

Manual de Programador

1. Introducción

Esta aplicación está diseñada para gestionar un sistema de compras y ventas, brindando una interfaz amigable tanto para los usuarios finales como para los administradores. La arquitectura de la aplicación sigue el enfoque de una SPA (Single Page Application), donde el frontend está separado del backend, comunicándose a través de API REST para la gestión de datos y autenticación. Los usuarios pueden navegar, agregar productos al carrito, y completar el proceso de compra. Mientras tanto, los administradores pueden gestionar inventarios, monitorear ventas y visualizar información actualizada en tiempo real.

Tecnologías utilizadas

- **React:** Se utiliza para el desarrollo del frontend, proporcionando una interfaz dinámica e interactiva que mejora la experiencia del usuario.
- **Vite:** Herramienta de desarrollo que permite un proceso de construcción más rápido y eficiente para React, asegurando tiempos de carga y desarrollo optimizados.
- **Express:** Framework de Node.js que gestiona el backend, sirviendo las APIs que conectan la base de datos con el frontend.
- **MySQL:** Base de datos relacional que almacena la información de productos, usuarios, ventas e inventarios.
- **Firebase:** Usado principalmente para la autenticación de usuarios, permitiendo tanto el registro tradicional como el acceso mediante Google.

Despliegue

- **Frontend (Vercel):** El frontend de la aplicación está desplegado en Vercel, una plataforma que facilita el despliegue continuo y el hosting de aplicaciones frontend optimizadas.
- **Backend y base de datos (Railway):** Tanto el backend desarrollado con Express como la base de datos MySQL están desplegados en Railway, un servicio que permite la gestión de servidores y bases de datos con facilidad, integrando tanto la API como el almacenamiento de datos en un solo lugar.

2. Instalación y Configuración

Requisitos previos.

Deberá de tener instalado lo que es la ultima versión de node.js y npm

git clone <https://github.com/AlexHernandez2698632494/Aguila-Azul-Enterprises-.git>

Abrir el archivo clonado

1. Después de clonar el repositorio, abrir terminal de comandos:
2. Navegar a la carpeta server e instalar dependencias:

```
cd server
```

```
npm install
```

3. Navegar a la carpeta client e instalar dependencias:

```
cd client
```

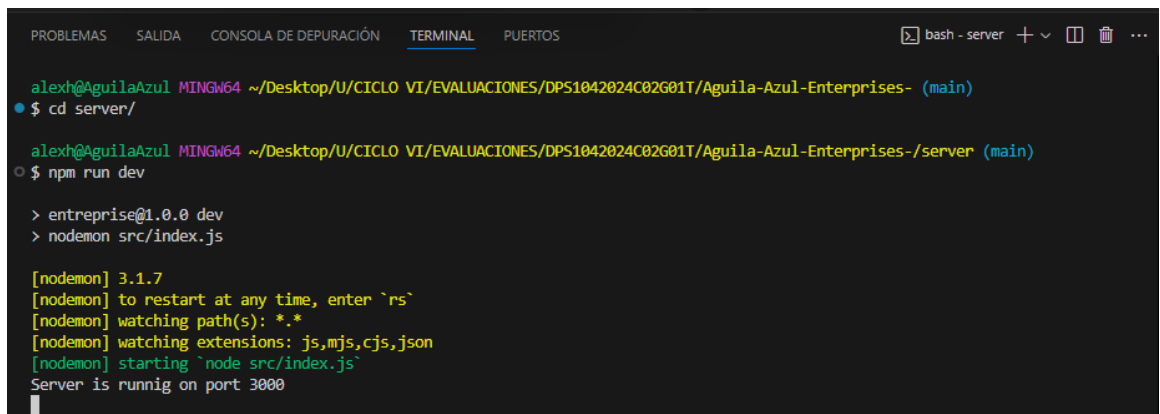
```
npm install
```

4. Descargar la base de datos:

El script de la base de datos la encontrara en la carpeta llamada "DATABASE" con el nombre "AAE.sql" correspondiente a mysql

