y el catálogo de artículos para comprar.
- Formulario de artículos: un formulario que permita registrar de forma rápida los campos requeridos para registrar un artículo.
- Catálogo de artículos: una rejilla donde cada elemento sea un artículo en exhibición, con opción para añadir al carrito.
- Carrito de compras: una página donde el usuario pueda consultar su carrito de compras y le permite realizar operaciones con éste tales como modificar cantidad y/o eliminar del carrito.

Para esto, se creó un proyecto utilizando *Create React App*, el cual provee de una estructura de archivos básica y la configuración inicial, para usarlo usamos el comando:

```
> npx create-react-app frontend --template typescript
Creating a new React app in /Users/humbertowoodly/Proyectos/frontend.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template-typescript ...

added 1435 packages in 13s

234 packages are looking for funding
  run 'npm fund' for details

Initialized a git repository.

Installing template dependencies using npm ...
added 40 packages, and changed 2 packages in 3s

234 packages are looking for funding
  run 'npm fund' for details

We detected TypeScript in your project (src/App.test.tsx) and created a tsconfig.json file for you.

Your tsconfig.json has been populated with default values.

Removing template package using npm ...

removed 1 package, and audited 1475 packages in 1s

234 packages are looking for funding
  run 'npm fund' for details

6 high severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

Created git commit.

Success! Created frontend at /Users/humbertowoodly/Proyectos/frontend
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd frontend
  npm start

Happy hacking!
```

La estructura de archivos provista por CRA es la siguiente:

```
> tree -I node_modules
.
├── README.md
├── package-lock.json
├── package.json
└── public
    ├── favicon.ico
    ├── index.html
    ├── logo192.png
    ├── logo512.png
    ├── manifest.json
    └── robots.txt
└── src
    ├── App.css
    ├── App.test.tsx
    ├── App.tsx
    ├── index.css
    ├── index.tsx
    ├── logo.svg
    ├── react-app-env.d.ts
    ├── reportWebVitals.ts
    └── setupTests.ts
    tsconfig.json

3 directories, 19 files
~/Proyectos/frontend main
> █
```

Comenzaremos a añadir los componentes dentro de **src/** y para cada uno tendremos que resolver problemas particulares.

1.2.2.1. Agregar Artículo

Este será el primer componente del que hablemos porque gracias a este componente fue que se decidió por usar el sistema de diseño Material UI así como otras librerías.

Al inicio del desarrollo de esta tarea, decidí preguntarle a ChatGPT 4 cuál sería el formulario más simple que se pudiera realizar con React, esta fue la conversación:

HU can you give me a material ui form in tsx that has the following fields and validation?

- nombre: string, less than 255 chars
- descripción: string, no more than 2048 chars, make it a textbox
- precio: float, more than 0, 2 decimal places, MXN currency format if that helps
- cantidad: int, more than 1
- foto: make it so that the user can upload an image and, when submitting the form, it gets serialized into base64

I want that when the user clicks on "Crear Producto", it validates the form (shows alert if it has issues), then it serializes the image and everything into JSON, sends it to POST localhost:8080/productos and, if the answer is a 200 OK, show a confirmation popup, if it's a 400 you should show a popup instructing the user to check the data, any other response should go into a popup saying the system found an internal issue.

thanks!

Sure, below is a sample form with Material UI components and validation using Formik and Yup. Please note that you need to have these dependencies installed for this form to work: `@material-ui/core`, `@material-ui/icons`, `formik`, `yup`, `axios` and `react`.

For the image serialization, this example uses the FileReader API to convert the image to a base64 string when the form is submitted. The image is then included in the JSON payload. Be aware that this method can result in large strings for large images.

```
tsx Copy code
import React, { useState } from 'react';
import { Button, TextField, Box } from '@material-ui/core';
import { useFormik } from 'formik';
import * as yup from 'yup'.
```

Be aware that this method can result in large strings for large images.

```
tsx Copy code

import React, { useState } from 'react';
import { Button, TextField, Box } from '@material-ui/core';
import { useFormik } from 'formik';
import * as yup from 'yup';
import axios from 'axios';

const validationSchema = yup.object({
  nombre: yup
    .string('Enter a name')
    .max(255, 'Name should not exceed 255 characters')
    .required('Name is required'),
  descripcion: yup
    .string('Enter a description')
    .max(2048, 'Description should not exceed 2048 characters')
    .required('Description is required'),
  precio: yup
    .number('Enter a price')
    .min(0, 'Price should be more than 0')
    .required('Price is required')
    .typeError('You must specify a number'),
  cantidad: yup
    .number('Enter a quantity')
    .min(1, 'Quantity should be more than 1')
    .required('Quantity is required')
    .integer('Quantity must be an integer')
    .typeError('You must specify a number'),
  foto: yup.mixed().required('A file is required')
});

const Form = () => {
  const [image, setImage] = useState(null);
  const [imagePreview, setImagePreview] = useState(null);

  const formik = useFormik({
    initialValues: {
      nombre: '',
      descripcion: '',
      precio: '',
      cantidad: '',
      foto: null,
    },
    validationSchema: validationSchema,
    onSubmit: (values, { setSubmitting }) => {
      const reader = new FileReader();
      reader.readAsDataURL(values.foto);
      reader.onloadend = () => {
        values.foto = reader.result;
        axios
          .post('https://api.example.com/items', values)
          .then((response) => {
            console.log(response.data);
            setSubmitting(false);
          })
          .catch((error) => {
            console.error(error);
            setSubmitting(false);
          });
      };
    }
  });

  return (
    <Formik {...formik} >
      <div>
        <TextField label="Name" name="nombre" />
        <TextField label="Description" name="descripcion" />
        <TextField label="Price" name="precio" />
        <TextField label="Quantity" name="cantidad" />
        <input type="file" name="foto" />
        <Button type="submit">Submit</Button>
      </div>
    </Formik>
  );
}

export default Form;
```

↻ Regenerate response » Continue generating

```
onSubmit: (values, { setSubmitting }) => {
  const reader = new FileReader();
  reader.readAsDataURL(values.foto);
  reader.onloadend = () => {
    values.foto = reader.result;
    axios
      .post('http://localhost:8080/productos', values)
      .then(response => {
        if (response.status === 200) {
          alert('Product created successfully!');
        }
        setSubmitting(false);
      })
      .catch(error => {
        if (error.response.status === 400) {
          alert('Please check the entered data');
        } else {
          alert('An internal error occurred');
        }
        setSubmitting(false);
      });
  };
},
));

const handleFileChange = (event) => {
  setImage(event.target.files[0]);
  setImagePreview(URL.createObjectURL(event.target.files[0]));
  formik.setFieldValue('foto', event.target.files[0]);
};

return (
<div>
  <form onSubmit={formik.handleSubmit}>
    <TextField
      fullWidth
      id="nombre"
      name="nombre"
      label="Nombre"
      value={formik.values.nombre}
      onChange={formik.handleChange}
      error={formik.touched.nombre && Boolean(formik.errors.nombre)}
      helperText={formik.touched.nombre && formik.errors.nombre}
    />
    <TextField
      fullWidth
      id="descripcion"
      name="descripcion"
      label="Descripcion"
      value={formik.values.descripcion}
      onChange={formik.handleChange}
      error={formik.touched.descripcion && Boolean(formik.errors.descripcion)}
    />
  </form>
</div>

```

```
>
<TextField
  fullWidth
  id="descripcion"
  name="descripcion"
  label="Descripcion"
  value={formik.values.descripcion}
  onChange={formik.handleChange}
  error={formik.touched.descripcion && Boolean(formik.errors.descripcion)}
  helperText={formik.touched.descripcion && formik.errors.descripcion}
/>
<TextField
  fullWidth
  id="precio"
  name="precio"
  label="Precio"
  value={formik.values.precio}
  onChange={formik.handleChange}
  error={formik.touched.precio && Boolean(formik.errors.precio)}
  helperText={formik.touched.precio && formik.errors.precio}
/>
<TextField
  fullWidth
  id="cantidad"
  name="cantidad"
  label="Cantidad"
  value={formik.values.cantidad}
  onChange={formik.handleChange}
  error={formik.touched.cantidad && Boolean(formik.errors.cantidad)}
  helperText={formik.touched.cantidad && formik.errors.cantidad}
/>
<input
  id="foto"
  name="foto"
  type="file"
  onChange={handleFileChange}
/>
{imagePreview && <img src={imagePreview} alt="preview" />}
<Button color="primary" variant="contained" fullWidth type="submit">
  Crear Producto
</Button>
</form>
</div>
);
};

export default Form;
```

This is a basic example and could be enhanced in many ways, for example by adding proper error handling and di

 Regenerate response

 Continue generating

```
        onChange={handleFileChange}
      />
      {imagePreview && <img src={imagePreview} alt="preview" />}
      <Button color="primary" variant="contained" fullWidth type="submit">
        Crear Producto
      </Button>
    </form>
  </div>
);
};

export default Form;
```

This is a basic example and could be enhanced in many ways, for example by adding proper error handling and displaying feedback to the user, using a more sophisticated component for image uploading, handling larger file uploads more efficiently, etc.

Also note that the way the image is handled here - by converting it to a base64 string - can result in very large strings if the images are large. Depending on the backend that you are using, there might be more efficient ways to handle image uploads, for example by uploading the image separately and just sending the URL or ID of the image with the form.

Lastly, this example assumes that the backend accepts JSON payloads and returns a status code of 200 for success, 400 for validation errors, and any other status code is considered an internal error. You would need to adjust this according to your actual backend implementation.

Fue así que decidí instalar Material UI y de aquí en adelante todo el desarrollo fue usando dicha librería. En términos del componente generado por ChatGPT, fue una buena pase para comenzar con el desarrollo, algunas cosas faltantes fueron:

- Que pudiera serializar la imagen en Base64.
- Que enviara bien los datos.
- Que usara la ruta completa.
- Que los componentes de Material UI estuvieran con configuración de espaciado correcta.

Para lograr esto, se utilizó GitHub Copilot dentro del IDE *Visual Studio Code*, con el cual se fueron solventando poco a poco los errores.

El código final del componente es el siguiente:

```
import React, { useState } from "react";
import { Button, TextField, Box, Container, Typography } from "@mui/material";
import { useFormik } from "formik";
import * as yup from "yup";
import axios from "axios";
import {
  CLIENT_ID_FUNCIONES_AZURE,
  CODIGO_FUNCIONES_AZURE,
  URL_FUNCIONES_AZURE,
} from "./Constantes";
import Articulo from "./Articulo";

// Esquema de validación para cada Artículo en el formulario.
const validationSchema = yup.object({
  nombre: yup
    .string()
    .max(64, "El nombre debe contener máximo 64 caracteres")
    .required("El nombre es requerido"),
  descripcion: yup
    .string()
    .max(512, "La descripción del artículo no debe tener más de 512 caracteres")
    .optional(),
  precio: yup
    .number()
    .min(0, "El precio debe ser mayor a 0.0")
    .required("El precio es requerido")
    .typeError("Debes introducir un número"),
  cantidad: yup
    .number()
    .min(1, "La cantidad de elementos en el almacén debe ser más de 1")
    .required("La cantidad es requerida")
    .integer("La cantidad debe ser un entero")
    .typeError("La cantidad debe ser un número"),
  fotografia: yup.mixed().optional(),
});

// Componente funcional que renderiza el formulario para agregar un nuevo artículo.
const AgregarArticulo: React.FC = () => {
  const [imagePreview, setImagePreview] = useState(null);

  // Función para convertir un archivo a base64.
  const convertirABase64 = (file: File): Promise<string> => {
    return new Promise((resolve, reject) => {
      const reader = new FileReader();
      reader.readAsDataURL(file);
      reader.onload = () => resolve(reader.result as string);
      reader.onerror = (error) => reject(error);
    });
  };
}
```

```
// Formik es una librería que nos permite manejar formularios de una manera más sencilla.
const formik = useFormik({
initialValues: {
id: 0,
nombre: "",
descripcion: "",
precio: 0.0,
cantidad: 1,
},
validationSchema: validationSchema,
// onSubmit es la función que se ejecuta cuando el usuario presiona el botón de submit.
onSubmit: async (values: Articulo, { setSubmitting }) => {
axios
.post(
`${URL_FUNCIONES_AZURE}/CrearArticulo?code=${CODIGO_FUNCIONES_AZURE}&client-id=${CLIENT_ID_FUNCIONES_AZURE}`,
values
)
.then((response) => {
console.log(response);
if (response.status === 201) {
alert(`Artículo ${values.nombre} agregado con éxito`);
}
setSubmitting(false);
})
.catch((error) => {
console.log(error);
if (error.code === "ECONNREFUSED") {
alert("No se pudo conectar con el servidor");
} else if (error.code === "ECONNABORTED") {
alert("Se agotó el tiempo de espera");
} else if (error.code === "ECONNRESET") {
alert("Se terminó la conexión con el servidor");
} else if (error.code === "ENOENT") {
alert("No se encontró el servidor");
} else if (error.code === "ERR_NETWORK") {
alert("Error de red");
} else if (error.response) {
if (error.response.status === 400) {
alert("Los datos ingresados no son válidos");
} else {
alert("Ocurrió un error al agregar el artículo");
}
}
setSubmitting(false);
});
},
});

const handleFileChange = async (event: any) => {
console.log("Archivo");
```

```
setImagePreview(URL.createObjectURL(event.target.files[0]) as any);
formik.setFieldValue(
"foto",
await convertirABase64(event.target.files[0])
);

};

return (
<Container>
<Box mt={4} mb={4}>
<Typography variant="h4" component="h1" gutterBottom>
Agregar un nuevo articulo
</Typography>
<Typography variant="body1" gutterBottom>
En esta sección podrás agregar un nuevo articulo al sistema.
</Typography>
<form onSubmit={formik.handleSubmit}>
<TextField
fullWidth
id="nombre"
name="nombre"
label="Nombre"
value={formik.values.nombre}
onChange={formik.handleChange}
error={formik.touched.nombre && Boolean(formik.errors.nombre)}
helperText={formik.touched.nombre && formik.errors.nombre}
sx={{ mb: 4 }}
/>
<TextField
fullWidth
id="descripcion"
name="descripcion"
label="Descripcion"
value={formik.values.descripcion}
onChange={formik.handleChange}
error={
formik.touched.descripcion && Boolean(formik.errors.descripcion)
}
helperText={formik.touched.descripcion && formik.errors.descripcion}
sx={{ mb: 4 }}
/>
<TextField
type="number"
fullWidth
id="precio"
name="precio"
label="Precio"
value={formik.values.precio}
onChange={formik.handleChange}
error={formik.touched.precio && Boolean(formik.errors.precio)}
helperText={formik.touched.precio && formik.errors.precio}
```

```
sx={{ mb: 4 }}  
/>  
<TextField  
  fullWidth  
  id="cantidad"  
  name="cantidad"  
  label="Cantidad"  
  value={formik.values.cantidad}  
  onChange={formik.handleChange}  
  error={formik.touched.cantidad && Boolean(formik.errors.cantidad)}  
  helperText={formik.touched.cantidad && formik.errors.cantidad}  
sx={{ mb: 4 }}  
/>  
<input  
  id="foto"  
  name="foto"  
  type="file"  
  onChange={handleFileChange}  
/>  
{imagePreview && (  
  <img src={imagePreview} alt="preview" width={200} height={200} />  
) }  
<Button color="primary" variant="outlined" fullWidth type="submit">  
  Agregar Artículo  
</Button>  
</form>  
</Box>  
<Box mb={12}>  
<Button href="/" variant="contained">  
  Volver a la página principal  
</Button>  
</Box>  
</Container>  
);  
};  
  
export default AgregarArticulo;
```

El código ya en ejecución visto desde mi computadora es el siguiente:



The screenshot shows a web page titled "Tarea 9 - Carrito de Compras". The main content is a form titled "Agregar un nuevo artículo". The form has four input fields: "Nombre" (Name), "Descripción" (Description), "Precio" (Price) with a value of "0", and "Cantidad" (Quantity) with a value of "1". Below the form is a file upload field labeled "Choose File" with the message "No file chosen". A blue button at the bottom right says "AGREGAR ARTÍCULO". At the bottom left is a blue button labeled "VOLVER A LA PÁGINA PRINCIPAL".

1.2.2.2. Rutas, encabezado y pié de página..

Como se puede observar, la página en cuestión ya tiene encabezado y pié de página, son dos componentes muy sencillos así que los veremos ambos junto con las rutas de la aplicación.

Esta configuración vive en **App.tsx** en la cual se observa el siguiente componente funcional:

```
const App: React.FC = () => {
  return (
    <React.Fragment>
      <AppBar position="relative">
        <Toolbar>
          <Typography variant="h6" component="div" sx={{ flexGrow: 1 }}>
            Tarea 9 - Carrito de Compras
          </Typography>
        </Toolbar>
      </AppBar>
      <Container>
        <Routes>
          <Route path="/" element={<Inicio />} />
          <Route path="/articulos" element={<Articulos />} />
          <Route path="/carrito" element={<CarritoDeCompras />} />
          <Route path="/agregar-articulo" element={<AgregarArticulo />} />
        </Routes>
      </Container>
      <Footer />
    </React.Fragment>
  );
};
```

En este componente suceden tres cosas muy importantes.

1. El encabezado es el componente de *AppBar* con *Toolbar* dentro de sí. Esto corresponde a la barra azul en la parte superior de todas las páginas.
2. Las rutas disponibles son 4:
 - a. */inicio*: página inicial, asociada a componente “Inicio”.
 - b. */articulos*: página de lista de artículos, asociada al componente “Articulos”.
 - c. */carrito*: página de listado de elementos en el carrito, asociada al componente “CarritoDeCompras”.
 - d. */agregar-articulo*: formulario para agregar un nuevo artículo, asociada al componente “AgregarArticulo”.
3. Renderiza el pié de página al final del contenido de cada componente asociado a una ruta, mediante el componente *Footer*.

El contenido de *Footer* fue generado a partir de la siguiente conversación con ChatGPT:

HU perfect! can you give me a footer component using material ui?

it only has to have my name, the current year, copyright mark, and my school id: 2016630495, add a cool emoji, up to you!

it will show in all pages so be creative

Sure, here's a simple example of a Footer component using Material-UI's `Container`, `Typography`, and `Box` components. For the emoji, I'll use a graduation cap 🎓 as it seems fitting with the school theme.

```
jsx
import React from 'react';
import { Container, Typography, Box } from '@mui/material';

const Footer = () => {
  const year = new Date().getFullYear();

  return (
    <Box sx={{ backgroundColor: '#f7f7f7', padding: 3, position: 'fixed', bottom: 0, width: '100%' }}>
      <Container maxWidth="sm">
        <Typography variant="body1" align="center">
          🎓 Your Name
        </Typography>
        <Typography variant="body2" color="text.secondary" align="center">
          {'Copyright © '}
          {year}
          {' '}
          {'|'}
          {'School ID: 2016630495'}
        </Typography>
      </Container>
    </Box>
  );
}

export default Footer;
```

This footer will be fixed at the bottom of the page and will have a grey background color. You can adjust these styles as needed. Also, don't forget to replace '`Your Name`' with your actual name.

El componente final de Footer es el siguiente:

```
const Footer: React.FC = () => {
  const year = new Date().getFullYear();

  return (
    <Box sx={{{
      bgcolor: "#f7f7f7",
      p: 1,
      position: "fixed",
      bottom: 0,
      width: "100%",
    }}>
    <Container>
      <Typography variant="body1" align="center">
        ⚡ Humberto Alejandro Ortega Alcocer
      </Typography>
      <Typography variant="body2" color="text.secondary" align="center">
        {"Copyright © "}
        {year} {" | "}
        {"Boleta: 2016630495 | Grupo: 4CV13"}
      </Typography>
    </Container>
  </Box>
);
```

Visualizándose así en todas las páginas:

💻 Tarea 9 - Carrito de Compras 📦

Carrito de Compras



Este es un carrito de compras simple basado en una arquitectura de backend serverless con Azure Functions y una base de datos MySQL.

Desarrollado por: Humberto Alejandro Ortega Alcocer

2016630495

Sistemas Distribuidos - 4CV13

[AGREGAR ARTÍCULO](#)

[VISUALIZAR ARTÍCULOS](#)

⚡ Humberto Alejandro Ortega Alcocer

Copyright © 2023 | Boleta: 2016630495 | Grupo: 4CV13

1.2.2.3. Página inicial.

Para desarrollar la página inicial se le preguntó a ChatGPT por un diseño simple y fácil de entender con los datos de identificación relevantes, de la siguiente forma:

HU hello! can you give me a react tsx for a simple index page for my homework? i am using material ui package and should be dead-simple:

- title: Tarea 9
- my name: Humberto Alejandro Ortega Alcocer
- subject: Sistemas Distribuidos - 4CV13

and two buttons:

- "Agregar Producto" should redirect to the /agregar-producto page
- "Visualizar Productos" should redirect to the /productos page

the design is up to you, but please use material ui package and keep it simple yet beautiful, thanks!



Sure, I'd be happy to help! Below is a simple React TSX code snippet for an index page with Material UI components. This uses the `Typography` component for the title, your name, and the subject. It also includes two `Button` components for the "Agregar Producto" and "Visualizar Productos" buttons, which use the `useHistory` hook from `react-router-dom` to navigate to the corresponding pages when clicked.

```
tsx
import React from 'react';
import { useHistory } from 'react-router-dom';
import { Typography, Button, Container, Box } from '@material-ui/core';

const HomePage: React.FC = () => {
  const history = useHistory();

  const handleAgregarProductoClick = () => {
    history.push('/agregar-producto');
  };

  const handleVisualizarProductosClick = () => {
    history.push('/productos');
  };

  return (
    <Container>
      <Box mt={4} mb={4}>
        <Typography variant="h4" component="h1" gutterBottom>
          Tarea 9
        </Typography>
        <Typography variant="h6" component="h2" gutterBottom>
          Humberto Alejandro Ortega Alcocer
        </Typography>
        <Typography>
```

```
<Typography variant="h4" component="h1" gutterBottom>
  Tarea 9
</Typography>
<Typography variant="h6" component="h2" gutterBottom>
  Humberto Alejandro Ortega Alcocer
</Typography>
<Typography variant="body1" component="p" gutterBottom>
  Sistemas Distribuidos - 4CV13
</Typography>
</Box>
<Box mt={4} mb={4}>
  <Button variant="contained" color="primary" onClick={handleAgregarPr
    Agregar Producto
  </Button>
</Box>
<Box mt={4} mb={4}>
  <Button variant="contained" color="primary" onClick={handleVisualiza
    Visualizar Productos
  </Button>
</Box>
</Container>
);
};

export default HomePage;
```

This is a quite simple yet neat design. You can adjust the styles as needed to fit your preferences. Please note that this example assumes you are using a router (like `react-router-dom`) in your application to handle navigation. If not, you will need to set this up.

