# 🚀 Guía de Instalación y Configuración Completa - PepsiCo Fleet Management

**Proyecto:** PepsiCo Fleet Management System

**Versión:** 1.0.0

**Fecha:** Octubre 15, 2024

**Equipo:** Joaquín Marín & Benjamin Vilches

---

## 📋 Índice

1. [Requisitos del Sistema](#requisitos-del-sistema)
2. [Instalación en Windows](#instalación-en-windows)
3. [Instalación en macOS](#instalación-en-macos)
4. [Instalación en Linux](#instalación-en-linux)
5. [Configuración de Base de Datos](#configuración-de-base-de-datos)
6. [Configuración de Variables de Entorno](#configuración-de-variables-de-entorno)
7. [Configuración de Desarrollo](#configuración-de-desarrollo)
8. [Configuración de Producción](#configuración-de-producción)
9. [Troubleshooting](#troubleshooting)
10. [Verificación de Instalación](#verificación-de-instalación)

---

## 1. Requisitos del Sistema

### 1.1 Requisitos Mínimos

#### Hardware

* \*\*CPU:\*\* 2 cores, 2.0 GHz
* \*\*RAM:\*\* 4 GB mínimo, 8 GB recomendado
* \*\*Disco:\*\* 10 GB de espacio libre
* \*\*Red:\*\* Conexión a internet para dependencias

#### Software

* \*\*Node.js:\*\* 20.0.0 o superior
* \*\*npm:\*\* 10.0.0 o superior
* \*\*Git:\*\* 2.40.0 o superior
* \*\*PostgreSQL:\*\* 15.0 o superior (opcional, se puede usar Neon/Supabase)

### 1.2 Requisitos Recomendados

#### Hardware

* \*\*CPU:\*\* 4 cores, 3.0 GHz
* \*\*RAM:\*\* 16 GB
* \*\*Disco:\*\* 50 GB SSD
* \*\*Red:\*\* Conexión estable a internet

#### Software

* \*\*Node.js:\*\* 20.18.0 LTS
* \*\*npm:\*\* 10.2.0
* \*\*Git:\*\* 2.42.0
* \*\*PostgreSQL:\*\* 15.4
* \*\*VS Code:\*\* Última versión
* \*\*Docker:\*\* 24.0.0 (opcional)

---

## 2. Instalación en Windows

### 2.1 Instalar Node.js

#### Opción 1: Descarga Directa

1. Ir a [nodejs.org](https://nodejs.org)
2. Descargar la versión LTS (20.18.0)
3. Ejecutar el instalador
4. Seguir las instrucciones del wizard
5. Verificar instalación:

node --version  
npm --version

#### Opción 2: Usando Chocolatey

# Instalar Chocolatey (si no está instalado)  
Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))  
  
# Instalar Node.js  
choco install nodejs  
  
# Verificar instalación  
node --version  
npm --version

#### Opción 3: Usando winget

# Instalar Node.js  
winget install OpenJS.NodeJS  
  
# Verificar instalación  
node --version  
npm --version

### 2.2 Instalar Git

#### Opción 1: Descarga Directa

1. Ir a [git-scm.com](https://git-scm.com)
2. Descargar Git para Windows
3. Ejecutar el instalador
4. Configurar Git:

git config --global user.name "Tu Nombre"  
git config --global user.email "tu.email@ejemplo.com"

#### Opción 2: Usando Chocolatey

choco install git

### 2.3 Instalar PostgreSQL (Opcional)

#### Opción 1: Descarga Directa

1. Ir a [postgresql.org](https://www.postgresql.org/download/windows/)
2. Descargar PostgreSQL 15
3. Ejecutar el instalador
4. Configurar contraseña del usuario postgres
5. Verificar instalación:

psql --version

#### Opción 2: Usando Chocolatey

choco install postgresql15

### 2.4 Instalar VS Code

#### Opción 1: Descarga Directa

1. Ir a [code.visualstudio.com](https://code.visualstudio.com)
2. Descargar VS Code
3. Ejecutar el instalador

#### Opción 2: Usando winget

winget install Microsoft.VisualStudioCode

### 2.5 Extensiones Recomendadas para VS Code

{  
 "recommendations": [  
 "ms-vscode.vscode-typescript-next",  
 "bradlc.vscode-tailwindcss",  
 "esbenp.prettier-vscode",  
 "ms-vscode.vscode-eslint",  
 "prisma.prisma",  
 "ms-vscode.vscode-json",  
 "ms-vscode.vscode-git",  
 "formulahendry.auto-rename-tag",  
 "christian-kohler.path-intellisense",  
 "ms-vscode.vscode-thunder-client"  
 ]  
}

---

## 3. Instalación en macOS

### 3.1 Instalar Node.js

#### Opción 1: Usando Homebrew (Recomendado)

# Instalar Homebrew (si no está instalado)  
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"  
  
# Instalar Node.js  
brew install node  
  
# Verificar instalación  
node --version  
npm --version

#### Opción 2: Descarga Directa

1. Ir a [nodejs.org](https://nodejs.org)
2. Descargar la versión LTS para macOS
3. Ejecutar el instalador
4. Verificar instalación:

node --version  
npm --version

#### Opción 3: Usando nvm (Node Version Manager)

# Instalar nvm  
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.0/install.sh | bash  
  
# Reiniciar terminal o ejecutar  
source ~/.bashrc  
  
# Instalar Node.js LTS  
nvm install --lts  
nvm use --lts  
  
# Verificar instalación  
node --version  
npm --version

### 3.2 Instalar Git

#### Usando Homebrew

brew install git  
  
# Configurar Git  
git config --global user.name "Tu Nombre"  
git config --global user.email "tu.email@ejemplo.com"

### 3.3 Instalar PostgreSQL (Opcional)

#### Usando Homebrew

# Instalar PostgreSQL  
brew install postgresql@15  
  
# Iniciar servicio  
brew services start postgresql@15  
  
# Crear base de datos  
createdb pepsico\_fleet  
  
# Verificar instalación  
psql --version

### 3.4 Instalar VS Code

#### Usando Homebrew

brew install --cask visual-studio-code

---

## 4. Instalación en Linux

### 4.1 Ubuntu/Debian

#### Instalar Node.js

# Actualizar sistema  
sudo apt update && sudo apt upgrade -y  
  
# Instalar Node.js usando NodeSource  
curl -fsSL https://deb.nodesource.com/setup\_20.x | sudo -E bash -  
sudo apt-get install -y nodejs  
  
# Verificar instalación  
node --version  
npm --version

#### Instalar Git

sudo apt install git -y  
  
# Configurar Git  
git config --global user.name "Tu Nombre"  
git config --global user.email "tu.email@ejemplo.com"

#### Instalar PostgreSQL (Opcional)

# Instalar PostgreSQL  
sudo apt install postgresql postgresql-contrib -y  
  
# Iniciar servicio  
sudo systemctl start postgresql  
sudo systemctl enable postgresql  
  
# Crear usuario y base de datos  
sudo -u postgres psql  
CREATE USER pepsico\_user WITH PASSWORD 'password123';  
CREATE DATABASE pepsico\_fleet OWNER pepsico\_user;  
GRANT ALL PRIVILEGES ON DATABASE pepsico\_fleet TO pepsico\_user;  
\q  
  
# Verificar instalación  
psql --version

### 4.2 CentOS/RHEL/Fedora

#### Instalar Node.js

# Instalar Node.js usando NodeSource  
curl -fsSL https://rpm.nodesource.com/setup\_20.x | sudo bash -  
sudo yum install -y nodejs  
  
# Verificar instalación  
node --version  
npm --version

#### Instalar Git

sudo yum install git -y

#### Instalar PostgreSQL (Opcional)

# Instalar PostgreSQL  
sudo yum install postgresql-server postgresql-contrib -y  
  
# Inicializar base de datos  
sudo postgresql-setup initdb  
  
# Iniciar servicio  
sudo systemctl start postgresql  
sudo systemctl enable postgresql

### 4.3 Arch Linux

#### Instalar Node.js

# Actualizar sistema  
sudo pacman -Syu  
  
# Instalar Node.js  
sudo pacman -S nodejs npm  
  
# Verificar instalación  
node --version  
npm --version

#### Instalar Git

sudo pacman -S git

#### Instalar PostgreSQL (Opcional)

# Instalar PostgreSQL  
sudo pacman -S postgresql  
  
# Inicializar base de datos  
sudo -u postgres initdb -D /var/lib/postgres/data  
  
# Iniciar servicio  
sudo systemctl start postgresql  
sudo systemctl enable postgresql

---

## 5. Configuración de Base de Datos

### 5.1 Opción 1: PostgreSQL Local

#### Crear Base de Datos

-- Conectar como postgres  
psql -U postgres  
  
-- Crear base de datos  
CREATE DATABASE pepsico\_fleet;  
  
-- Crear usuario  
CREATE USER pepsico\_user WITH PASSWORD 'password123';  
  
-- Asignar permisos  
GRANT ALL PRIVILEGES ON DATABASE pepsico\_fleet TO pepsico\_user;  
  
-- Salir  
\q

#### Configurar Variables de Entorno

DATABASE\_URL="postgresql://pepsico\_user:password123@localhost:5432/pepsico\_fleet?schema=public"

### 5.2 Opción 2: Neon (Recomendado para Desarrollo)

#### Crear Cuenta en Neon

1. Ir a [neon.tech](https://neon.tech)
2. Crear cuenta gratuita
3. Crear nuevo proyecto
4. Copiar connection string

#### Configurar Variables de Entorno

DATABASE\_URL="postgresql://username:password@ep-xxx.us-east-1.aws.neon.tech/pepsico\_fleet?sslmode=require"

### 5.3 Opción 3: Supabase

#### Crear Proyecto en Supabase

1. Ir a [supabase.com](https://supabase.com)
2. Crear cuenta
3. Crear nuevo proyecto
4. Ir a Settings > Database
5. Copiar connection string

#### Configurar Variables de Entorno

DATABASE\_URL="postgresql://postgres:password@db.xxx.supabase.co:5432/postgres"

---

## 6. Configuración de Variables de Entorno

### 6.1 Backend (.env)

# ============================================================================  
# DATABASE CONFIGURATION  
# ============================================================================  
DATABASE\_URL="postgresql://username:password@localhost:5432/pepsico\_fleet?schema=public"  
  
# ============================================================================  
# SERVER CONFIGURATION  
# ============================================================================  
PORT=3000  
NODE\_ENV=development  
FRONTEND\_URL=http://localhost:5173  
  
# ============================================================================  
# JWT CONFIGURATION  
# ============================================================================  
JWT\_SECRET="cambiar-en-produccion-por-algo-muy-seguro-y-largo-minimo-32-caracteres"  
JWT\_EXPIRES\_IN=15m  
JWT\_REFRESH\_EXPIRES\_IN=7d  
  
# ============================================================================  
# CORS CONFIGURATION  
# ============================================================================  
CORS\_ORIGIN=http://localhost:5173  
CORS\_CREDENTIALS=true  
  
# ============================================================================  
# RATE LIMITING  
# ============================================================================  
RATE\_LIMIT\_WINDOW\_MS=900000  
RATE\_LIMIT\_MAX\_REQUESTS=100  
  
# ============================================================================  
# LOGGING  
# ============================================================================  
LOG\_LEVEL=info  
LOG\_FILE\_PATH=./logs  
  
# ============================================================================  
# EMAIL CONFIGURATION (Opcional)  
# ============================================================================  
SMTP\_HOST=smtp.gmail.com  
SMTP\_PORT=587  
SMTP\_USER=tu\_email@gmail.com  
SMTP\_PASS=tu\_app\_password  
SMTP\_FROM=noreply@pepsico.cl  
  
# ============================================================================  
# FILE UPLOAD (Opcional)  
# ============================================================================  
MAX\_FILE\_SIZE=10485760  
UPLOAD\_PATH=./uploads  
  
# ============================================================================  
# CLOUDINARY (Opcional)  
# ============================================================================  
CLOUDINARY\_CLOUD\_NAME=  
CLOUDINARY\_API\_KEY=  
CLOUDINARY\_API\_SECRET=

### 6.2 Frontend (.env)

# ============================================================================  
# API CONFIGURATION  
# ============================================================================  
VITE\_API\_URL=http://localhost:3000/api  
  
# ============================================================================  
# APP CONFIGURATION  
# ============================================================================  
VITE\_APP\_NAME=Gestión de Flota PepsiCo  
VITE\_APP\_VERSION=1.0.0  
  
# ============================================================================  
# ENVIRONMENT  
# ============================================================================  
VITE\_NODE\_ENV=development  
  
# ============================================================================  
# FEATURES FLAGS (Opcional)  
# ============================================================================  
VITE\_ENABLE\_ANALYTICS=false  
VITE\_ENABLE\_DEBUG=true

---

## 7. Configuración de Desarrollo

### 7.1 Clonar Repositorio

# Clonar repositorio  
git clone https://github.com/tu-usuario/pepsico-fleet-management.git  
cd pepsico-fleet-management  
  
# Verificar que estás en la rama correcta  
git branch  
git checkout main

### 7.2 Instalar Dependencias

# Instalar dependencias del proyecto raíz  
npm install  
  
# O instalar por separado  
cd backend && npm install  
cd ../frontend && npm install

### 7.3 Configurar Base de Datos

cd backend  
  
# Copiar archivo de variables de entorno  
cp env.example.txt .env  
  
# Editar variables de entorno  
# (Editar .env con tus credenciales)  
  
# Generar cliente Prisma  
npm run db:generate  
  
# Ejecutar migraciones  
npm run db:migrate  
  
# (Opcional) Poblar con datos de prueba  
npm run db:seed  
  
# (Opcional) Abrir Prisma Studio  
npm run db:studio

### 7.4 Configurar Frontend

cd frontend  
  
# Copiar archivo de variables de entorno  
cp env.example.txt .env  
  
# Editar variables de entorno  
# (Editar .env con la URL de la API)

### 7.5 Ejecutar en Modo Desarrollo

#### Opción 1: Ejecutar Todo desde la Raíz

# Desde la raíz del proyecto  
npm run dev

#### Opción 2: Ejecutar por Separado

**Terminal 1 - Backend:**

cd backend  
npm run dev

**Terminal 2 - Frontend:**

cd frontend  
npm run dev

### 7.6 Verificar Instalación

#### Verificar Backend

# Health check  
curl http://localhost:3000/health  
  
# API endpoints  
curl http://localhost:3000/api/auth/login

#### Verificar Frontend

# Abrir navegador  
open http://localhost:5173

---

## 8. Configuración de Producción

### 8.1 Variables de Entorno de Producción

#### