UNIVERSIDAD AUTÓNOMA DE SINALOA

LICENCIATURA EN INFORMATICA

SISTEMA DE GESTIÓN DE INVENTARIO EN AWS EC2

Despliegue en la Nube de una Aplicación PHP

Materia: Cómputo en la Nube

Profesor: Javier Alonso Muro

Integrantes del equipo: Bustillos Landa Guimel, Cazares Rochin Ricardo, Orduño Angulo Gustavo, Vega Garcia Susana Valentina

Fecha de entrega: 9 de enero de 2026

UAS CON VISIÓN DE FUTURO 2025

1. INTRODUCCIÓN

Este documento describe el diseño, implementación y despliegue de un sistema web de gestión de inventario desarrollado en PHP y MySQL, migrado desde un entorno local a infraestructura en la nube en Amazon Web Services.

El objetivo principal fue demostrar la capacidad de desplegar y monitorear una aplicación real en EC2 cumpliendo con criterios de seguridad, disponibilidad, optimización de costos y buenas prácticas de administración en la nube.

La solución implementada utiliza una única instancia Amazon EC2 ejecutando una pila LAMP (Linux, Apache, MariaDB y PHP), Security Groups para controlar el acceso, CloudWatch para monitoreo en tiempo real, y se mantiene completamente dentro del AWS Free Tier para optimizar costos sin comprometer la funcionalidad.

---

2. DESCRIPCIÓN DE LA APLICACIÓN

El sistema de gestión de inventario es una aplicación web que permite administrar múltiples aspectos operativos de un negocio pequeño o mediano, las funcionalidades principales incluyen:

Módulo de Autenticación y Usuarios: - Registro e inicio de sesión de usuarios con asignación de roles (administrador, empleado, etc.). - Gestión de perfiles y cambio de contraseña. - Control de acceso basado en roles para restricción de funcionalidades.

Módulo de Inventario: - Crear, editar y eliminar productos en el catálogo. - Visualizar stock disponible y niveles de inventario. - Notificaciones automáticas cuando el stock es bajo. - Consultas y reportes de inventario.

Módulo de Empleados: - Registro y administración de empleados. - Asignación de roles y permisos. - Visualización de datos de contacto y departamento.

Módulo de Proveedores: - Administración de proveedores y datos de contacto. - Histórico de órdenes de compra a proveedores. - Seguimiento de entregas.

Módulo de Órdenes y Pruebas: - Creación y gestión de órdenes de compra. - Registro de pruebas de calidad en nuevos productos. - Seguimiento del estado de órdenes.

Análisis y Reportes: - Análisis de ventas y tendencias. - Generación de reportes visuales (gráficos de tendencias, stock por categoría, etc.). - Exportación de datos.

La aplicación utiliza PHP del lado del servidor para la lógica de negocio, MySQL/MariaDB para persistencia de datos, y Apache como servidor web. La interfaz de usuario incluye formularios para entrada de datos, tablas para visualización y gráficos para análisis.

3. ARQUITECTURA EN AWS

3.1 Descripción General de la Arquitectura

La arquitectura del proyecto se construye sobre una única instancia Amazon EC2 que ejecuta un sistema operativo Amazon Linux 2023 con una pila LAMP completa (Linux como sistema operativo, Apache como servidor web, MariaDB como base de datos relacional, y PHP como lenguaje de programación del lado del servidor).

Componentes principales:

- **Instancia EC2:** Ejecuta Apache, PHP y MariaDB.
- **Security Group:** Controla el tráfico de red permitiendo SSH (puerto 22) desde la IP del administrador y HTTP (puerto 80) desde cualquier lugar. No expone puertos de base de datos ni otros servicios al exterior.
- **MariaDB:** Se ejecuta localmente en la misma instancia, accesible únicamente desde localhost. El usuario dedicado inventario_user tiene permisos únicamente sobre la base de datos inventario_sistema.
- **CloudWatch:** Recopila métricas de la instancia EC2 (CPU, red, disco) y permite crear alarmas para monitorear la salud del sistema.
- **Dirección IP Elástica:** Se asignó una IP pública estática a la instancia para garantizar que la aplicación sea accesible desde Internet con una dirección fija.

3.2 Flujo de Solicitudes

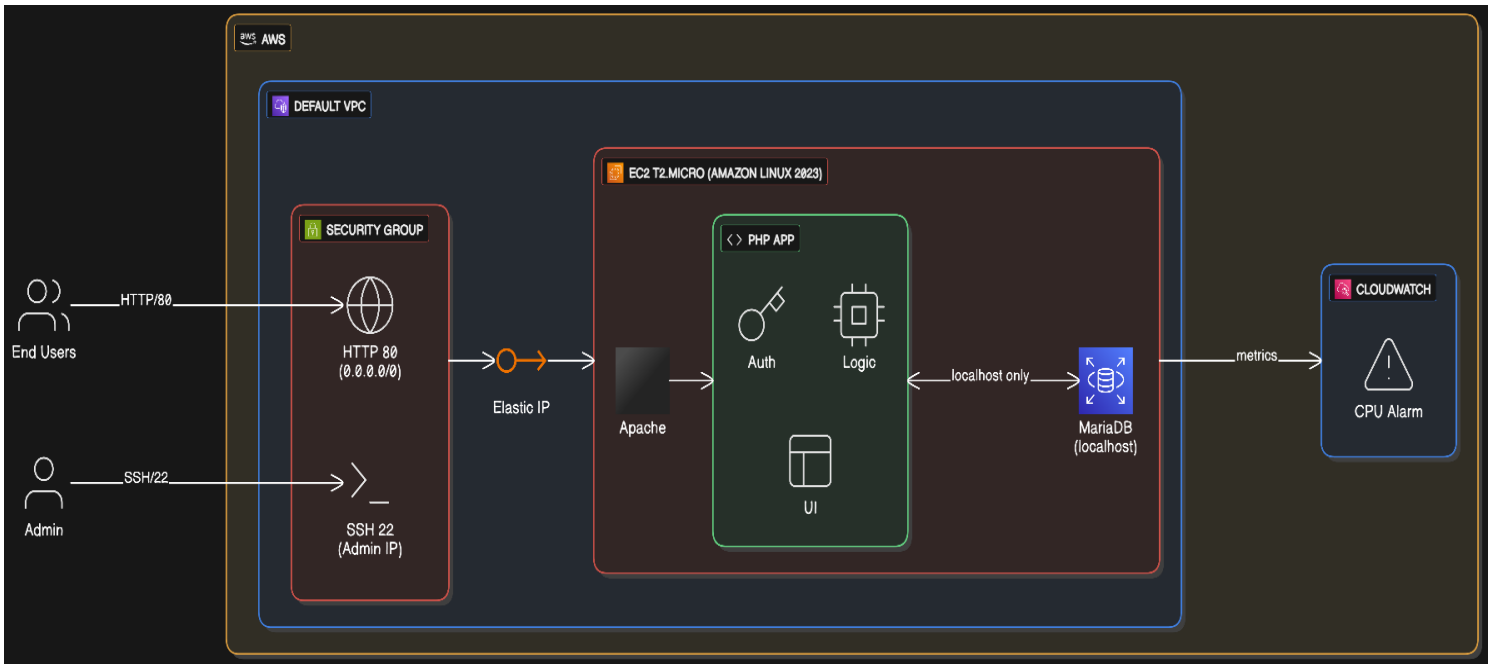
1. Un usuario abre su navegador web e ingresa la dirección IP pública de la instancia EC2 (ej. http://18.218.123.105).
2. El navegador envía una solicitud HTTP que cruza Internet y llega al Security Group de la instancia.
3. El Security Group valida que el puerto 80 está permitido y deja pasar la solicitud.
4. Apache recibe la solicitud en la instancia EC2 y enruta al archivo PHP correspondiente (ej. login.php, inventario_list.php).
5. El código PHP ejecuta la lógica de negocio, accede a MariaDB localmente para recuperar o guardar datos.
6. Apache devuelve una respuesta HTML/CSS/JavaScript al navegador del usuario.
7. En paralelo, CloudWatch recopila métricas de CPU, red y otras variables de la instancia para monitoreo.