Luego de modificaciones pertinentes para corregir detalles de diseño estéticos, el código final a partir de esa respuesta es:

```
const Inicio: React.FC = () => {
return (
<Container>
<Box mt={4} mb={4}>
<Typography variant="h4" component="h1" gutterBottom>
  Carrito de Compras
</Typography>
<Typography variant="body1" component="p" gutterBottom>
  Este es un carrito de compras simple basado en una arquitectura de
  backend serverless con Azure Functions y una base de datos MySQL.
</Typography>
<Typography variant="body1" component="p" gutterBottom>
  Desarrollado por: Humberto Alejandro Ortega Alcocer
</Typography>
```

```
<Typography variant="body1" component="p" gutterBottom>  
2016630495  
</Typography>  
<Typography variant="body1" component="p" gutterBottom>  
Sistemas Distribuidos - 4CV13  
</Typography>  
</Box>  
<Box mt={4} mb={4}>  
<Button href="/agregar-articulo" variant="contained" color="primary">  
Aregar Artículo  
</Button>  
</Box>  
<Box mt={4} mb={4}>  
<Button href="/articulos" variant="contained" color="primary">  
Visualizar Artículos  
</Button>  
</Box>  
</Container>  
) ;  
} ;
```

Y se observa en ejecución de la siguiente manera:

Tarea 9 - Carrito de Compras

Este es un carrito de compras simple basado en una arquitectura de backend serverless con Azure Functions y una base de datos MySQL.
Desarrollado por: Humberto Alejandro Ortega Alcocer
2016630495
Sistemas Distribuidos - 4CV13

[AGREGAR ARTÍCULO](#)

[VISUALIZAR ARTÍCULOS](#)

👉 Humberto Alejandro Ortega Alcocer
Copyright © 2023 | Boleta: 2016630495 | Grupo: 4CV13

1.2.2.4. Catálogo de artículos.

Para el catálogo de artículos se sostuvo la siguiente conversación con ChatGPT:

The screenshot shows a conversation between a user (HU) and ChatGPT. The user asks for a React TypeScript (tsx) file for a simple responsive grid layout to display products. The AI responds by providing a basic example of such a component, which includes importing React, useState, useEffect, and axios, and defining a function that fetches products from an endpoint and handles adding them to a cart with a POST request. The AI also includes error handling for a 400 status code.

```
tsx
import React, { useState, useEffect } from 'react';
import { Button, Grid, Typography, Card, CardContent, CardMedia, TextField } from '@mui/material';
import axios from 'axios';

const Productos = () => {
  const [productos, setProductos] = useState([]);

  useEffect(() => {
    axios.get('/productos')
      .then(response => setProductos(response.data))
      .catch(error => console.error('Error fetching products:', error));
  }, []);

  const handleAddToCarrito = (producto, cantidad) => {
    axios.post('/agregar_producto_a_carrito', { producto, cantidad })
      .then(response => alert('Product added to cart'))
      .catch(error => {
        if (error.response.status === 400) {
          alert(error.response.data);
        } else {
          console.error(error);
        }
      });
  };
}
```

Se puede observar muy claramente que equivocadamente usé la palabra “Producto” en lugar de “Artículo”, esto terminó involucrando una labor de “Buscar y reemplazar” los textos para que el componente se refiriera a Artículos y no a productos.

```
const handleAddToCarrito = (producto, cantidad) => {
  axios.post('/agregar_producto_a_carrito', { producto, cantidad })
    .then(response => alert('Product added to cart'))
    .catch(error => {
      if (error.response.status === 400) {
        alert(error.response.data);
      } else {
        console.error('Error adding product to cart:', error);
      }
    });
};

return (
  <Grid container spacing={2}>
    <Grid item xs={12}>
      <Typography variant="h2">Productos</Typography>
      <Button variant="contained" color="primary" href="/carrito">Go to Cart</Button>
    </Grid>

    {productos.map(producto => (
      <Grid item xs={12} sm={6} md={4} lg={3} key={producto.id}>
        <Card>
          <CardMedia
            component="img"
            height="140"
            image={producto.photo}
            alt={producto.name}
          />
          <CardContent>
            <Typography variant="h5" component="div">{producto.name}</Typography>
            <TextField type="number" defaultValue={1} />
            <Button variant="contained" color="primary" onClick={() => handleAddToCarrito(producto, cantidad)}>Add to Cart</Button>
          </CardContent>
        </Card>
      </Grid>
    )));
};

export default Productos;
```

This example includes a fetch to `'/productos'` on initial render using the `'useEffect'` hook, which sets the products to state. For each product, a Card component is rendered, which includes the product image, name, a quantity input, and a button that calls the `'handleAddToCarrito'` function, which sends a POST request to `'/agregar_producto_a_carrito'`. This function includes error handling that alerts the user in the case of a 400 response.

Please note that this is a very basic example and you will likely need to modify this to fit your exact needs. Furthermore, remember to replace the placeholders (like `'producto.id'`, `'producto.name'`, and `'producto.photo'`) with your actual data fields.

This code assumes that you've already installed and set up Material-UI, axios, and React in your project. If you haven't done that, you can install them using npm or yarn:

```
graphql
Copy code

npm install @mui/material @emotion
 Regenerate response
 Continue generating
```

Este componente provisto por ChatGPT fue de gran ayuda pues sólamente bastaron un par de adecuaciones para que funcionara y sirvió de base para crear la vista de carrito de compras también.

El funcionamiento en general del componente es el siguiente:

1. Al iniciarse llama al backend para obtener el listado de artículos.
2. Con este listado de artículos se renderiza la malla de artículos.
3. Se muestra un título y el campo para filtrar los artículos.
 - a. Si se comienza a escribir en este campo se volverá a llamar al backend para obtener el listado de artículos pero proporcionando el texto de usuario para que se use en la búsqueda en la Base de Datos.
4. Para cada artículo
 - a. Se muestra la fotografía.
 - b. Se muestra el nombre.
 - c. Un campo de cantidad.
 - d. Botón de “Añadir al carrito de compras”
 - e. Botón de “Ver más información”
 - i. Este deberá mostrar una ventana con la descripción y los datos completos del producto.
5. Se mostrará un botón de “Volver al menú principal” para regresar a la página de inicio.

Realizando las modificaciones apropiadas para que el componente reflejara este flujo de ejecución

El código final después de las modificaciones es:

```
const Articulos: React.FC = () => {
  // Variables que manejaremos con el estado de react.
  const [carritos, setCarritos]: [Carrito[], any] = useState([]);

  // Esta función convierte un arreglo de Articulos a un arreglo de Carritos.
  const articulosACarritos = (articulos: Articulo[], cantidad: number = 1) => {
    return articulos.map((articulo: Articulo) => {
      return {
        articulo_id: articulo.id,
        cantidad: cantidad,
        articulo: {
          id: articulo.id,
          nombre: articulo.nombre,
          descripcion: articulo.descripcion,
          precio: articulo.precio,
          cantidad: articulo.cantidad - cantidad,
          fotografia: articulo.fotografia,
        },
      };
    });
  };
}
```

```
// Esta función se ejecuta en cuanto se carga el componente, podría ser como un
// constructor, aquí hago el request para obtener los Artículos y formar
// objetos Carrito para cada uno a modo de controlar el flujo más fácil.
useEffect(() => {
  axios
    .get(
      `${URL_FUNCIONES_AZURE}/ObtenerArticulos?code=${CODIGO_FUNCIONES_AZURE}&client-id=${CLIENT_ID_FUNCIONES_AZURE}`
    )
    .then((response) => {
      // Actualizamos el estado de los carritos.
      setCarritos(articulosACarritos(response.data));
    })
    .catch((error) => console.error("Error fetching products:", error));
}, []);

/**
 * Función para mostrar más información de un artículo.
 */
const handleVerMasInformacion = (carrito: Carrito) => {
  alert(`Nombre: ${carrito?.articulo?.nombre}\nDescripción:
${carrito?.articulo?.descripcion}\nPrecio: ${carrito?.articulo?.precio}\nCantidad:
${carrito?.articulo?.cantidad}`);
};

// Función para buscar un artículo por nombre.
const handleBuscarArticulo = (event: React.ChangeEvent<HTMLInputElement>) => {
  event.preventDefault();
  const nombre = event.target.value;
  axios
    .get(
      `${URL_FUNCIONES_AZURE}/ObtenerArticulos?nombre=${nombre}&code=${CODIGO_FUNCIONES_AZURE}&client-id=${CLIENT_ID_FUNCIONES_AZURE}`
    )
    .then((response) => setCarritos(articulosACarritos(response.data)))
    .catch((error) => console.error("Error fetching products:", error));
};

const handleCantidadChange = (
  event: React.ChangeEvent<HTMLInputElement>,
  carrito: Carrito
) => {
  // Obtenemos la cantidad del input.
  const cantidad: number = parseInt(event.target.value);

  // Asignamos la cantidad seleccionada por el usuario.
  carrito.cantidad = cantidad;
```

```
// Actualizamos el estado de los carritos.  
const carritosActualizados = carritos.map((carritoActual: Carrito) => {  
  if (carritoActual.articulo_id === carrito.articulo_id) {  
    return carrito;  
  } else {  
    return carritoActual;  
  }  
});  
setCarritos(carritosActualizados);  
};  
  
// Función para añadir un articulo al carrito.  
const handleAddToCarrito = (carrito: Carrito) => {  
  console.log("Carrito a enviar", carrito);  
  if (!carrito.articulo) {  
    alert("No se encontró el articulo, intenta de nuevo.");  
  } else {  
    // Validamos que la cantidad esté disponible en el articulo.  
    if (carrito.cantidad <= carrito.articulo.cantidad) {  
      axios  
.post(`  
        ${URL_FUNCIONES_AZURE}/AgregarCarrito?code=${CODIGO_FUNCIONES_AZURE}&client-id=${CLIENT_ID}  
        ${FUNCIONES_AZURE}`,  
        {  
          articulo_id: carrito.articulo_id,  
          cantidad: carrito.cantidad,  
        }  
      )  
.then((response) => {  
        const articulo: Articulo = response.data;  
  
        // Enviamos un mensaje al usuario.  
        alert(`  
          Se añadieron ${carrito.cantidad} ${carrito?.articulo?.nombre}(s) al carrito (quedan  
          ${articulo.cantidad} disponibles).`  
      );  
  
      // Actualizamos el estado de los carritos.  
      const carritosActualizados = carritos.map(  
        (carritoActual: Carrito) => {  
          if (carritoActual.articulo_id === carrito.articulo_id) {  
            return {  
              articulo_id: carrito.articulo_id,  
              cantidad: 1,  
              articulo,  
            };  
          } else {  
            return carritoActual;  
          }  
        }  
      );  
    }  
  }  
};
```

```
setCarritos(carritosActualizados);
})
.catch((error) => {
if (error.response && error.response.status) {
if (error.response.status === 404) {
alert("El artículo no existe en la BD.");
}
if (error.response.status === 400) {
alert("No hay suficiente cantidad de este artículo.");
} else {
alert(
"El artículo ya se encontraba en el carrito, modifíquelo ahí."
);
}
} else {
alert("Ocurrió un error al añadir el artículo al carrito.");
console.error("Error añadiendo producto al carrito:", error);
}
});
} else {
alert(
`No hay suficiente cantidad de este artículo, el máximo es ${carrito.articulo.cantidad}`
);
}
}
};

return (
<Grid container spacing={2} mt={4} mb={12}>
<Grid item xs={12} sm={6}>
<Typography variant="h4">Artículos disponibles</Typography>
<Typography variant="body1" gutterBottom>
En esta sección podrás ver los artículos disponibles para comprar.
</Typography>
<Button variant="contained" color="primary" href="/carrito">
Ir al carrito de compras
</Button>
</Grid>
<Grid item xs={12} sm={6}>
<Typography variant="subtitle1">Búsqueda de artículos:</Typography>
<Input
fullWidth
onChange={handleBuscarArticulo}
placeholder={"Escribe el nombre del artículo a buscar..."}
/>
</Grid>

{carritos.map((carrito: Carrito) => (
<Grid
item
textAlign="center"
```

```
xs={12}
sm={6}
md={4}
lg={3}
key={(carrito?.articulo?.id)}
>
<Card>
<CardMedia
component="img"
height="140"
src={carrito?.articulo?.fotografia}
alt={carrito?.articulo?.nombre}
/>
<CardContent>
<Typography variant="h5" component="div">
{carrito?.articulo?.nombre} - ${carrito?.articulo?.precio} MXN
</Typography>
<Button
color="primary"
onClick={() => handleVerMasInformacion(carrito)}
>
Ver más información
</Button>
<TextField
fullWidth
type="number"
defaultValue={1}
inputProps={{
min: 1,
max: carrito?.articulo?.cantidad,
}}
onChange={(event: ChangeEvent<HTMLInputElement>) =>
handleCantidadChange(event, carrito)
}
/>
<hr />
<Button
variant="contained"
color="primary"
onClick={() =>
handleAddToCarrito(
carritos.find(
(carritoActual: Carrito) =>
carritoActual.articulo?.id === carrito.articulo?.id
) || carrito
)
}
>
Agregar al carrito
</Button>
</CardContent>
```

```
</Card>
</Grid>
) }

<Grid item xs={12}>
<Button variant="contained" color="primary" href="/">
Regresar a menú principal
</Button>
</Grid>
</Grid>
);
};
```

Se observa de la siguiente forma en ejecución:

The screenshot shows a user interface for a shopping cart. At the top, there's a blue header bar with the text "Tarea 9 - Carrito de Compras". Below the header, on the left, is a section titled "Artículos disponibles" with the sub-instruction "En esta sección podrás ver los artículos disponibles para comprar.". A blue button labeled "IR AL CARRITO DE COMPRAS" is visible. On the right, there's a search bar with the placeholder "Búsqueda de artículos:" and the instruction "Escribe el nombre del artículo a buscar...". The main content area displays a product card for "Práctico 1". The card includes a small thumbnail image, the product name "Práctico 1", its price "\$12.50 MXN", a "VER MÁS INFORMACIÓN" link, a quantity input field containing "1", and a blue "AGREGAR AL CARRITO" button. At the bottom of the page, there's a footer with the text "Humberto Alejandro Ortega Alcocer" and "Copyright © 2023 | Boleta: 2016630495 | Grupo: 4CV13".

1.2.2.5. Carrito de compras.

Para el carrito de compras ya no fue necesario consultar a ChatGPT puesto que en el desarrollo del componente anterior ya se tenía claridad sobre la API de Material UI para generar estos tipos de *layouts*. Para el carrito de compras, el flujo de ejecución es el siguiente:

1. Obtener el listado de artículos y cantidades de cada uno presentes en el carrito de compras desde el backend.
2. Para cada elemento del carrito de compras...
 - a. Mostrar su nombre.
 - b. Mostrar la cantidad de artículos en un campo de texto.
 - c. Mostrar el precio unitario.
 - d. Mostrar el precio total (cantidad x precio unitario).
 - e. Mostrar botón “Eliminar del carrito”.

3. Mostrar total (suma de los totales anteriores).
4. Mostrar botón de “Eliminar todos los artículos del carrito”.
5. Mostrar botón de “Volver a catálogo de artículos”.

El código resultante es el siguiente:

```
const CarritoDeCompras: React.FC = () => {
  const [carritos, setCarritos]: [Carrito[], any] = useState([]);

  useEffect(() => {
    axios
      .get(`/${URL_FUNCIONES_AZURE}/ObtenerCarrito?code=${CODIGO_FUNCIONES_AZURE}&client-id=${CLIENT_ID_FUNCIONES_AZURE}`)
      .then((response) => setCarritos(response.data))
      .catch((error) => console.error("Error fetching products:", error));
  }, []);

  // Función para cuando el usuario modifica la cantidad de un artículo en el carrito.
  const handleModificarCantidad = async (carrito: Carrito, cantidad: number) => {
    try {
      // Actualizamos la cantidad del artículo en el carrito.
      await axios.post(`/${URL_FUNCIONES_AZURE}/EditarCarrito?code=${CODIGO_FUNCIONES_AZURE}&client-id=${CLIENT_ID_FUNCIONES_AZURE}`,
        {
          articulo_id: carrito.articulo_id,
          cantidad,
        }
      );

      // Actualizamos el estado de los carritos.
      setCarritos(
        await axios
          .get(`/${URL_FUNCIONES_AZURE}/ObtenerCarrito?code=${CODIGO_FUNCIONES_AZURE}&client-id=${CLIENT_ID_FUNCIONES_AZURE}`)
          .then((response) => response.data)
        );
    } catch (error) {
      console.error("Error fetching products:", error);
    }
  };

  // Función para cuando el usuario da click en el botón de eliminar un artículo del carrito.
  const handleEliminarArticulo = (carrito: Carrito,
```

```
preconfirmacion: boolean = false
) => {
// Confirmación del usuario.
const confirmacionUsuario: boolean =
preconfirmacion ||
window.confirm(
"¿Estás seguro de que quieres eliminar este artículo del carrito?" +
"\n" +
carrito?.articulo?.nombre
);

if (confirmacionUsuario) {
// Eliminamos el artículo del carrito.
axios
.post(
`${URL_FUNCIONES_AZURE}/EliminarCarrito?code=${CODIGO_FUNCIONES_AZURE}&client-id=${CLIENT_ID_FUNCIONES_AZURE}`,
{
articulo_id: carrito.articulo_id,
}
)
.then(() =>
// Actualizamos el estado de los carritos.
axios
.get(
`${URL_FUNCIONES_AZURE}/ObtenerCarrito?code=${CODIGO_FUNCIONES_AZURE}&client-id=${CLIENT_ID_FUNCIONES_AZURE}`
)
.then((response) => setCarritos(response.data))
)
.catch((error) => console.error("Error fetching products:", error));
}
};

// Función para cuando el usuario da click en el botón de eliminar todos los artículos del carrito.
const handleEliminarCarrito = async () => {
// Confirmación del usuario.
const confirmacionUsuario: boolean = window.confirm(
"¿Estás seguro de que quieres eliminar todos los artículos del carrito?"
);

if (confirmacionUsuario) {
carritos.forEach((carrito: Carrito) => {
handleEliminarArticulo(carrito, true);
});
}
};

return (
<Grid container spacing={2} mt={4} mb={12}>
```

```
<Grid item xs={12}>
<Typography variant="h4" component="h1" gutterBottom>
Carrito de compras
</Typography>
<Typography variant="body1" gutterBottom>
En esta sección podrás ver los artículos que has añadido al carrito de compras, la cantidad y el precio de cada uno.
</Typography>
</Grid>
{carritos.map((carrito: Carrito) => (
<Grid item textAlign="right" xs={12} key={carrito.articulo_id}>
<Card>
<CardMedia
component="img"
height="140"
src={carrito?.articulo?.fotografia}
alt={carrito?.articulo?.nombre}
/>
<CardHeader title={carrito?.articulo?.nombre || "Desconocido"} />
<CardContent>
<Typography variant="body1" gutterBottom>
Cantidad:{" "}
<TextField
type="number"
variant="standard"
inputProps={{ min: 1, max: carrito?.articulo?.cantidad || 0 }}
value={carrito.cantidad}
onChange={(event) => {
handleModificarCantidad(
carrito,
parseInt(event.target.value)
);
}}
/>
</Typography>
<Typography variant="body1" gutterBottom>
Precio unitario: ${carrito?.articulo?.precio || 0} MXN
</Typography>
<Typography variant="body1" gutterBottom>
Total: ${(carrito?.articulo?.precio || 0) * carrito.cantidad}{ " "}
MXN
</Typography>
<Button
onClick={() => handleEliminarArticulo(carrito)}
color="warning"
variant="contained"
>
Eliminar artículo del carrito
</Button>
</CardContent>
</Card>
```

```
</Grid>
))

<Grid item textAlign="center" xs={12}>
<Typography variant="h5" component="h2" gutterBottom>
Total: $ {carritos
.reduce((a, b) => a + b.cantidad * (b?.articulo?.precio || 0), 0)
.toLocaleString("es-MX") } " " }
MXN(" ")
<Button
variant="contained"
color="error"
onClick={handleEliminarCarrito}
>
Vaciar carrito de compras.
</Button>
</Typography>
<Button variant="contained" color="primary" href="/articulos">
Volver a listado de artículos
</Button>
</Grid>
</Grid>
);
};
```

Y se ve así en ejecución:

Tarea 9 - Carrito de Compras

Carrito de compras

En esta sección podrás ver los artículos que has añadido al carrito de compras, la cantidad y el precio de cada uno.

Artículo	Cantidad	Precio unitario	Total
Prueba1	1	\$12.50 MXN	\$12.5 MXN

Total: \$12.5 MXN [VACIAR CARRITO DE COMPRAS](#)

[VOLVER A LISTADO DE ARTÍCULOS](#)

1.2.2.6. Constantes y variables de entorno.

Como puede observarse en varias partes del código, cree un archivo llamado **Constantes.ts** en el que definí algunas constantes de operación importantes para la aplicación.

El contenido de dicho archivo es el siguiente:

```
// URL de la API de Artículos.  
export const URL_FUNCIONES_AZURE: string = process.env.URL_FUNCIONES_AZURE ||  
"https://t9-af-2016630495.azurewebsites.net/api" || "http://localhost:7071/api";  
  
// Código y Client ID de la API de Artículos.  
export const CODIGO_FUNCIONES_AZURE: string = process.env.CODIGO_FUNCIONES_AZURE || "";  
export const CLIENT_ID_FUNCIONES_AZURE: string = process.env.CLIENT_ID_FUNCIONES_AZURE ||  
"default";
```

Así puedo proveer del código de Azure mediante una variable de entorno y dinámicamente cambiar todos sus usos en los distintos componentes.

1.3. Despliegue de aplicación.

Para desplegar la aplicación debemos asegurarnos que cada uno de los elementos tenga la configuración inicial válida requerida por el software, comenzaremos por configurar la base de datos, luego haremos el backend y finalmente el frontend.

1.3.1. Despliegue de base de datos.

Lo primero será crear una base de datos (lógica) nueva en nuestra instancia de MySQL en Azure, para esto iremos a la página inicial de la instancia:

The screenshot shows the Azure portal interface for managing a MySQL database. The left sidebar lists various service categories like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Learning center, Databases, Connect, Server parameters, Replication, Maintenance, High availability, Backup and restore, Advisor recommendations, Locks, Power Platform, Power BI, Identity, Data encryption, Authentication, Monitoring, Alerts, and Metrics. The main content area is titled 't9-bd-2016630495' and shows the 'Overview' tab selected. It displays basic information about the database server, including its name (t9-bd-2016630495.mysql.database.azure.com), location (East US), and creation date (2023-06-10). A 'Getting started' section provides links for Learn, Allow access, Connect, and Samples. Below this, sections for 'Configure your database' (Maintenance schedule, Server parameters, Backup and restore), 'Get ready for production use' (Enable Replication and high availability), and 'Samples' are visible. The top right corner shows the user's name (horteaga1500@alumno.ipn.mx) and the IP address (10.0.0.11).

Haremos click en el apartado *Databases* en el menú izquierdo y observaremos que solo hay bases de datos default:

The screenshot shows the Microsoft Azure portal interface for a MySQL flexible server named "t9-bd-2016630495". The left sidebar is open, showing various service links like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Learning center, and Settings. Under the Databases section, there is a list of existing databases:

Name	Character set	Collation	Schema type
mysql	utf8mb4	utf8mb4_0900...	System
information_schema	utf8mb3	utf8mb3_gener...	System
performance_schema	utf8mb4	utf8mb4_0900...	System
sys	utf8mb4	utf8mb4_0900...	System

Haremos click en *+Add* y se nos mostrará un formulario en el cual crearemos nuestra base de datos con el nombre “tarea9”:

The screenshot shows the same Microsoft Azure MySQL flexible server interface. This time, the "Create Database" dialog box is open over the list of databases. The dialog has fields for Name (set to "tarea9"), Character set (set to "utf8"), and Collation (set to "utf8_general_ci").

Una vez creada se mostrará en el listado:

The screenshot shows the Azure portal interface for a MySQL flexible server. On the left, there's a sidebar with various management options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Learning center, Compute + storage, Networking, and Databases. The 'Databases' section is selected. In the main content area, it says 'You can create, view and delete MySQL databases on this server. Note that you cannot delete any system schemas such as mysql, sys, information_schema, performance_schema. You can connect to the database using MySQL client or Power BI'. Below this, a table lists existing databases:

Name	Character set	Collation	Schema type
mysql	utf8mb4	utf8mb4_0900...	System
information_schema	utf8mb3	utf8mb3_gener...	System
performance_schema	utf8mb4	utf8mb4_0900...	System
sys	utf8mb4	utf8mb4_0900...	System
tarea9	utf8mb3	utf8mb3_gener...	User

To the right, there's a 'Notifications' panel with a message: 'Successfully created MySQL database.' and 'Successfully created "tarea9" MySQL database.' A timestamp indicates it was 'a few seconds ago'.

También podremos observarla desde DBeaver:

The screenshot shows the DBeaver interface with a tree view of the database structure. The root node is 't9-bd-2016630495.mysql.database.azure.com - t9-bd-2016630495'. Under it, there's a 'Databases' node which has a child node 'sys'. Below 'sys' is another node 'tarea9', which is highlighted with a gray background. Other nodes under 'Databases' include 'Users', 'Administer', and 'System Info'. There are also collapsed nodes for 'Tables', 'Views', 'Procedures', and 'Triggers'.

Para desplegar la base de datos, bastará con aplicar nuestro *script* de inicialización:

```
-- Sistemas Distribuidos - Tarea 9
-- Script para generar la base de datos requerida por la tarea.
-- Humberto Alejandro Ortega Alcocer
-- 2016630495

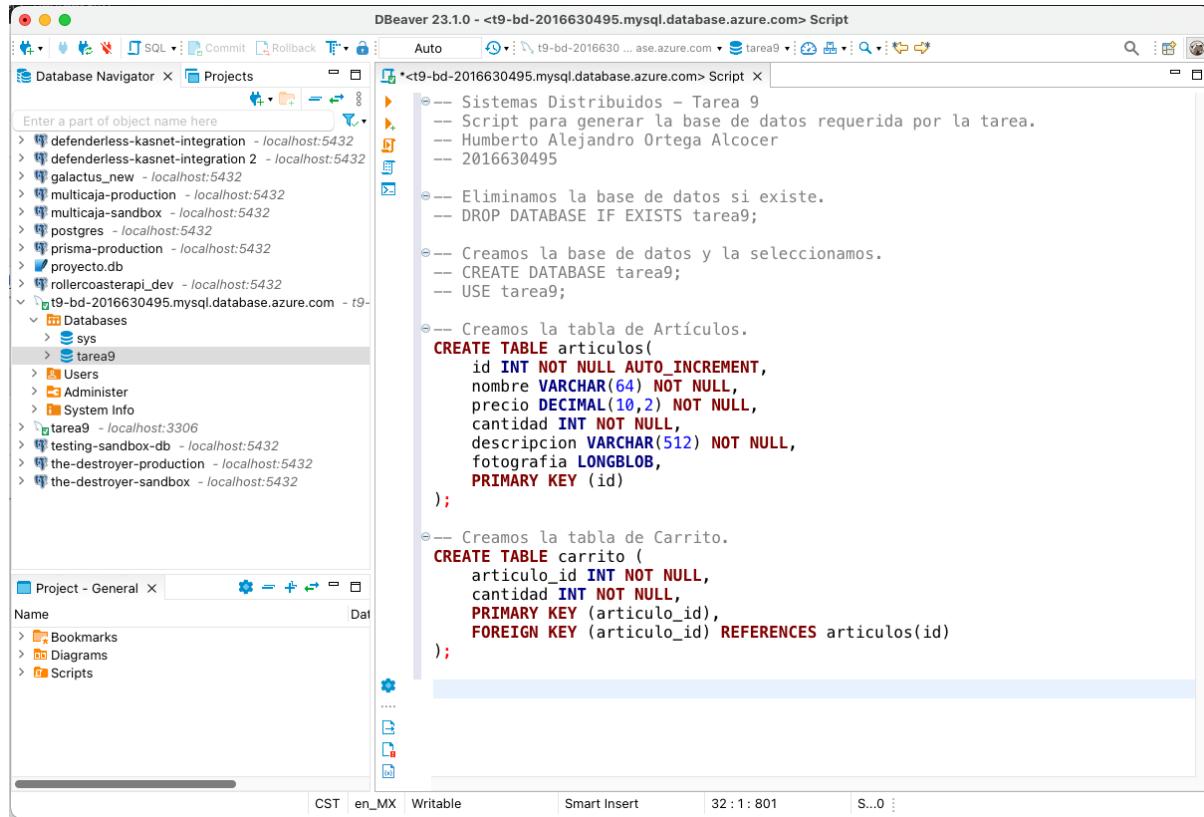
-- Eliminamos la base de datos si existe.
DROP DATABASE IF EXISTS tarea9;

-- Creamos la base de datos y la seleccionamos.
CREATE DATABASE tarea9;
USE tarea9;

-- Creamos la tabla de Artículos.
CREATE TABLE articulos(
id INT NOT NULL AUTO_INCREMENT,
nombre VARCHAR(64) NOT NULL,
precio DECIMAL(10,2) NOT NULL,
cantidad INT NOT NULL,
descripcion VARCHAR(512) NOT NULL,
```

```
fotografia LONGBLOB,  
PRIMARY KEY (id)  
);  
  
-- Creamos la tabla de Carrito.  
CREATE TABLE carrito (  
articulo_id INT NOT NULL,  
cantidad INT NOT NULL,  
PRIMARY KEY (articulo_id),  
FOREIGN KEY (articulo_id) REFERENCES articulos(id)  
);
```

Este lo copiaremos directamente en DBeaver conectados a la instancia de MySQL en Azure y habiendo seleccionado la base de datos correspondiente a la tarea, comentando la sección de creación y selección de la base de datos pues DBeaver ya nos probó eso implícitamente:



Y lo ejecutaremos:

```

-- -- Creamos la tabla de Artículos.
CREATE TABLE articulos(
    id INT NOT NULL AUTO_INCREMENT,
    nombre VARCHAR(64) NOT NULL,
    precio DECIMAL(10,2) NOT NULL,
    cantidad INT NOT NULL,
    descripcion VARCHAR(512) NOT NULL,
    fotografía LONGBLOB,
    PRIMARY KEY (id)
);

-- -- Creamos la tabla de Carrito.
CREATE TABLE carro (
    articulo_id INT NOT NULL,
    cantidad INT NOT NULL,
    PRIMARY KEY (articulo_id),
    FOREIGN KEY (articulo_id) REFERENCES articulos(id)
);

```

Como se puede observar, las tablas existen. Ahora verificaremos la estructura de cada una. Esta es la estructura de un Artículo:

Column Name	# Data Type	Not Null	Auto Increment	Key
1 st id	1 int	[v]		P
abc nombre	2 varchar(64)	[v]		
123 precio	3 decimal(10,2)	[v]		
123 cantidad	4 int	[v]		
abc descripcion	5 varchar(512)	[v]		
fotografia	6 longblob	[]		

Y esta es la estructura de la tabla Carrito:

DBeaver 23.1.0 - carrito

Table Name: carrito

Engine: InnoDB

Auto Increment: 0

Charset: utf8mb3

Collation: utf8mb3_general_ci

Columns

Column Name	# Data Type	Not Null	Auto Increment	Key	Default
121 articulo_id	1 int	[v]	[]	PRI	
123 cantidad	2 int	[v]	[]		

El diagrama Entidad-Relación generado por DBeaver es el siguiente:

DBeaver 23.1.0 - tarea9

carrito

articulos

Properties

ER Diagram

```

    graph LR
        carrito[carrito] -- "121 articulo_id" --> articulos[articulos]
        articulos[id]
    
```

Lo cual es correcto. Con esto hemos finalizado la configuración de nuestro servidor de bases de datos.

1.3.2. Despliegue de backend.

Para desplegar el backend, primero deberemos instalar las herramientas de línea de comandos de Azure:

```
brew install azure-cli
→ Downloading https://formulae.brew.sh/api/formula.azure.json
#####
#### 100.0%
→ Downloading https://formulae.brew.sh/api/cask.azure.json
#####
#### 100.0%
Warning: azure-cli 2.49.0 is already installed and up-to-date.
To reinstall 2.49.0, run:
  brew reinstall azure-cli
```

Con esto instalado, iremos a nuestra aplicación de funciones en Azure:

The screenshot shows the Microsoft Azure portal's Overview page for a Function App named "T9-AF-2016630495". The left sidebar contains navigation links for Home, Function App, Overview, Functions, Deployment, Settings, and more. The main content area shows the following details:

- Resource group:** SistemasDistribuidosResourceGroup
- Status:** Running
- Location:** East US
- Subscription:** Azure for Students
- Subscription ID:** 45fd6f75-3ae1-4e2b-95b5-02150d67b8d2
- Tags:** Click here to add tags

Below this, there are three metric charts:

- Memory working set:** Shows usage from 0B to 100B over time from 5 PM UTC-06:00 to 5:15 PM UTC-06:00. Value: 0B.
- Function Execution Count:** Shows execution counts from 0 to 100 over the same time period. Value: 0.
- MB Milliseconds:** Shows execution times in milliseconds from 0 to 100 over the same time period. Value: 0.

Iremos a la sección de *Functions*:

The screenshot shows the Microsoft Azure portal's Functions section for the same Function App ("T9-AF-2016630495"). The left sidebar is identical to the previous screenshot. The main content area shows the following interface:

- A search bar at the top.
- A table header with columns: Name, Trigger, Status, and Monitor.
- A message: "No results."

Seleccionaremos la opción + Create y en el formulario que aparece del lado derecho seleccionaremos la opción Any editor + Core Tools:

The screenshot shows the Microsoft Azure Functions portal interface. On the left, there's a sidebar with various options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), Functions (selected), App keys, App files, Proxies, Deployment (Deployment slots, Deployment Center), Settings (Configuration, Authentication, Application Insights, Identity, Backups, Custom domains, Certificates, Networking, Scale up (App Service plan), Scale out, Locks), and App Service plan. The main area shows a list of functions under 'T9-AF-2016630495 | Functions' with a search bar and buttons for '+ Create', 'Refresh', and 'Delete'. On the right, a modal window titled 'Create function' is open. It has sections for 'Select development environment' (with 'Any editor + Core Tools' selected), 'Install dependencies' (with a command line input field containing 'npm install -g azure-functions-core-tools@4 --unsafe-perm true'), 'Create an Azure Functions project' (with a command line input field containing 'func init'), 'Create a function' (with a command line input field containing 'func new'), 'Run your function project locally' (with a command line input field containing 'func start'), and 'Deploy your code to Azure' (with a command line input field containing 'func azure functionapp publish T9-AF-2016630495'). A 'Close' button is at the bottom right of the modal.

Ahí se describen los pasos para iniciar desde cero, al hacer scroll observaremos la opción Deploy your code to Azure:

This screenshot is identical to the one above, showing the 'Create function' dialog in the Microsoft Azure Functions portal. The 'Development environ...' dropdown is still set to 'Any editor + Core Tools'. The right side of the screen displays the same instructions for installing Core Tools and creating an Azure Functions project using the 'func' command. At the very bottom of the right panel, the 'Deploy your code to Azure' section is visible, containing the command 'func azure functionapp publish T9-AF-2016630495'.

Antes de ejecutar este comando, tendremos que iniciar sesión en la herramienta de línea de comandos de Azure, de la siguiente forma:

```
> az login
A web browser has been opened at https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize. Please continue the login in the web browser. If no web browser is available or if the web browser fails to open, use device code flow with 'az login --use-device-code'.
[{"cloudName": "AzureCloud",
"homeTenantId": "REDACTED",
"id": "REDACTED",
"isDefault": true,
"managedByTenants": [],
"name": "Azure for Students",
"state": "Enabled",
"tenantId": "REDACTED",
"user": {
"name": "hortegaa1500@alumno.ipn.mx",
"type": "user"
}
}]
```

Utilicé el color rojo para tapar la información sensible ya que esta es una autorización a mi cuenta de alumno.

Realizado esto, ahora sí ejecutaremos el comando presente en la página de las funciones:

```
> func azure functionapp publish T9-AF-2016630495
Getting site publishing info ...
Uploading package ...
Uploading 4.15 MB [#####
Upload completed successfully.
Deployment completed successfully.
Syncing triggers ...
Functions in T9-AF-2016630495:
    AgregarCarrito - [httpTrigger]
        Invoke url: https://t9-af-2016630495.azurewebsites.net/api/agregarcarrito

    CrearArticulo - [httpTrigger]
        Invoke url: https://t9-af-2016630495.azurewebsites.net/api/creararticulo

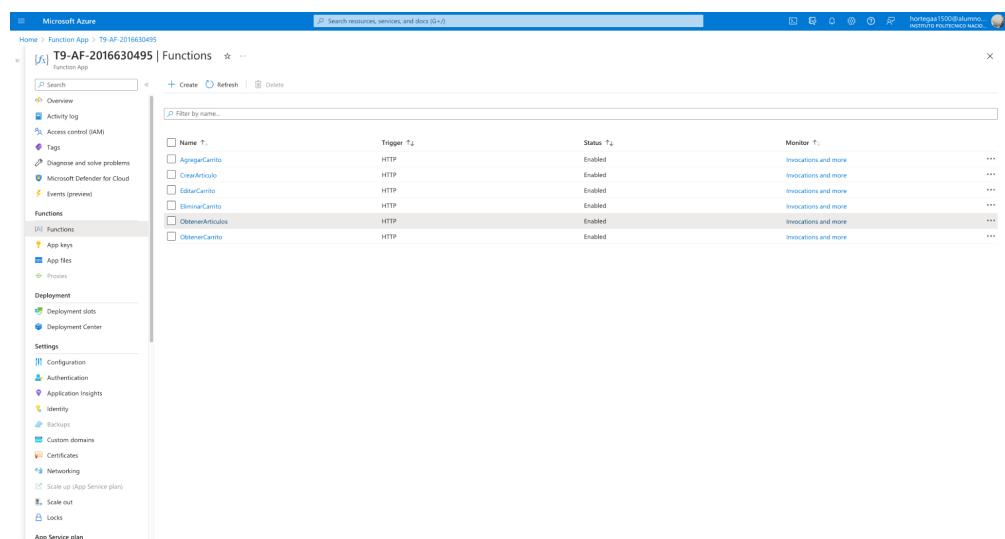
    EditarCarrito - [httpTrigger]
        Invoke url: https://t9-af-2016630495.azurewebsites.net/api/editarcarrito

    EliminarCarrito - [httpTrigger]
        Invoke url: https://t9-af-2016630495.azurewebsites.net/api/eliminarcarrito

    ObtenerArticulos - [httpTrigger]
        Invoke url: https://t9-af-2016630495.azurewebsites.net/api/obtenerarticulos

    ObtenerCarrito - [httpTrigger]
        Invoke url: https://t9-af-2016630495.azurewebsites.net/api/obtenercarrito
```

Ahora si hacemos click en *Refresh* en la página de Azure veremos nuestras funciones:



Para realizar una prueba, usaremos la función ObtenerArticulos para garantizar que funciona correctamente:

The screenshot shows the Azure Functions Overview page for the 'ObtenerArticulos' function. Key details include:

- Resource group:** SistemasDistribuidosResourceGroup
- Location:** East US
- Subscription:** Azure for Students
- Subscription ID:** a1d9d675-3ae1-4e2b-95b5-02150d67b8d2
- Function app:** T9-AF-2016630495
- Tags:** Click here to add tags

Execution metrics for the last 24 hours:

Metric	Value
Total Execution Count	0
Successful Execution Count	0
Failed Execution Count	0
Http 2xx	0
Http 4xx	0
Http 5xx	0

Daremos click sobre *Get Function Url* con la cual se nos generará una URL que ya cuenta con un código de autenticación provisto por Azure para acceder a la misma:

The screenshot shows the Azure Functions Overview page for the 'ObtenerArticulos' function. Key details include:

- Resource group:** SistemasDistribuidosResourceGroup
- Location:** East US
- Subscription:** Azure for Students
- Subscription ID:** a1d9d675-3ae1-4e2b-95b5-02150d67b8d2
- Function app:** T9-AF-2016630495
- Tags:** Click here to add tags

Execution metrics for the last 24 hours:

Metric	Value
Total Execution Count	0
Successful Execution Count	0
Failed Execution Count	0
Http 2xx	0
Http 4xx	0
Http 5xx	0

Usando la URL en mi navegador puedo observar que me responde con un artículo de prueba que inserté en la base de datos mientras hacía pruebas:



```
[{"id": 1,  
 "nombre": "Prueba1",  
 "descripcion": "Esta es una prueba",  
 "precio": "12.50",  
 "cantidad": 5,  
 "fotografia": "data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAAQABAAAD/2wCEAAAYGBgYHBgcICAcKCwoLCg8ODAwODxYQERAREBYiFRKVFRkVlh4kHhw"}]
```

Es importante notar que todas las URLs que serán llamadas por el frontend deberán contar con la llave correspondiente para poder funcionar.

1.3.3. Despliegue de frontend.

Para desplegar nuestro frontend en React, lo primero será ejecutar el comando **npm run build** el cual genera, en la carpeta **dist** un *minificado*, es decir, una versión de la aplicación que consiste solamente en HTML, CSS y JS. Ejecutar este comando se ve de la siguiente manera:

```
> npm run build  
  
> frontend-tarea-9@0.1.0 build  
> react-scripts build  
  
Creating an optimized production build ...  
Compiled successfully.  
  
File sizes after gzip:  
  
 148.97 kB  build/static/js/main.4b6a4f03.js  
  1.78 kB    build/static/js/787.48dceaf5.chunk.js  
   264 B     build/static/css/main.e6c13ad2.css  
  
The project was built assuming it is hosted at /.  
You can control this with the homepage field in your package.json.  
  
The build folder is ready to be deployed.  
You may serve it with a static server:  
  
  npm install -g serve  
  serve -s build  
  
Find out more about deployment here:  
  
  https://cra.link/deployment
```

Los archivos generados son los siguientes:

```
> tree build
build
├── asset-manifest.json
├── favicon.ico
├── index.html
├── logo192.png
├── logo512.png
└── manifest.json
    └── robots.txt
        └── static
            ├── css
            │   └── main.e6c13ad2.css
            │   └── main.e6c13ad2.css.map
            └── js
                └── 787.48dceaf5.chunk.js
                └── 787.48dceaf5.chunk.js.map
                └── main.4b6a4f03.js
                └── main.4b6a4f03.js.LICENSE.txt
                └── main.4b6a4f03.js.map

4 directories, 14 files
```

Ahora copiaremos estos archivos en nuestra máquina virtual usando el comando de **scp**:

```
> scp -i ..../../.azure/humbertowoodys-azure.pem -r ./build/* 20.231.41.187:/home/humbertowoodys/sitio/
asset-manifest.json
favicon.ico
index.html
logo192.png
logo512.png
manifest.json
robots.txt
main.e6c13ad2.css.map
main.e6c13ad2.css
main.4b6a4f03.js.map
main.4b6a4f03.js.LICENSE.txt
787.48dceaf5.chunk.js.map
787.48dceaf5.chunk.js
main.4b6a4f03.js
```

100%	517	6.1KB/s	00:00
100%	3870	45.7KB/s	00:00
100%	818	9.6KB/s	00:00
100%	5347	60.4KB/s	00:00
100%	9664	114.8KB/s	00:00
100%	492	5.9KB/s	00:00
100%	67	0.8KB/s	00:00
100%	584	6.8KB/s	00:00
100%	337	3.9KB/s	00:00
100%	2169KB	3.3MB/s	00:00
100%	2245	26.3KB/s	00:00
100%	10KB	129.3KB/s	00:00
100%	4613	54.8KB/s	00:00
100%	471KB	1.8MB/s	00:00

Y, en la máquina virtual verificaremos haber recibido todos los archivos:

```
humbertowoodys@T9-FE-2016630495:~$ tree sitio
sitio
├── asset-manifest.json
├── favicon.ico
├── index.html
├── logo192.png
└── logo512.png
    └── manifest.json
    └── robots.txt
        └── static
            ├── css
            │   └── main.e6c13ad2.css
            │   └── main.e6c13ad2.css.map
            └── js
                └── 787.48dceaf5.chunk.js
                └── 787.48dceaf5.chunk.js.map
                └── main.4b6a4f03.js
                └── main.4b6a4f03.js.LICENSE.txt
                └── main.4b6a4f03.js.map

3 directories, 14 files
```

1.3.3.1. Instalando NGINX

El servidor web elegido para desplegar en nuestra máquina virtual fue nginx porque es el que uso en mi trabajo y me es muy fácil de manipular, es un servidor muy similar a Apache.