5. Ejecutar el backend

Para poder ejecutar la api debera primero estar en la carpeta "server" podrá ejecutar el comando "cd server" y luego digitar el comando "npm run dev" en consola vera el mensaje "server is running on port 3000"



```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS
bash - server + - [] ...

alexh@AguilaAzul MINGW64 ~/Desktop/U/CICLO VI/EVALUACIONES/DPS1042024C02G01T/Aguila-Azul-Enterprises- (main)
$ cd server/

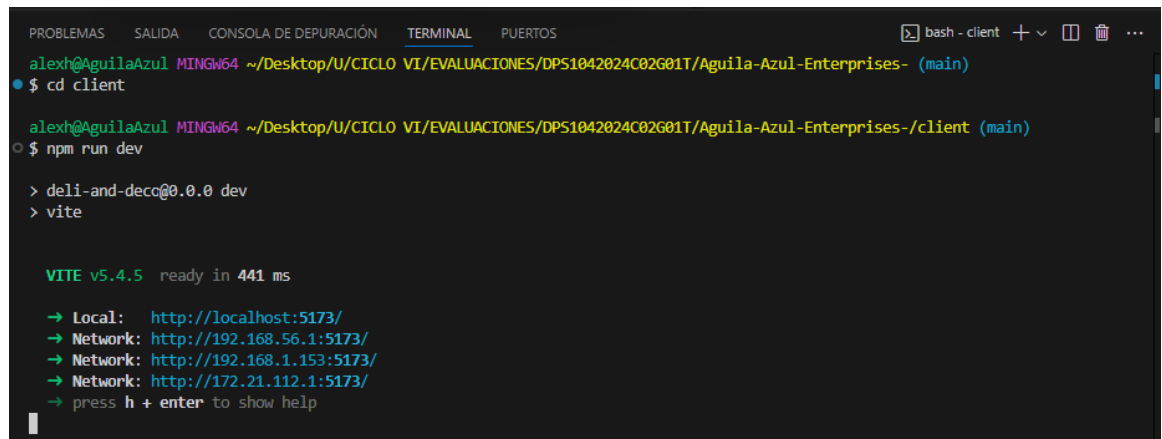
alexh@AguilaAzul MINGW64 ~/Desktop/U/CICLO VI/EVALUACIONES/DPS1042024C02G01T/Aguila-Azul-Enterprises-/server (main)
$ npm run dev

> enterprise@1.0.0 dev
> nodemon src/index.js

[nodemon] 3.1.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node src/index.js`
Server is running on port 3000
```

6. Ejecutar el frondEnd

Para poder ejecutar el cliente debera primero estar en la carpeta “client” podrá ejecutar el comando “cd client” y luego digitar el comando “npm run dev” en consola vera el mensaje con los diferentes enlaces que puede para ver el cliente



```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS
alexh@AguilaAzul MINGW64 ~/Desktop/U/CICLO VI/EVALUACIONES/DPS1042024C02G01T/Aguila-Azul-Enterprises- (main)
• $ cd client
alexh@AguilaAzul MINGW64 ~/Desktop/U/CICLO VI/EVALUACIONES/DPS1042024C02G01T/Aguila-Azul-Enterprises-/client (main)
○ $ npm run dev
> deli-and-deco@0.0.0 dev
> vite

VITE v5.4.5  ready in 441 ms
→ Local:   http://localhost:5173/
→ Network: http://192.168.56.1:5173/
→ Network: http://192.168.1.153:5173/
→ Network: http://172.21.112.1:5173/
→ press h + enter to show help
```