3.3 Diagrama de Arquitectura

Ventajas de esta arquitectura: - **Simplicidad:** Un solo servidor que contiene toda la aplicación, fácil de entender y mantener. - **Costo:** Usa solo recursos dentro del AWS Free Tier. - **Seguridad:** La base de datos no está expuesta a Internet; solo se accede localmente. - **Escalabilidad inicial:** t3.micro es suficiente para la carga del proyecto, y puede escalarse a tipos mayores si es necesario.

---

4. PROCESO DE DESPLIEGUE



4.1 Preparación del Proyecto Local

Antes de migrar a AWS, se realizaron los siguientes cambios en el código:

- **Creación de archivo de configuración centralizado (db.php):** Se concentraron todas las credenciales y parámetros de conexión a la base de datos en un único archivo reutilizable por toda la aplicación:

```php
<?php
$host = 'localhost';
$db   = 'inventario_sistema';
$user = 'inventario_user';
$pass = 'Inventario123!';

$conexion = new mysqli($host, $user, $pass, $db);
```

```php
if ($conexion->connect_error) {
    die('Error de conexión: ' . $conexion->connect_error);
}
?>
```

- **Exportación de la base de datos:** Se exportó la base de datos completa desde phpMyAdmin local en formato SQL (archivo inventario_sistema.sql).
- **Revisión de compatibilidad:** Se verificó que todo el código PHP fuese compatible con PHP 8.x (versión que viene en Amazon Linux 2023).

4.2 Creación de la Cuenta AWS y Activación del Free Tier

1. Se creó una cuenta en https://aws.amazon.com/ con correo personal.
2. Se registraron datos personales y se verificó con tarjeta de débito/crédito.
3. Se seleccionó el plan de soporte Basic.
4. Se verificó que la cuenta estaba dentro de los límites del AWS Free Tier:
   - 750 horas mensuales de instancia t2.micro/t3.micro.
   - 30 GB de almacenamiento EBS.
   - 1 GB de tráfico de datos saliente (primeras líneas).

4.3 Creación de la Instancia EC2 y Configuración LAMP

Paso 1: Lanzar la instancia - Accedimos a la consola EC2 → “Launch instance”. - Configuración: - **Nombre:** proyecto-muro-aws - **AMI:** Amazon Linux 2023 **Tipo de instancia:** t3.micro - **Key Pair:** Se creó un nuevo par de claves proyecto-muro-key.pem y se descargó para acceso SSH. - **Security Group:** Se creó con reglas: - SSH (22): desde IP del administrador (177.228.99.252/32) - HTTP (80): desde 0.0.0.0/0 (cualquier dirección) - MySQL (3306): No expuesto - **Almacenamiento:** 8 GB de volumen EBS (gp2), suficiente para la aplicación. - La instancia fue lanzada y quedó en estado “running” en pocos minutos.

Paso 2: Conexión SSH a la instancia

```
ssh -i /c/Users/gus_o/proyecto-muro-key.pem ec2-user@18.218.123.105
```

Se accedió con éxito a la instancia de Linux en el servidor.

Paso 3: Instalación de LAMP

- Actualización del sistema: bash sudo yum update -y

- Instalación de Apache, MariaDB y PHP: bash sudo yum install -y httpd mariadb105-server php php-mysqlnd

- Inicialización de servicios: bash sudo systemctl start httpd sudo systemctl enable httpd sudo systemctl start mariadb sudo systemctl enable mariadb

4.4 Migración de Código y Base de Datos

Paso 1: Subida de archivos de la aplicación

Los archivos PHP se transfirieron desde la máquina local a la instancia EC2 usando el comando scp:

```
scp -i /c/Users/gus_o/proyecto-muro-key.pem -r /c/Users/gus_o/OneDrive/Desktop/app/* ec2-user@18.218.123.105:/home/ec2-user/app
```

Esto copió todos los archivos PHP, CSS e imágenes a /home/ec2-user/app en el servidor.

Luego, se movieron al DocumentRoot de Apache:

```
sudo rm -rf /var/www/html/*
sudo cp -r /home/ec2-user/app/* /var/www/html/
sudo chown -R apache:apache /var/www/html
sudo systemctl restart httpd
```

Paso 2: Subida y configuración de la base de datos

El archivo SQL se transfirió a la instancia:

```
scp -i /c/Users/gus_o/proyecto-muro-key.pem /c/Users/gus_o/OneDrive/Desktop/DB/inventario_sistema.sql ec2-user@18.218.123.105:/home/ec2-user/backup.sql
```

Se ejecutó el asistente de seguridad de MariaDB:

```
sudo mysql_secure_installation
```

Respuestas configuradas: - Contraseña root: Se estableció una contraseña fuerte. - Eliminar usuarios anónimos: Sí. - Deshabilitar acceso root remoto: Sí. - Eliminar base de datos de prueba: Sí.