Para instalarlo, primero actualizaremos los repositorios locales y luego instalaremos usando el comando **sudo apt install nginx**:

```
humberto@T9-FE-2016630495:~$ sudo apt update && sudo apt install nginx
[75/1909]
Hit:1 http://azure.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://azure.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:4 http://azure.archive.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:5 http://azure.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [2627 kB]
Get:6 http://azure.archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [1070 kB]
Fetched 4032 kB in 1s (5005 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
42 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  fontconfig fonts-dejavu-core libfontconfig libgd3 libjbig0 libjpeg-turbo8 libjpeg8 libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter libnginx-mod-mail libnginx-mod-stream
  libtiff5 libwebp libxml2 nginx nginx-common nginx-core
Suggested packages:
  libgd-tools fcgiwrap nginx-doc ssl-cert
The following NEW packages will be installed:
  fontconfig fonts-dejavu-core libfontconfig libgd3 libjbig0 libjpeg-turbo8 libjpeg8 libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter libnginx-mod-mail libnginx-mod-stream
  libtiff5 libwebp libxml2 nginx nginx-common nginx-core
0 upgraded, 17 newly installed, 0 to remove and 42 not upgraded.
Need to get 2436 kB of archives.
After this operation, 7919 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://azure.archive.ubuntu.com/ubuntu focal/main amd64 fonts-dejavu-core all 2.37-1 [1041 kB]
Get:2 http://azure.archive.ubuntu.com/ubuntu focal/main amd64 fontconfig-config all 2.13.1-2ubuntu3 [28.8 kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu focal-updates/main amd64 libfontconfig1 amd64 2.13.1-2ubuntu3 [114 kB]
Get:4 http://azure.archive.ubuntu.com/ubuntu focal-updates/main amd64 libjpeg-turbo8 amd64 2.0.3-0ubuntu1.20.04.3 [118 kB]
```

Al finalizar la instalación trataremos de mostrar el estado del servicio de nginx:

```
humberto@T9-FE-2016630495:~$ systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2023-06-11 01:48:02 UTC; 30s ago
       Docs: man:nginx(8)
   Main PID: 3605 (nginx)
      Tasks: 3 (limit: 4699)
     Memory: 5.6M
        CGroup: /system.slice/nginx.service
                  └─3605 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
                     ├─3606 nginx: worker process
                     ├─3607 nginx: worker process
                     ├─3608 nginx: worker process

Jun 11 01:48:02 T9-FE-2016630495 systemd[1]: Starting A high performance web server and a reverse proxy server...
Jun 11 01:48:02 T9-FE-2016630495 systemd[1]: Started A high performance web server and a reverse proxy server.
humberto@T9-FE-2016630495:~$
```

1.3.3.2. Configurando NGINX.

Lo primero será permitir el acceso completo a nginx en nuestro cortafuegos del sistema, con el siguiente comando:

```
humberto@T9-FE-2016630495:~$ sudo ufw allow 'Nginx Full'  
Rules updated  
Rules updated (v6)
```

Luego crearemos un archivo de configuración para nuestra aplicación, la llamaremos “tarea9” y lo abriremos para edición con Vim:

```
humberto@T9-FE-2016630495:~$ sudo touch /etc/nginx/sites-available/tarea9  
humberto@T9-FE-2016630495:~$ sudo vim /etc/nginx/sites-available/tarea9
```

Dado que queremos que nuestro sitio sea alcanzable por IP pública únicamente (no usaremos DNS), en la configuración usaremos la IP pública como el nombre del servidor, y apuntaremos la ruta inicial a nuestra carpeta dónde acabamos de copiar los contenidos de nuestro build, de igual forma, se tratará de resolver mediante URI las peticiones pero, en caso de no encontrar ningún match, usará nuestro **index.html**.

Ahora colocaremos la configuración de nginx correspondiente:

```
server {  
    listen 80;  
    server_name 20.231.41.187;  
  
    location / {  
        root /home/humbertwoody/sitio;  
        try_files $uri /index.html;  
    }  
}
```

Ahora crearemos un enlace simbólico a la carpeta **/etc/nginx/sites-enabled**, a modo de que nginx pueda reconocer la configuración y la aplique:

```
humberto@T9-FE-2016630495:~$ sudo ln -s /etc/nginx/sites-available/tarea9 /etc/nginx/sites-enabled/  
humberto@T9-FE-2016630495:~$ ls /etc/nginx/sites-enabled/  
default tarea9
```

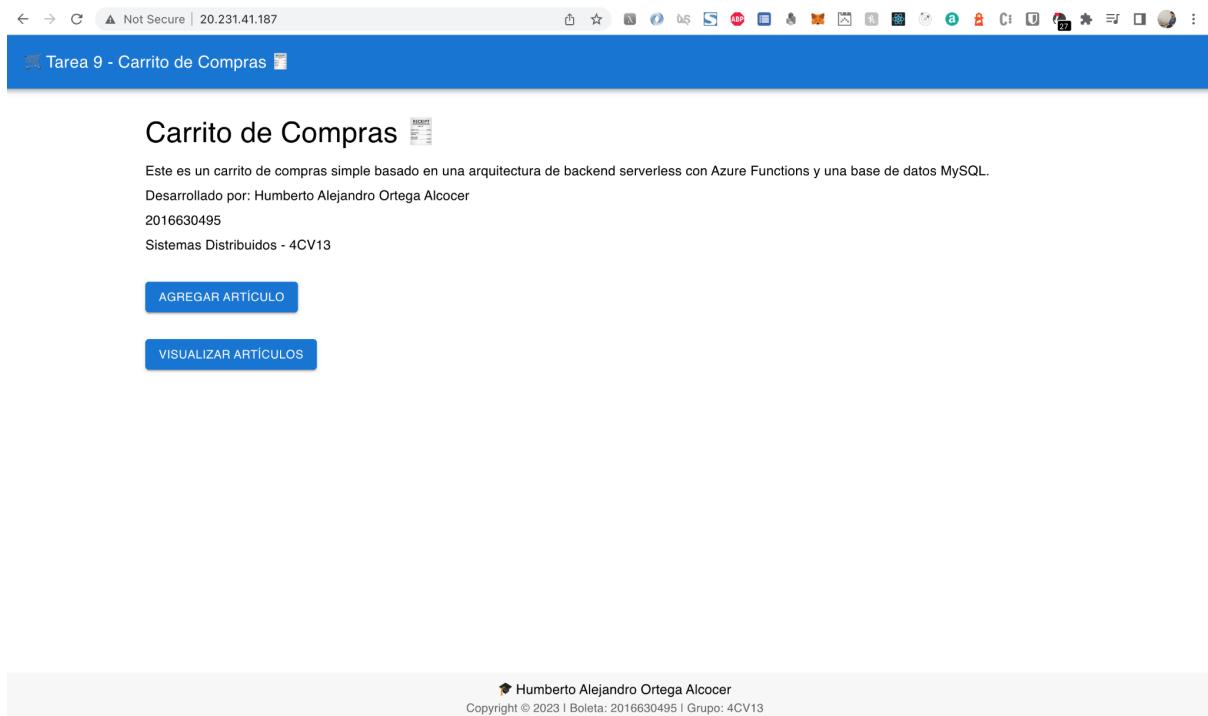
Ahora probaremos que nuestra configuración es válida usando el comando **sudo nginx -t**:

```
humberto@T9-FE-2016630495:~$ sudo nginx -t  
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Reiniciaremos el servicio:

```
humberto@T9-FE-2016630495:~$ sudo systemctl reload nginx  
humberto@T9-FE-2016630495:~$ sudo systemctl status nginx  
● nginx.service - A high performance web server and a reverse proxy server  
  Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)  
  Active: active (running) since Sun 2023-06-11 01:48:02 UTC; 1min ago  
    Docs: man:nginx(8)  
  Process: 3695 (nginx) Main PID: 3695 (nginx)  
     Tasks: 3 (limit: 4699)  
    Memory: 5.9M  
   CGroup: /system.slice/nginx.service  
           ├─3695 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;  
           ├─4105 nginx: worker process  
           ├─4106 nginx: worker process  
           └─4107 nginx: worker process  
  
Jun 11 01:48:02 T9-FE-2016630495 systemd[1]: Starting A high performance web server and a reverse proxy server...  
Jun 11 01:48:02 T9-FE-2016630495 systemd[1]: Started A high performance web server and a reverse proxy server.  
Jun 11 02:02:13 T9-FE-2016630495 systemd[1]: Reloading A high performance web server and a reverse proxy server.  
Jun 11 02:02:13 T9-FE-2016630495 systemd[1]: Reloaded A high performance web server and a reverse proxy server.
```

Ahora introduciremos nuestra IP pública en el navegador (usando el protocolo HTTP) y debemos ver nuestra aplicación de la siguiente manera:



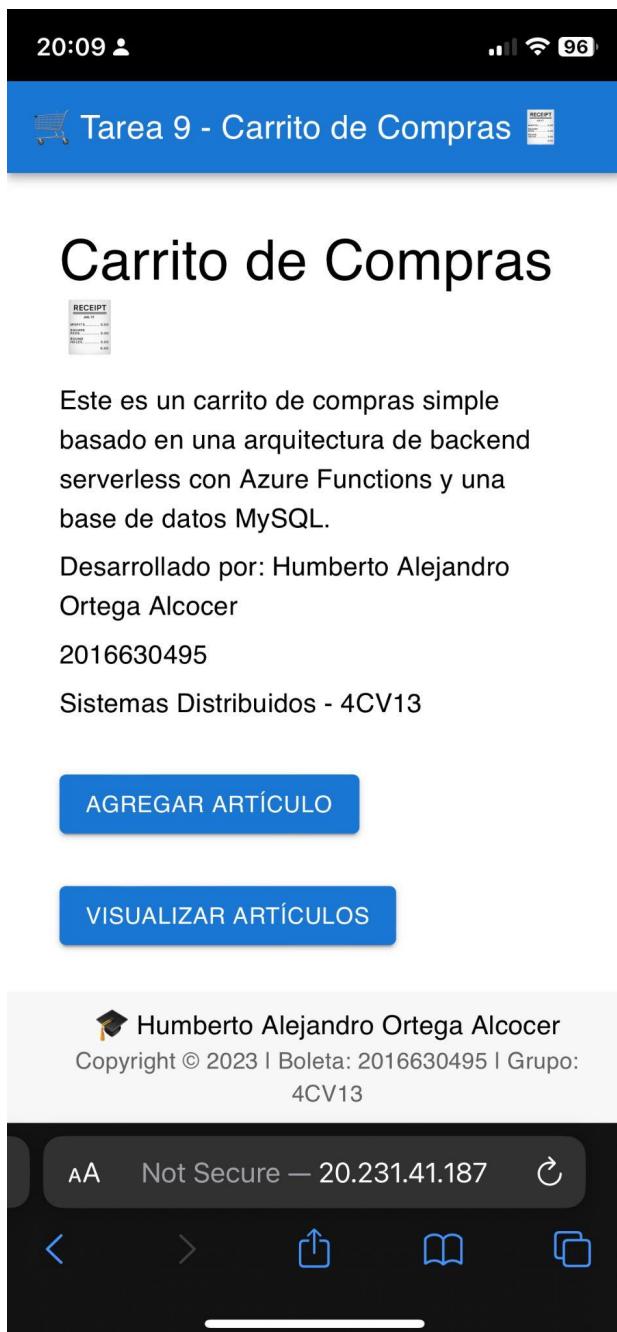
Con esto queda totalmente funcional el despliegue de la Tarea 9. La siguiente sección será dónde se realicen las pruebas funcionales.

2. Pruebas de Funcionamiento

En esta sección se realizarán las pruebas de funcionamiento de la tarea según las especificaciones del profesor. Para cada prueba se mostrará la pantalla del dispositivo móvil realizando la operación y se anexarán capturas de la base de datos mostrando los cambios así como los logs correspondientes a la ejecución en el backend.

2.1. Página principal.

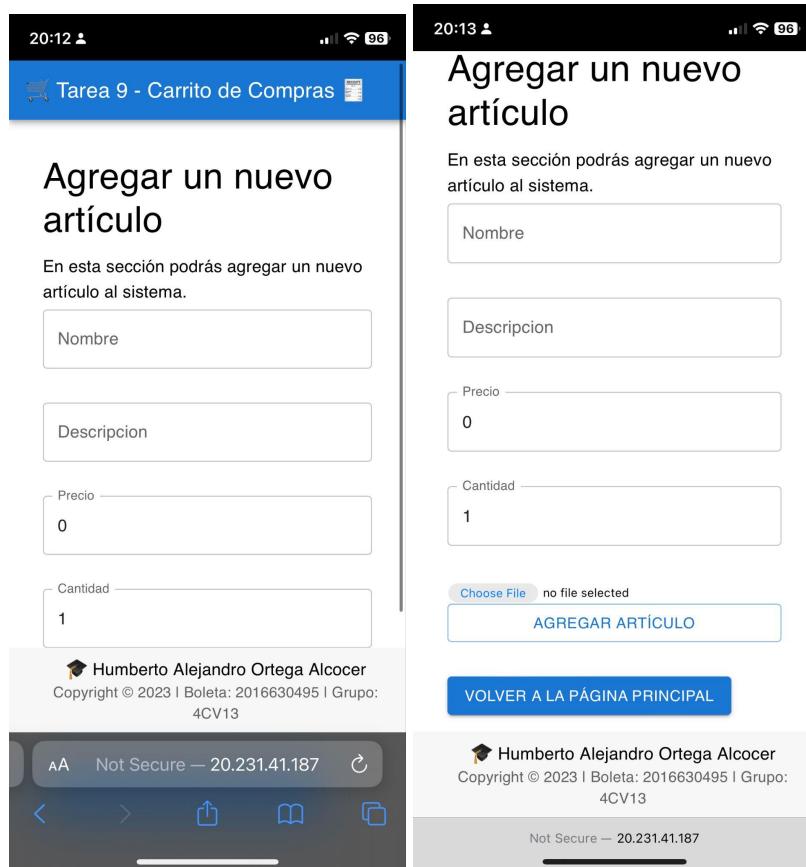
El cliente deberá desplegar inicialmente un menú con dos opciones: Captura de artículo y Compra de artículos.



2.2. Captura de artículo.

Al seleccionar la opción "Captura de artículo" el cliente deberá desplegar la pantalla "Captura de artículos" la cual deberá permitir capturar el nombre del artículo, la descripción del artículo, el precio, la cantidad en almacén y la fotografía del artículo. Los datos de los artículos se deberán guardar en una tabla llamada "articulos". Cada artículo deberá tener un ID auto-incremental.

2.2.1. Formulario sin datos.



2.2.2. Formulario con datos inválidos.

The screenshot shows a mobile application interface for adding a new article. At the top, there is a header with the time (20:14), signal strength, battery level (96%), and a user icon. Below the header, the title "Agregar un nuevo artículo" is displayed in large bold text. A descriptive text below the title states: "En esta sección podrás agregar un nuevo artículo al sistema." The form consists of four input fields, each with a red border indicating validation errors:

- Nombre:** An empty input field. Below it, the error message "El nombre es requerido" is displayed in red.
- Descripción:** An empty input field.
- Precio:** An empty input field. Below it, the error message "El precio es requerido" is displayed in red.
- Cantidad:** An empty input field. Below it, the error message "La cantidad es requerida" is displayed in red.

At the bottom of the screen, there is a footer section containing a graduation cap icon, the name "Humberto Alejandro Ortega Alcocer", and copyright information: "Copyright © 2023 | Boleta: 2016630495 | Grupo: 4CV13". Below the footer is a navigation bar with icons for back, forward, search, and other functions.

2.2.3. Formulario con datos válidos.

20:16 12 96

artículo al sistema.

Nombre

Descripción

Precio

Cantidad

Choose File  image.jpg

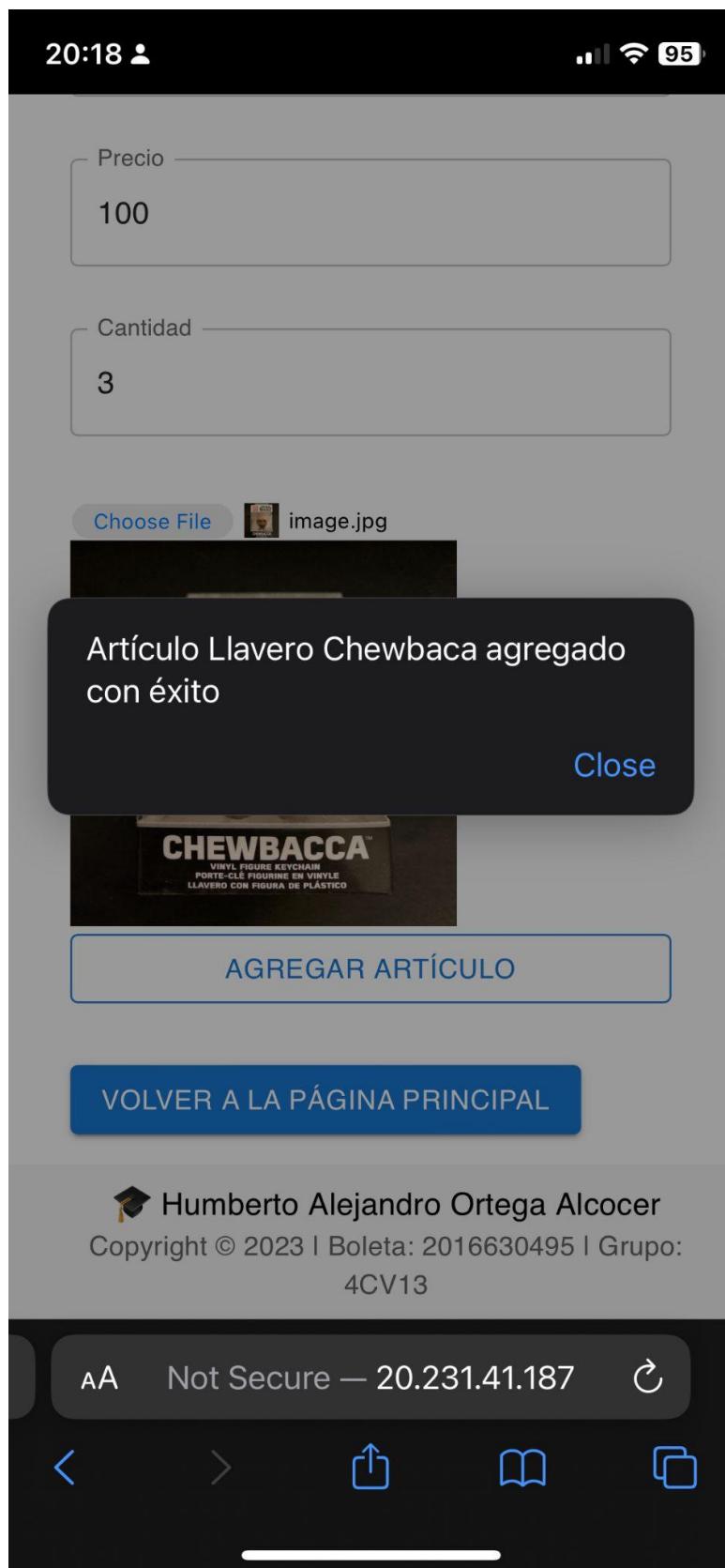


AGREGAR ARTÍCULO

🎓 Humberto Alejandro Ortega Alcocer
Copyright © 2023 | Boleta: 2016630495 | Grupo:
4CV13

Not Secure — 20.231.41.187

2.2.4. Respuesta exitosa de backend.



2.2.5. Estado de la base de datos.

	id	nombre	precio	cantidad	descripcion	fotografia
1	Prueba1	12.5	4	Esta es una prueba	data:image/jpeg;base64,/9j/4AAQSkQ...	
2	Llavero Chewbacca	100	3	Un hermoso llavero de Chewbacca	data:image/jpeg;base64,/9j/4AAQSkQ...	

2.3. Compra de artículos.

Al seleccionar la opción "Compra de artículos" el cliente deberá desplegar la pantalla "Compra de artículos" la cual deberá permitir al usuario buscar artículos ingresando una palabra la cual se buscará en los campos "nombre" y "descripcion" de la tabla "articulos". La búsqueda se deberá realizar utilizando una instrucción SELECT con LIKE.

2.3.1. Pantalla principal.

Artículos disponibles
En esta sección podrás ver los artículos disponibles para comprar.
[IR AL CARRITO DE COMPRAS](#)

Búsqueda de artículos:
Escribe el nombre del artículo a buscar...

CASO PRÁCTICO 1: Prueba1 - \$12.50 MXN
VER MÁS INFORMACIÓN

Humberto Alejandro Ortega Alcocer
Copyright © 2023 | Boleta: 2016630495 | Grupo: 4CV13

Not Secure — 20.231.41.187

2.3.2. Filtrando por texto.

The screenshot shows a mobile application interface. At the top, there is a header with the time "20:26", signal strength, battery level at 95%, and a profile icon. Below the header, the title "Artículos disponibles" is displayed in large bold letters. A subtext below it reads "En esta sección podrás ver los artículos disponibles para comprar." A blue button labeled "IR AL CARRITO DE COMPRAS" is visible. The search query "Chew" is entered into a search bar. Below the search bar, a product card for a "Llavero Chewbacca" is shown, featuring a photo of a Chewbacca figurine. The product details are: "Llavero Chewbacca - \$100.00 MXN". A blue link "VER MÁS INFORMACIÓN" is present. A quantity selector box contains the number "1". At the bottom of the screen, there is a footer with a graduation cap icon and the name "Humberto Alejandro Ortega Alcocer". It also includes copyright information: "Copyright © 2023 | Boleta: 2016630495 | Grupo: 4CV13". The bottom navigation bar includes icons for back, forward, upload, bookmark, and share.

2.3.3. Removiendo filtro.

20:27 1 95

Búsqueda de artículos:
Escribe el nombre del artículo a buscar...

CASO PRÁCTICO 1

Institución de gobierno. Se estima un tamaño aproximado de desarrollo de 100,000 líneas de código fuente, contando con 21 módulos principales de tamaño promedio. El cliente ha manifestado un estricto respeto en los requerimientos además de exigir una total discreción con la información manejada, por lo tanto, nos hará la petición formal de firmar una carta de confidencialidad. El equipo de desarrollo cuenta con muy buenas evaluaciones en el CMM, además de contar con muy alta experiencia en lo relacionado al producto a desarrollar. El cliente también desea que todos los módulos tengan un nivel de reusabilidad muy alto.

Prueba1 - \$12.50 MXN

VER MÁS INFORMACIÓN

1

AGREGAR AL CARRITO



Llavero Chewbacca - \$100.00

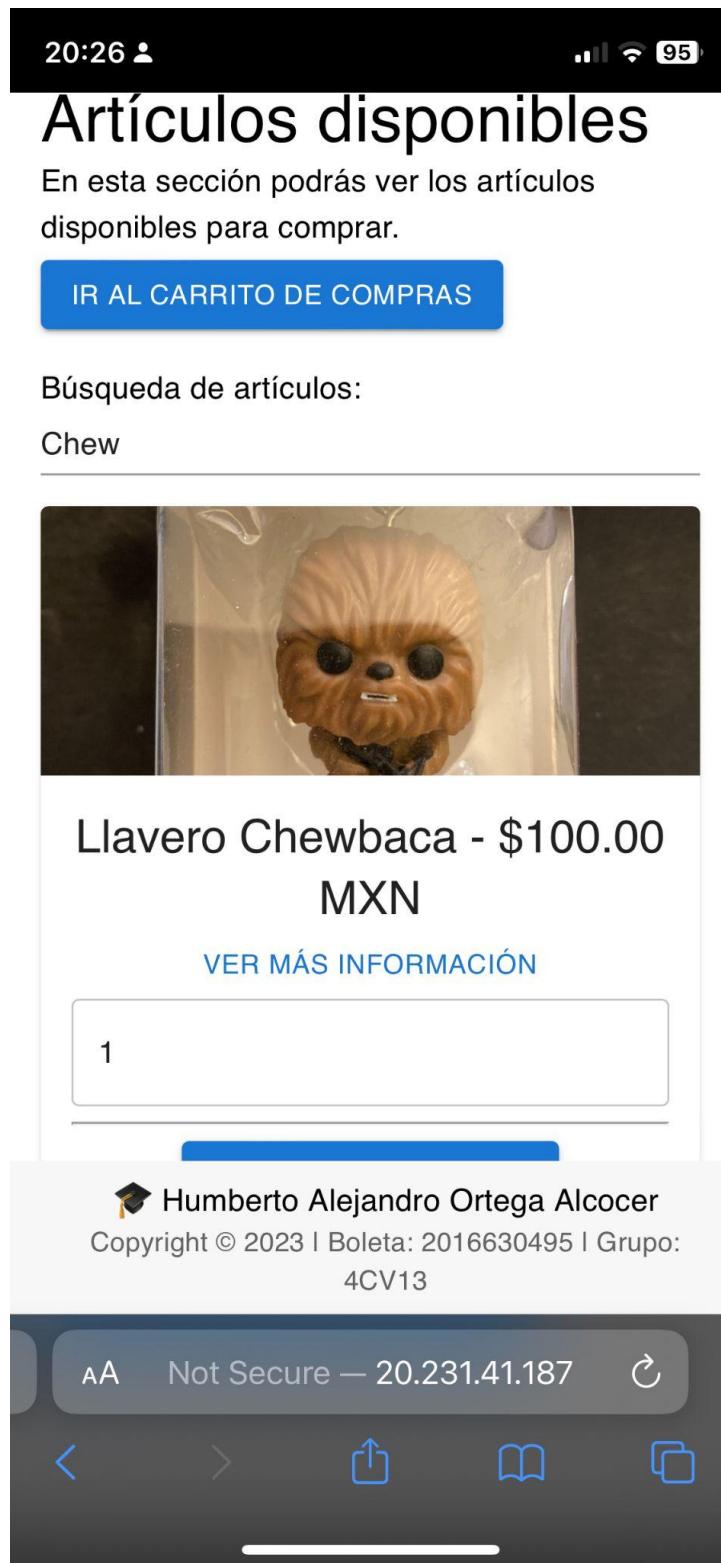
🎓 Humberto Alejandro Ortega Alcocer
Copyright © 2023 | Boleta: 2016630495 | Grupo:
4CV13

Not Secure — 20.231.41.187

2.4. Búsqueda de artículos.

Los datos de los artículos (fotografía, nombre y precio) que resulten de una búsqueda se deberán desplegar en la pantalla "Compra de artículos".

2.4.1. Filtrando por texto.



2.3.3. Removiendo filtro.

20:27 1 95

Búsqueda de artículos:
Escribe el nombre del artículo a buscar...

CASO PRÁCTICO 1

Institución de gobierno. Se estima un tamaño aproximado de desarrollo de 100,000 líneas de código fuente, contando con 21 módulos principales de tamaño promedio. El cliente ha manifestado un estricto respeto en los requerimientos además de exigir una total discreción con la información manejada, por lo tanto, nos hará la petición formal de firmar una carta de confidencialidad. El equipo de desarrollo cuenta con muy buenas evaluaciones en el CMM, además de contar con muy alta experiencia en lo relacionado al producto a desarrollar. El cliente también desea que todos los módulos tengan un nivel de reusabilidad muy alto.

Prueba1 - \$12.50 MXN

VER MÁS INFORMACIÓN

1

AGREGAR AL CARRITO



Llavero Chewbacca - \$100.00

🎓 Humberto Alejandro Ortega Alcocer
Copyright © 2023 | Boleta: 2016630495 | Grupo:
4CV13

Not Secure — 20.231.41.187

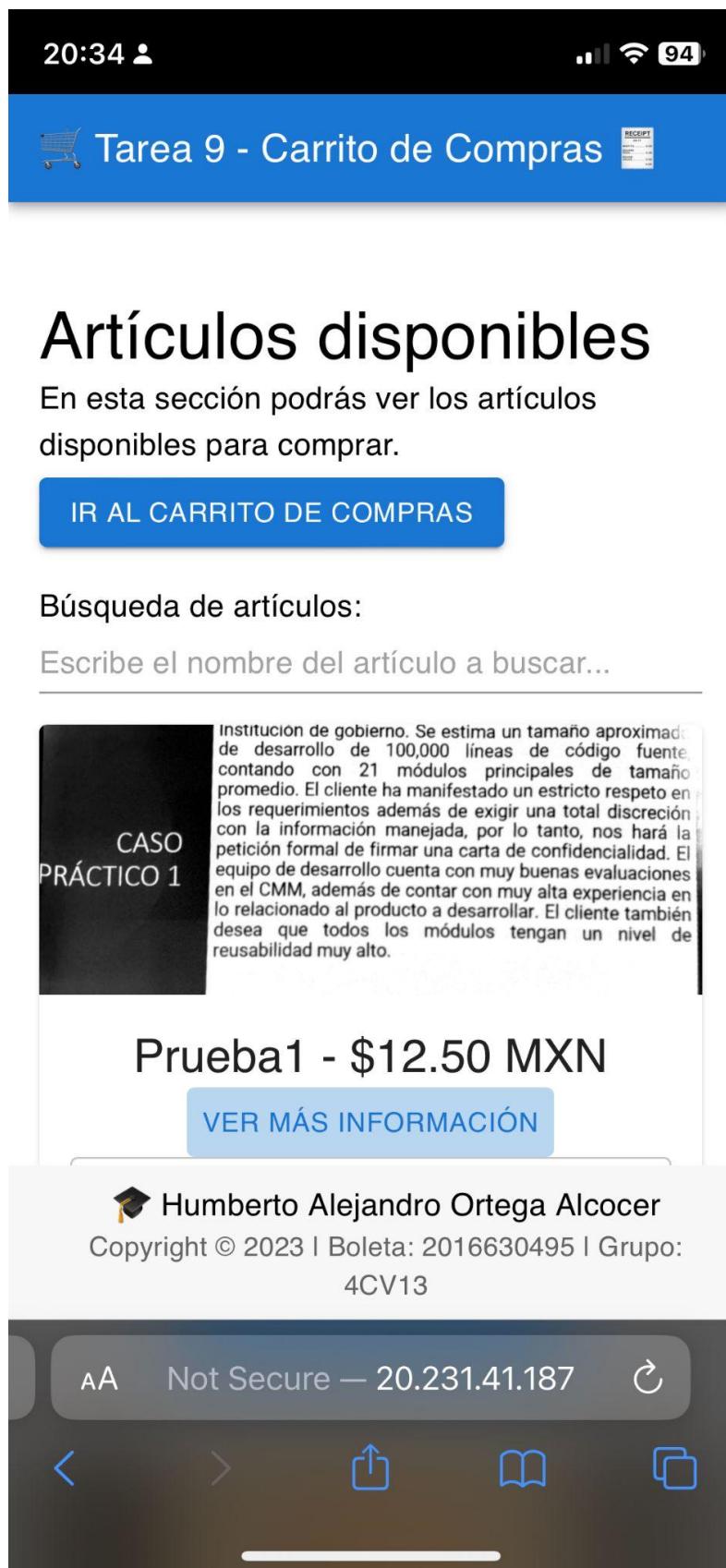
2.5. Descripción de artículos.

Para cada artículo resultado de la búsqueda, se deberá desplegar un botón de "Compra" y un campo de "Cantidad" con un valor default igual a 1. Para cada artículo se deberá desplegar un botón "Descripción", al dar clic a este botón se deberá desplegar una ventana con la descripción del artículo y un botón "Cerrar" para cerrar ésta ventana.

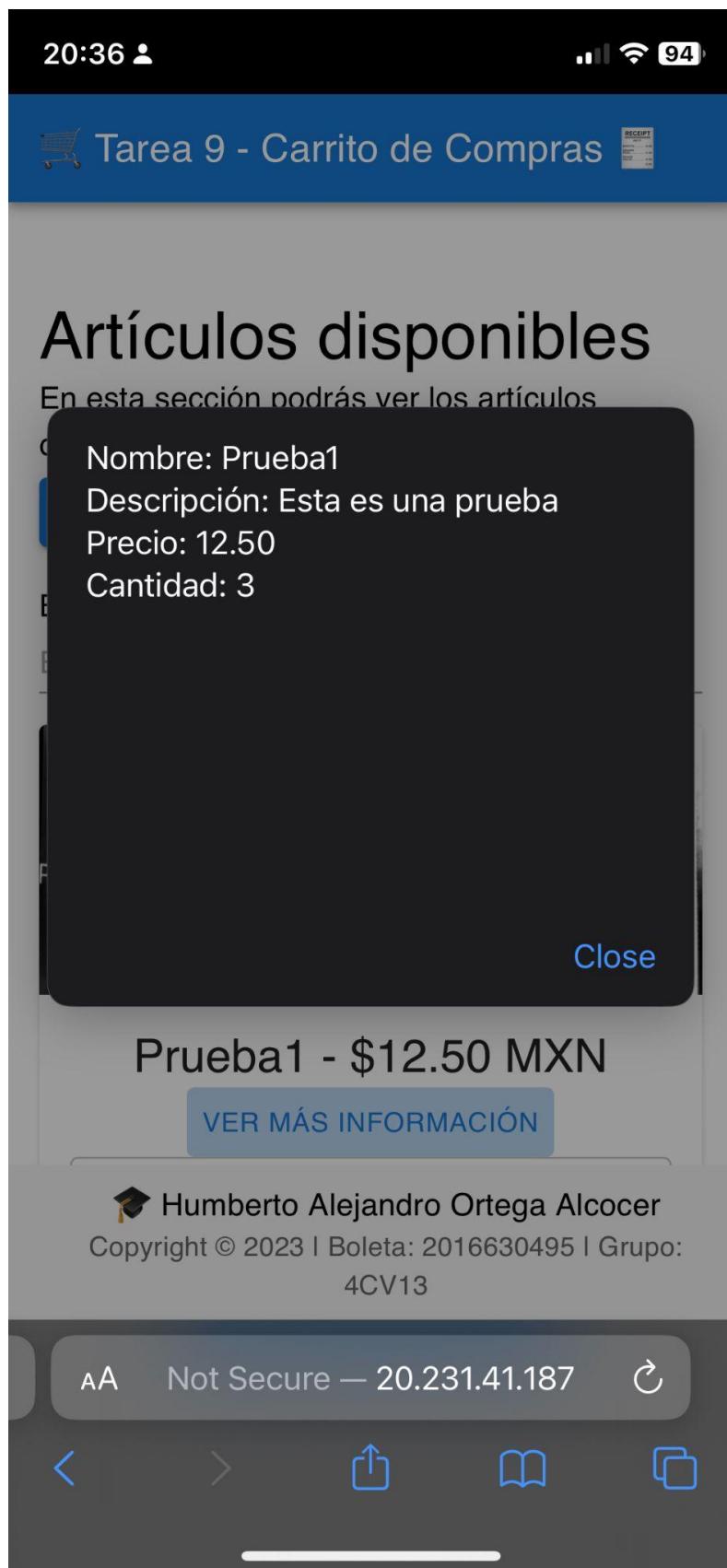
2.5.1. Pantalla principal de artículos.



2.5.2. Presionando botón de “Ver más información”.



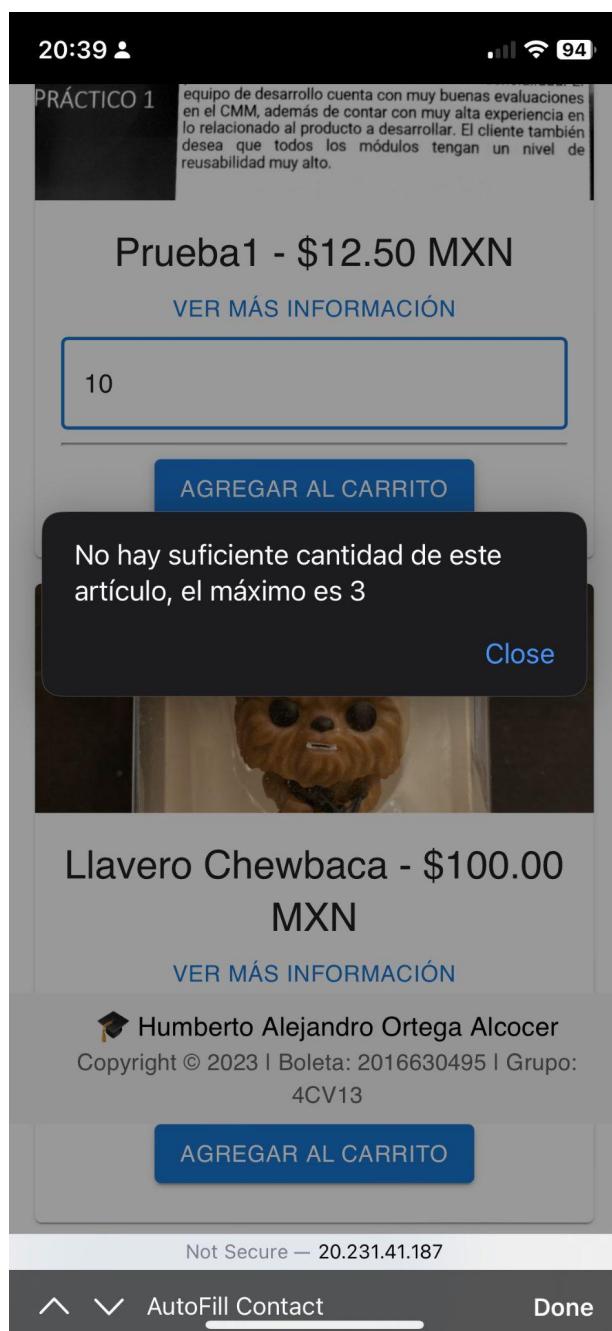
2.5.3. Ventana de “más información”.



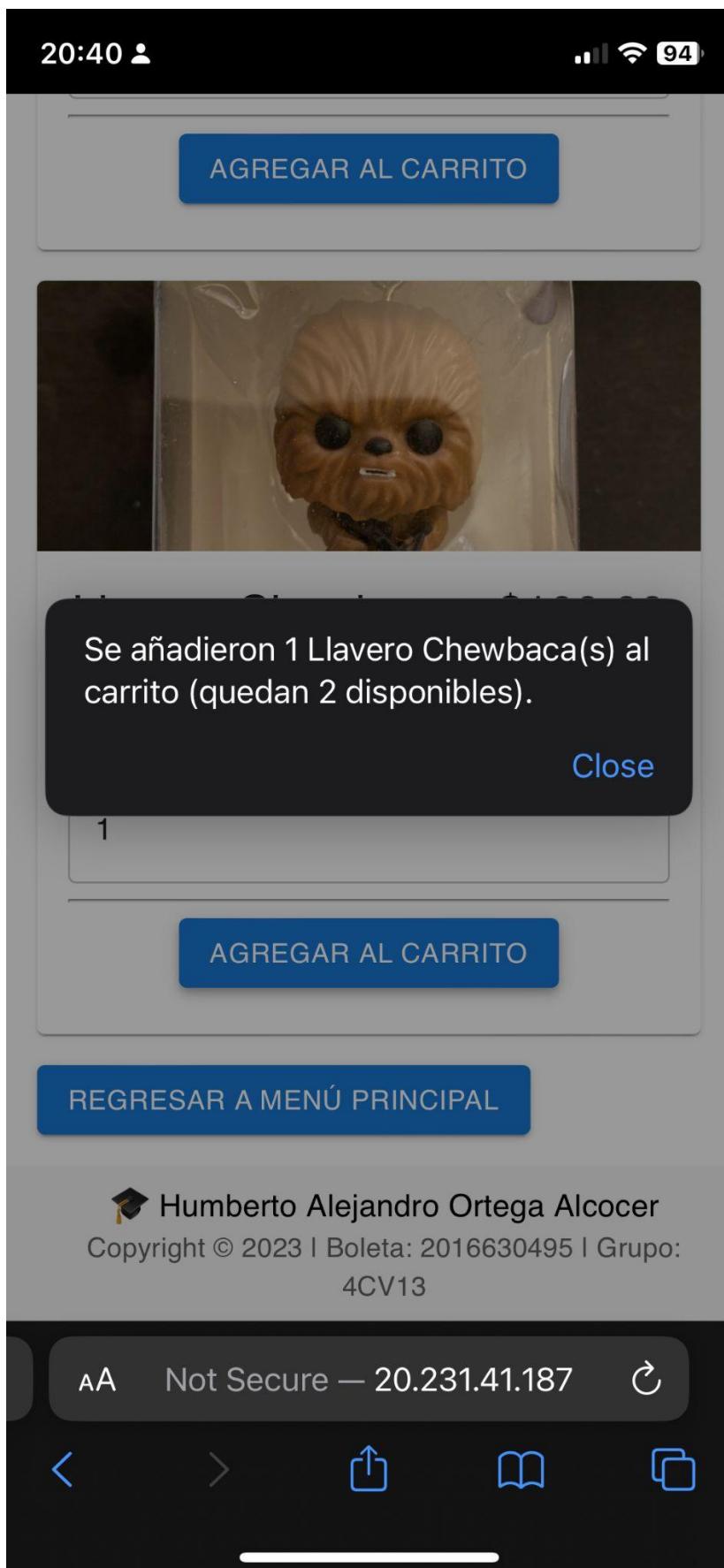
2.6. Agregar al carrito con validación.

Cuando el usuario presione el botón de "Compra", si la cantidad de artículos a comprar es menor o igual a la cantidad de artículos en la tabla "articulos", se deberá insertar en una tabla llamada "carrito_compra" el ID del artículo y la cantidad, así mismo se restará la cantidad solicitada de la cantidad en la tabla de "artículos". Si la cantidad de artículos a comprar es mayor a la cantidad de artículos en la tabla "articulos" se deberá desplegar un mensaje indicando al usuario el número de artículos disponibles. El INSERT a la tabla "carrito_compra" y el UPDATE a la tabla "artículos" se deberán realizar dentro de una transacción.

2.6.1. Validación de cantidad máxima de artículos.



2.6.2. Agregando al carrito correctamente.



2.6.3. Estado de la base de datos.

The screenshot shows the DBeaver 23.1.0 interface with two open tables: 'articulos' and 'carrito'.
Table articulos:
Properties | Data | ER Diagram |
t9-bd-2016630495.mysql.database.azure.com | Databases | tarea9 | Tables |
articulos | Enter a SQL expression to filter results | Grid | Text | Record |
Fields: id, nombre, precio, cantidad.
Data:
1 Prueba1 12.5 16K
2 Llavero Chewbacca 100 16K
Table carrito:
Properties | Data | ER Diagram |
t9-bd-2016630495.mysql.database.azure.com | Databases | tarea9 | Tables |
carrito | Enter a SQL expression to filter results | Grid | Text | Record |
Fields: articulo_id, cantidad.
Data:
1 1 1
2 2 1
... 2 row(s) fetched - 1.279s (40ms fetch), on 2023-06-10 at 20:19:25
CST en_MX
... 2 row(s) fetched - 72ms, on 2023-06-10 at 20:41:37

2.6.4. Operaciones sucediendo en la misma transacción.

The screenshot shows a code editor with a dark theme, displaying a file named `index.ts`. The code is written in TypeScript and contains logic for managing a shopping cart. It includes comments in Spanish explaining the steps:

- Verificamos que se haya encontrado al menos uno.
- Creamos la transacción.
- Agregamos el artículo al carrito.
- Quitamos la cantidad de artículos del inventario.
- Restamos la cantidad de artículos del inventario.
- Hacemos commit de la transacción.
- Regresamos el artículo.

The code uses Sequelize's query interface to perform database operations within a transaction. It includes type annotations for `QueryTypes` and `Replacements`.