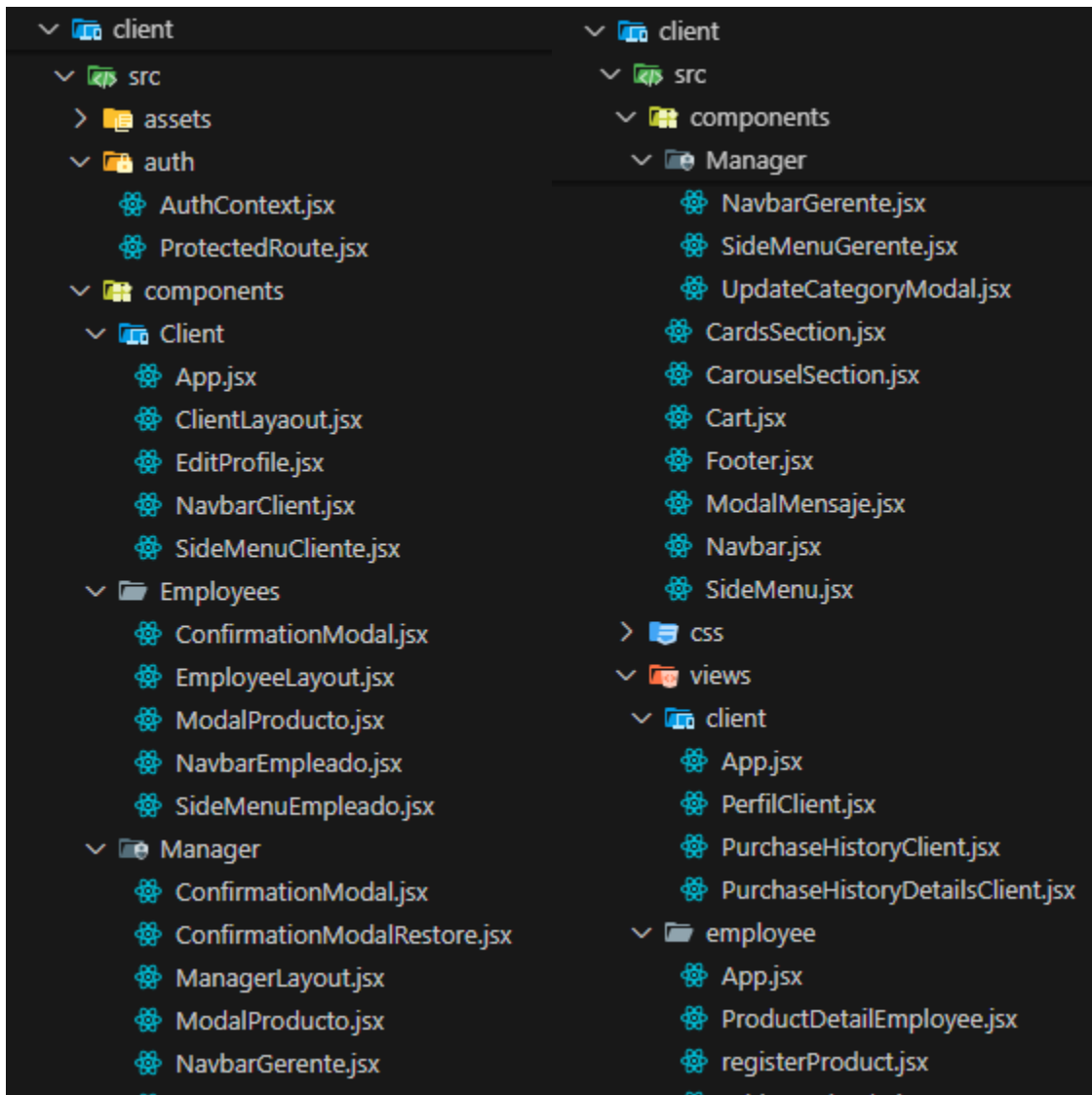
Configuración de variables de entorno.

