

# Plataforma de E-Commerce

Bienvenido a la Plataforma de E-Commerce! Este proyecto está construido de la siguiente manera: - Angular, Javascript, Typescript, HTML y CSS para el frontend. - Django y Python para el backend. - La BD esta configurada para PostgreSQL en su versión 16.0. - Utiliza un server Nginx para la seguridad de la red y el uso de SSL. - Está contenedorizado utilizando Docker y Docker Compose.

## Tabla de Contenidos

- [Instalación](#)
  - [Instalando Docker Compose](#)
    - [Windows](#)
    - [macOS](#)
    - [Linux](#)
- [Ejecutar los Servicios](#)
- [Estructura del Proyecto](#)
- [Tecnologías usadas](#)
- [Descripción del Proyecto](#)
- [Seguridad](#)
- [Chatbot](#)
- [Despliegue en AWS](#)

## Instalación

### Instalando Docker Compose

#### Windows

1. **Instalar Docker Desktop :**
  - Descarga e instala Docker Desktop desde [Docker Hub](#).
  - Sigue las instrucciones de instalación y asegúrate de que Docker Desktop esté ejecutándose.
2. **Verificar la Instalación :**
  - Abre el Símbolo del sistema (cmd) o PowerShell.
  - Ejecuta el siguiente comando para verificar la instalación:  
`docker-compose --version`

#### macOS

1. **Instalar Docker Desktop :**
  - Descarga e instala Docker Desktop desde [Docker Hub](#).
  - Sigue las instrucciones de instalación y asegúrate de que Docker Desktop esté ejecutándose.
2. **Verificar la Instalación :**
  - Abre la Terminal.
  - Ejecuta el siguiente comando para verificar la instalación:  
`docker-compose --version`

#### Linux

1. **Instalar Docker :**
  - Sigue las instrucciones para instalar Docker desde [la documentación oficial de Docker](#).
2. **Instalar Docker Compose :**

- Ejecuta los siguientes comandos para descargar e instalar Docker Compose:  

```
sudo curl -L "https://github.com/docker/compose/releases/download/  
sudo chmod +x /usr/local/bin/docker-compose
```

### 3. Verificar la Instalación:

- Ejecuta el siguiente comando para verificar la instalación:  

```
docker-compose --version
```

## Ejecutar los Servicios

Una vez que Docker Compose esté instalado, sigue estos pasos para iniciar los servicios:

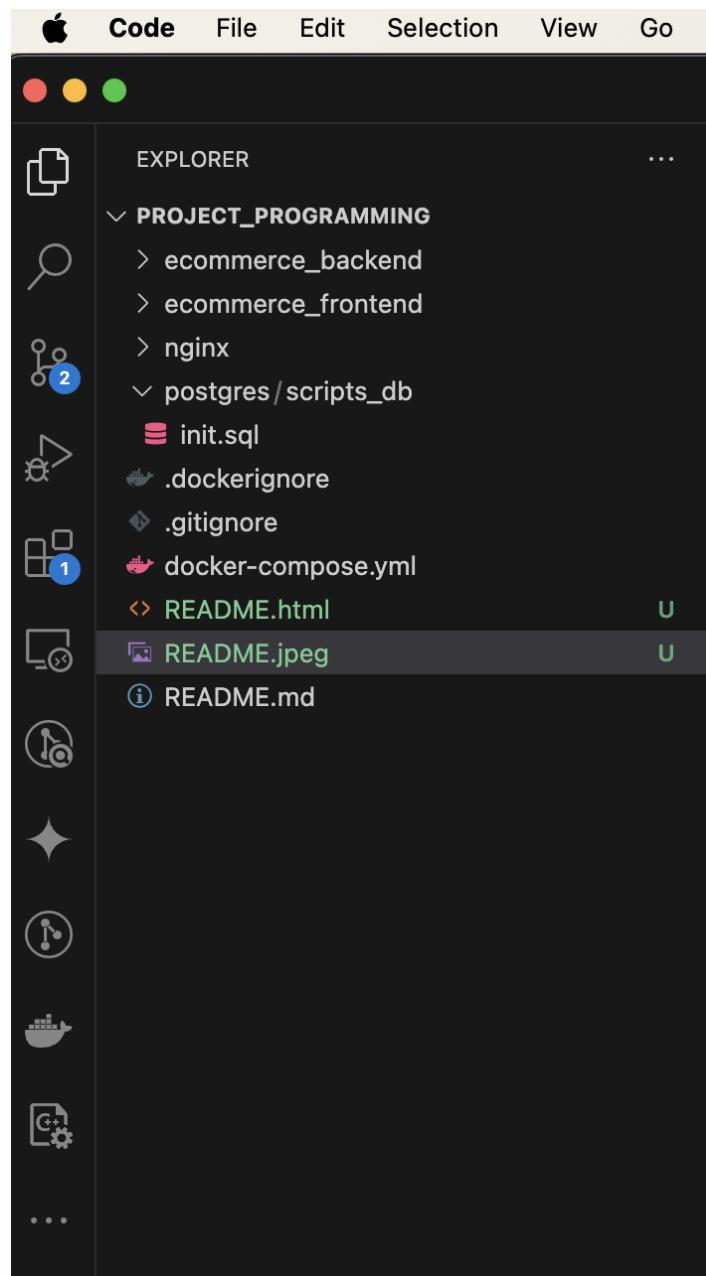
### 1. Clonar el Repositorio:

```
git clone https://github.com/rcalezreyes/project_programming.git  
cd project_programming
```

