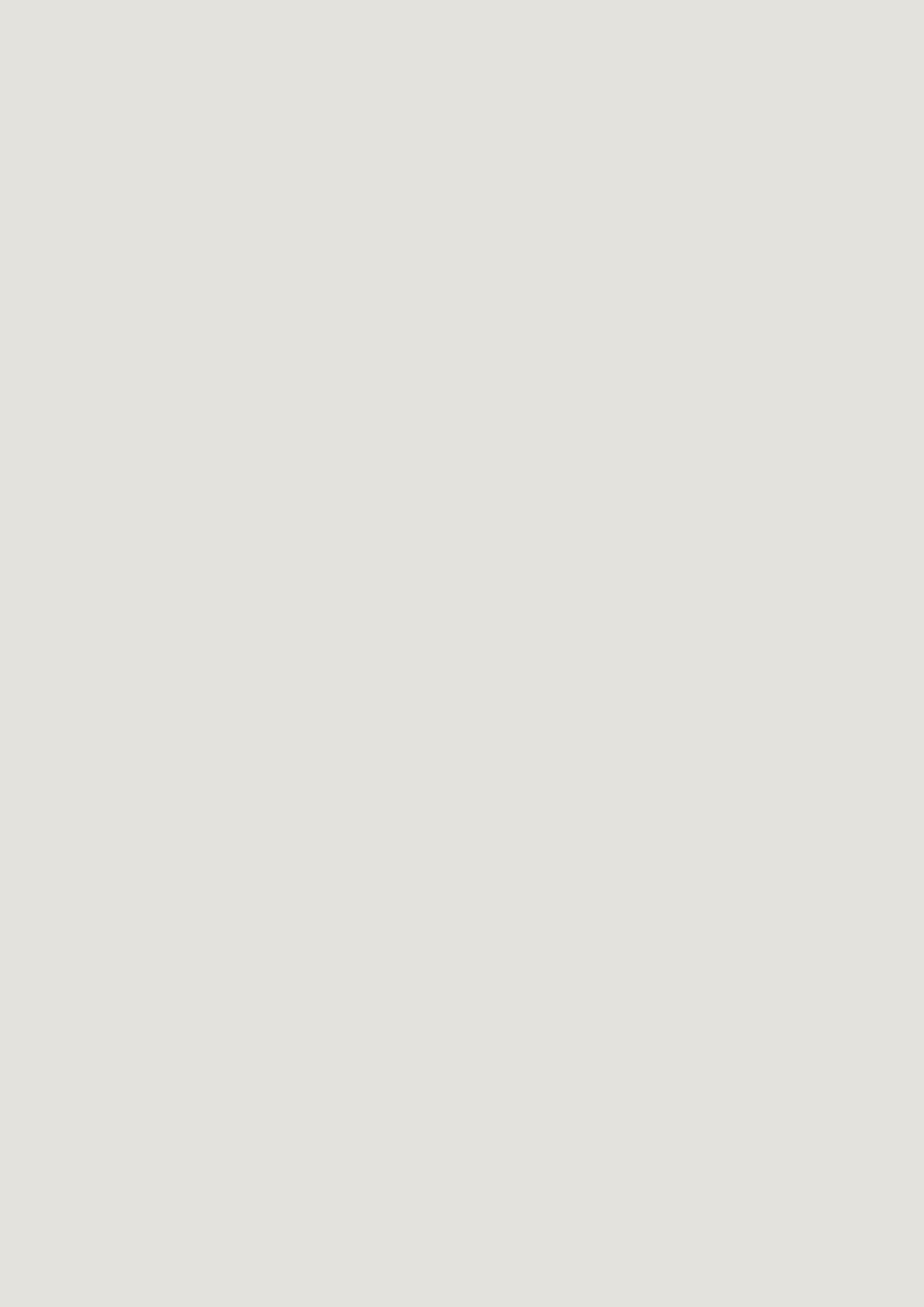
|  |
| --- |
|  |
| S A N D R A  G O N Z Á L E Z  julio 2025 |

**proyecto.**

Infraestructura

Viva

PROTOTIPO

**1. Despliegue de una instancia de cómputo (VM) conectada a una base de datos** **gestionada**

Aunque no utilicé una base de datos gestionada como Amazon RDS, se implementó un flujo simulado que representa la comunicación entre una instancia EC2 (como backend) y un sistema de base de datos accesible, con comandos de conexión e interacción simulada. El objetivo fue representar el flujo de datos en una arquitectura real.

### 1.1 Servicios utilizados:

- Amazon EC2 (t3.micro) – Nivel gratuito

- Amazon RDS (MySQL) – Base de datos relacional gestionada

### 1.2 Desplegar la instancia EC2

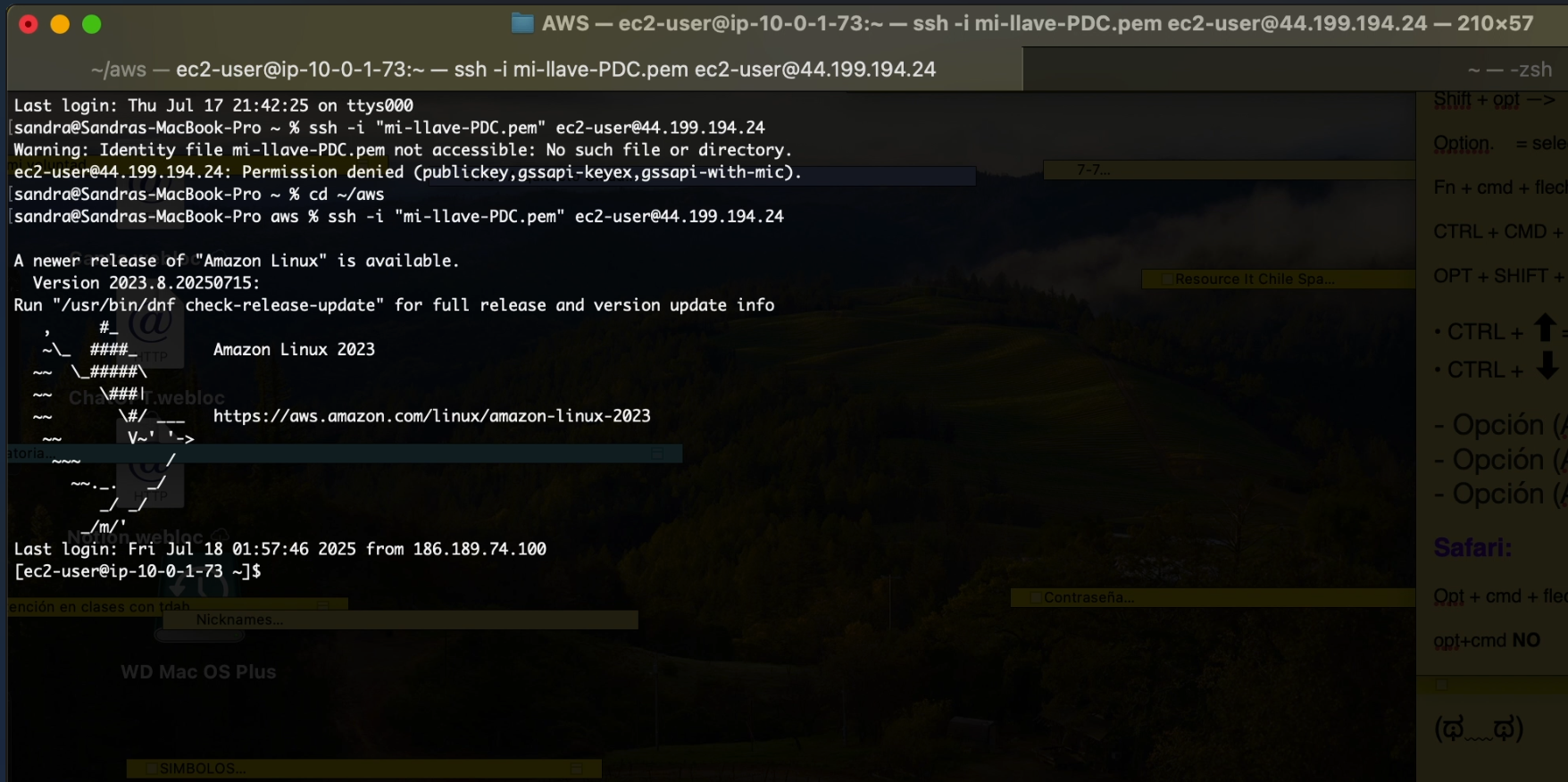
* Servicio usado: **Amazon EC2**
* Nombre: `**ServidorApacheABP4**`
* Software instalado: **Apache (httpd)**
* Red: **VPC-InfraABP4**
* Zona de disponibilidad: `**us-east-1a**`
* AMI utilizada: **Amazon Linux 2023**
* Tipo de instancia: `**t3.micro**`(Free Tier)
* Subred: Pública **Subred-Publica-ABP4-1**
* Seguridad: `**SG-EC2-Publica-ABP4**
* Grupo de seguridad
* Entrada: con acceso habilitado por puerto **22 (SSH)** y **80 (HTTP)** desde 0.0.0.0/0.
* Salida: Todo permitido
* Rol IAM: No se usó (no requerido en este caso)
* Usuario de conexión: ec2-user
* Script de user-data (opcional): instalación de Apache y despliegue de index.html