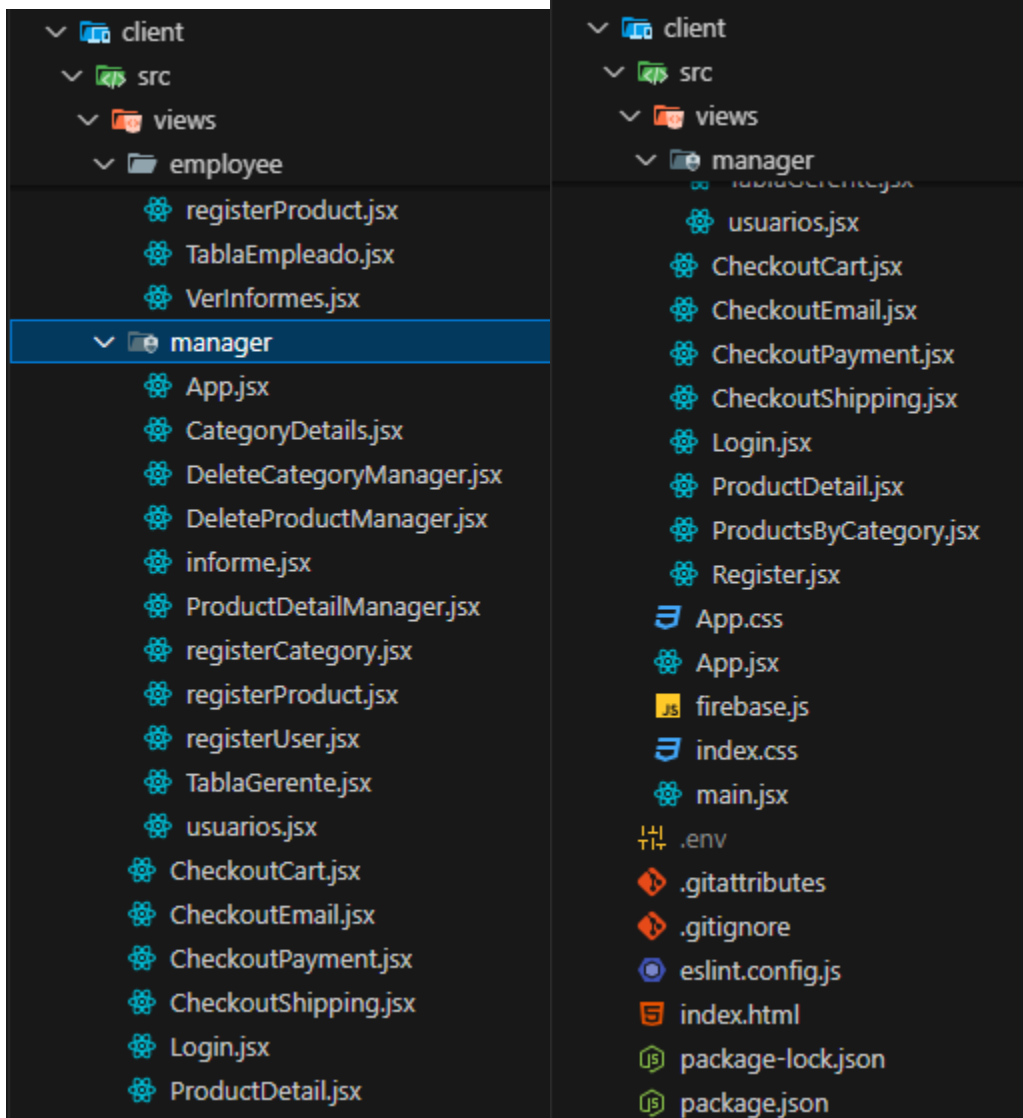
Las variables de entorno que debe crear son las siguientes

- a) Server:
 - host:'localhost'
 - port:3306
 - user:"
 - password:"
 - database:'aae'
- b) Cliente:
 - VITE_API_URL="",VITE_FIREBASE_API_KEY="",
 - VITE_FIREBASE_AUTH_DOMAIN="",
 - VITE_FIREBASE_PROJECT_ID="",
 - VITE_FIREBASE_STORAGE_BUCKET="",
 - VITE_FIREBASE_MESSAGING_SENDER_ID="",
 - VITE_FIREBASE_APP_ID=""

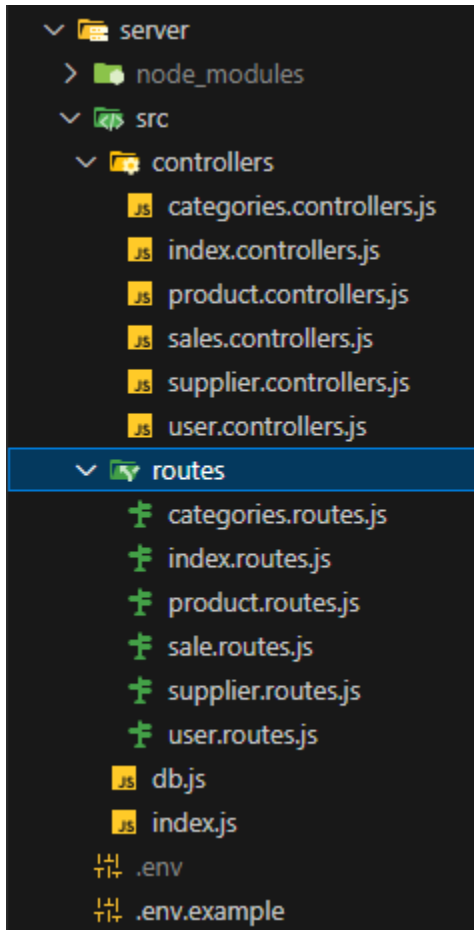
3. Estructura del Proyecto

1. Estructura de carpetas y archivos del frontend (React/Vite).





2. Estructura de carpetas y archivos del backend (Express).



4. Configuración de la Base de Datos

Detalles sobre la base de datos MySQL podrá encontrar el Scripts SQL para crear y poblar la base de datos en la carpeta “DATABASE”

5. Autenticación y Roles

Descripción de los roles: gerente, cliente y empleado.