```
1 TS index.ts X
2 tareas > tarea-9 > backend > AgregarCarrito > TS index.ts > httpTrigger
3
4     replacements: { articulo_id: articulo.id },
5
6 );
7
8     // Verificamos que se haya encontrado al menos uno.
9     if (carritos.length === 0) {
10        // Creamos la transaccion.
11        const transaction = await sequelize.transaction();
12
13        // Agregamos el articulo al carrito.
14        await sequelize.query(
15            "INSERT INTO carrito (articulo_id, cantidad) VALUES (:articulo_id, :cantidad)",
16            {
17                type: QueryTypes.INSERT,
18                replacements: {
19                    articulo_id: articulo.id,
20                    cantidad: agregarArticuloCarritoDto.cantidad,
21                },
22                transaction,
23            }
24        );
25
26        // Quitamos la cantidad de articulos del inventario.
27        await sequelize.query(
28            "UPDATE articulos SET cantidad = cantidad - :cantidad WHERE id = :id",
29            {
30                type: QueryTypes.UPDATE,
31                replacements: {
32                    id: articulo.id,
33                    cantidad: agregarArticuloCarritoDto.cantidad,
34                },
35                transaction,
36            }
37        );
38
39        // Restamos la cantidad de articulos del inventario.
40        articulo.cantidad -= agregarArticuloCarritoDto.cantidad;
41
42        // Hacemos commit de la transaccion.
43        await transaction.commit();
44
45        // Regresamos el articulo.
46        context.res = {
47            status: 200,
48            body: articulo,
49        };
50    }
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
```

2.7. Carrito de compra.

La pantalla de "Compra de artículos" deberá disponer de un botón "Carrito de compra" el cual deberá desplegar una pantalla "Artículos en el carrito" con la lista de artículos en la tabla "carrito_compra", incluyendo una pequeña imagen del artículo, nombre del artículo, cantidad, precio y costo (cantidad x precio). Así mismo, en la ventana "Artículos en el carrito" se deberá desplegar el total de la compra.

2.7.1. Botón de carrito de compra.



2.7.2. Página principal de carrito de compras.

The screenshot shows a mobile application interface for a shopping cart. At the top, there is a blue header bar with the text "Tarea 9 - Carrito de Compras" and a shopping cart icon. The main title "Carrito de compras" is displayed prominently in large black font. Below the title, a descriptive text block states: "En esta sección podrás ver los artículos que has añadido al carrito de compras, la cantidad y el precio de cada uno." To the left of this text is a dark rectangular box containing the text "CASO PRÁCTICO 1". To the right of the text block is a detailed description of a project requirement: "Institución de gobierno. Se estima un tamaño aproximado de desarrollo de 100,000 líneas de código fuente, contando con 21 módulos principales de tamaño promedio. El cliente ha manifestado un estricto respeto en los requerimientos además de exigir una total discreción con la información manejada, por lo tanto, nos hará la petición formal de firmar una carta de confidencialidad. El equipo de desarrollo cuenta con muy buenas evaluaciones en el CMM, además de contar con muy alta experiencia en lo relacionado al producto a desarrollar. El cliente también desea que todos los módulos tengan un nivel de reusabilidad muy alto." Below the description, the word "Prueba1" is displayed. Further down, the quantity "Cantidad: 1" and unit price "Precio unitario: \$12.50 MXN" are shown, followed by the total "Total: \$12.5 MXN". At the bottom of the screen, there is a footer with the text "Humberto Alejandro Ortega Alcocer" and "Copyright © 2023 | Boleta: 2016630495 | Grupo: 4CV13". The bottom of the screen also shows a standard mobile browser toolbar with icons for back, forward, search, and refresh.

2.7.3. Elementos mostrados por artículo en el carrito.

The screenshot shows a mobile application interface. At the top, there is a black header bar with the time "20:48", a profile icon, signal strength, battery level at 93%, and a "Carrito de compras" title. Below the header, a dark sidebar on the left contains the text "CASO PRÁCTICO 1". The main content area displays a product item with the following details:

- Nombre:** Prueba1
- Cantidad:** 1
- Precio unitario:** \$12.50 MXN
- Total:** \$12.5 MXN

Below the product details is an orange button labeled "ELIMINAR ARTÍCULO DEL CARRITO". Underneath the main content area, there is a small image of a Chewbacca figurine in a clear plastic case. At the bottom of the screen, there is a footer bar containing the following information:

- A graduation cap icon followed by the name "Humberto Alejandro Ortega Alcocer"
- Copyright © 2023 I Boleta: 2016630495 I Grupo:
- 4CV13

At the very bottom of the screen, it says "Not Secure — 20.231.41.187".

2.7.4. Total de la compra.

20:49    93%

Total: \$12.5 MXN

[ELIMINAR ARTÍCULO DEL CARRITO](#)



Llavero Chewbacca

Cantidad: 1

Precio unitario: \$100.00 MXN

Total: \$100 MXN

[ELIMINAR ARTÍCULO DEL CARRITO](#)

Total: \$112.5 MXN

[VACIAR CARRITO DE COMPRAS.](#)

[VOLVER A LISTADO DE ARTÍCULOS](#)

 Humberto Alejandro Ortega Alcocer
Copyright © 2023 | Boleta: 2016630495 | Grupo:
4CV13

Not Secure — 20.231.41.187

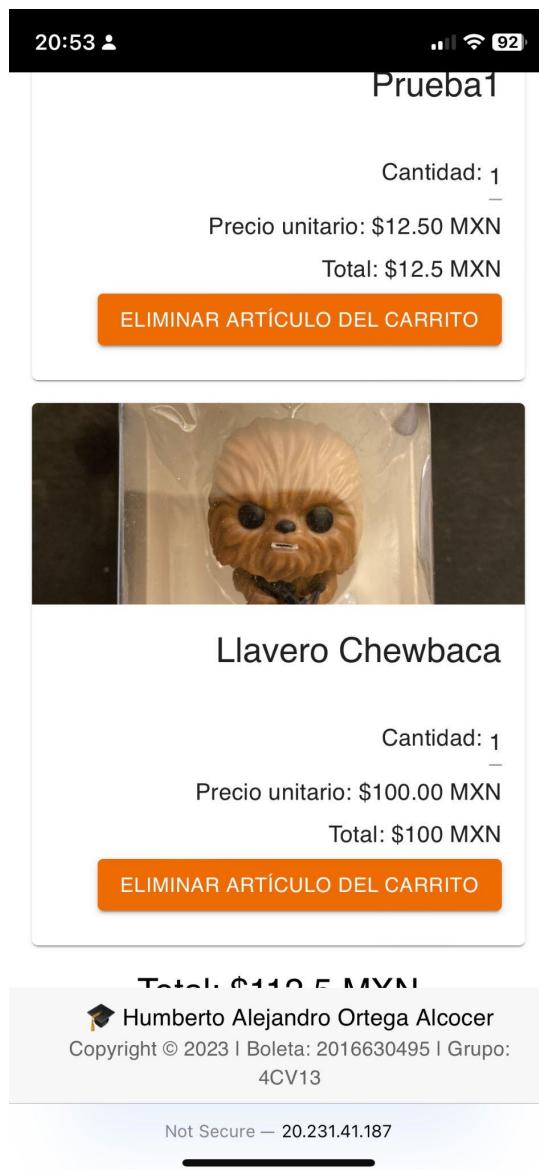
2.8. Eliminar artículo de carrito de compra.

Para cada artículo en la pantalla "Artículos en el carrito" se deberá incluir un botón "Eliminar artículo", al dar clic a este botón se deberá eliminar el artículo del carrito de compra, borrando el registro correspondiente de la tabla "carrito_compra" y agregando la cantidad de los artículos en la tabla "articulos".

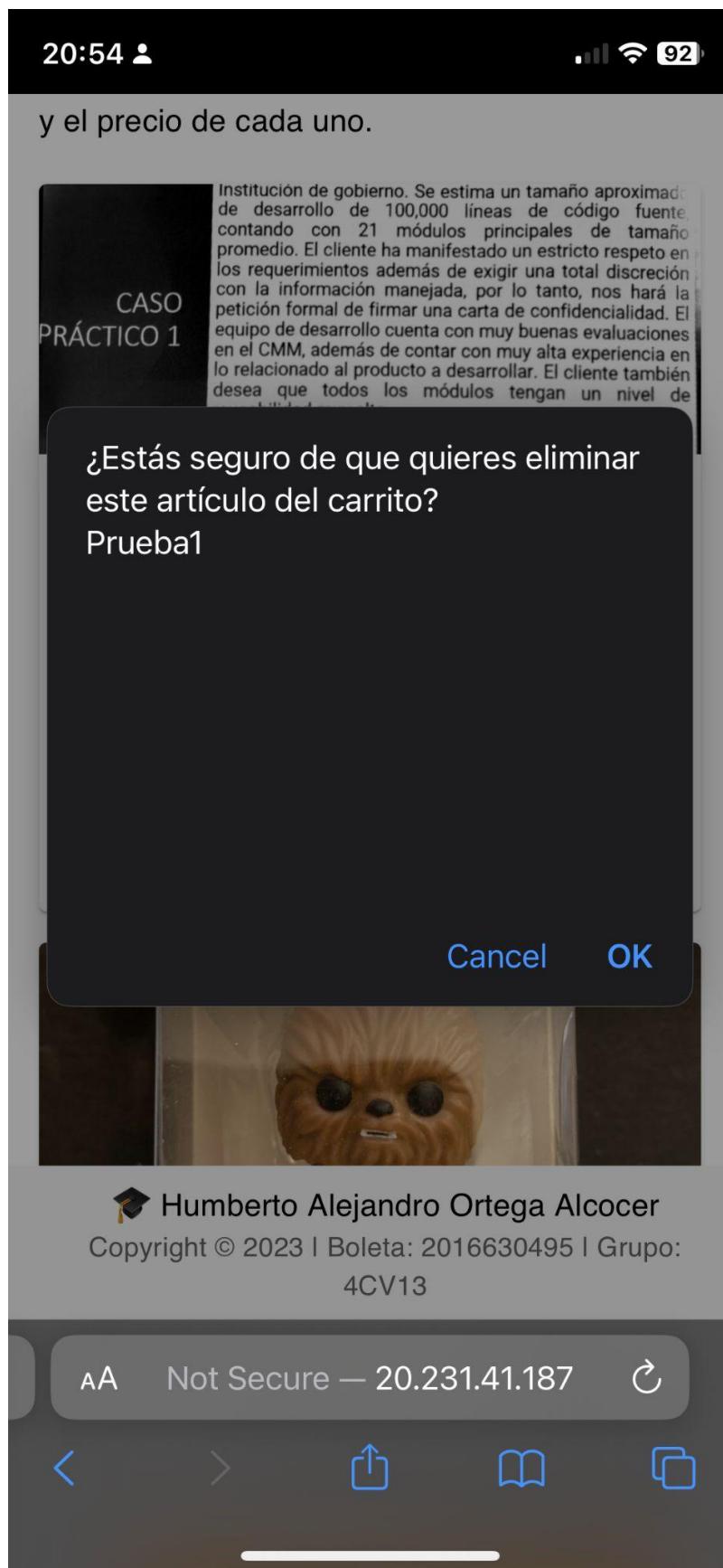
El borrado (DELETE) del artículo de la tabla "carrito_compra" y la actualización (UPDATE) de la tabla "articulos" deberán realizarse dentro de una transacción.

Se deberá desplegar una ventana preguntando al usuario si está seguro de eliminar el artículo del carrito de compra, incluyendo el nombre del artículo, la cantidad de artículos y dos botones "Si" y "No"; si el usuario presiona el botón "Si" se deberá eliminar el artículo del carrito de compra.

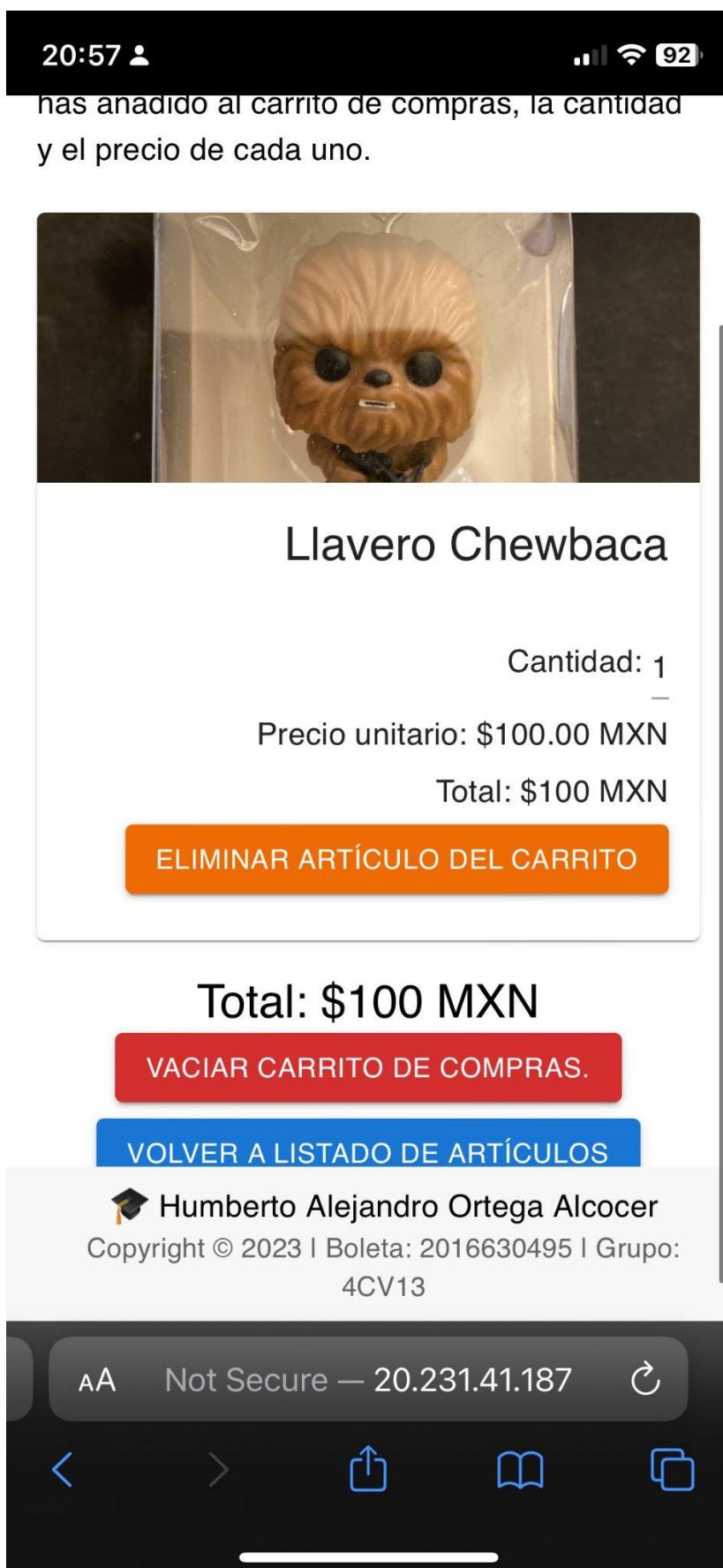
2.8.1. Botón de "Eliminar artículo".



2.8.2. Ventana de confirmación para eliminar artículo.



2.8.3. Carrito sin elemento eliminado.



2.8.3. Parte del código que usa transacciones.



The screenshot shows a code editor window with the file `index.ts` open. The code is written in TypeScript and uses Sequelize for database operations. It performs a series of steps: it checks if the cart is empty, creates a transaction, adds the article back to the cart, removes the article from the cart, executes the transaction, responds with a 200 status, and finally closes the connection. Error handling is included to respond with a 500 status if an error occurs.