Se creó la base de datos y usuario dedicado:

```
sudo mysql -u root -p
```

Dentro de MariaDB:

```
CREATE DATABASE inventario_sistema CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;

CREATE USER 'inventario_user'@'localhost' IDENTIFIED BY 'Inventario123!';

GRANT ALL PRIVILEGES ON inventario_sistema.* TO 'inventario_user'@'localhost';
```

```
FLUSH PRIVILEGES;
```

Se importó el archivo SQL:

```
mysql -u inventario_user -p inventario_sistema < /home/ec2-user/backup.sql
```

Se ingresó la contraseña Inventario123! y la base de datos se importó exitosamente con todas las tablas y datos.

Paso 3: Configuración de la conexión en PHP

Se editó el archivo db.php en /var/www/html:

```
sudo nano /var/www/html/db.php
```

Contenido final:

```
<?php
$host = 'localhost';
$db   = 'inventario_sistema';
$user = 'inventario_user';
$pass = 'Inventario123!';

$conexion = new mysqli($host, $user, $pass, $db);

if ($conexion->connect_error) {
    die('Error de conexión: ' . $conexion->connect_error);
}
?>
```

Se reinició Apache para aplicar cambios:

```
sudo systemctl restart httpd
```

Al final verificamos si funciono accediendo a http://18.218.123.105, la aplicación cargó correctamente mostrando la página de login y permitiendo acceso a todos los módulos con conectividad a la base de datos funcionando.

---

5. SEGURIDAD

5.1 Gestión de Identidad y Acceso (IAM)

Principio aplicado: No usar la cuenta raíz de AWS para tareas cotidianas.

Se creó un usuario IAM dedicado para la administración del proyecto:

- **Nombre:** admin-proyecto-aws

- **Tipo de acceso:** Acceso a consola de administración AWS

- **Políticas asignadas:**

- AmazonEC2FullAccess (permitir crear, modificar, eliminar instancias EC2)

- CloudWatchFullAccess (permitir crear y revisar dashboards y alarmas)

Este usuario se utiliza para todas las operaciones en la consola AWS en lugar de la cuenta raíz, lo que reduce el riesgo de exposición accidental de credenciales raíz y permite auditoría de quién hace qué en la cuenta.

5.2 Control de Acceso de Red (Security Groups)

Reglas de Entrada (Inbound):

Protocolo	Puerto	Origen	Propósito
TCP	22	177.228.99.252/32	SSH desde IP del administrador
TCP	80	0.0.0.0/0	HTTP desde cualquier lugar
-	3306	(No expuesto)	Base de datos solo local

Explicación: - **SSH (22):** Limitado a la dirección IP del administrador para que solo el equipo autorizado pueda conectarse por terminal remota a la instancia.

- **HTTP (80):** Abierto al público (0.0.0.0/0) para que usuarios finales accedan a la aplicación web desde cualquier lugar.

- **MariaDB (3306):** No se expone al exterior; la base de datos solo se accede desde procesos PHP en la misma instancia (conexión local).

---

6. MONITOREO Y DISPONIBILIDAD

6.1 Dashboard en CloudWatch

Se creó un panel de monitoreo en Amazon CloudWatch para visualizar en tiempo real el estado de la instancia EC2:

Métricas monitoreadas:

- **CPU Utilization:** Porcentaje de CPU usado por la instancia. Métrica clave para detectar sobrecarga. -

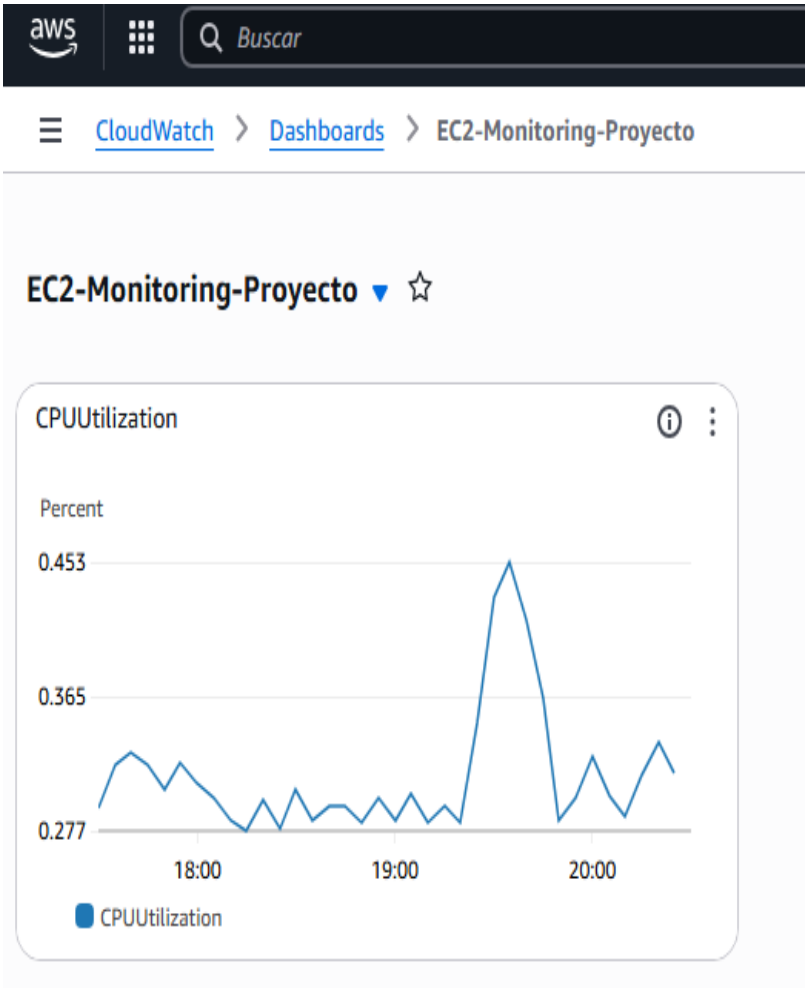
Dashboard “EC2-Monitoring-Proyecto”:

- Se agregó un widget tipo “línea” que muestra la métrica CPUUtilization de la instancia en los últimos 60 minutos.

- Actualización automática cada 1 minuto.

- Permite identificar rápidamente si la instancia está bajo presión o funcionando con recursos normales.

Captura del dashboard:



6.2 Alarmas de Disponibilidad

Se configuró una alarma para alertar cuando la CPU supera umbrales peligrosos:

Alarma "EC2-CPU-High-Proyecto":

- **Métrica:** CPUUtilization de la instancia EC2

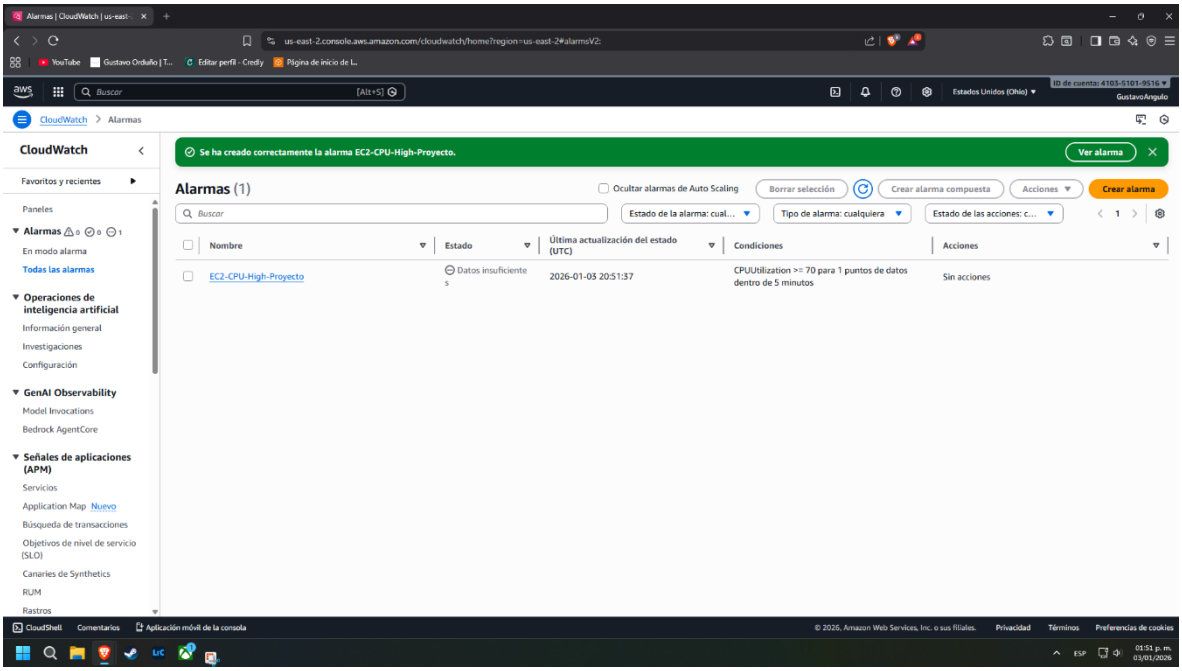
- **Estadística:** Promedio

- **Periodo:** 5 minutos

- **Umbral:** CPU >= 70%

Lógica: Si durante 5 minutos consecutivos el promedio de CPU está en o por encima del 70%, la alarma cambia a estado "ALARM" y se dispara una notificación. Esto indica que la instancia está bajo carga alta y puede que necesite optimización o escalamiento.

Captura de la alarma:



---

7. COSTOS Y OPTIMIZACIÓN

7.1 Uso del AWS Free Tier

La solución está completamente diseñada para operar sin costos significativos gracias al AWS Free Tier:

Límites del Free Tier (12 meses para nuevas cuentas):

Servicio	Límite	Uso del Proyecto	Estado
EC2 t2.micro/t3.micro	750 horas/mes	~720 horas/mes (1 instancia siempre encendida)	✓ Dentro
EBS (almacenamiento)	30 GB/mes	8 GB	✓ Dentro
EBS (IOPS)	1 millón/mes	~50,000/mes	✓ Dentro
CloudWatch (métricas)	10 métricas/mes	2-3 métricas	✓ Dentro

Servicio	Límite	Uso del Proyecto	Estado
Datos salientes	1 GB/mes	~100 MB/mes	✓ Dentro
IP públicas	Gratis (si está en uso)	1	✓ Dentro

Conclusión: El proyecto se mantiene completamente dentro de los límites gratuitos, resultando en $0 USD de costo mensual mientras la cuenta esté activa en el Free Tier.

7.2 Decisiones de Arquitectura para Optimizar Costos

1. **Una sola instancia EC2 (t3.micro):**
   - Consolida todos los servicios en una máquina.
2. **Base de datos local (MariaDB) en lugar de RDS:**
   - RDS tendría costo mensual incluso en Free Tier.
   - MariaDB en la instancia EC2 no agrega costo.
3. **Volumen EBS pequeño (8 GB) en lugar de mayor:**
   - 30 GB disponibles en Free Tier.
   - 8 GB es más que suficiente para código PHP y base de datos.
4. **Sin servicios adicionales:**
   - Solo lo esencial para el proyecto.

---

8. REPOSITORIO

El código fuente del proyecto está disponible en el repositorio GitHub:

URL: https://github.com/LinkTrifuerza/SistemaDeInventarioFidel

El repositorio está pensado para otra materia para la que se creó el proyecto, sin embargo, contiene todos los archivos necesarios para desplegar la app en aws como lo hicimos en este proyecto.

Contenido del repositorio: - Carpeta raíz llamada app con todos los archivos PHP de la aplicación (login.php, dashboard.php, inventario_list.php, etc.). - Carpeta /img con imágenes y recursos visuales. – Carpeta DB con archivo inventario_sistema.sql con estructura e inserciones iniciales de la base de datos. - Archivo README.md con instrucciones de instalación local. – Carpeta PNG que incluye los diagramas de la aplicación que se muestran en el README.

Además de este repositorio creamos otro mas simple en donde solo esta el proyecto php y la db:

URL: https://github.com/LinkTrifuerza/Proyecto-aws-muro

---

9. JUSTIFICACIÓN DE DECISIONES TÉCNICAS

Por qué Amazon EC2 con LAMP

Se eligió **Amazon EC2** como servicio principal de cómputo porque:

1. **Flexibilidad y control:** EC2 permite configurar una pila LAMP idéntica a la del entorno local XAMPP para que fuera más fácil la migración del código sin cambios significativos.
2. **Familiaridad:** Durante el curso se estudió en profundidad cómo crear instancias EC2, instalar y configurar servicios, lo que nos permitió aplicar directamente los conocimientos adquiridos.
3. **Costo en Free Tier:** La instancia t3.micro está incluida en el AWS Free Tier, esto no permitió un costo cero.

Por qué una instancia única con LAMP integrado

Se utilizo MariaDB en la misma instancia EC2 en lugar de usar Amazon RDS porque:

1. **Simplicidad:** Una sola máquina concentra toda la aplicación, lo hace mas simple.
2. **Costo:** RDS tiene costos mensuales incluso en Free Tier, MariaDB ejecutándose en EC2 no agrega costo.
3. **Suficiente para la carga:** t3.micro con MariaDB local es totalmente adecuado.

Por qué CloudWatch para monitoreo

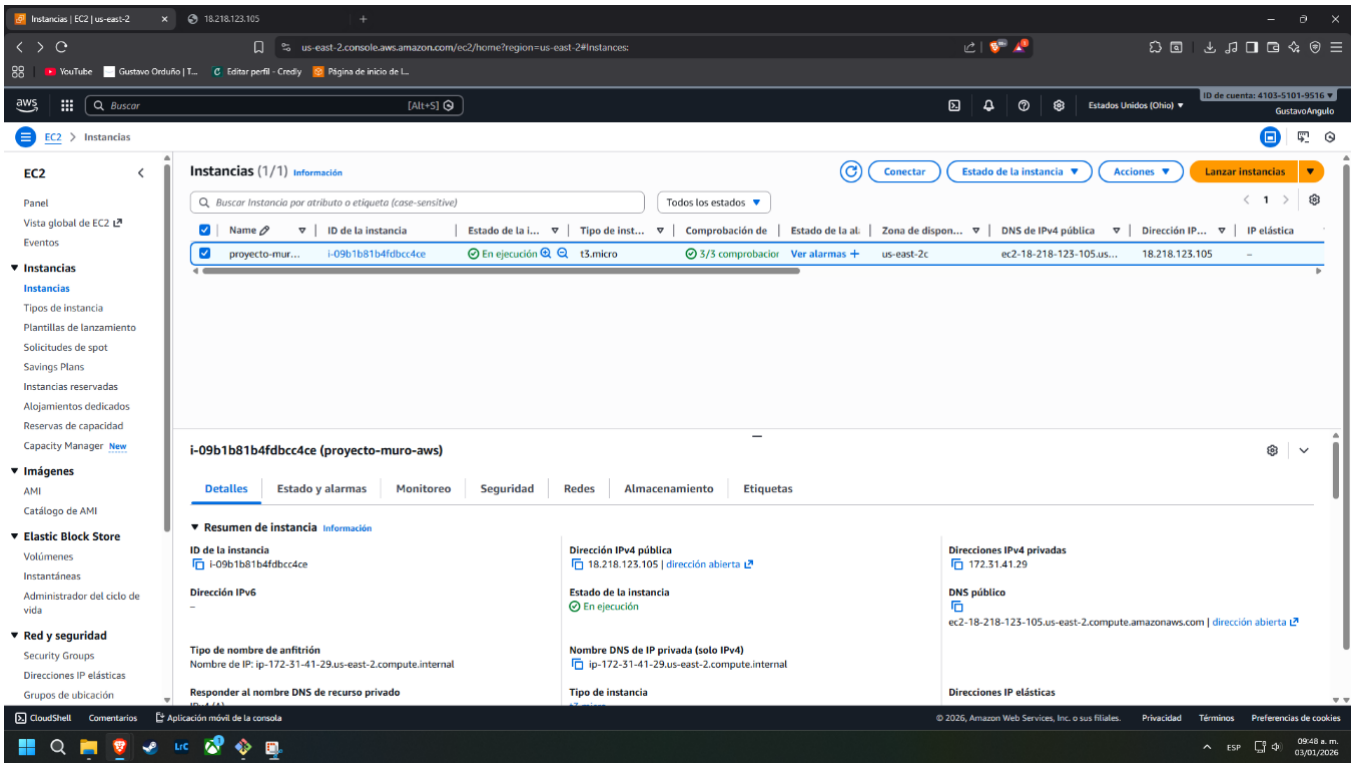
Se eligió **Amazon CloudWatch** para monitoreo porque:

1. **Integrado nativamente:** EC2 envía métricas automáticamente a CloudWatch sin configuración adicional.
2. **Sin costo:** Las métricas básicas están incluidas en el Free Tier.
3. **Visibilidad en tiempo real:** Los gráficos y alarmas permiten detectar rápidamente anomalías en la instancia.
4. **Registro histórico:** CloudWatch mantiene datos de métricas durante 15 meses, permitiendo análisis y tendencias a largo plazo.