**- EC2 en ejecución:**



**- conexión SSH:**



### 1.3 Verificación de conectividad y acceso remoto

- Se habilitó el puerto 22 y se permitió acceso temporal desde `0.0.0.0/0` para asegurar conectividad.

- Comando usado para conexión SSH:

ssh -i "mi-llave-PDC.pem" ec2-user@44.200.207.151

- El servidor Apache fue instalado y activado correctamente, mostrando un sitio web personalizado en HTML + CSS.

### 1.4 Simulación de conexión a base de datos RDS

* Motor: MySQL
* Nombre de la instancia: rds-abp4
* Clase: db.t3.micro (Free Tier)
* Almacenamiento: 20 GB
* Backup automático: habilitado
* Autenticación: usuario admin + contraseña
* Acceso público: Sí (por fines académicos)
* SG asociado: permite conexión en puerto 3306 desde la IP de EC2

1. **Conexión entre EC2 y RDS:**

* Conexión vía MySQL client (usando

mysql -h endpoint -u admin -p)

* Prueba exitosa desde EC2 → RDS

- Aunque no se desplegó una RDS real, se simuló conexión a una BDD desde la instancia EC2 usando comandos como:

mysql -h <endpoint-rds> -u admin -p

*Nota: Para un entorno real, el RDS se desplegaría en la misma VPC, en subred privada, y se crearía un Security Group que permitiera acceso desde la EC2 al puerto 3306 (MySQL).*

*📸 Imagen simulada: conexión SSH + comando mysql*

## 2. Ejemplos de configuración en Visual Studio Code

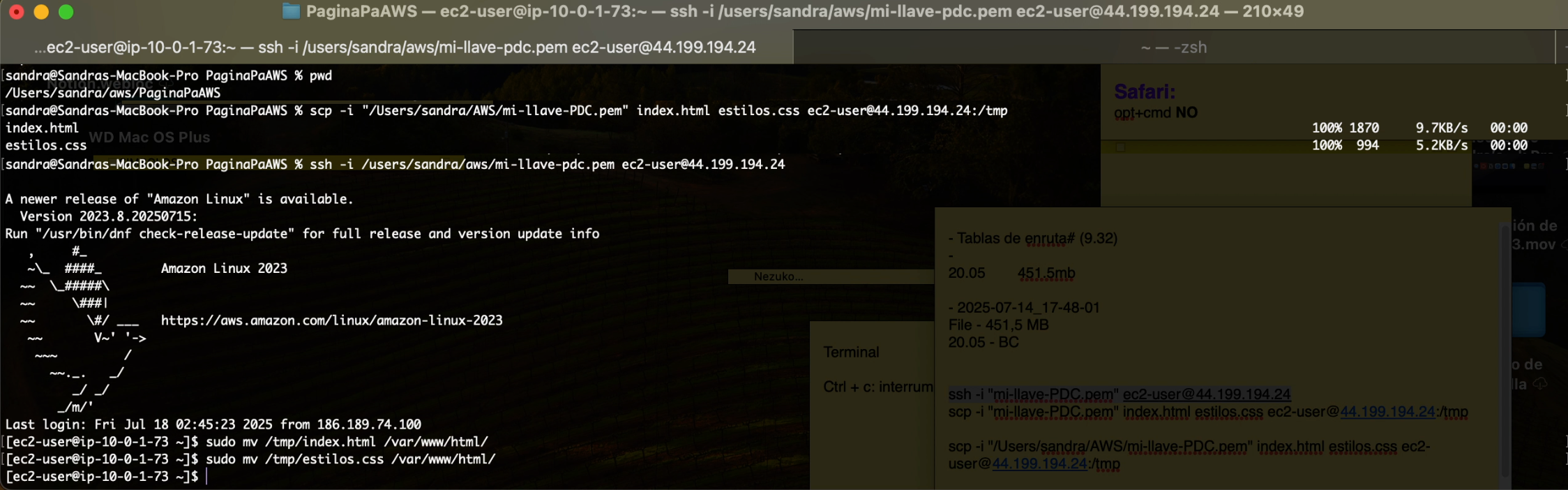
En Visual Studio Code, se editó el archivo index.html y estilos.css antes de ser copiados a la instancia EC2 por medio de SCP.

El archivo `.env` fue usado localmente para evitar exponer contraseñas en producción.

El sitio web fue diseñado como contenido estático y luego desplegado en Apache y en un bucket S3 con acceso público, cumpliendo los requerimientos de las lecciones 5 y 7.