```
ts index.ts M X
tareas > tarea-9 > backend > EliminarCarrito > ts index.ts > [o] httpTrigger
67     if (carritos.length ≤ 0) {
68         context.res = {
69             status: 404,
70             body: "No existe el artículo en el carrito de compras",
71         };
72     } else {
73         // Crear una transacción.
74         const transaction: Transaction = await sequelize.transaction();
75
76         // Agregar de vuelta las unidades del carrito al artículo.
77         await sequelize.query(
78             "UPDATE articulos SET cantidad = cantidad + :cantidad WHERE id = :id",
79             {
80                 replacements: [
81                     { id: carritos[0].articulo_id, cantidad: carritos[0].cantidad },
82                 ],
83                 type: QueryTypes.UPDATE,
84             },
85             transaction,
86         );
87     }
88
89     // Eliminar el artículo del carrito de compras.
90     await sequelize.query(
91         "DELETE FROM carrito WHERE articulo_id = :articulo_id",
92         {
93             replacements: { articulo_id: id },
94             type: QueryTypes.DELETE,
95             transaction,
96         },
97     );
98
99     // Ejecutar la transacción.
100    await transaction.commit();
101
102    // Responder con un 200.
103    context.res = {
104        status: 200,
105        body: "Artículo eliminado del carrito de compras",
106    };
107 }
108 // Cerrar la conexión a la base de datos.
109 await sequelize.close();
110 } catch (error) {
111     context.res = {
112         status: 500,
113         body: {
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
818
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
897
898
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
917
918
919
919
920
921
922
923
924
925
926
927
927
928
929
929
930
931
932
933
934
935
936
937
937
938
939
939
940
941
942
943
944
945
946
947
947
948
949
949
950
951
952
953
954
955
956
957
957
958
959
959
960
961
962
963
964
965
966
967
967
968
969
969
970
971
972
973
974
975
976
977
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1046
1047
1048
1048
1049
1050
1051
1052
1053
1054
1055
1056
1056
1057
1058
1058
1059
1060
1061
1062
1063
1064
1065
1065
1066
1067
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1075
1076
1077
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1094
1095
1096
1096
1097
1098
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1113
1114
1115
1115
1116
1117
1117
1118
1119
1119
1120
1121
1122
1123
1123
1124
1125
1125
1126
1127
1127
1128
1129
1129
1130
1131
1132
1133
1133
1134
1135
1135
1136
1137
1137
1138
1139
1139
1140
1141
1142
1143
1143
1144
1145
1145
1146
1147
1147
1148
1149
1149
1150
1151
1152
1153
1153
1154
1155
1155
1156
1157
1157
1158
1159
1159
1160
1161
1162
1163
1163
1164
1165
1165
1166
1167
1167
1168
1169
1169
1170
1171
1172
1173
1173
1174
1175
1175
1176
1177
1177
1178
1179
1179
1180
1181
1182
1183
1183
1184
1185
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1193
1194
1195
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1203
1204
1205
1205
1206
1207
1207
1208
1209
1209
1210
1211
1212
1213
1213
1214
1215
1215
1216
1217
1217
1218
1219
1219
1220
1221
1222
1223
1223
1224
1225
1225
1226
1227
1227
1228
1229
1229
1230
1231
1232
1233
1233
1234
1235
1235
1236
1237
1237
1238
1239
1239
1240
1241
1242
1243
1243
1244
1245
1245
1246
1247
1247
1248
1249
1249
1250
1251
1252
1253
1253
1254
1255
1255
1256
1257
1257
1258
1259
1259
1260
1261
1262
1263
1263
1264
1265
1265
1266
1267
1267
1268
1269
1269
1270
1271
1272
1273
1273
1274
1275
1275
1276
1277
1277
1278
1279
1279
1280
1281
1282
1283
1283
1284
1285
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1293
1294
1295
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1303
1304
1305
1305
1306
1307
1307
1308
1309
1309
1310
1311
1312
1313
1313
1314
1315
1315
1316
1317
1317
1318
1319
1319
1320
1321
1322
1323
1323
1324
1325
1325
1326
1327
1327
1328
1329
1329
1330
1331
1332
1333
1333
1334
1335
1335
1336
1337
1337
1338
1339
1339
1340
1341
1342
1343
1343
1344
1345
1345
1346
1347
1347
1348
1349
1349
1350
1351
1352
1353
1353
1354
1355
1355
1356
1357
1357
1358
1359
1359
1360
1361
1362
1363
1363
1364
1365
1365
1366
1367
1367
1368
1369
1369
1370
1371
1372
1373
1373
1374
1375
1375
1376
1377
1377
1378
1379
1379
1380
1381
1382
1383
1383
1384
1385
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1393
1394
1395
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1403
1404
1405
1405
1406
1407
1407
1408
1409
1409
1410
1411
1412
1413
1413
1414
1415
1415
1416
1417
1417
1418
1419
1419
1420
1421
1422
1423
1423
1424
1425
1425
1426
1427
1427
1428
1429
1429
1430
1431
1432
1433
1433
1434
1435
1435
1436
1437
1437
1438
1439
1439
1440
1441
1442
1443
1443
1444
1445
1445
1446
1447
1447
1448
1449
1449
1450
1451
1452
1453
1453
1454
1455
1455
1456
1457
1457
1458
1459
1459
1460
1461
1462
1463
1463
1464
1465
1465
1466
1467
1467
1468
1469
1469
1470
1471
1472
1473
1473
1474
1475
1475
1476
1477
1477
1478
1479
1479
1480
1481
1482
1483
1483
1484
1485
1485
1486
1487
1487
1488
1489
1489
1490
1491
1492
1493
1493
1494
1495
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1503
1504
1505
1505
1506
1507
1507
1508
1509
1509
1510
1511
1512
1513
1513
1514
1515
1515
1516
1517
1517
1518
1519
1519
1520
1521
1522
1523
1523
1524
1525
1525
1526
1527
1527
1528
1529
1529
1530
1531
1532
1533
1533
1534
1535
1535
1536
1537
1537
1538
1539
1539
1540
1541
1542
1543
1543
1544
1545
1545
1546
1547
1547
1548
1549
1549
1550
1551
1552
1553
1553
1554
1555
1555
1556
1557
1557
1558
1559
1559
1560
1561
1562
1563
1563
1564
1565
1565
1566
1567
1567
1568
1569
1569
1570
1571
1572
1573
1573
1574
1575
1575
1576
1577
1577
1578
1579
1579
1580
1581
1582
1583
1583
1584
1585
1585
1586
1587
1587
1588
1589
1589
1590
1591
1592
1593
1593
1594
1595
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1603
1604
1605
1605
1606
1607
1607
1608
1609
1609
1610
1611
1612
1613
1613
1614
1615
1615
1616
1617
1617
1618
1619
1619
1620
1621
1622
1623
1623
1624
1625
1625
1626
1627
1627
1628
1629
1629
1630
1631
1632
1633
1633
1634
1635
1635
1636
1637
1637
1638
1639
1639
1640
1641
1642
1643
1643
1644
1645
1645
1646
1647
1647
1648
1649
1649
1650
1651
1652
1653
1653
1654
1655
1655
1656
1657
1657
1658
1659
1659
1660
1661
1662
1663
1663
1664
1665
1665
1666
1667
1667
1668
1669
1669
1670
1671
1672
1673
1673
1674
1675
1675
1676
1677
1677
1678
1679
1679
1680
1681
1682
1683
1683
1684
1685
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1693
1694
1695
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1703
1704
1705
1705
1706
1707
1707
1708
1709
1709
1710
1711
1712
1713
1713
1714
1715
1715
1716
1717
1717
1718
1719
1719
1720
1721
1722
1723
1723
1724
1725
1725
1726
1727
1727
1728
1729
1729
1730
1731
1732
1733
1733
1734
1735
1735
1736
1737
1737
1738
1739
1739
1740
1741
1742
1743
1743
1744
1745
1745
1746
1747
1747
1748
1749
1749
1750
1751
1752
1753
1753
1754
1755
1755
1756
1757
1757
1758
1759
1759
1760
1761
1762
1763
1763
1764
1765
1765
1766
1767
1767
1768
1769
1769
1770
1771
1772
1773
1773
1774
1775
1775
1776
1777
1777
1778
1779
1779
1780
1781
1782
1783
1783
1784
1785
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1793
1794
1795
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1803
1804
1805
1805
1806
1807
1807
1808
1809
1809
1810
1811
1812
1813
1813
1814
1815
1815
1816
1817
1817
1818
1819
1819
1820
1821
1822
1823
1823
1824
1825
1825
1826
1827
1827
1828
1829
1829
1830
1831
1832
1833
1833
1834
1835
1835
1836
1837
1837
1838
1839
1839
1840
1841
1842
1843
1843
1844
1845
1845
1846
1847
1847
1848
1849
1849
1850
1851
1852
1853
1853
1854
1855
1855
1856
1857
1857
1858
1859
1859
1860
1861
1862
1863
1863
1864
1865
1865
1866
1867
1867
1868
1869
1869
1870
1871
1872
1873
1873
1874
1875
1875
1876
1877
1877
1878
1879
1879
1880
1881
1882
1883
1883
1884
1885
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1893
1894
1895
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1903
1904
1905
1905
1906
1907
1907
1908
1909
1909
1910
1911
1912
1913
1913
1914
1915
1915
1916
1917
1917
1918
1919
1919
1920
1921
1922
1923
1923
1924
1925
1925
1926
1927
1927
1928
1929
1929
1930
1931
1932
1933
1933
1934
1935
1935
1936
1937
1937
1938
1939
1939
1940
1941
1942
1943
1943
1944
1945
1945

```

2.8.4. Estado de la base de datos.

2.8.4.1. Antes de la operación.

The screenshot shows the DBeaver interface with two tables open: 'articulos' and 'carrito'. The 'articulos' table has two rows: Prueba1 (id=1) and Llavero Chewbacca (id=2). The 'carrito' table has one row: articulo_id=1, cantidad=1. The database connection is t9-bd-2016630495.mysql.database.azure.com.

id	nombre	precio	cantidad
1	Prueba1	12.5	4
2	Llavero Chewbacca	100	2

articulo_id	cantidad
1	1

2.8.4.1. Despues de la operación.

The screenshot shows the DBeaver interface with the same two tables. The 'articulos' table remains the same. The 'carrito' table now has one row: articulo_id=2, cantidad=1. The database connection is t9-bd-2016630495.mysql.database.azure.com.

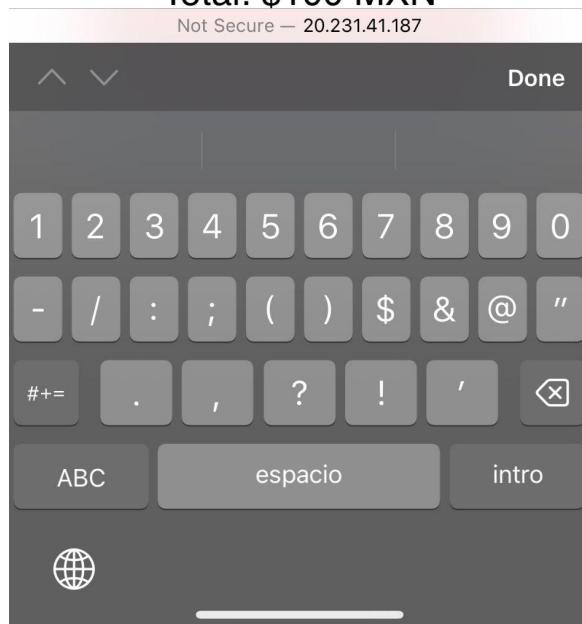
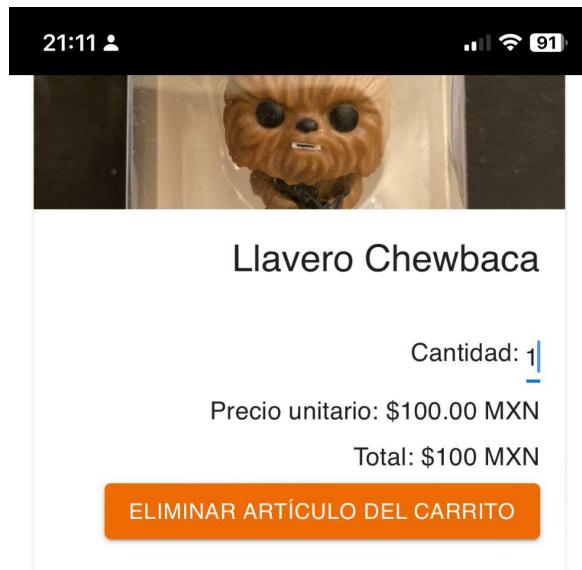
id	nombre	precio	cantidad
1	Prueba1	12.5	5
2	Llavero Chewbacca	100	2

articulo_id	cantidad
2	1

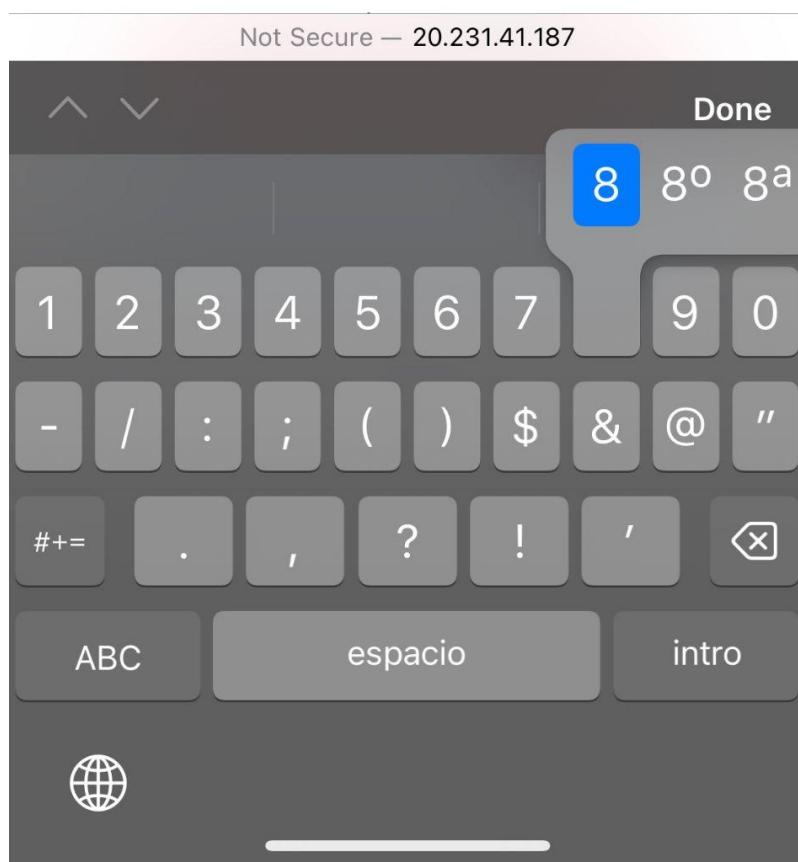
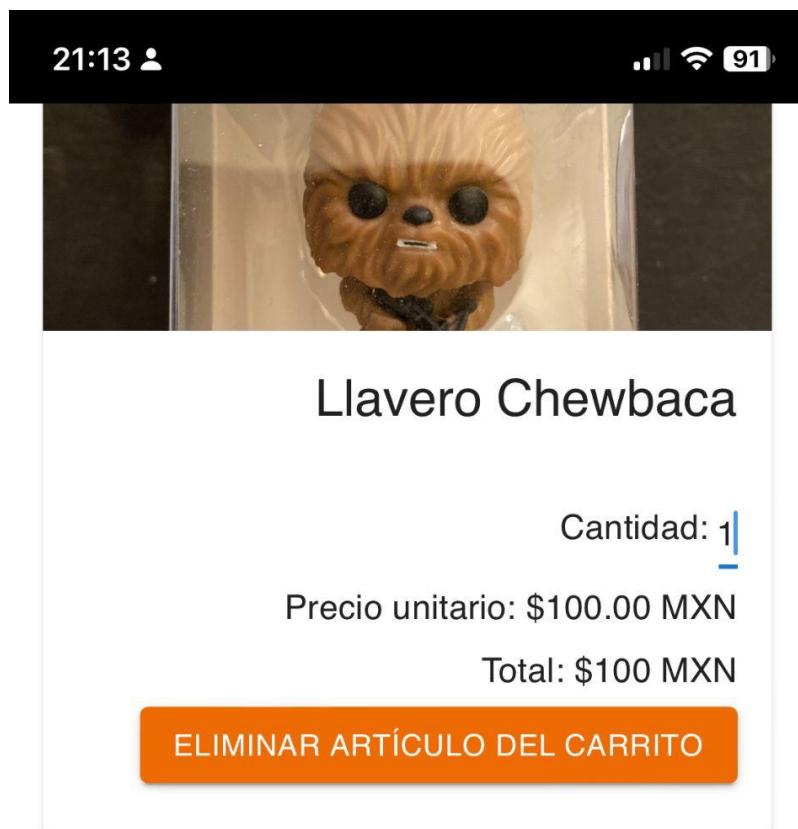
2.9. Actualizar cantidad de artículo en carrito de compra.

Para cada artículo en la pantalla "Artículos en el carrito" se deberá incluir un control que permita incrementar o decrementar la cantidad del artículo en el carrito. Al incrementar la cantidad se deberá verificar si hay suficientes artículos en la tabla "artículos", si no es así se deberá mostrar una ventana indicando esta situación. Al incrementar o decrementar la cantidad del artículo en el carrito se deberá actualizar (UPDATE) la cantidad en la tabla "articulos" y en la tabla "carrito_compra" dentro de una transacción.

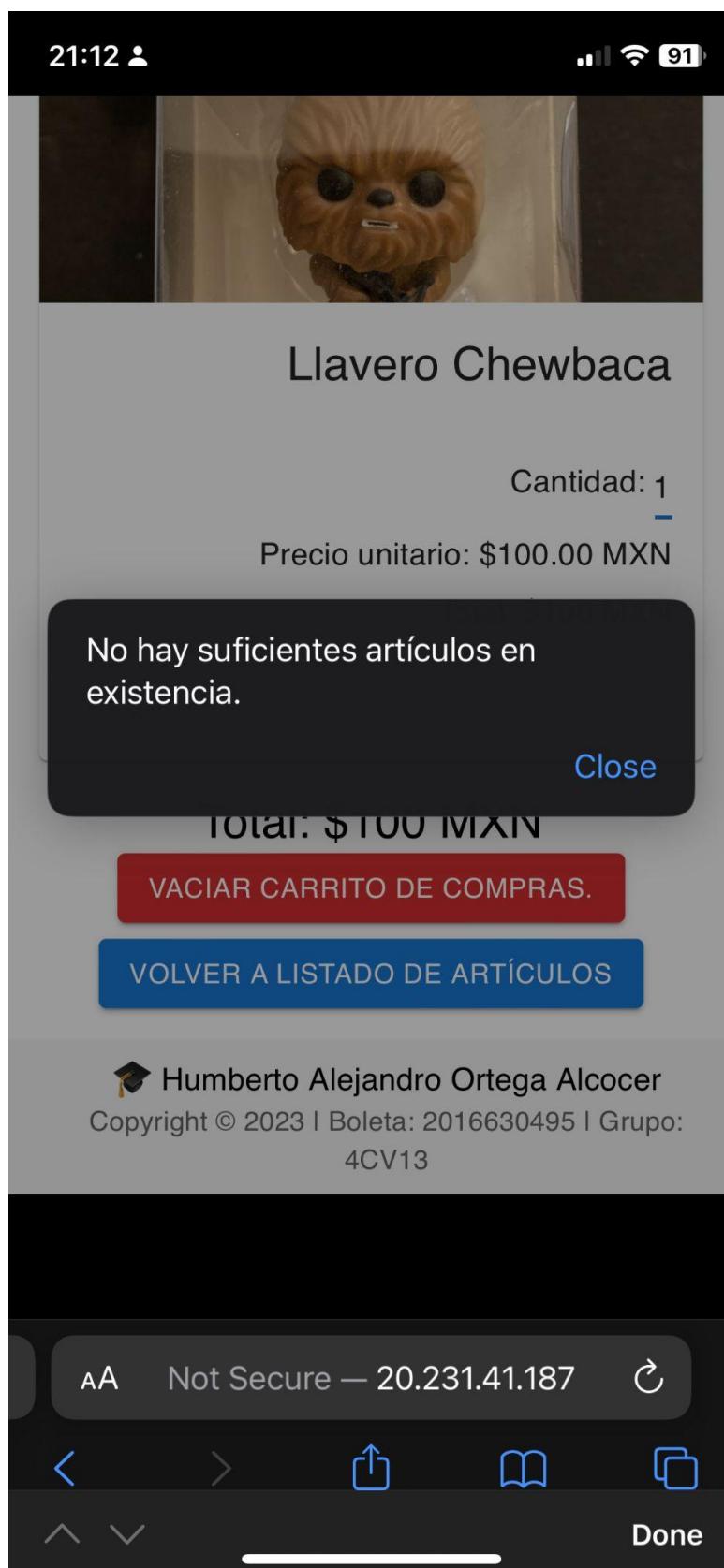
2.9.1 Campo para introducir la cantidad a modificar.



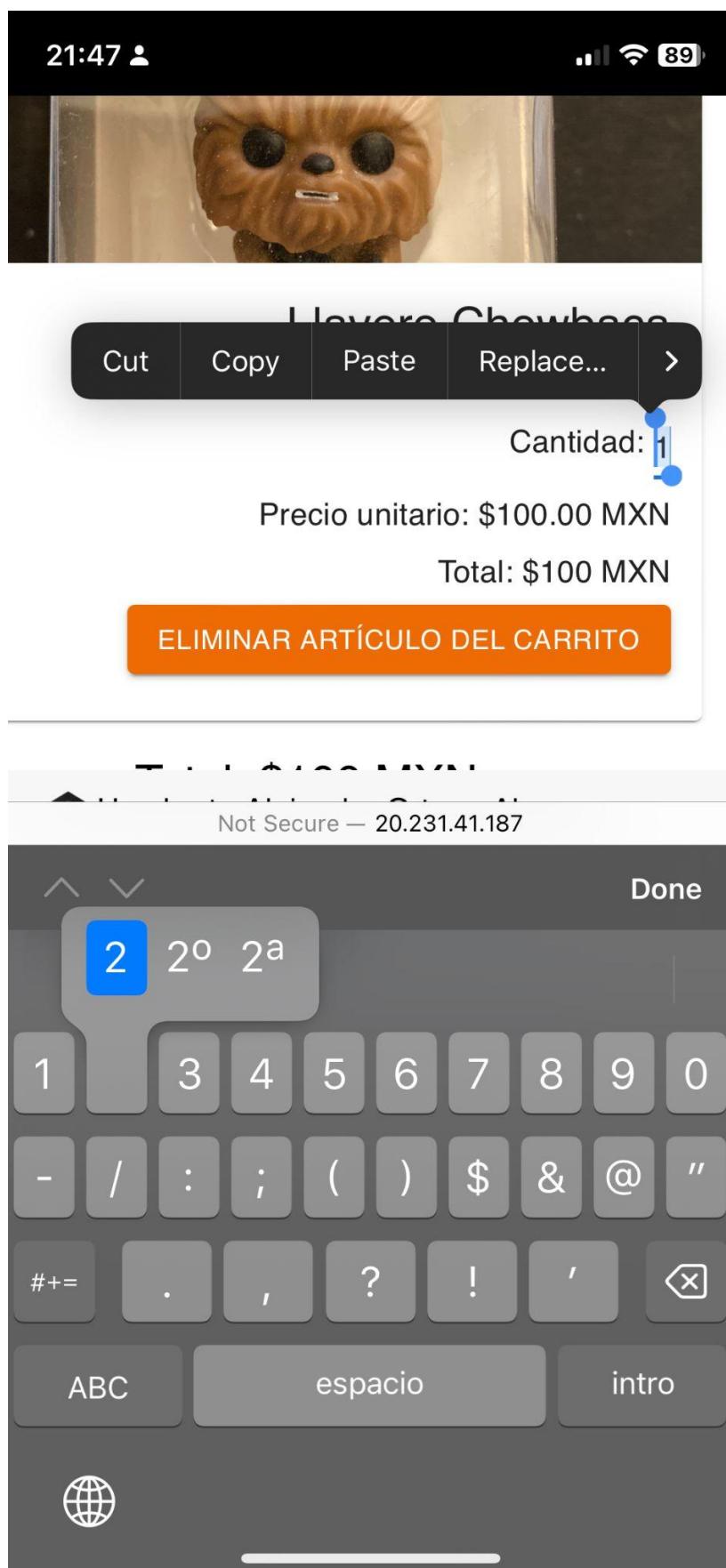
2.9.2. Introduciendo una cantidad inválida.



2.9.3. Ventana de validación incorrecta de cantidad de artículos.



2.9.5. Cambiar cantidad a un valor válido.



2.9.5. Código relevante a las transacciones.

Las operaciones a realizar difieren según sea el caso que el usuario quiera agregar elementos o quitarlos del carrito.

2.9.5.1. En caso de que sea aumentar la cantidad de elementos en el carrito.



The screenshot shows a code editor with a dark theme, displaying a file named `index.ts`. The code is written in TypeScript and contains logic for updating a shopping cart. It includes comments in Spanish explaining the purpose of each step. The code uses `sequelize` for database queries and `httpTrigger` annotations. The logic involves checking if there are enough items in stock, creating a transaction, updating the cart's quantity, and then updating the item's quantity in the database. Finally, it commits the transaction and retrieves the updated cart.

```
ts index.ts M x
tareas > tarea-9 > backend > EditarCarrito > Ts index.ts > [e] httpTrigger
  ...
  95
  96   if (carrito.cantidad < solicitud.cantidad) {
  97     // El usuario agregó más artículos al carrito. You, yesterday • tarea 9
  98     // Validamos que haya suficientes artículos en existencia.
  99     if (articulo.cantidad < solicitud.cantidad) {
100       context.res = {
101         status: 400,
102         body: "No hay suficientes artículos en existencia.",
103       };
104     } else {
105       // Creamos una transacción.
106       const transaction = await sequelize.transaction();
107
108       // Actualizamos la cantidad de artículos en el carrito.
109       await sequelize.query(
110         "UPDATE carrito SET cantidad = :cantidad WHERE articulo_id = :articulo_id",
111         {
112           replacements: {
113             cantidad: solicitud.cantidad,
114             articulo_id: solicitud.articulo_id,
115           },
116           type: QueryTypes.UPDATE,
117           transaction,
118         }
119       );
120
121       // Actualizamos la cantidad de artículos en existencia.
122       await sequelize.query(
123         "UPDATE artículos SET cantidad = :cantidad WHERE id = :id",
124         {
125           replacements: {
126             cantidad: articulo.cantidad -
127               (solicitud.cantidad - carrito.cantidad),
128             id: solicitud.articulo_id,
129           },
130           type: QueryTypes.UPDATE,
131           transaction,
132         }
133       );
134
135       // Confirmamos la transacción.
136       await transaction.commit();
137
138
139       // Obtenemos el carrito de compras.
140       const carritoRespuesta: Carrito = {
```

2.9.5.2. En caso de que sea disminuir la cantidad de elementos en el carrito.

The screenshot shows a code editor with a dark theme, displaying a file named `index.ts`. The code is part of a Node.js application using Express and Sequelize. It handles a POST request to update the quantity of items in a shopping cart. The code uses transactions to ensure data consistency. It updates the cart and then updates the inventory. Finally, it commits the transaction and retrieves the updated cart data.