En la base de datos encontrara la tabla usuarios en la cual hay un campo que se llama “NivelUsuario” en la cual solo se pueden ingresar datos 0 ó 1 ó 2 en 0 es para gerentes; el 1 es para empleados y el 2 es para clientes el login es para los tres roles pero el register es únicamente para clientes además se encuentra un botón para poder iniciar sesión con Google el cual se a configurado previamente firebase console obteniendo la configuracion necesario que dice Google las claves que se agregaron en el archivo .env en client y a la vez se configura el acceso directo de authentication y agregar iniciar con Google además si desea desplegar el proyecto deberá configurar los dominios para configurar los dominios

deberá ir a configuración luego en dominios autorizados y ahí agregar el dominio que le proporcione vercel

6. Rutas y Controladores en la API

Listado de las rutas principales (incluyendo las de autenticación).

Las rutas que se tienen son

1. Categorías:
 - `router.get('/categories',getCategories)`
 - `router.get('/categories/restore',getCategoriesRestore)`
 - `router.get('/category/:id',getCategory)`
 - `router.post('/category',createCategory)`
 - `router.put('/category/:id',updateCategory)`
 - `router.delete('/category/:id',deleteCategory)`
 - `router.delete('/category/restore/:id',restoreCategory)`
2. productos:
 - `router.get("/productsAll", getProducts);`
 - `router.get("/products", getProductsActive);`
 - `router.get("/productsI", getProductsInaActive);`
 - `router.get("/product/:id", getProduct);`
 - `router.get("/product/category/:id", getProductCategory);`
 - `router.get("/salesReport", getSalesReport);`
 - `router.get("/product/:id/inventory", getProductInventory);`
 - `router.post("/product", createProducts);`
 - `router.put("/product/:id", updateProducts);`
 - `router.delete("/product/:id", deleteProducts);`
 - `router.delete("/productRestore/:id", restoreProducts);`
3. ventas:
 - `router.post('/ventas', createVenta);`
 - `router.get('/ventas', getVentas);`
 - `router.get('/ventas/:id', getVenta);`
 - `router.get('/ventas/usuario/:usuario', getVentasByUsuario)`
 - `router.put('/ventas/:id/estado', updateEstadoEnvio);`
4. proveedores:
 - `router.get("/suppliers", getSuppliers);`
 - `router.get("/supplier/:id", getSupplier);`
 - `router.post("/supplier", createSuppliers);`
 - `router.put("/supplier/:id", updateSuppliers);`
 - `router.delete("/supplier/:id", deleteSuppliers);`
5. usuarios:

- `router.post("/register", registerUser);`
- `router.post("/login", loginUser);`
- `router.get("/users", getUsers);`
- `router.get("/users/:id", getUserById);`
- `router.put("/users/:id", updateUser);`
- `router.delete("/users/:id", deleteUser);`
- `router.put("/users/change-password/:id", changeUserPassword);`
- `router.get("/users/username/:usuario", getUserByUsername);`

7. Componentes y Vistas del Frontend

Descripción de las vistas principales de la aplicación.

Se podrá observar que esta configurado que hay vistas y componentes que son general para todo usuario y carpeta por cada rol con los archivos necesarios para sus componentes y vistas