### 2. Ejecución del sistema:

Para levantar el sistema en ambiente docker solo tiene que ejecutar este comando desde la carpeta de la aplicación: sh docker-compose up -d --build Para detener el servicio: sh docker-compose down Finalmente accediendo a la URL: <https://127.0.0.1/> puede levantar la aplicación en ambiente docker

## Estructura del proyecto



Estructura

Muestra las carpetas de las 4 imágenes del proyecto: backend, frontend, nginx y postgres En el caso de nginx, se deben generar los certificados si se correrá local. Estos son los comandos:  
openssl genpkey -algorithm RSA -out nginx/certificates/DNS/dns\_key.pem

```
openssl req -new -key nginx/certificates/DNS/dns_key.pem -x509 -days  
365 -out nginx/certificates/DNS/dns_cert.crt
```

```
openssl pkcs12 -export -out nginx/certificates/DNS/dns_cert.pfx -inkey  
nginx/certificates/DNS/dns_key.pem -in  
nginx/certificates/DNS/dns_cert.crt
```

```
openssl pkcs12 -in nginx/certificates/DNS/dns_cert.pfx -out  
nginx/certificates/DNS/dns_cert.pem -nodes -clcerts
```

```
openssl pkcs12 -in nginx/certificates/DNS/dns_cert.pfx -out  
nginx/certificates/DNS/dns_key.pem -nodes -nocerts
```

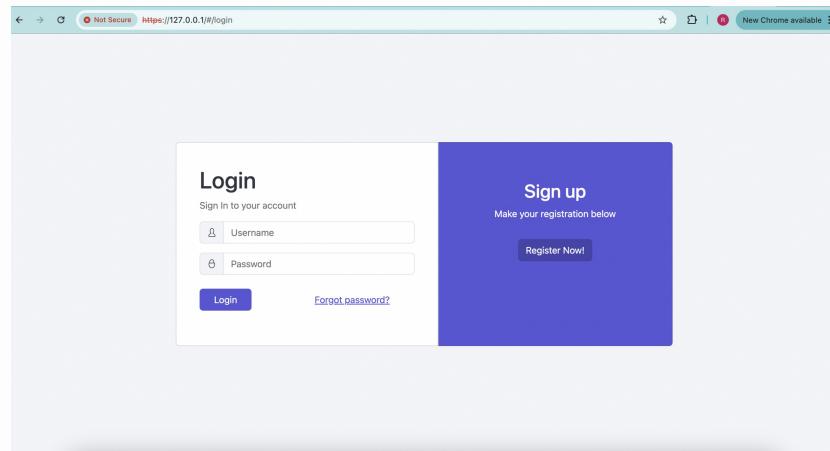
```
openssl pkcs12 -in nginx/certificates/DNS/dns_cert.pfx -out
```

```
nginx/certificates/DNS/dns_chain.pem -nodes -nokeys -clcerts
```

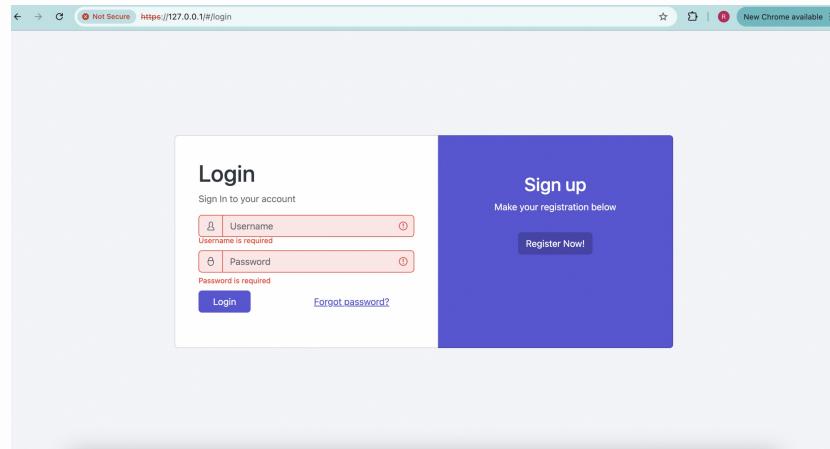
## Tecnologías usadas

Lenguajes de Programación: HTML5, Javascript y Python  
Frameworks: Angular y Django  
Estilos: Material UI y CSS  
Otras librerías: Sweetalert (Swal)

## Descripción del proyecto



Esta es nuestra página de inicio, cuenta con el login, y el acceso mediante https protocol.



