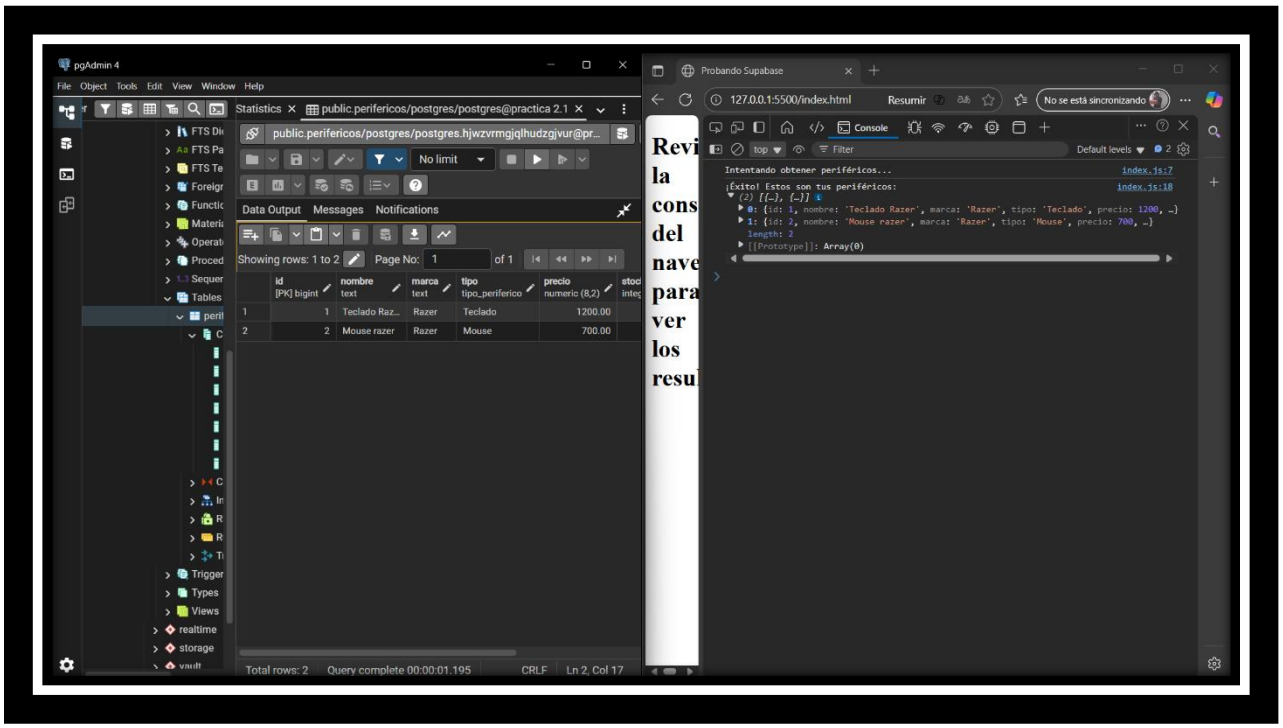
INSTITUTO POLITECNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

Práctica 3.1:"Alojamiento gratuito de bases de datos y aplicaciones web"

Mora Rodriguez Diego-3CV2

07/10/2025



Decisiones Tecnológicas

Para el desarrollo de nuestro proyecto, la selección de las herramientas y tecnologías fue un paso fundamental, guiado por los requerimientos específicos de la aplicación y la eficiencia del desarrollo.

Para la practica pondremos de prueba una tienda de periféricos, simplemente para tener una tabla rápida de crear.

En cuanto a la base de datos, se optó por utilizar PostgreSQL a través de la plataforma Supabase. Para un inventario de periféricos, es crucial mantener una estructura de datos fija y consistente, con relaciones claras entre entidades como productos, categorías y proveedores. PostgreSQL garantiza la integridad referencial y la consistencia de los datos mediante un esquema estricto, lo cual es más complejo de gestionar en un modelo basado en documentos como el de MongoDB. Supabase fue elegido como la plataforma que gestiona nuestra instancia de PostgreSQL porque ofrece una capa de servicios de backend que acelera significativamente el desarrollo. Nos proporcionó una API REST y de GraphQL automática sobre nuestra base de datos, autenticación de usuarios integrada y un sistema de seguridad robusto, lo que nos permitió enfocarnos en la lógica del frontend sin tener que construir un backend desde cero.