```
TS index.ts M X
tareas > tarea-9 > backend > EditarCarrito > ts index.ts > [o] httpTrigger
142     cantidad: solicitud.cantidad,
143     articulo: articulo,
144     };
145
146     // Regresamos el carrito de compras.
147     context.res = {
148         status: 200,
149         body: carritoRespuesta,
150     };
151 }
152 } else {
153     // El usuario quitó artículos del carrito.
154     // Creamos una transacción.
155     const transaction = await sequelize.transaction();
156
157     // Actualizamos la cantidad de artículos en el carrito.
158     await sequelize.query(
159         "UPDATE carrito SET cantidad = :cantidad WHERE articulo_id = :articulo_id",
160         {
161             replacements: {
162                 cantidad: solicitud.cantidad,
163                 articulo_id: solicitud.articulo_id,
164             },
165             type: QueryTypes.UPDATE,
166             transaction,
167         }
168     );
169
170     // Actualizamos la cantidad de artículos en existencia.
171     await sequelize.query(
172         "UPDATE articulos SET cantidad = :cantidad WHERE id = :id",
173         {
174             replacements: {
175                 cantidad: articulo.cantidad + (carrito.cantidad - solicitud.cantidad),
176                 id: solicitud.articulo_id,
177             },
178             type: QueryTypes.UPDATE,
179             transaction,
180         }
181     );
182
183     // Confirmamos la transacción.
184     await transaction.commit();| You, yesterday • tarea 9
185
186
187     // Obtenemos el carrito de compras.
188     const carritoRespuesta: Carrito = {
```

2.9.6. Estado de la base de datos.

2.9.6.1. Estado de la base de datos antes de la operación.

The screenshot shows the DBBeaver interface with two open tables: 'articulos' and 'carrito'.
Table 'articulos':
Columns: id, nombre, precio, cantidad, descripcion, fotografía.
Data:
id	nombre	precio	cantidad	descripcion	fotografía
1	Prueba1	12.5	5	Esta es una prueba	data:image/jpeg;base64,/9
2	Llavero Chewbacca	100	2	Un hermoso llavero de Chewbacca	data:image/jpeg;base64,/9
Table 'carrito':					
Columns: articulo_id, cantidad.					
Data:					
articulo_id	cantidad				
1	1				
2	2				
The bottom status bar indicates: 2 row(s) fetched - 855ms (5ms fetch), on 2023-06-10 at 21:44:58.

2.9.6.2. Estado de la base de datos después de la operación.

The screenshot shows the DBBeaver interface with the same two tables: 'articulos' and 'carrito'.
Table 'articulos':
Columns: id, nombre, precio, cantidad, descripcion, fotografía.
Data:
id	nombre	precio	cantidad	descripcion	fotografía
1	Prueba1	12.5	5	Esta es una prueba	data:image/jpeg;base64,/9
2	Llavero Chewbacca	100	2	Un hermoso llavero de Chewbacca	data:image/jpeg;base64,/9
Table 'carrito':					
Columns: articulo_id, cantidad.					
Data:					
articulo_id	cantidad				
1	2				
The bottom status bar indicates: 1 row(s) fetched - 64ms, on 2023-06-10 at 21:49:04.

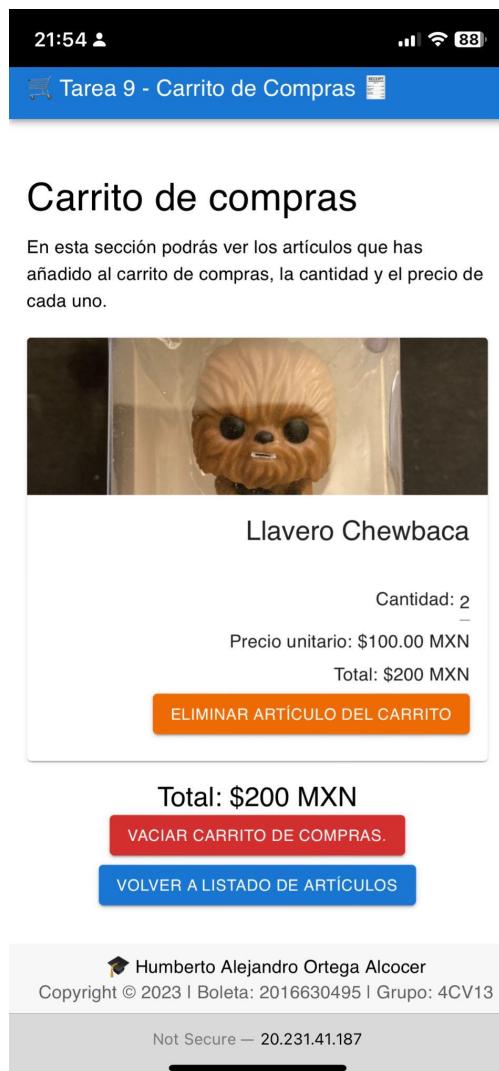
2.10. Eliminar todos los artículos en el carrito de compra.

La pantalla "Artículos en el carrito" deberá tener un botón "Eliminar carrito de compra", al dar clic a este botón se deberá regresar la cantidad de cada artículo a la tabla "articulos" y se deberá borrar todos los registros de la tabla "carrito_compra".

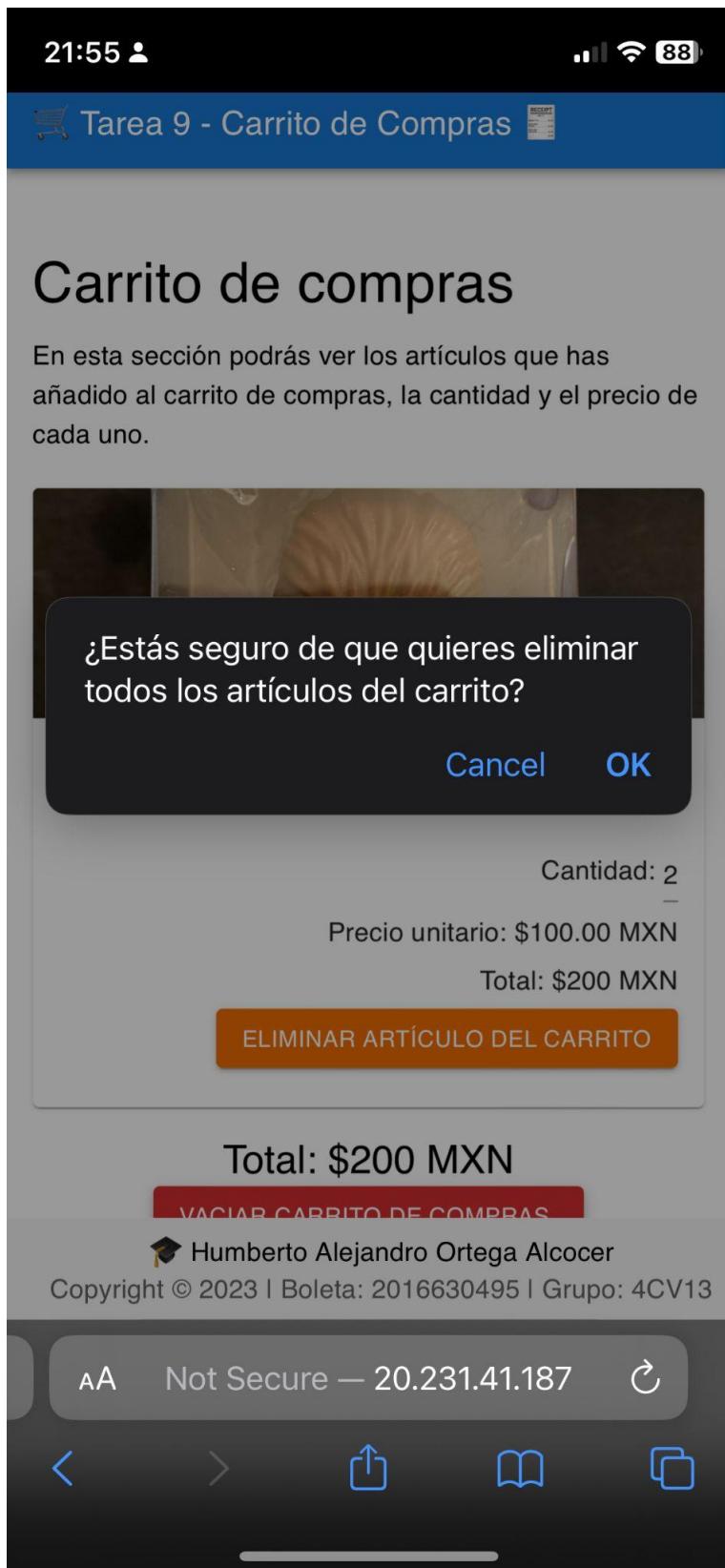
Se deberá desplegar una ventana preguntando al usuario si está seguro de eliminar el carrito de compra, incluyendo dos botones "Si" y "No"; si el usuario presiona el botón "Si" se deberá eliminar el carrito de compra.

Las actualizaciones (UPDATE) de la tabla "articulos" y la eliminación (DELETE) de los registros de la tabla "carrito_compra" deberán realizarse dentro de una transacción.

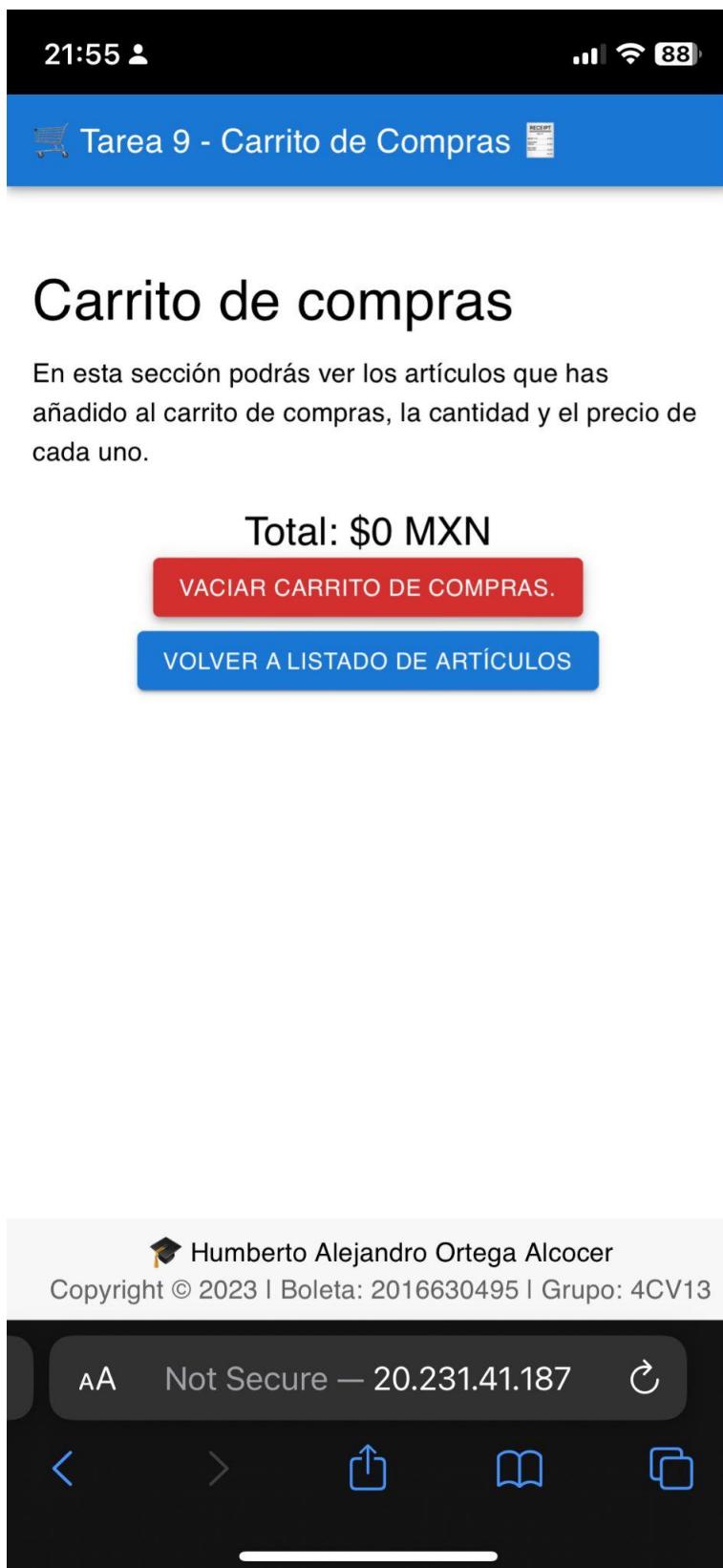
2.10.1. Botón de "eliminar carrito de compra".



2.10.2. Ventana de confirmación de la eliminación del carrito de compra.



2.10.3. Carrito de compras después de la operación.



2.10.3. Código relacionado a las transacciones.

The screenshot shows a code editor with a dark theme, displaying a file named `index.ts`. The code is part of a Node.js application using Express and Sequelize. It handles a POST request to remove an item from a shopping cart. The code starts by validating if the item exists in the cart. If it does, it creates a transaction, updates the item's quantity in the database, and then removes the item from the cart. Finally, it commits the transaction and sends a success response. If an error occurs during the process, it catches the error and sends a general error response.

```
ts index.ts M ×
tareas > tarea-9 > backend > EliminarCarrito > ts index.ts [o] httpTrigger
65
66 // Validamos que el artículo exista.
67 if (carritos.length <= 0) {
68   context.res = {
69     status: 404,
70     body: "No existe el artículo en el carrito de compras",
71   };
72 } else {
73   // Crear una transacción.
74   const transaction = await sequelize.transaction();
75
76   // Agregar de vuelta las unidades del carrito al artículo.
77   await sequelize.query(
78     "UPDATE articulos SET cantidad = cantidad + :cantidad WHERE id = :id",
79     {
80       replacements: [
81         { id: carritos[0].articulo_id, cantidad: carritos[0].cantidad },
82       ],
83       type: QueryTypes.UPDATE,
84       transaction,
85     }
86   );
87
88   // Eliminar el artículo del carrito de compras.
89   await sequelize.query(
90     "DELETE FROM carrito WHERE articulo_id = :articulo_id",
91     {
92       replacements: [{ articulo_id: id }],
93       type: QueryTypes.DELETE,
94       transaction,
95     }
96   );
97
98   // Ejecutar la transacción.
99   await transaction.commit();
100
101 // Responder con un 200.
102 context.res = {
103   status: 200,
104   body: "Artículo eliminado del carrito de compras",
105 };
106
107 // Cerrar la conexión a la base de datos.
108 await sequelize.close();
109
110 } catch (error) {
111   context.res = {
```

2.10.4. Estado de la base de datos.

2.10.4.1. Estado de la base de datos antes de la operación.

The screenshot shows the DBeaver interface with two tables open:

- articulos** table:
 - Grid view: 2 rows (id 1, nombre 'Prueba1', precio 12.5, cantidad 5, descripcion 'Esta es una prueba'), (id 2, nombre 'Llavero Chewbacca', precio 100, cantidad 2, descripcion 'Un hermoso llavero')
 - Text view: 2 rows (id 1, nombre 'Prueba1', precio 12.5, cantidad 5, descripcion 'Esta es una prueba'), (id 2, nombre 'Llavero Chewbacca', precio 100, cantidad 2, descripcion 'Un hermoso llavero')
- carrito** table:
 - Grid view: 1 row (articulo_id 2, cantidad 2)
 - Text view: 1 row (articulo_id 2, cantidad 2)

Bottom status bar: 200 rows fetched - 673ms (14ms fetch), on 2023-06-10 at 21:49:08

2.10.4.2. Estado de la base de datos después de la operación.

The screenshot shows the DBeaver interface with the same two tables, but the 'articulos' table has been updated:

- articulos** table:
 - Grid view: 2 rows (id 1, nombre 'Prueba1', precio 12.5, cantidad 5, descripcion 'Esta es una prueba'), (id 2, nombre 'Llavero Chewbacca', precio 100, cantidad 4, descripcion 'Un hermoso llavero')
 - Text view: 2 rows (id 1, nombre 'Prueba1', precio 12.5, cantidad 5, descripcion 'Esta es una prueba'), (id 2, nombre 'Llavero Chewbacca', precio 100, cantidad 4, descripcion 'Un hermoso llavero')
- carrito** table:
 - Grid view: No data
 - Text view: No data

Bottom status bar: 200 rows fetched - 1.161s (18ms fetch), on 2023-06-10 at 21:58:00

2.11. Botón para volver a la vista de artículos.

La pantalla "Artículos en el carrito" deberá tener un botón "Seguir comprando" el cual deberá permitir regresar a la pantalla "Compra de artículos".

2.11.1. Botón en página de carrito.



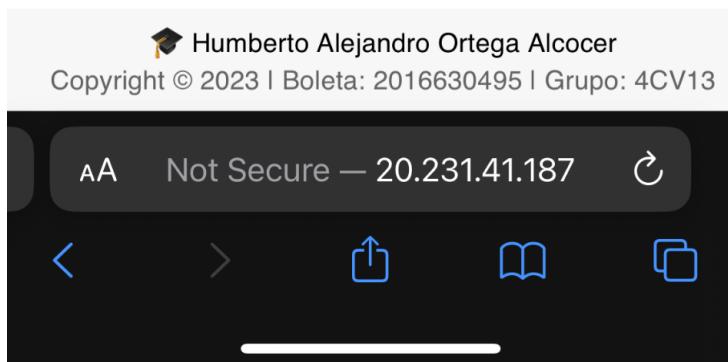
Carrito de compras

En esta sección podrás ver los artículos que has añadido al carrito de compras, la cantidad y el precio de cada uno.

Total: \$0 MXN

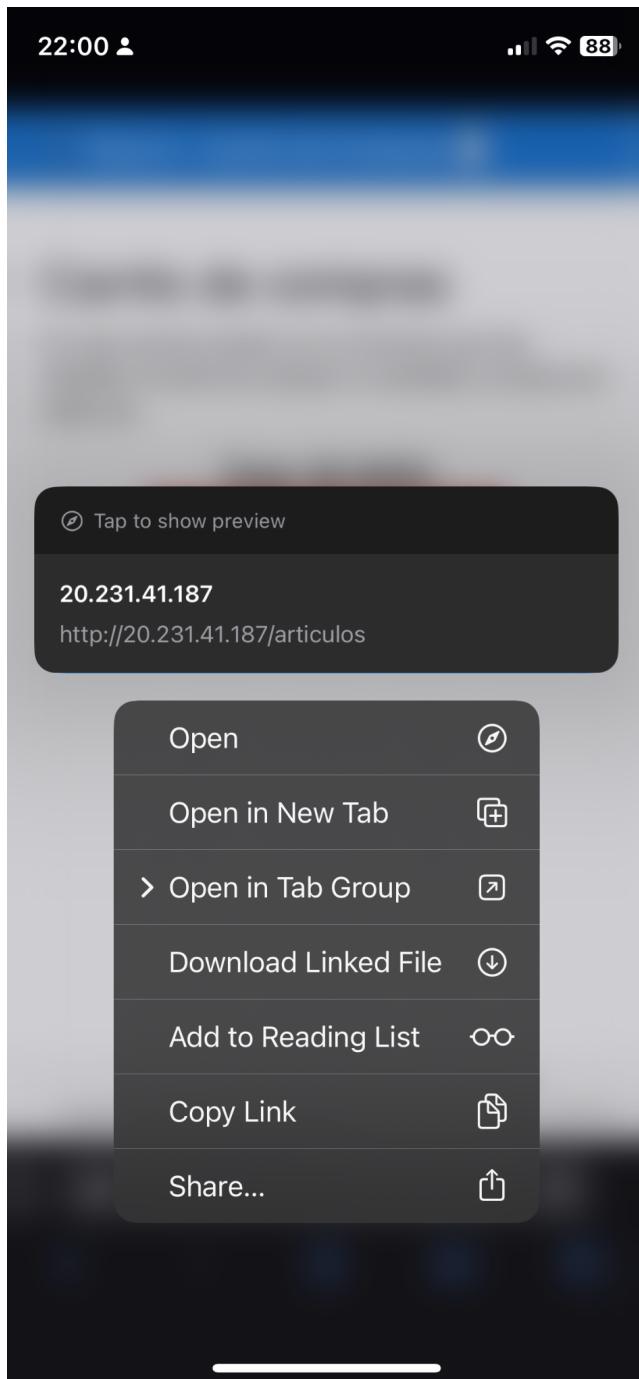
VACIAR CARRITO DE COMPRAS.

VOLVER A LISTADO DE ARTÍCULOS



2.11.2. Prueba de URL de redireccionamiento.

En iOS, al mantener presionado un botón de una página web aparece un menú contextual con la URL a la que dirige así como acciones para guardarlo o abrirlo en otra pestaña y demás.



3. Conclusión

En el desarrollo de esta práctica se tuvo que echar mano de conocimientos de toda la carrera. El cambio de utilizar una versión del código en un servidor como lo fue con Java a utilizar pequeñas funciones *Serverless* en Node.js fue un gran reto puesto que no sólamente la traducción entre los lenguajes supuso una barrera, sino que, además, entender el marco de trabajo de las funciones de Azure de forma local fue muy entretenido.

La tarea 7 me dejó con mal sabor de boca por el frontend que utilicé, aproveché esta entrega para enmendar mi error. Y mi teoría fue correcta, es más simple usar React en estas situaciones que tratar de usar JavaScript *solito*. El poder manipular el estado de los componentes y los distintos errores y complejidades que supone interactuar con servicios web me ayudó a que mi código estuviese más limpio y ordenado.

Desplegar toda la aplicación en general también supuso un reto, hubo que utilizar múltiples pedazos de infraestructura de Azure para lograrlo: almacenamiento, funciones, bases de datos, máquinas virtuales, un montón de direcciones IP, apertura de puertos, gestión de accesos, etcétera.

Me dió mucho gusto poder terminar la tarea 9 porque siento que mi front de la 7 estaba muy feo, además de que varias funciones ahí tenían implementaciones poco elegantes.

Creo que fue muy divertido escribir este reporte en lo particular, me ayudó a comprender de mejor forma el esfuerzo que requiere desplegar cosas y cómo puedo navegar los proveedores de nube interconectando servicios.

– Humberto Alejandro Ortega Alcocer.