Se muestra un ejemplo de la validacion del login usando las librerias de validación de Material UI.

Not Secure https://127.0.0.1/#/register

## Register

Create your account

First Name:  Last Name:

Username:  Email:

Password:  Repeat password:

Address:

Phone:  City:

Zip Code:  Select country:

Se muestra el formulario desarrollado en Angular con Material UI para el registro

Not Secure https://127.0.0.1/#/register

## Register

Create your account

First Name:  (Required)

Last Name:  (Required)

Username:  (Required)

Email:  (Email is required)

Password:  (Required)

Repeat password:  (Confirm password is required)

Address:  (Required)

Phone:  (Phone is required)

City:  (City is required)

Zip Code:  (Required)

Select country:  (Country is required)

Se muestra un ejemplo de las validaciones de tipo required

Not Secure https://127.0.0.1/#/register

## Register

Create your account

First Name: Roberto Carlos Last Name: Gonzalez Reyes

Username: rcglezreyes@gmail.com Email: rcglezreyes@gmail.com  
Email already exists in Users  
Email already exists in Customers

Password: ..... Repeat password: .....

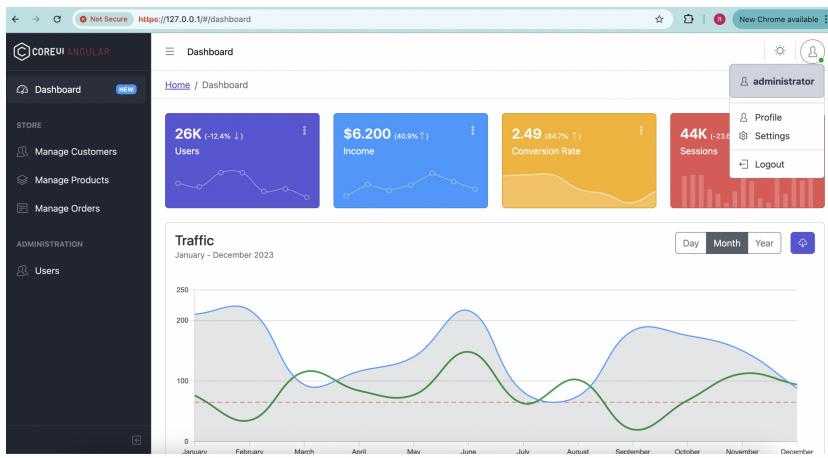
Address: 850 W 74th Street Apt 207

Phone: 7868677370 City: Hialeah

Zip Code: sdfds (Required)

Select country: United States

Algunos ejemplos de validacion personalizadas: En el caso del email, ahí se observa como se realiza una subscripción a un endpoint de nuestro backend que devuelve la existencia o no del email en las tablas Customer y User. En el caso del zip code, se customizó una validación para que solo aceptara valores numéricos. En el caso de los campos passwords, se personalizó una validación para cuando ambos no coincidieran.



El admin del proyecto tendrá acceso a: Manage Customers, Manage Products, Manage Orders y User.

#	Image	Name	Price	Stock	Available	Category	Actions
1		Apple Newton Max	\$705.48	4	YES	Electronics	
2		Beef Vit	\$29.99	35	YES	Food	
3		Best Alarm System	\$99.14	10	YES	Electronics	
4		Best T-Shirt	\$10.79	107	YES	Clothing	

Módulo de Manage Products List

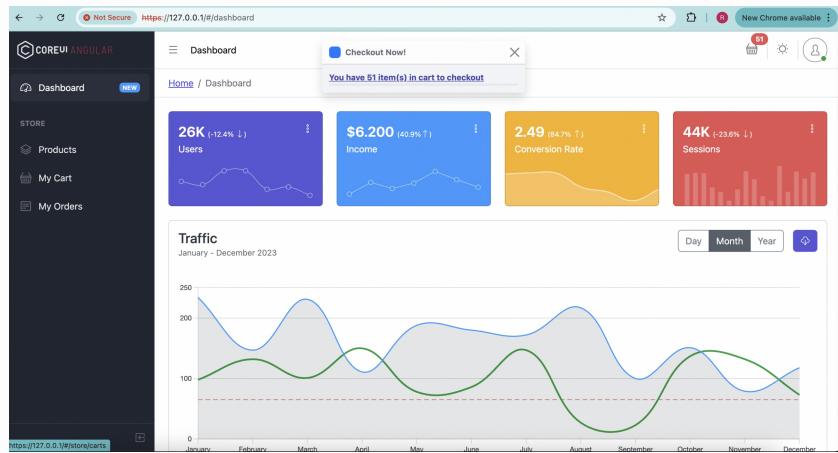
The screenshot shows the 'Add Product' form in the CoreUI Angular store. It includes fields for 'Name' (Apple Newton Max), 'Price' (\$705.48), 'Stock' (4), 'Category' (Select category), 'Description' (Apple Newton Max), and an 'Image' section with a file input field ('Choose File') and a preview area showing a smartphone image. There are 'Save' and 'Cancel' buttons at the bottom.

Módulo de Manage Products Add

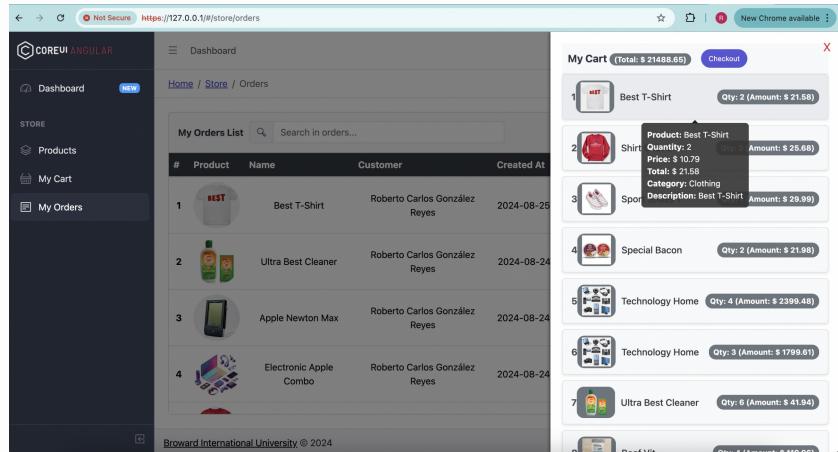
The screenshot shows the 'Edit Product' form for the 'Apple Newton Max' item. It displays the current values: Name (Apple Newton Max), Price (\$705.48), Stock (4), Category (Select category), Description (Apple Newton Max), and an 'Image' section with a file input field ('Choose File') and a preview area showing a smartphone image. There are 'Update' and 'Cancel' buttons at the bottom.

## Módulo de Manage Products Edit

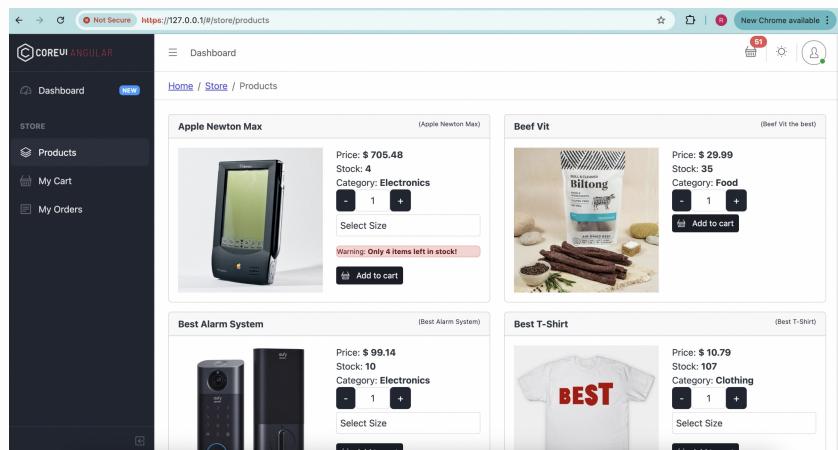
Mostramos un ejemplo del CRUD de Manage Products.



El customer del proyecto tendrá acceso a: Products (con otro formato), My Cart y My Orders. Se mostrará un Toast en el top-center de la página principal indicándole al customer que tiene artículos pendientes en su carrito (cart), con un link directo al checkout.



Aquí el customer podrá ver su lista de artículos en su carrito, así como el detalle de cada uno al hacer mouse over (tooltip) y el monto total de esa selección. También tendrá un link a checkout directamente.



Aquí se observa la vista de productos para los clientes, en este caso el cliente podrá seleccionar la cantidad (no más de lo que hay en stock), podrá seleccionar el size (en caso que el producto lo tenga) y le mostrará un warning para aquellos productos que tienen baja disponibilidad. En el botón Add to Cart, podrá añadirlo directamente a su carrito.

Unselect	#	Product	Name	Created At	Quantity	Price	Amount	Actions
<input checked="" type="checkbox"/>	1		Best T-Shirt	2024-08-25 00:34	2	\$ 10.79	\$ 21.58	
<input checked="" type="checkbox"/>	2		Shirt Coat 2	2024-08-25 00:33	2	\$ 12.84	\$ 25.68	
<input checked="" type="checkbox"/>	3		Sport Shoes 1	2024-08-24 01:26	1	\$ 29.99	\$ 29.99	
<input checked="" type="checkbox"/>	4		Special Bacon	2024-08-22 23:32	2	\$ 10.99	\$ 21.98	

Módulo de Cart

Unselect	#	Product	Name	Created At	Quantity	Price	Amount	Actions
<input checked="" type="checkbox"/>	1		Best T-Shirt	2024-08-25 00:34	2	\$ 10.79	\$ 21.58	
<input checked="" type="checkbox"/>	2		Shirt Coat 2	2024-08-25 00:33	2	\$ 12.84	\$ 25.68	
<input checked="" type="checkbox"/>	3		Sport Shoes 1	2024-08-24 01:26	1	\$ 29.99	\$ 29.99	
<input checked="" type="checkbox"/>	4		Special Bacon	2024-08-22 23:32	2	\$ 10.99	\$ 21.98	

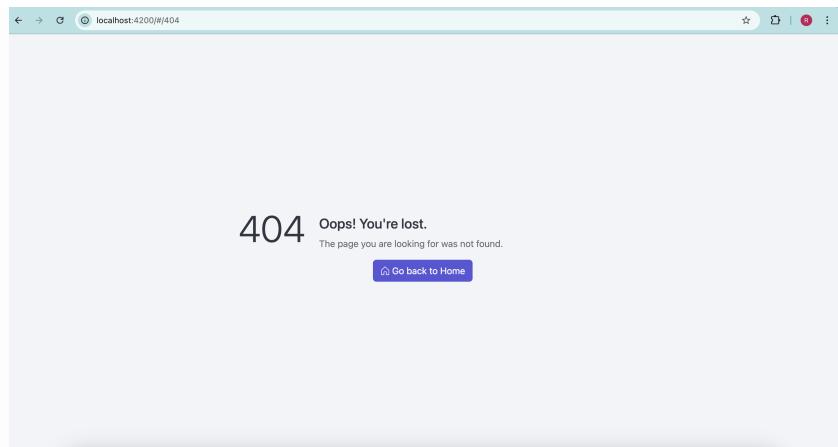
Módulo de Cart 2

Aquí se observa la vista del carrito del customer, aquí en la primera columna puede seleccionar o deseleccionar todos, o uno por uno. El monto real se va reflejando dinámicamente en la parte superior. Y al dar click en checkout, los productos seleccionados automáticamente se convierten en orders.

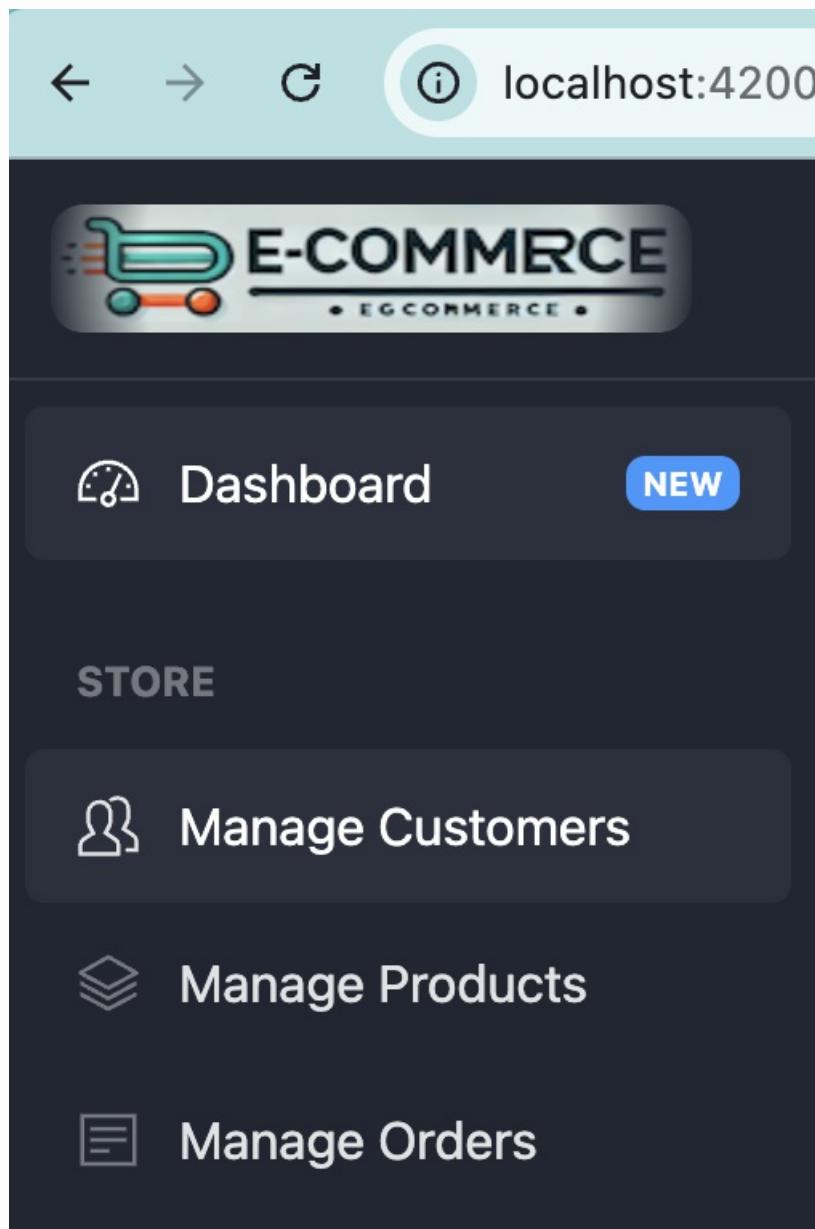
#	Product	Name	Customer	Created At	Amount	Delivery At
1		Best T-Shirt	Roberto Carlos González Reyes	2024-08-25 12:58	\$ 32.37	6215 Monroe Place, West New York, 07093
2		Ultra Best Cleaner	Roberto Carlos González Reyes	2024-08-24 18:31	\$ 20.97	6215 Monroe Place, West New York, 07093
3		Apple Newton Max	Roberto Carlos González Reyes	2024-08-24 18:30	\$ 1410.96	6215 Monroe Place, West New York, 07093
4		Electronic Apple Combo	Roberto Carlos González Reyes	2024-08-24 18:28	\$ 3292.68	6215 Monroe Place, West New York, 07093

Aquí se observan las órdenes del customer y el monto total gastado en los productos que ha comprado.

## Seguridad



Cuando se intenta acceder a una página a la cual no se tiene permiso o no existe, se muestra este error 404 con un link de redirección hacia el dashboard. Por ejemplo, si el usuario de role user intenta acceder al Manage Customers, se mostrará este error.



## ADMINISTRATION

 Users

Accesos para el rol de administrator



 Dashboard

NEW

STORE

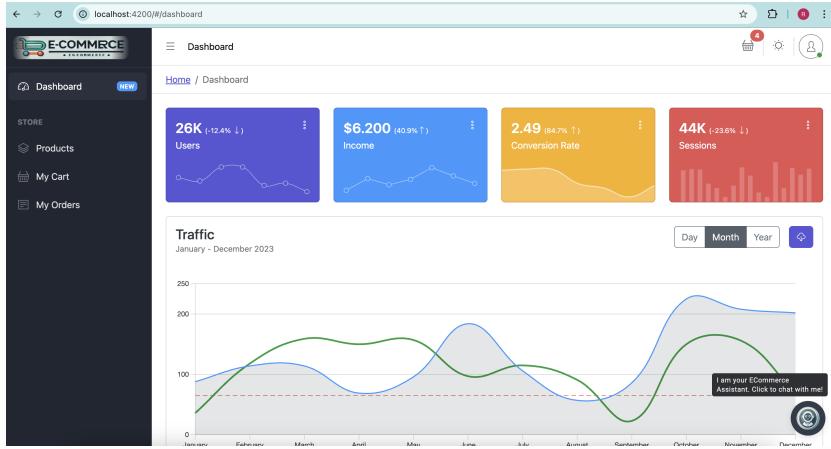
 Products

 My Cart

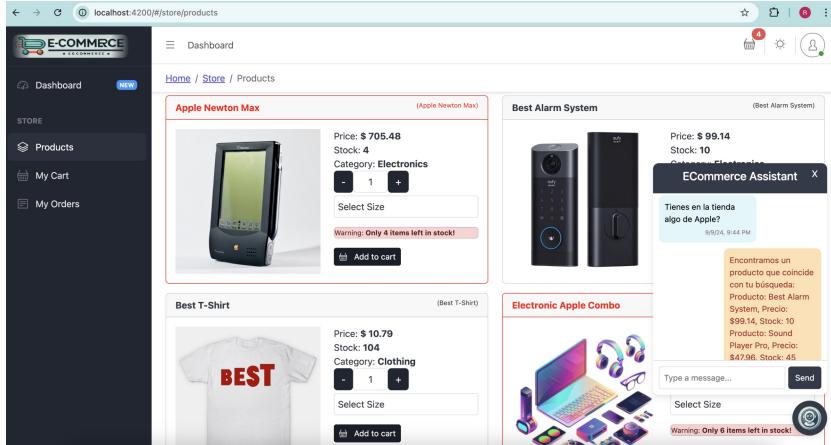
 My Orders

Accesos para el rol de user

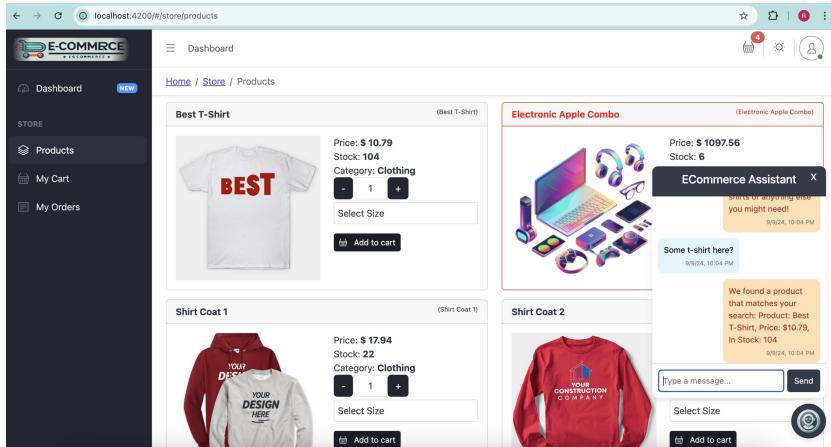
**Chatbot**



Se creó un chatbot usando la API de OpenAI con su modelo gpt-4o-mini. Dicho componente es la parte inferior derecha de la pantalla con un zIndex superior para que sea visible y accesible en todos los componentes de la aplicación



Se creó una función que devuelve en dependencia del mensaje escrito por el usuario la información de los productos relacionados con el mismo, su precio y stock. Dicha función también detecta el idioma



Ejemplo de un mensaje en inglés.