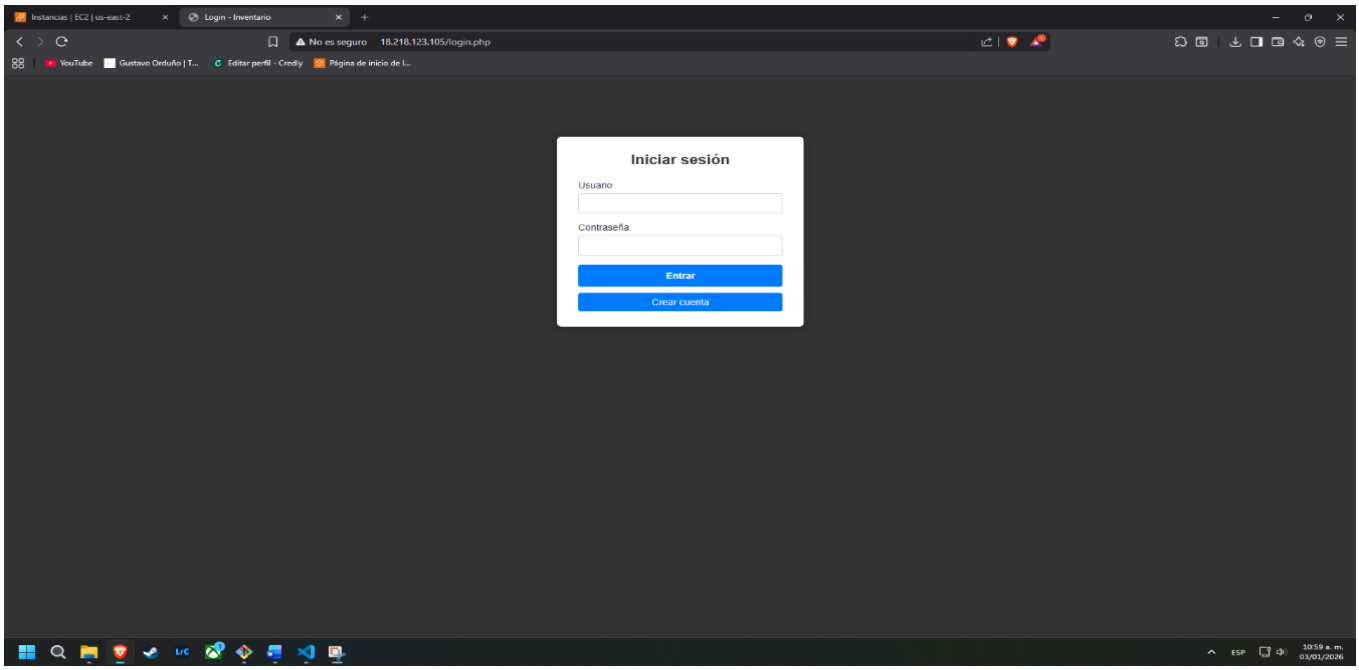
ANEXOS

Anexo A: Capturas de Pantalla

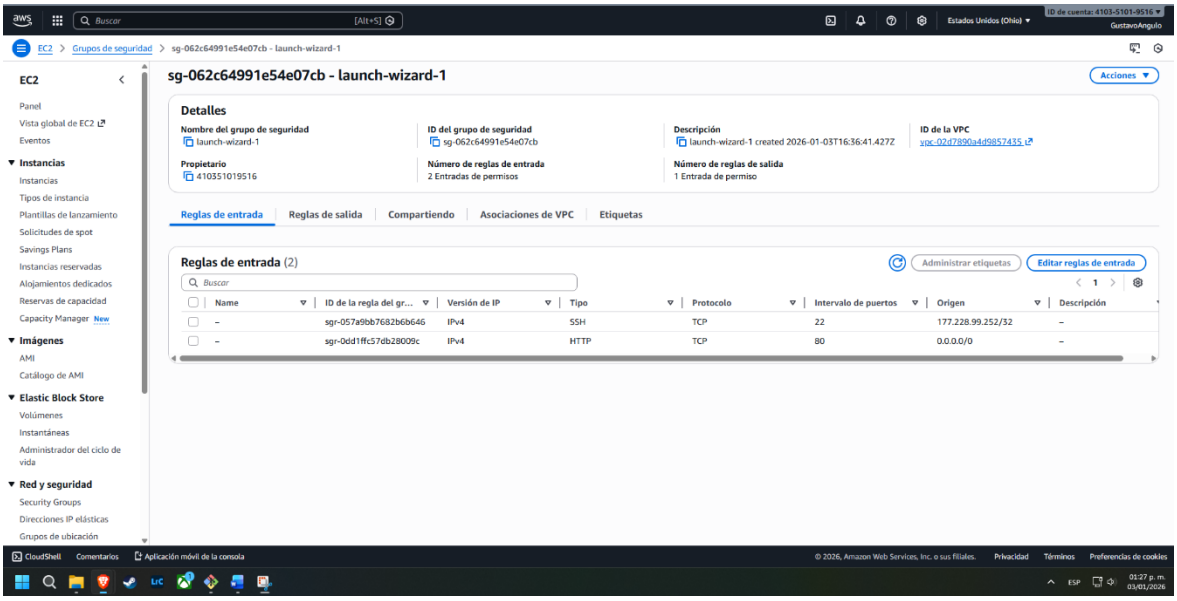
Captura 1: Consola EC2



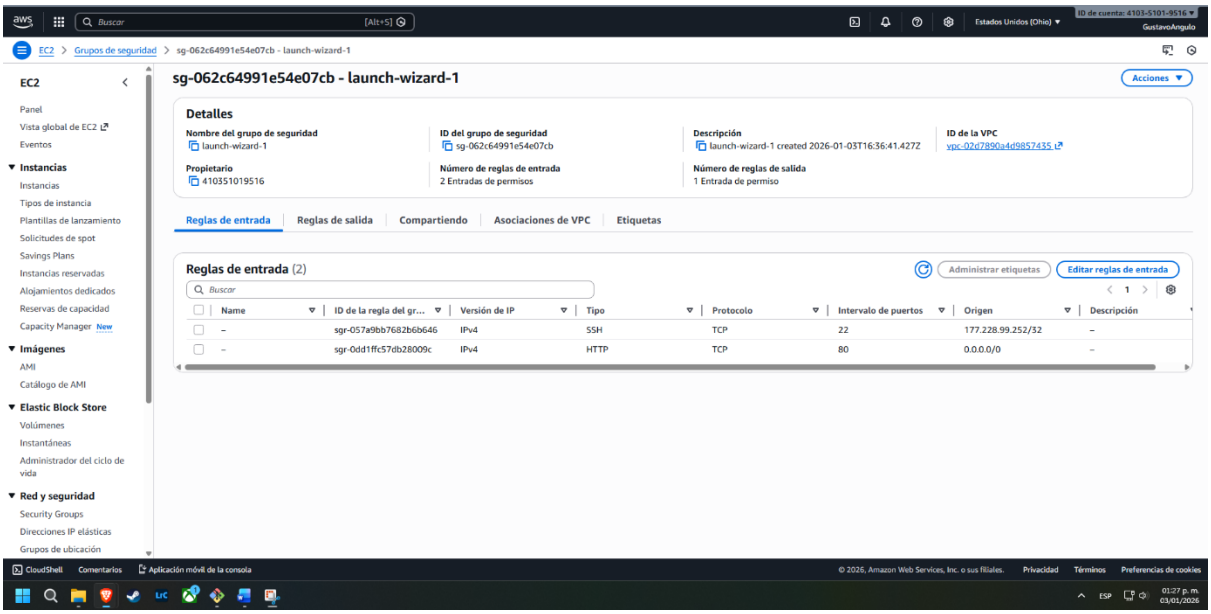
Captura 2: Aplicación en navegador



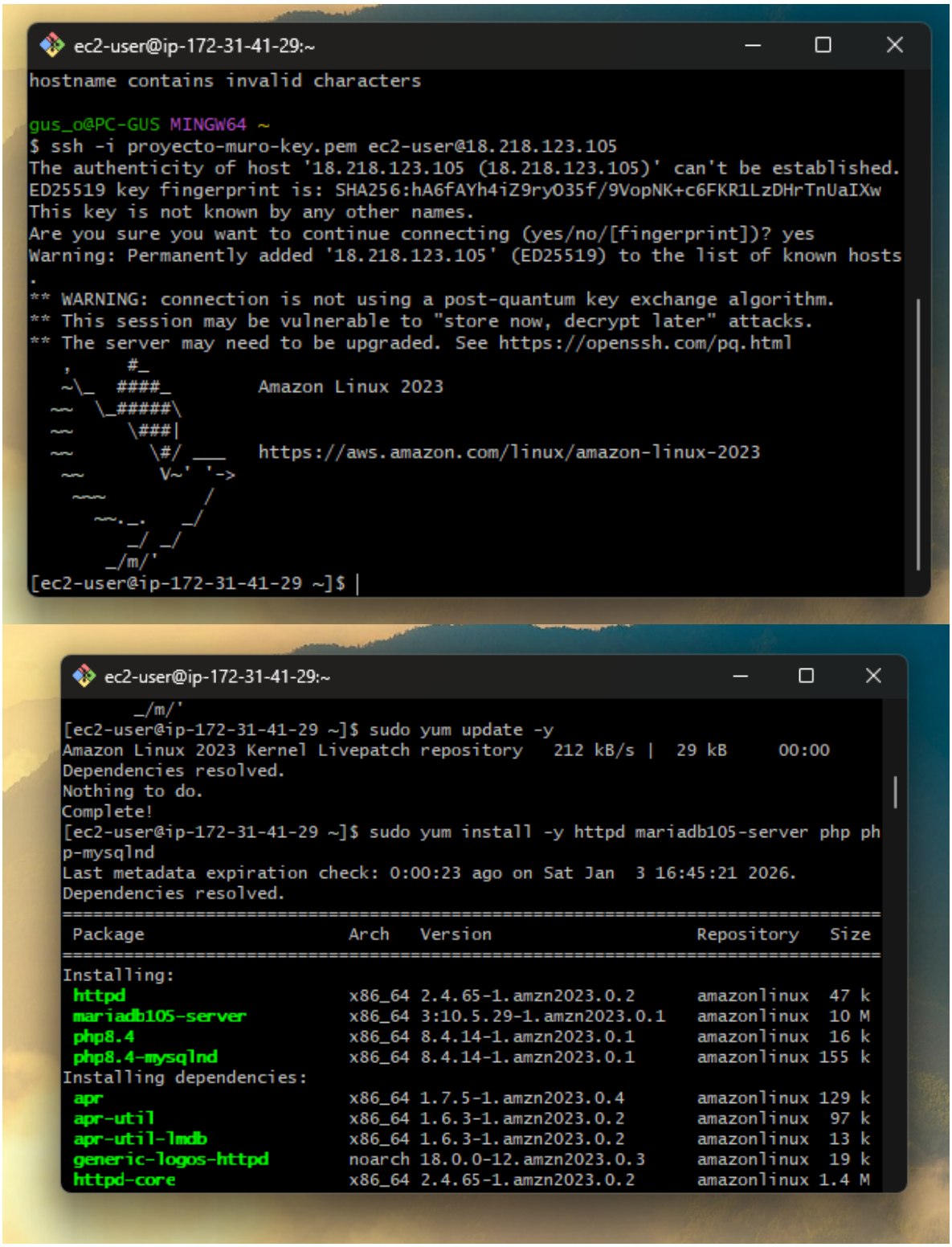
Captura 3: Security Group



Captura 4: Creación de usuario en IAM



Anexo B: Comandos principales ejecutados

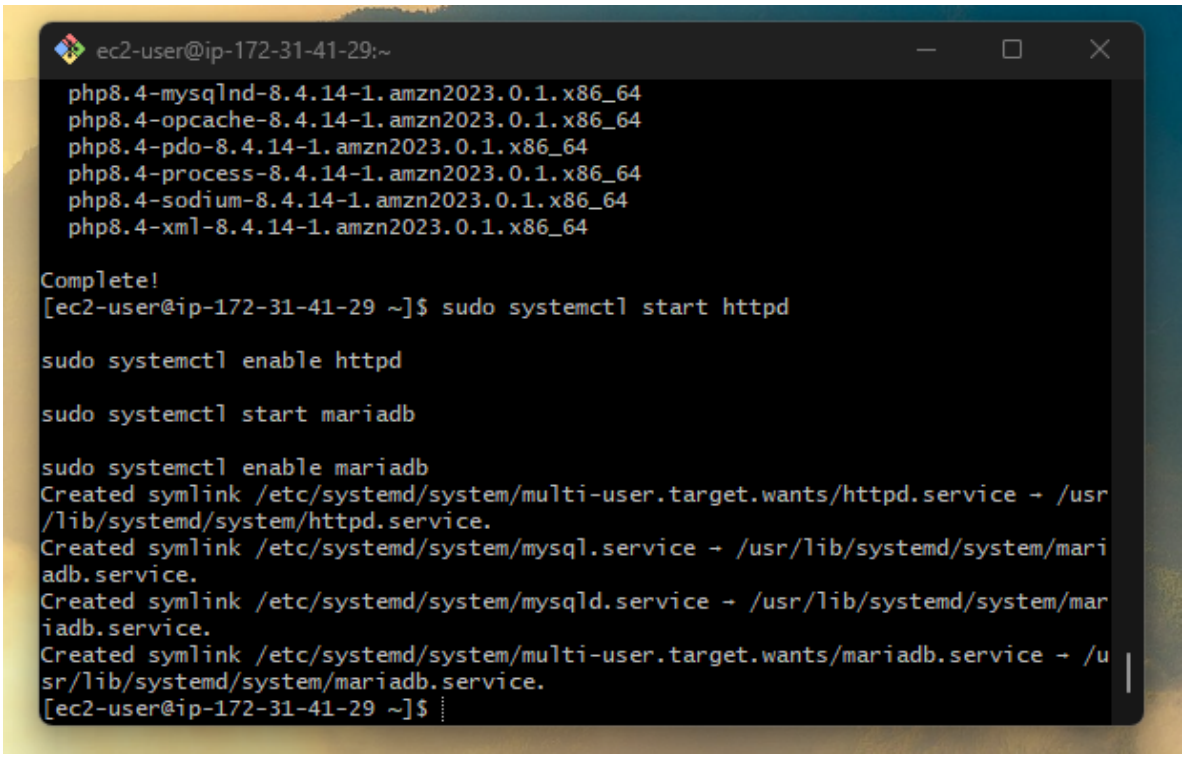


```
MINGW64:/c/Users/gus_o
gus_o@PC-GUS MINGW64 ~
$ chmod 400 proyecto-muro-key.pem

gus_o@PC-GUS MINGW64 ~
$ scp -i /c/Users/gus_o/proyecto-key.pem -r /c/Users/gus_o/OneDrive/Desktop/app/
* ec2-user@18.218.123.105:/home/ec2-user/app
Warning: Identity file /c/Users/gus_o/proyecto-key.pem not accessible: No such f
ile or directory.
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
ec2-user@18.218.123.105: Permission denied (publickey,gssapi-keyex,gssapi-with-m
ic).
scp: Connection closed

gus_o@PC-GUS MINGW64 ~
$ scp -i /c/Users/gus_o/proyecto-muro-key.pem -r /c/Users/gus_o/OneDrive/Desktop
/app/* ec2-user@18.218.123.105:/home/ec2-user/app
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
analisis_ventas.php                           100% 3770    46.9KB/s   00:00
auth_middleware.php                           100%  278     3.6KB/s   00:00
cambiar_password.php                          100% 2044    25.9KB/s   00:00
dashboard.php                                 100% 1293    16.1KB/s   00:00
db.php                                        100%  408     5.2KB/s   00:00
empleados_form.php                            100% 2442    30.5KB/s   00:00
empleados_list.php                            100% 2945    37.2KB/s   00:00
empleados_save.php                            100% 1423    17.7KB/s   00:00
footer.php                                    100%   23     0.3KB/s   00:00
header.php                                    100%  392     5.0KB/s   00:00
analisis.png                                  100%   15KB 188.7KB/s   00:00
empleados.png                                 100%   36KB 454.3KB/s   00:00
inventario.png                                100%  106KB 679.9KB/s   00:00
Proveedores.png                               100%   22KB 283.0KB/s   00:00
pruebas.png                                   100%   21KB 272.9KB/s   00:00
tickets_ordenes.png                           100%   20KB 258.2KB/s   00:00
tickets_pruebas.png                           100%   21KB 262.8KB/s   00:00
index.php                                     100%  153     2.0KB/s   00:00
inventario_delete.php                         100%  321     4.1KB/s   00:00
inventario_form.php                           100% 3199    40.3KB/s   00:00
inventario_list.php                           100% 2681    33.4KB/s   00:00
inventario_save.php                           100% 4300    53.8KB/s   00:00
login.php                                     100% 1406    17.7KB/s   00:00
logout.php                                    100%   90     1.2KB/s   00:00
navbar.php                                    100% 1096    13.9KB/s   00:00
no_autorizado.php                             100%  246     3.2KB/s   00:00
notificaciones_list.php                       100% 1274    16.1KB/s   00:00
orden_proveedor_form.php                      100% 3460    43.7KB/s   00:00
orden_proveedor_guardar.php                   100% 2089    26.4KB/s   00:00
perfil.php                                    100% 1771    22.1KB/s   00:00
perfil_update.php                             100%  455     5.8KB/s   00:00
proveedores_form.php                          100% 1844    22.9KB/s   00:00
proveedores_list.php                          100% 3026    38.3KB/s   00:00
proveedores_save.php                          100% 1010    12.8KB/s   00:00
pruebas_carrito.php                           100% 3588    45.3KB/s   00:00
pruebas_guardar.php                           100% 1686    21.3KB/s   00:00
register.php                                  100% 4784    60.4KB/s   00:00
stock_notificaciones.php                      100% 2122    26.8KB/s   00:00
styles.css                                    100% 4687    59.2KB/s   00:00
tickets_ordenes.php                           100% 4596    58.0KB/s   00:00
tickets_pruebas.php                           100% 3851    48.7KB/s   00:00

gus_o@PC-GUS MINGW64 ~
$ scp -i /c/Users/gus_o/proyecto-muro-key.pem /c/Users/gus_o/OneDrive/Desktop/DB
/inventario_sistema.sql ec2-user@18.218.123.105:/home/ec2-user/backup.sql
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
inventario_sistema.sql                        100%   11KB 144.1KB/s   00:00
```


ec2-user@ip-172-31-41-29:~
php8.4-mysqlnd-8.4.14-1.amzn2023.0.1.x86_64
php8.4-opcache-8.4.14-1.amzn2023.0.1.x86_64
php8.4-pdo-8.4.14-1.amzn2023.0.1.x86_64
php8.4-process-8.4.14-1.amzn2023.0.1.x86_64
php8.4-sodium-8.4.14-1.amzn2023.0.1.x86_64
php8.4-xml-8.4.14-1.amzn2023.0.1.x86_64
Complete!
[ec2-user@ip-172-31-41-29 ~]$ sudo systemctl start httpd
sudo systemctl enable httpd
sudo systemctl start mariadb
sudo systemctl enable mariadb
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
Created symlink /etc/systemd/system/mysql.service → /usr/lib/systemd/system/mariadb.service.
Created symlink /etc/systemd/system/mysqld.service → /usr/lib/systemd/system/mariadb.service.
Created symlink /etc/systemd/system/multi-user.target.wants/mariadb.service → /usr/lib/systemd/system/mariadb.service.
[ec2-user@ip-172-31-41-29 ~]$