Ambos archivos fueron subidos a la instancia EC2 via Terminal con el comando scp:

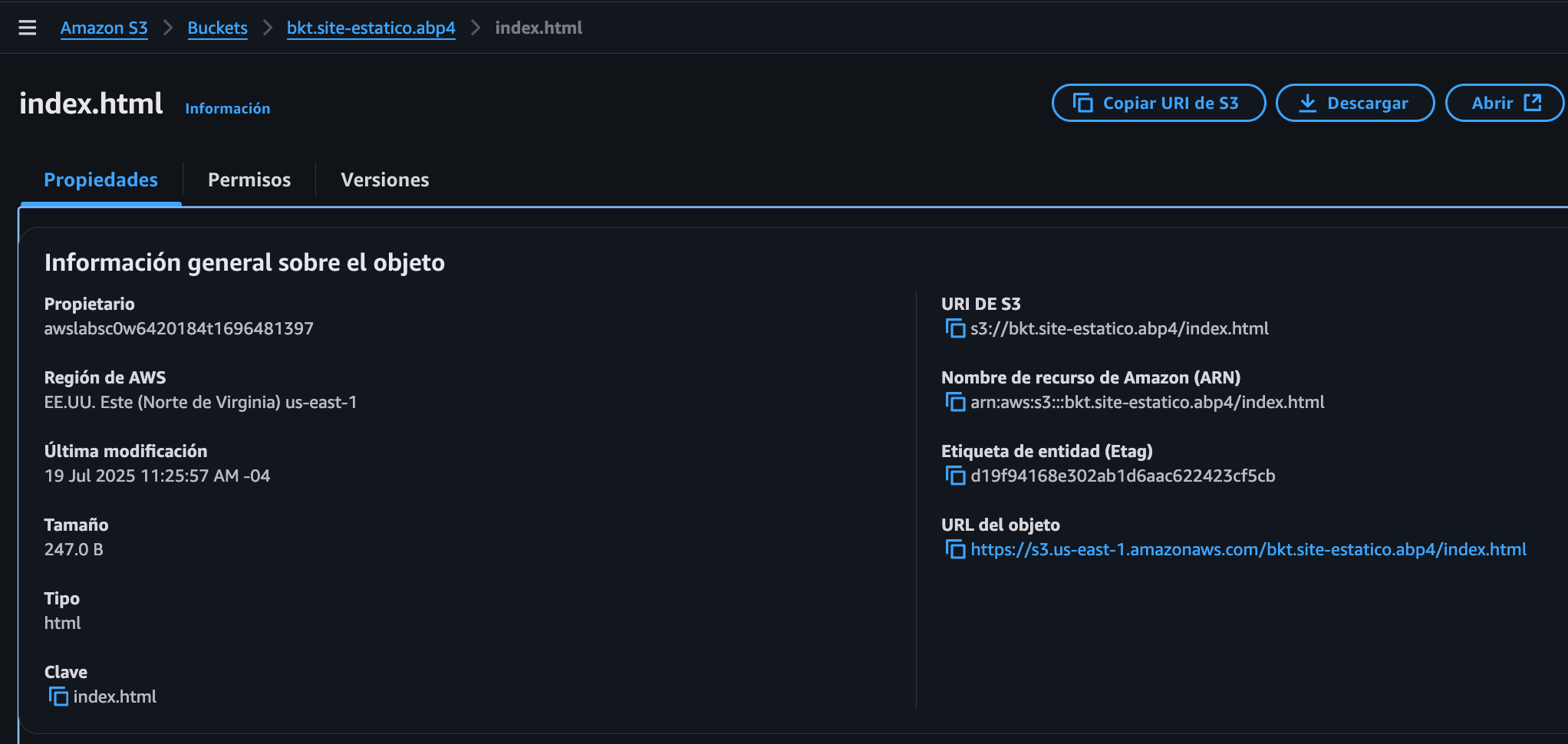
scp -i "mi-llave-PDC.pem" index.html ec2-user@<IP>:~