## Despliegue en AWS

Route 53 > Hosted zones > reyes-fullstack-dev.com

**Hosted zone details**

**Records (2)**

Name	Type	Value
ecommerce-frontend	A	10.0.0.1
ecommerce-backend	A	10.0.0.2

Se creó un dominio en AWS para el despliegue de nuestra aplicación con una zona host llamada `ecommerce.reyes-fullstack-dev.com`.

EC2 > Load balancers

Name	DNS name	State	VPC ID	Availability Zones	Type
ecommerce-frontend	ecommerce-frontend-109...	Active	vpc-0e627a537e62f20...	3 Availability Zones	application
ecommerce-backend	ecommerce-backend-1866...	Active	vpc-0e627a537e62f20...	3 Availability Zones	application

Se crearon dos load balancers para el despliegue de nuestros servicios, frontend y backend.

EC2 > Target groups

Name	ARN	Port	Protocol	Target type
backend-target-group	arn:aws:elasticloadbalancing:us-east-1:627a537e62f20...	8000	HTTP	IP
frontend-target-group	arn:aws:elasticloadbalancing:us-east-1:627a537e62f20...	4200	HTTP	IP

Se crearon dos target groups para el despliegue de nuestros servicios, frontend y backend. Uno escuchará por el puerto 4200 y el otro por el 8000.

**Amazon Elastic Container Registry**

**Private repositories**

Repository name	URI	Created at	Tag immutability	Encryption type
programming-the-internet/backend-app	351981951783.dkr.ecr.us-east-1.amazonaws.com/programming-the-internet/backend-app	September 07, 2024, 10:15:58 (UTC-04)	Mutable	AES-256
programming-the-internet/frontend-app	351981951783.dkr.ecr.us-east-1.amazonaws.com/programming-the-internet/frontend-app	September 07, 2024, 10:15:18 (UTC-04)	Mutable	AES-256

Se crearon dos ECR (Elastic Container Registry) para guardar las imágenes de los repos, backend y frontend.

**Amazon Elastic Container Service**

**Clusters**

Cluster	Services	Tasks	Container Instances	CloudWatch monitoring
ecommerce-cluster	2	0 Pending   2 Running	0 EC2	Default

Se creó un cluster en ECS (Elastic Container Service) para nuestros dos servicios.

**Services**

Service	Tasks
Draining	Active: 2   Pending: -   Running: 2

**Encryption**

Managed storage	Fargate ephemeral storage
-	-

**Services**

Service (2) Info	Create
Filter launch type: Any launch type   Filter service type: Any service type	Manage tags   Update   Delete service
Service name: ecommerce-backend-app-service   ARN: arn:aws:ecr:us-east-1:351981951783:repository/programming-the-internet/backend-app   Status: Active   Service type: REPLICAS   Deployments and tasks: 1/1 Tasks running	
Service name: ecommerce-frontend-app-service   ARN: arn:aws:ecr:us-east-1:351981951783:repository/programming-the-internet/frontend-app   Status: Active   Service type: REPLICAS   Deployments and tasks: 1/1 Tasks running	

Se crearon dos servicios para las dos task definitions de nuestra aplicación.

The screenshot shows the AWS CloudShell interface with the following details:

- Navigation Bar:** Includes links for CloudShell, Feedback, Documentation, Discover products, Subscriptions, and the AWS logo.
- Region:** us-east-1.console.aws.amazon.com
- Services:** Services dropdown menu with options like Route 53, Elastic Container Registry, CloudFormation, RDS, EC2, IAM, and S3.
- Region:** N. Virginia
- Page Content:** The main content area is titled "Amazon Elastic Container Service > Task definitions". It displays a table of task definitions:

Task definition	Status of last revision
ecommerce-backend-task	ACTIVE
ecommerce-frontend-task	ACTIVE