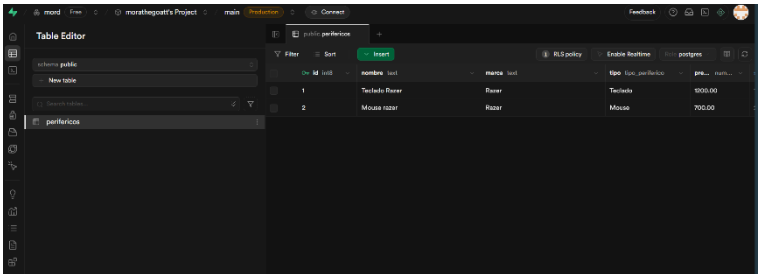


Ilustración 1 Visualización de la base de datos desde la plataforma supabase

Para el entorno de desarrollo y hosting inicial, se utilizó un servidor local a través de la extensión Live Server en Visual Studio Code. Esta elección nos brindó un ciclo de desarrollo extremadamente rápido, con recarga automática en el navegador al guardar cambios, lo que fue indispensable para la resolución de problemas de conexión. Nos permitió tener un control total sobre el entorno y obtener retroalimentación inmediata sin las complejidades de un proceso de despliegue continuo, que será implementado en una etapa posterior del proyecto.

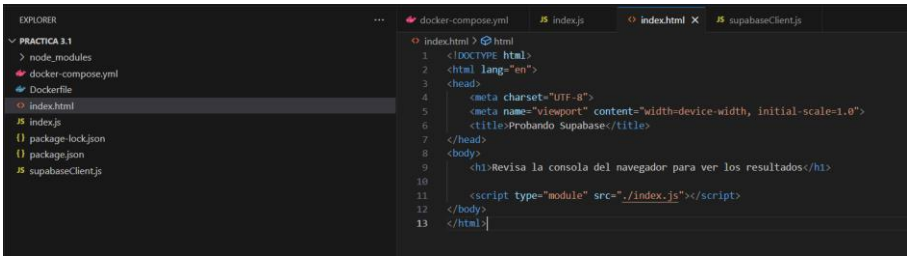


Ilustración 2 Creación y configuración del entorno, se observa la creación de la página de web básica para la visualización de la base de datos

El stack tecnológico para el cliente se mantuvo deliberadamente simple y fundamental. Se utilizó JavaScript vainilla, junto con HTML5 y CSS3. La elección de

no utilizar un framework como React o Vue en esta etapa inicial fue para establecer una prueba de concepto sólida, enfocándonos directamente en la comunicación con la API de Supabase sin abstracciones adicionales. La librería principal fue el cliente oficial de Supabase, @supabase/supabase-js, seleccionada por su documentación exhaustiva y por ser la herramienta recomendada oficialmente para interactuar con los servicios de la plataforma de manera segura y eficiente desde el navegador.

Complementando este stack, para la gestión y verificación directa de la base de datos, se utilizó pgAdmin 4. Esta herramienta nos permitió conectarnos a la instancia de PostgreSQL en Supabase desde un entorno local, cumpliendo así con el objetivo de la práctica de interactuar con la base de datos tanto a través de su API remota como mediante una conexión directa para tareas administrativas.

Arquitectura de solución y proceso de implementación

El proceso de implementación se centró en establecer una conexión funcional y segura entre una aplicación cliente y nuestra base de datos en Supabase, para luego estructurar y gestionar los datos de manera programática.

Ilustración 3 Configuración del deployment

La configuración inicial de la base de datos se realizó a través del panel de control de Supabase. Se creó una tabla inicial llamada "Productos" utilizando la interfaz gráfica para familiarizarnos con la plataforma. Posteriormente, para demostrar un manejo más técnico y escalable, se utilizó el Editor de SQL integrado. Se redactó y ejecutó un script SQL para crear una nueva tabla llamada "perifericos", la cual incluía la definición de tipos de datos específicos como ENUM para categorías, columnas con restricciones NOT NULL y valores por defecto, y una llave primaria autoincremental. Este enfoque nos permitió definir una estructura de datos robusta y precisa desde el inicio.

El proceso de conexión entre la aplicación y la base de datos fue el núcleo de la implementación. Se configuró un archivo cliente en JavaScript para inicializar la conexión utilizando la URL del proyecto y la clave API de tipo "anon" proporcionadas por Supabase. Estas credenciales se almacenaron en variables para ser utilizadas por la librería @supabase/supabase-js, la cual gestiona la comunicación con el backend. La implementación de la primera consulta para leer datos de la tabla fue directa, utilizando los métodos from() y select() proporcionados por la librería.

Durante el despliegue y la ejecución en el servidor local, nos enfrentamos a una serie de problemas que fueron cruciales para nuestro aprendizaje. El primer gran

obstáculo fue un error de red, net::ERR_NAME_NOT_RESOLVED. Inicialmente, se asumió que era un problema de configuración del código o del servidor. Sin embargo, tras un proceso de depuración sistemático que incluyó el uso de la herramienta ping desde la línea de comandos, se descubrió que la URL del proyecto almacenada en nuestras variables tenía un par de caracteres incorrectos. La lección aprendida fue la importancia de verificar las credenciales y puntos de conexión letra por letra.

Una vez corregida la URL, el siguiente problema fue un bloqueo por políticas de CORS (Cross-Origin Resource Sharing), que luego derivó en que la API devolvía un resultado exitoso pero con un arreglo vacío [], a pesar de que la tabla sí contenía datos. Este fue el desafío más significativo. La investigación nos llevó a descubrir el sistema de seguridad a nivel de fila (Row Level Security o RLS) de Supabase, el cual, por defecto, deniega todo acceso a los datos hasta que se creen políticas explícitas. La solución consistió en navegar a la sección de autenticación y políticas de la tabla en el panel de Supabase y redactar una nueva política de RLS utilizando una plantilla. Se creó una política que permitía explícitamente la operación de SELECT (lectura) para todos los usuarios. Al guardar esta política, la aplicación finalmente pudo leer y mostrar los datos correctamente en la consola del navegador, validando así todo el flujo de conexión.

Ilustración 4

Instrucciones de uso

Cómo acceder a la aplicación

Dado que el proyecto se encuentra en una fase de prueba de concepto, el acceso se realiza a través de un entorno de desarrollo local. Para ejecutar la aplicación, es necesario tener Visual Studio Code con la extensión "Live Server" instalada. Los pasos son los siguientes: primero, se debe abrir la carpeta del proyecto en el editor; segundo, hacer clic derecho sobre el archivo index.html y seleccionar la opción "Open with Live Server". Esto iniciará un servidor local y abrirá automáticamente la aplicación en el navegador web predeterminado, generalmente en una dirección como http://127.0.0.1:5500.

Para acceder localmente a la base de datos mediante pgAmin4, simplemente es ir al apartado de base de datos, click derecho y agregar un servidor, al crearlo tenemos que ir a el apartado de “conexiones” debemos de poner unos link que nos proporciona supabase, así como la contraseña y el host.

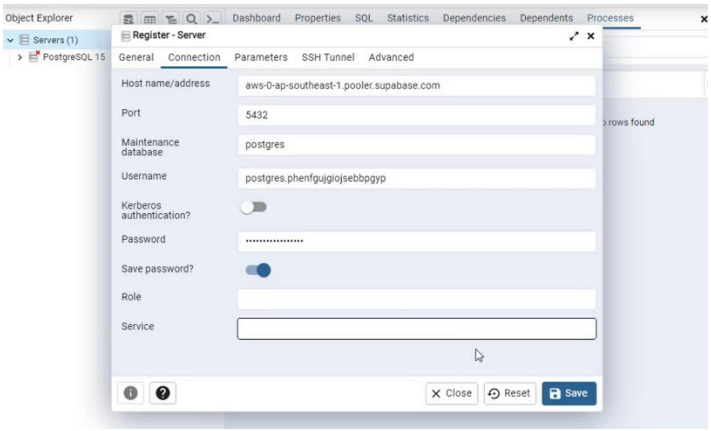


Ilustración 5 Registrar base de datos en pgAdmin4

Una vez con esto se conectara a la base de datos, tendremos que desglosar el menú y veremos ya la base de datos en el apartado “public”, desde ahí ya podemos visualizar y/o editar localmente haciendo uso de la interfaz grafica o SQL.