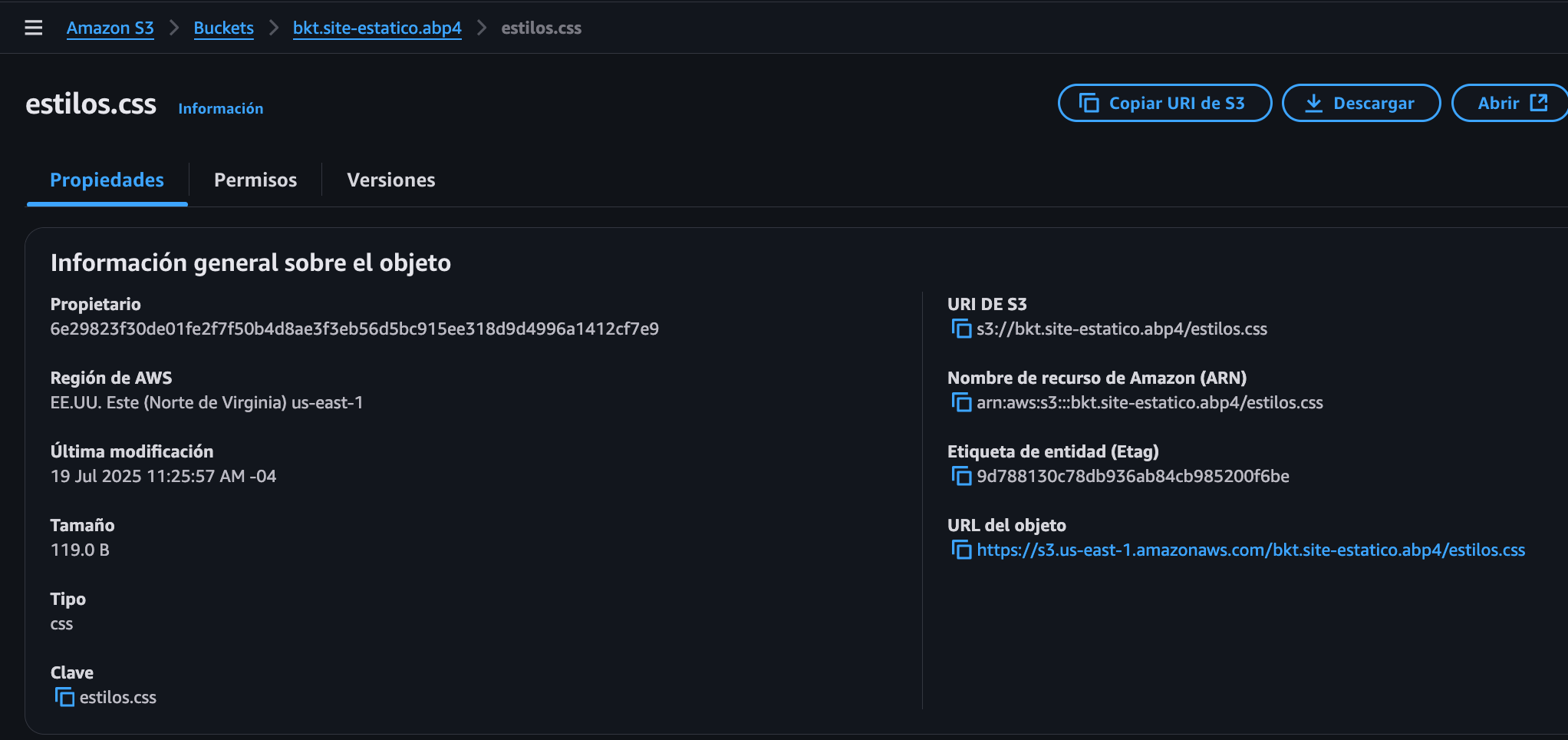
**

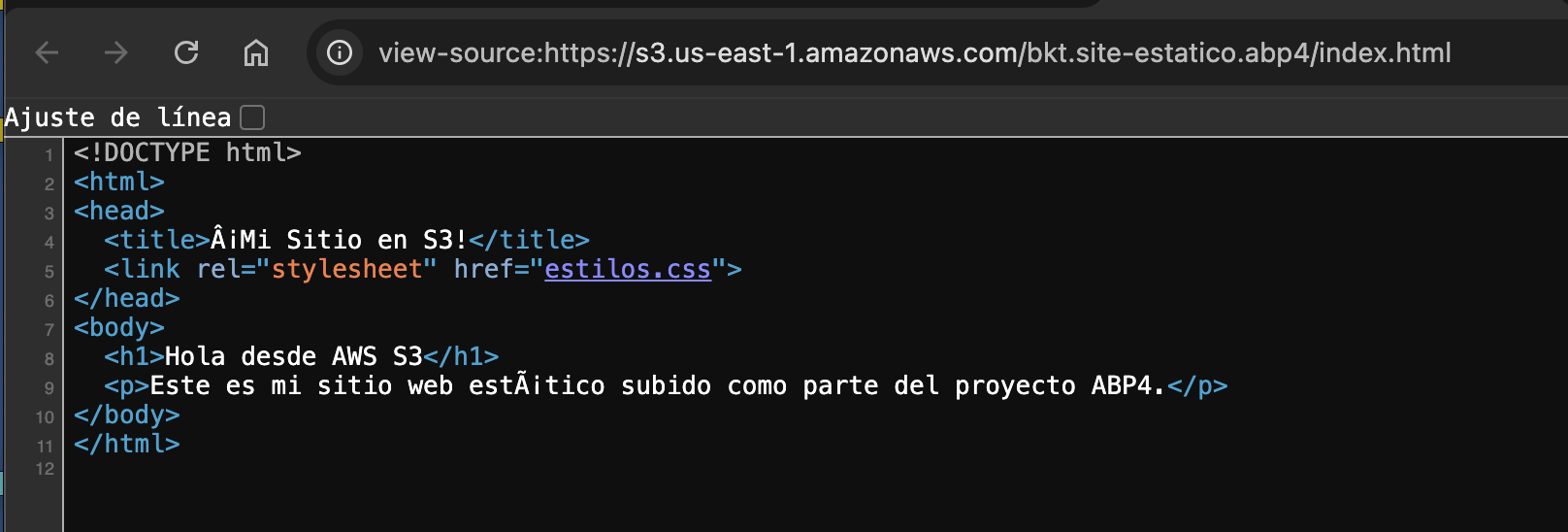
Luego se movieron a la ruta de archivos /var/www/html/ para exponerlos vía Apache:

sudo mv index.html /var/www/html/

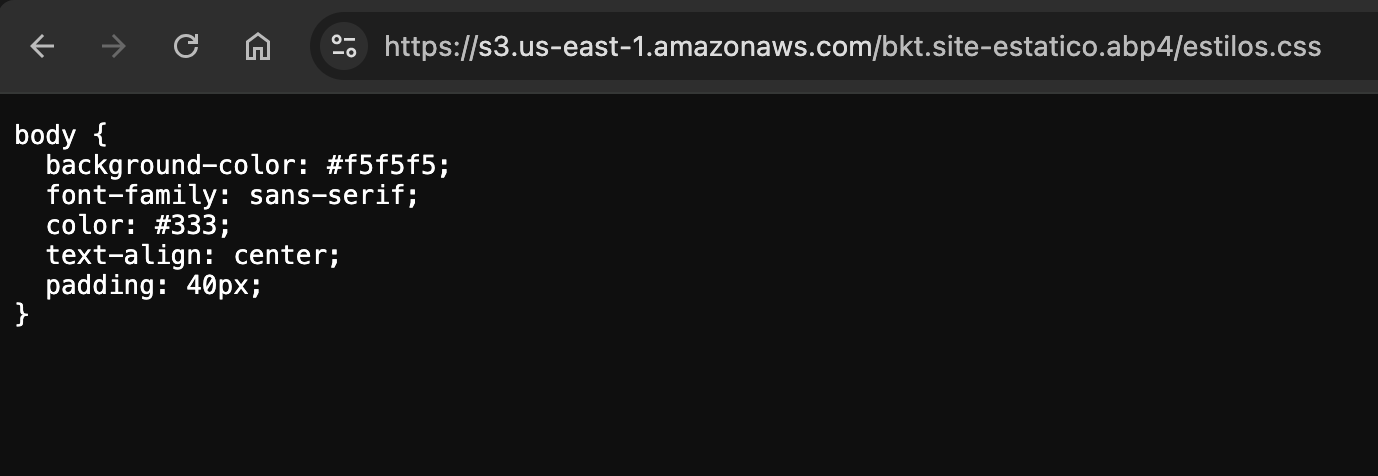
sudo mv estilos.css /var/www/html/

**

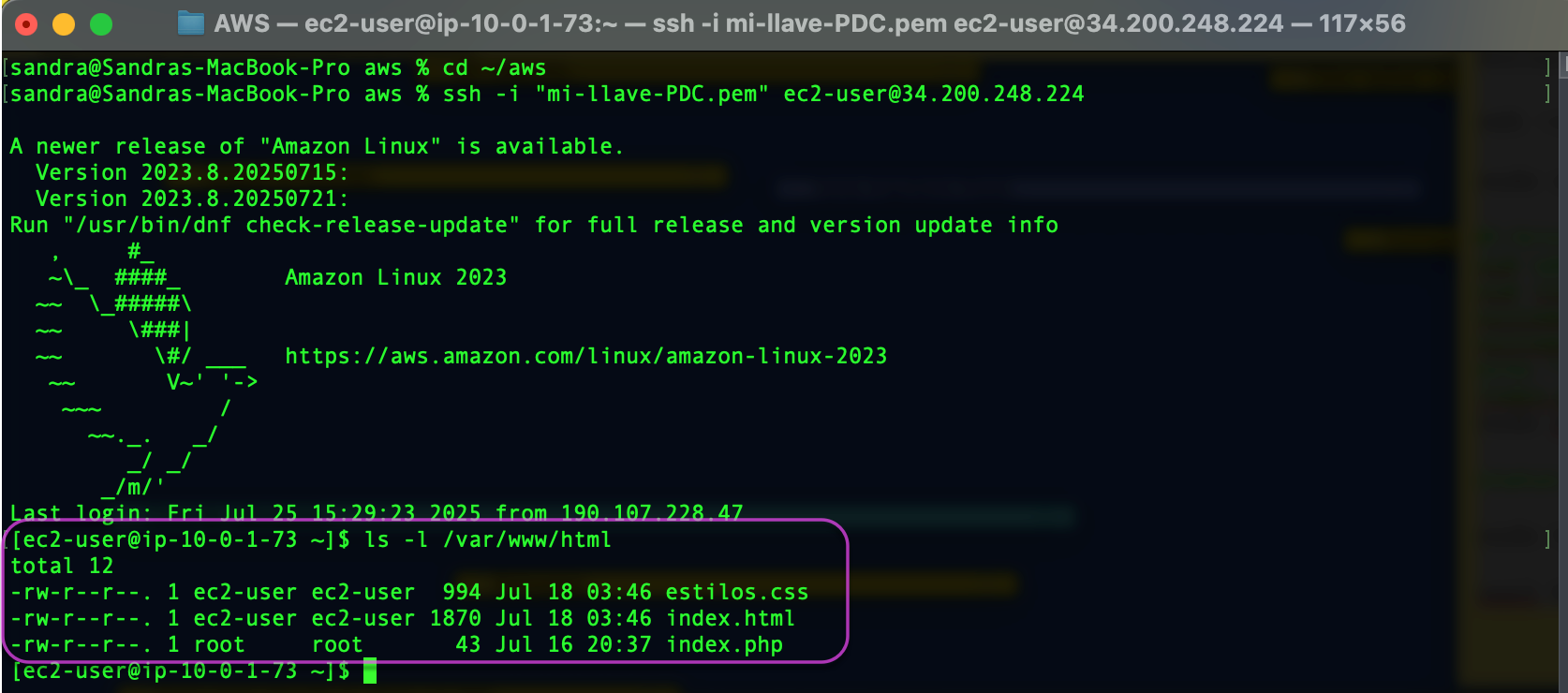
**

*Archivo index.html*

*Archivo estilos.css*



Archivos en el servidor EC2 (ls /var/www/html):