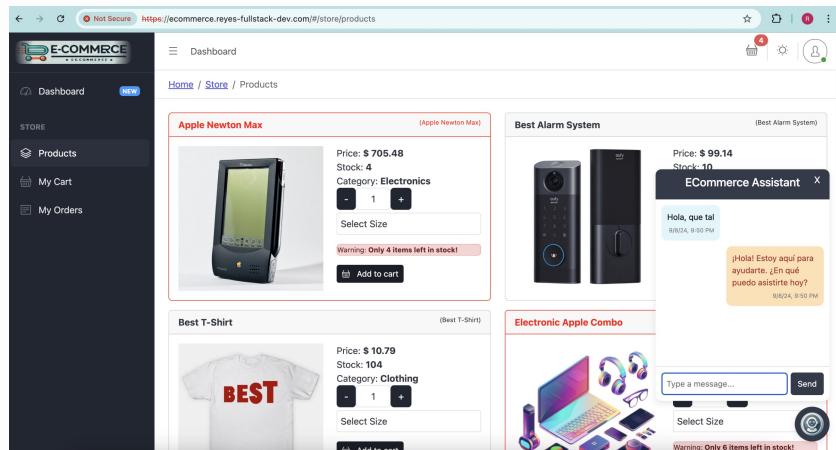
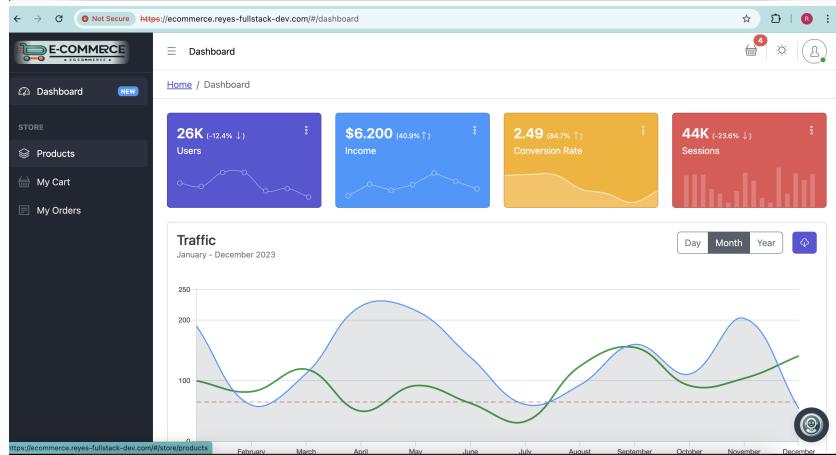
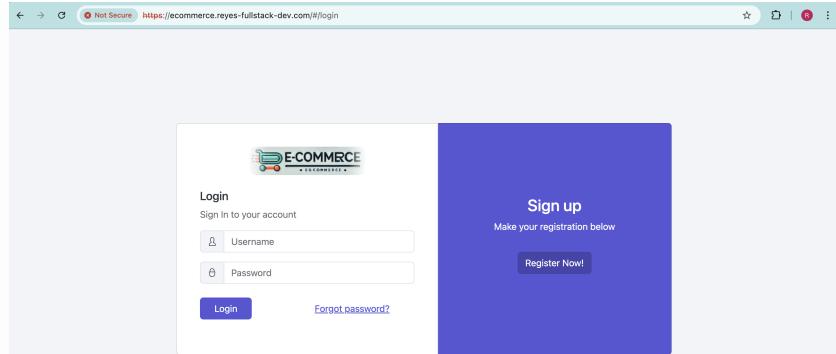
Filtering options include "Filter by status" (Active) and "Create new task definition".

Se crearon dos `task definitions` con listeners HTTP por el puerto 80, ancladas a las imágenes respectivas de cada ECR y con cada uno de sus respectivos `load balancers`.

Amazon RDS		Connectivity & security		
	Endpoint & port	Networking	Security	
Dashboard	Endpoint	Availability Zone	VPC security groups	
Databases	database-1.cdfsaq228w6gu.us-east-1.rds.amazonaws.com	us-east-1a	default (sp-057eda022623a2c8)	
Query Editor	Port	VPC	Active	
Performance insights	5432	Subnet group	Publicly accessible	
Snapshots		default-vpc-0e627a357e62f2026	Yes	
Exports in Amazon S3		Subnets	Certificate authority	
Automated backups		subnet-0d52832ad261b289	Info	
Reserved instances		subnet-07381c406909dfa11c	rds-ca-rsa2048-g1	
Proxies		subnet-071bb0d6258de0dd83		
Subnet groups		subnet-0ee5b16159d0dc77	Certificate authority date	
Parameter groups		subnet-0deef1e61716076a1	May 25, 2061, 19:34 (UTC-04:00)	
Option groups		subnet-0c4c5b9f12bf22ef	DB instance certificate expiration date	
Custom engine versions			September 07, 2025, 10:16 (UTC-04:00)	
Zero-ETL integrations	Network type			
New	IPv4			

The screenshot shows the pgAdmin 4 interface connected to a PostgreSQL database named 'PostgresAWS'. The 'Dependencies' and 'Dependents' tabs are visible at the top. The main pane displays the 'General' tab of the connection configuration, with fields for Host name/address, Port, Maintenance database, Username, Kerberos authentication, Role, and Service. Below this, a table lists various database objects such as tables, functions, and sequences. The 'Dependencies' tab is active, showing a list of dependencies for the selected object.

Se creó una BD en Postgres en su versión 16, usando el servicio de AWS BDS



Se observa nuestra aplicación corriendo en AWS.