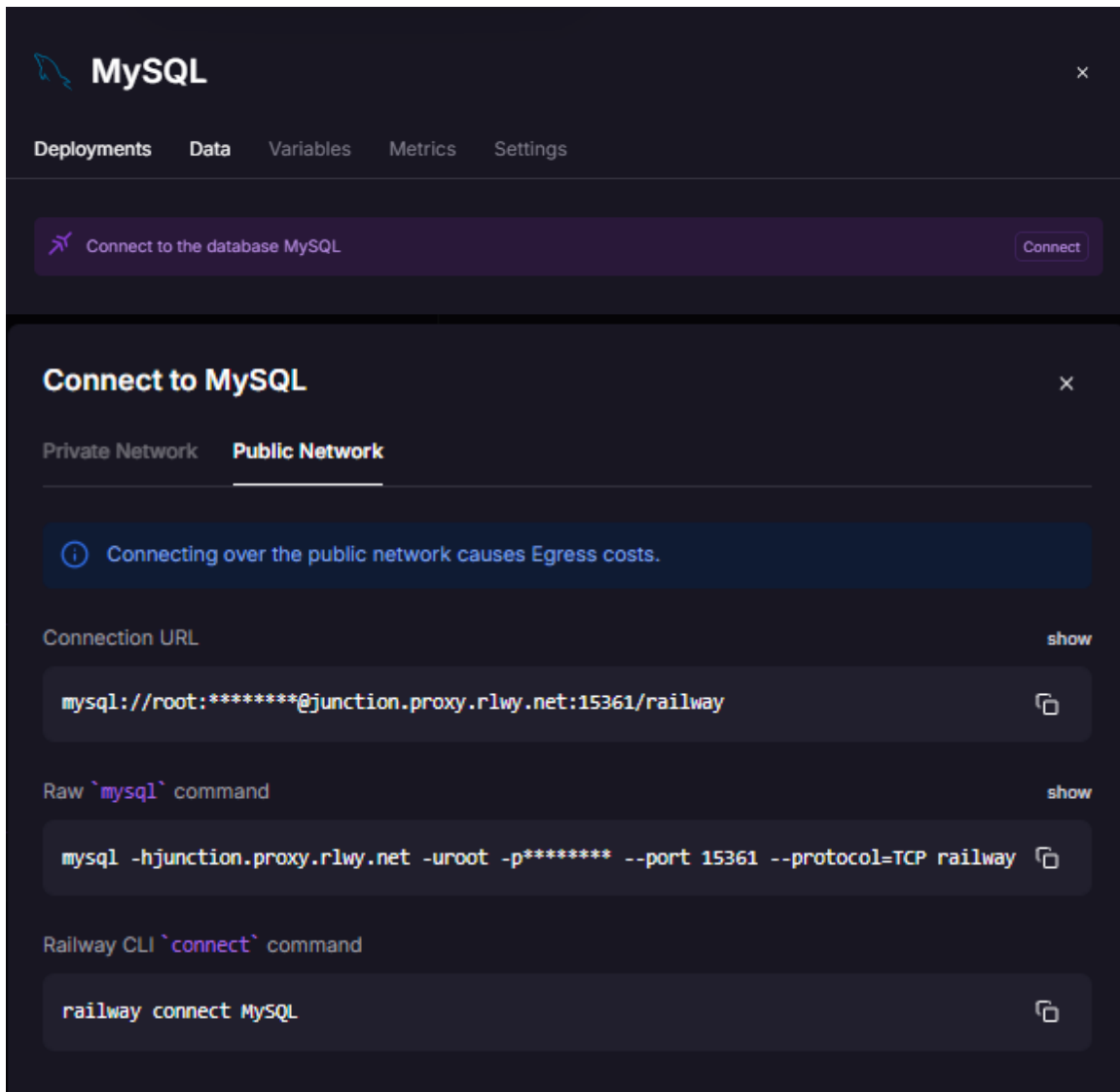
8. Despliegue

Desplegar el frontend en Vercel.

Para poder desplegar en vercel únicamente se necesita subir el proyecto a github luego crear la cuenta en vercel con la cuenta de github y seguir los pasos muy importante es antes de desplegar agregar las variables de entorno

Desplegar el backend y la base de datos en Railway.

Railway trabaja también con github se sube el proyecto a github muy parecido a vercel y se colocan las variables de entorno solo que en este caso son los nombres que se mencionan en el apartado de instalación y configuración y su valor lo proporcionara railway cuando crea la base de datos en data le sale un mensaje morado con un botón que dice connect le damos ahí y después public network la segunda opción.



9. Pruebas y Depuración

Cómo realizar pruebas en la API (Postman, Jest, etc.).

Puede colocar localhost:3000/la ruta y según el método get,post,put o delete como ejemplo localhost:3000/categories esto muestra todas las categorías

Para el frondend son las que le dan en la terminal cuando lo ejecuta

```
→ Local: http://localhost:5173/
→ Network: http://192.168.56.1:5173/
→ Network: http://192.168.1.153:5173/
→ Network: http://172.21.112.1:5173/
→ press h + enter to show help
```