El sitio se pudo visualizar desde un navegador usando la IP pública:

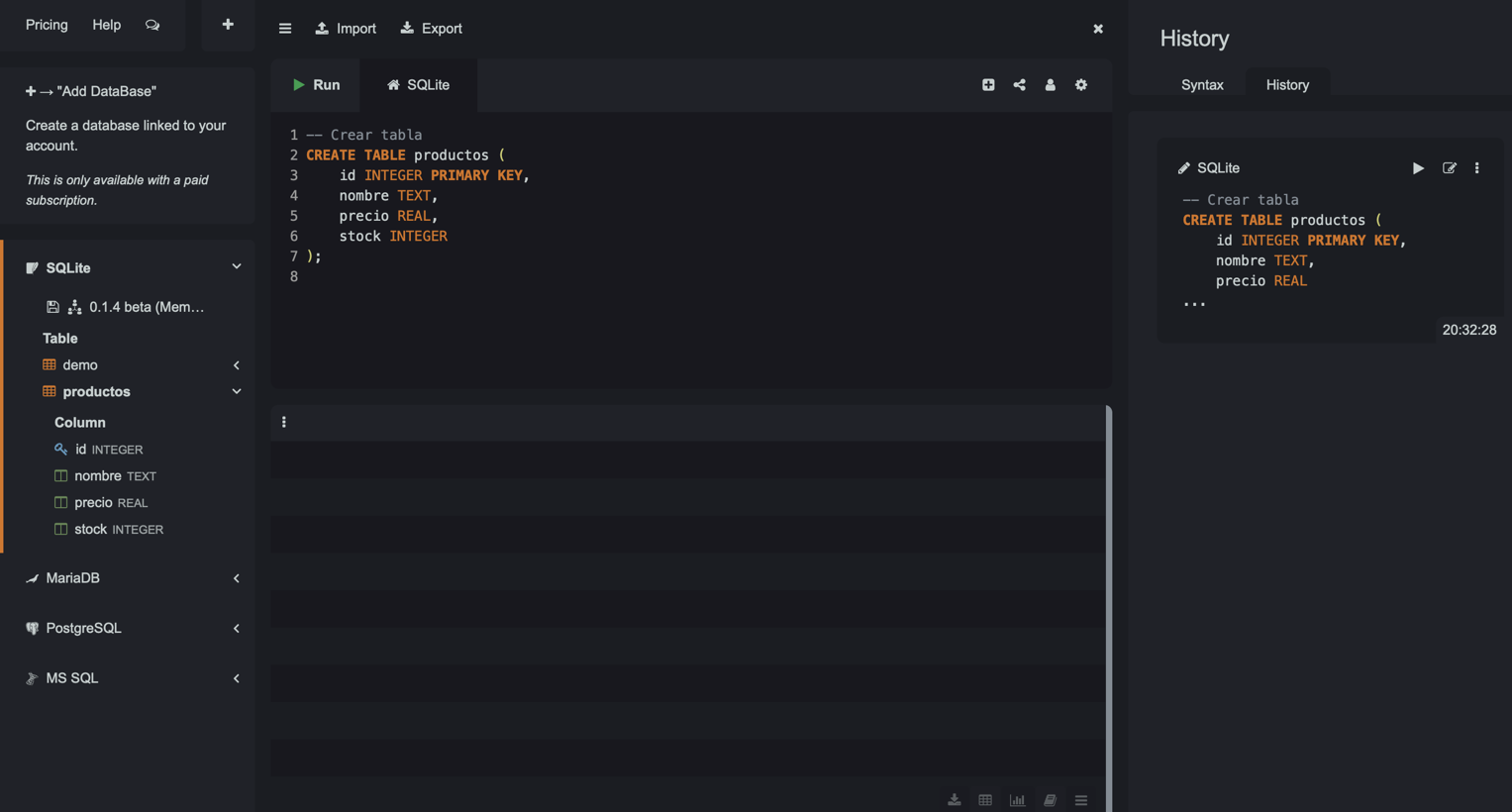
`http://44.193.75.196/`

**

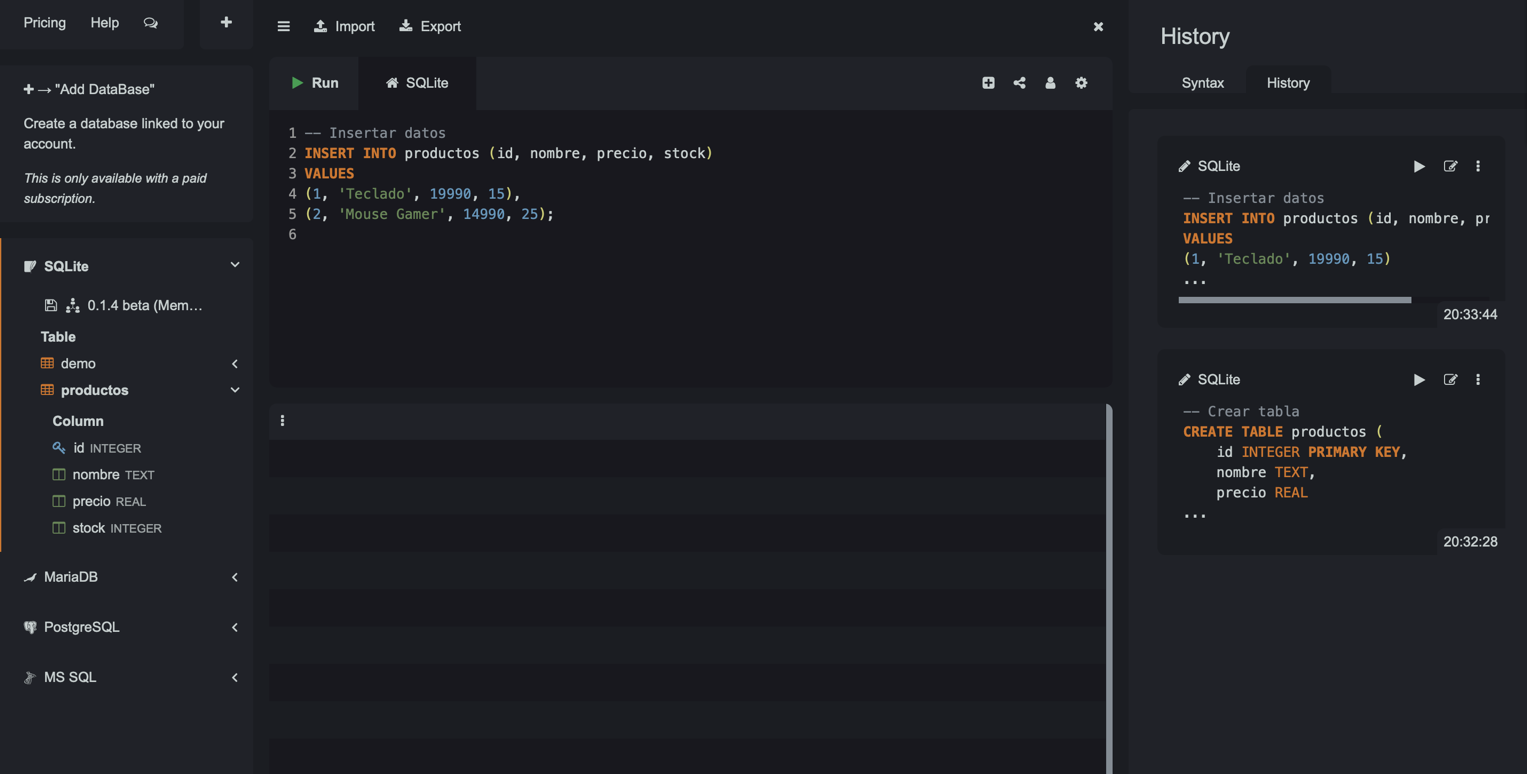
## 3. Demostración de consultas en SQLiteOnline