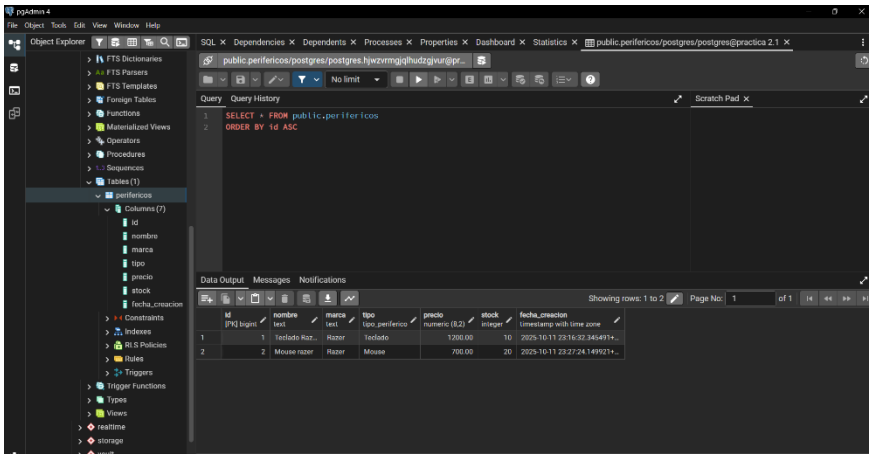


Ilustración 6 Base de datos mediante la interfaz grafica en pgAdmin

Funcionalidades principales

La funcionalidad central de esta implementación es establecer y validar la comunicación en tiempo real con la base de datos remota alojada en Supabase. Al cargar la aplicación en el navegador, esta ejecuta automáticamente un script que realiza una consulta de tipo SELECT a la tabla perifericos. La aplicación no cuenta con una interfaz gráfica para mostrar los datos; en su lugar, el resultado de la consulta se imprime directamente en la consola de desarrollador del navegador. Para visualizar los datos, el usuario debe abrir las herramientas de desarrollador (usualmente con la tecla F12) y navegar a la pestaña "Consola". El propósito de esta funcionalidad es servir como una prueba de concepto, demostrando que el flujo de datos desde el frontend hasta la base de datos es exitoso y está correctamente e

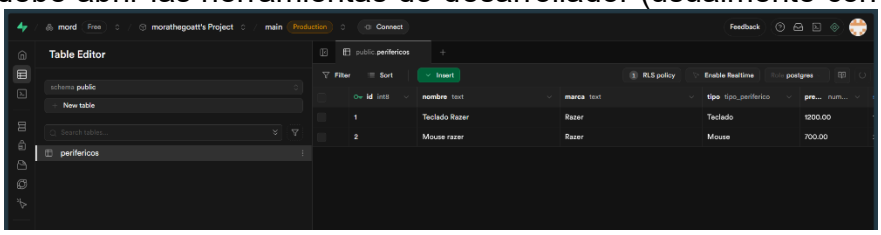


Ilustración 7 Base de datos desde supabase

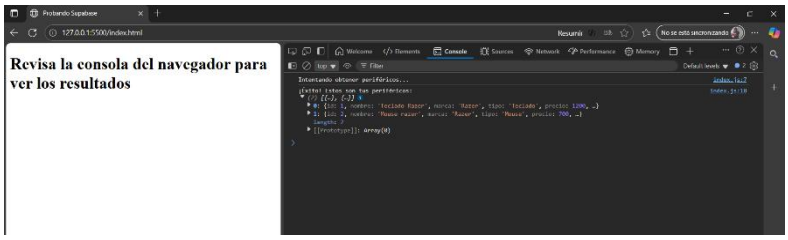


Ilustración 8 Visualización de las tablas desde la pagina web (URL pública)

Credenciales de prueba

Para la funcionalidad actual de la aplicación, no se requieren credenciales de prueba. La política de seguridad a nivel de fila (RLS) configurada en la tabla perifericos permite el acceso de lectura pública a todos los usuarios de forma anónima. Esto significa que cualquier persona que acceda a la aplicación puede ver el listado de productos sin necesidad de iniciar sesión o proporcionar credenciales. En futuras iteraciones del proyecto, se implementará un sistema de autenticación de usuarios a través de Supabase Auth, momento en el cual se definirán roles y credenciales específicas para operaciones de escritura, como crear, actualizar o eliminar registros.

Bibliografía:

Dey, R. (2025). *Live Server* [Extensión de Visual Studio Code]. Visual Studio Marketplace. https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer

Microsoft Corporation. (2025). *Visual Studio Code* [Software]. https://code.visualstudio.com

Supabase, Inc. (2025). *Supabase* [Plataforma como servicio]. https://supabase.com

The pgAdmin Development Team. (2025). *pgAdmin 4* [Software]. https://www.pgadmin.org

The PostgreSQL Global Development Group. (2025). *PostgreSQL* [Software]. https://www.postgresql.org