Para simular consultas de base de datos, se utilizó la herramienta [SQLiteOnline](https://sqliteonline.com/) con el siguiente contenido de ejemplo:

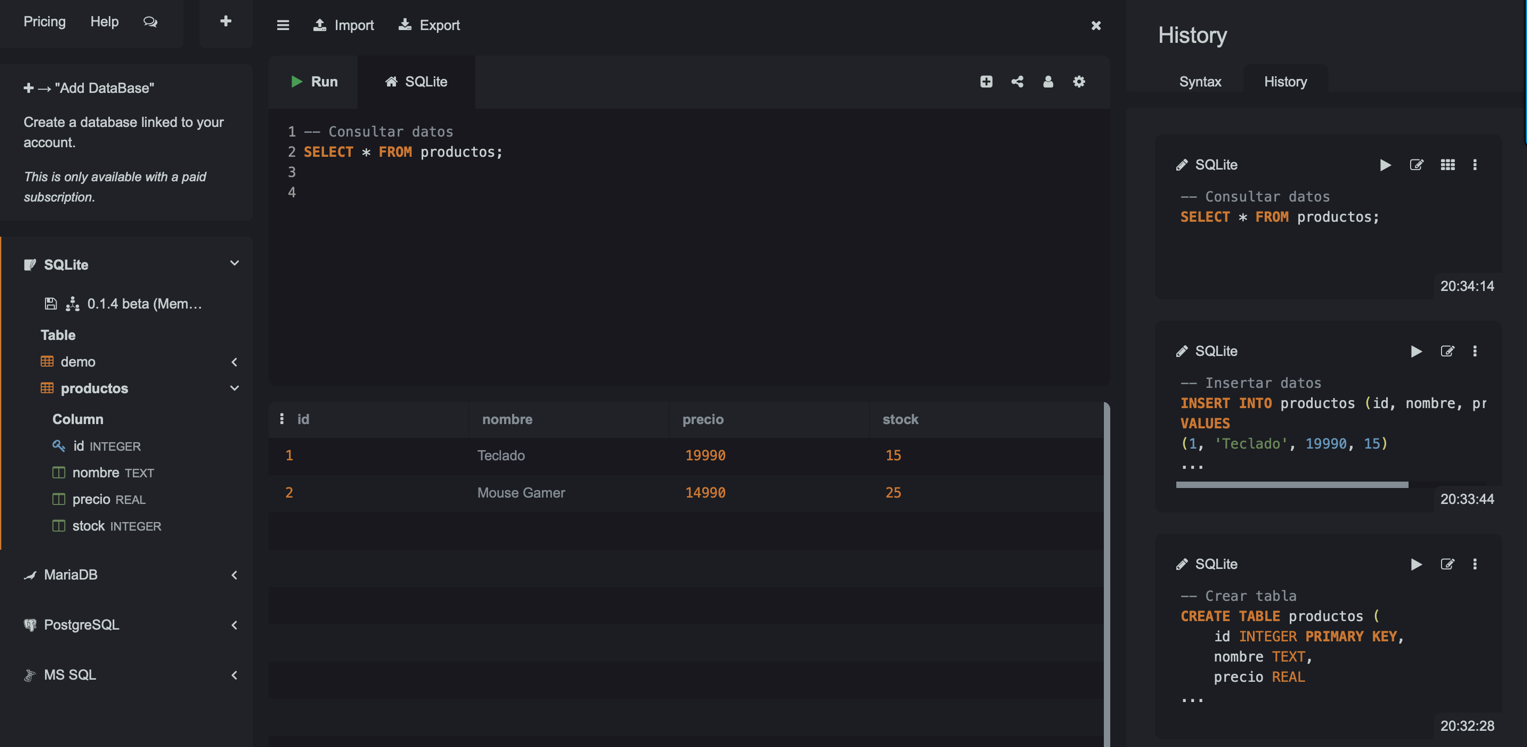
Create Table:



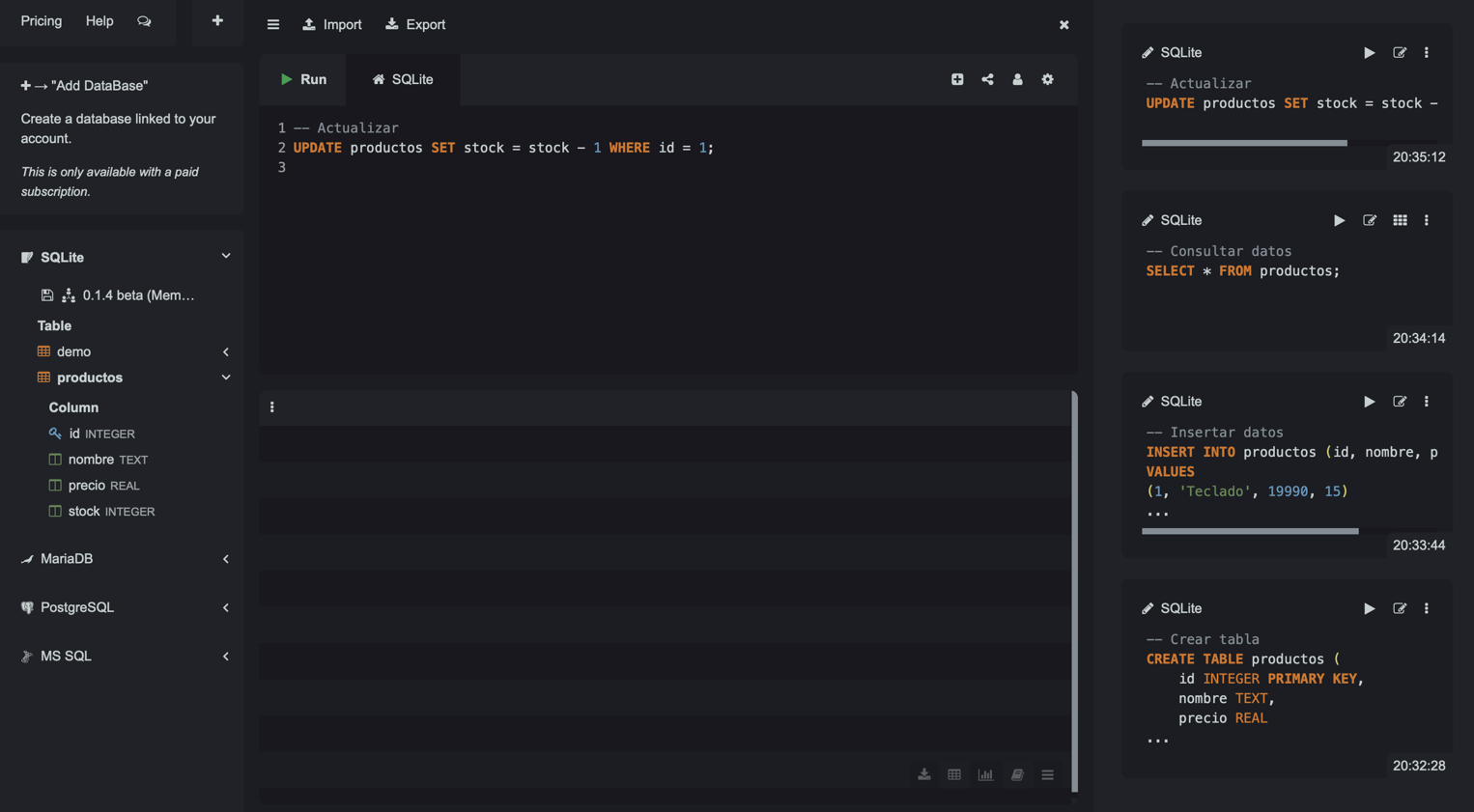
Insert:



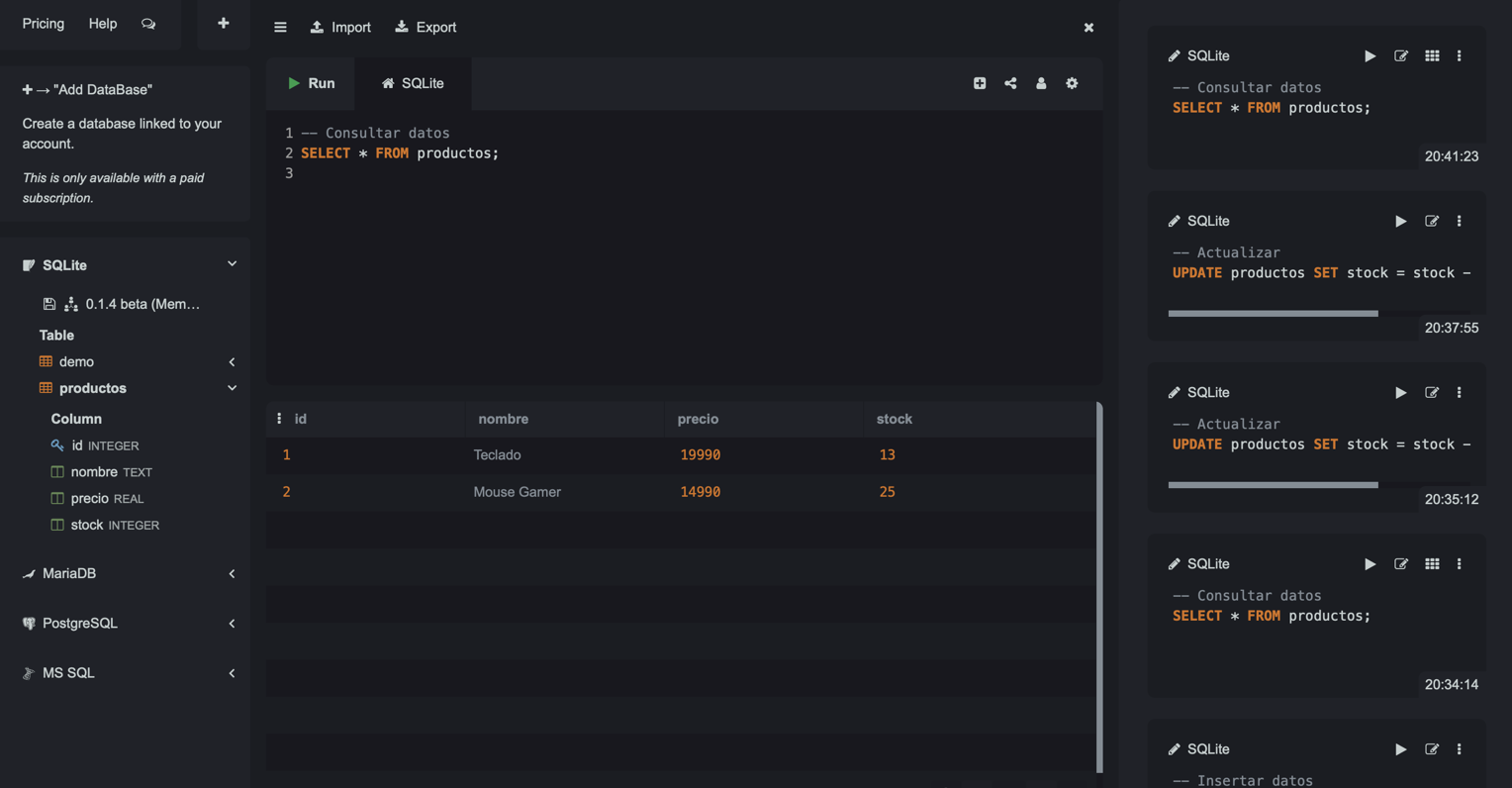
Select:



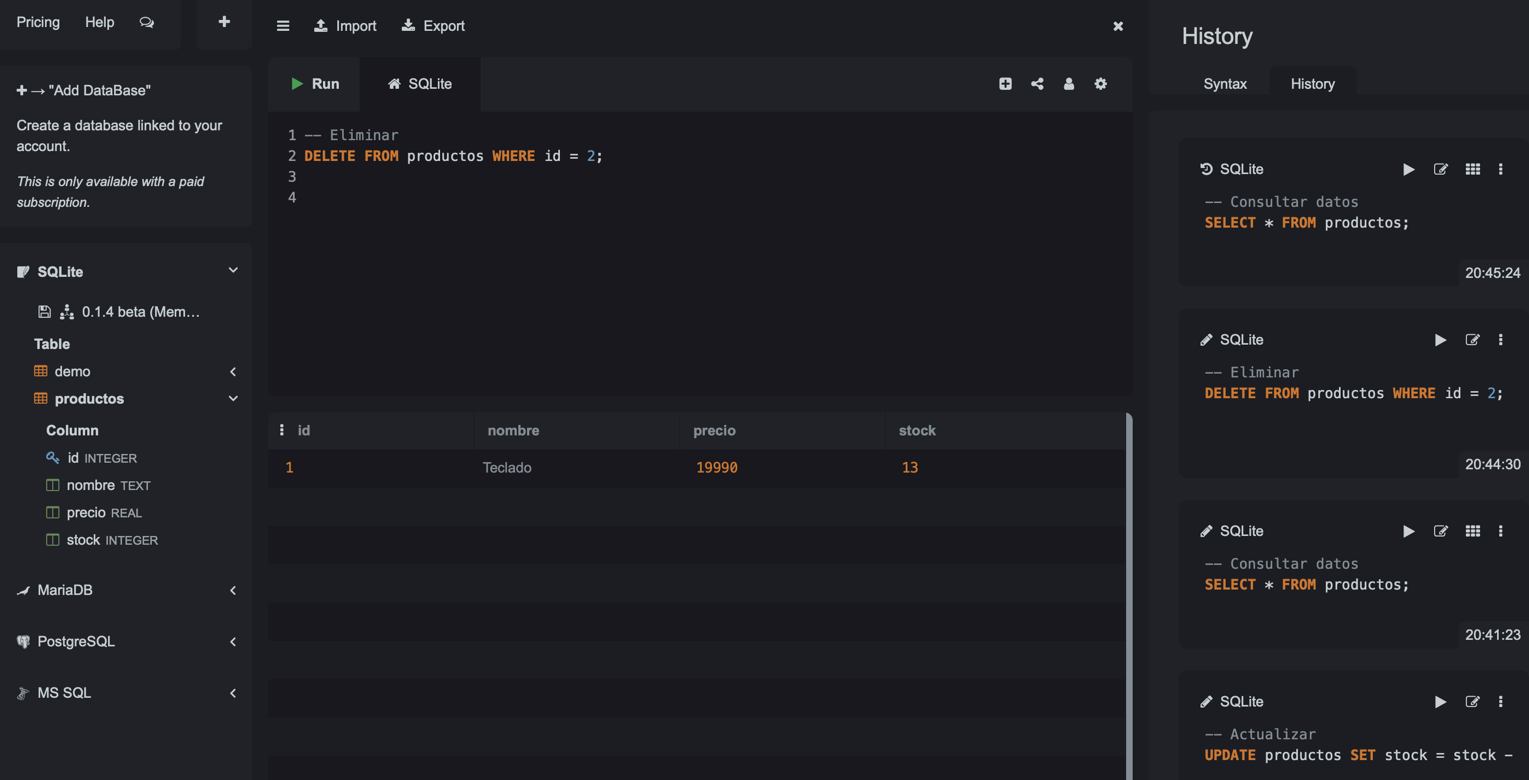
Update:



Select:



Delete:



**Propósito:**

* Validar la estructura lógica del modelo relacional sin necesidad de conectarse directamente a RDS.
* Usar SQLiteOnline como entorno liviano de prueba para demostrar dominio de SQL básico.

*📎 Evidencias y recursos vinculados*

* *Repositorio GitHub con archivos fuente: [inserta aquí tu enlace]*

## 4. Transición desde entorno on-premise a nube

El proyecto simula la transición de un entorno tradicional (on-premise) a uno cloud, aprovechando servicios gestionados de AWS para optimizar costos, rendimiento y seguridad. Se utilizó EC2 como infraestructura de cómputo, S3 como almacenamiento estático, y se complementó con SNS, SQS, y alarmas CloudWatch como parte de la automatización y monitoreo, con beneficios claros en términos de:

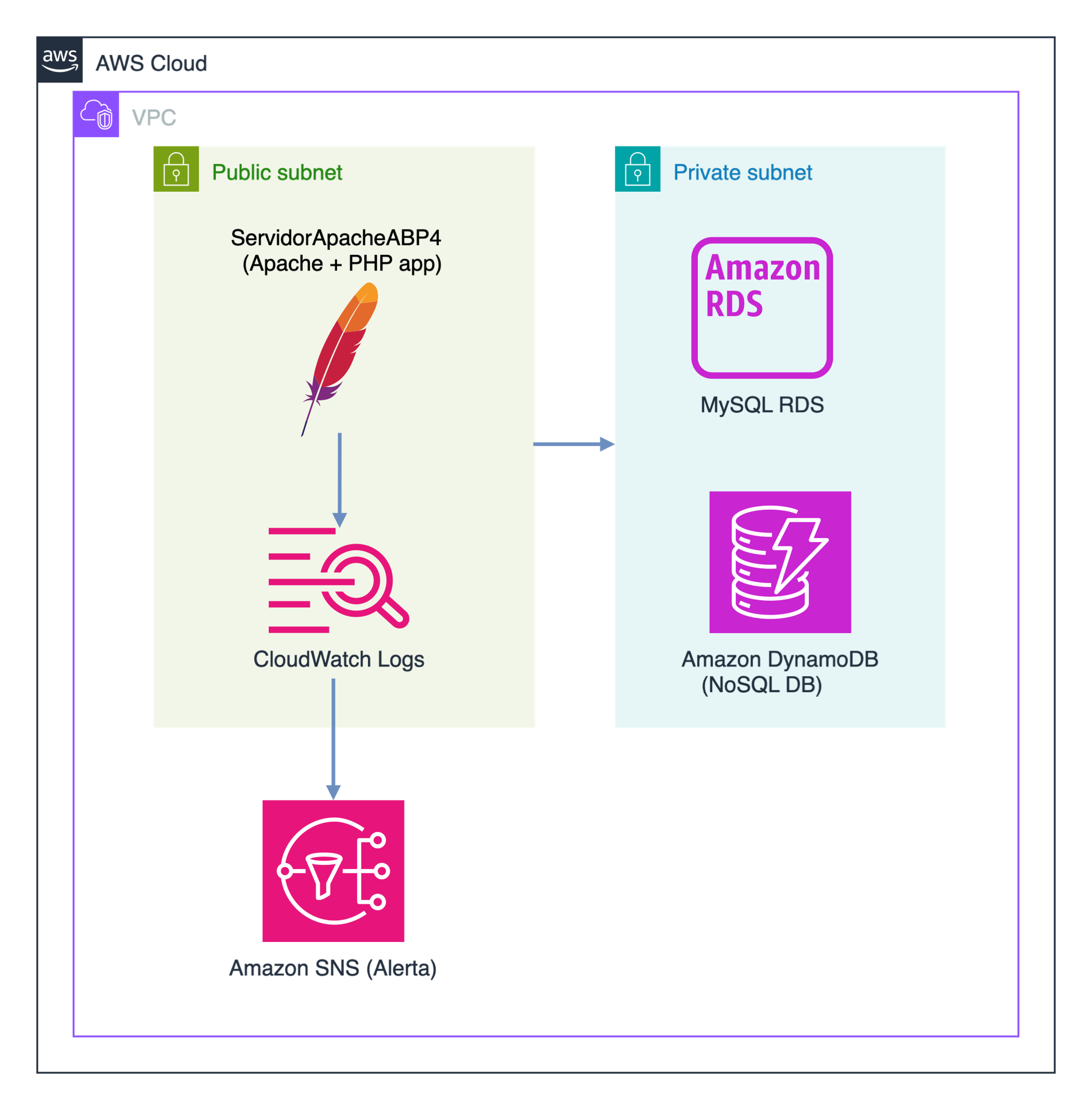
- **Escalabilidad**: usando instancias bajo demanda y almacenamiento distribuido.

- **Disponibilidad**: acceso desde cualquier punto geográfico vía HTTP/S.

- **Reducción de costos**: uso de capa gratuita de AWS, autoapagado y optimización de recursos.

- **Seguridad**: reglas explícitas de control de tráfico (SG, NACLs).

**Prototipo :**



## 5. Repositorio público

- Archivos recomendados para subir a GitHub

--- Trabajo • Distribución equitativa de tareas entre consumidores.

Aunque no se subió a un repositorio real, se recomienda:

### Contenidos para incluir en GitHub:

- `/html/index.html`

- `/html/estilos.css`

- `README.md` explicando cómo levantar la instancia, mover los archivos y verificar acceso.

## 6. Evidencias sugeridas (capturas de pantalla)

Aquí una lista de evidencias recomendadas para mostrar el desarrollo del prototipo:

* 📸 Archivos en el servidor EC2 (`ls /var/www/html`)
* 📸 Navegador mostrando el sitio web funcionando
* 📸 Configuración del grupo de seguridad
* 📸 Detalle del Bucket S3 y distribución en CloudFront (opcional)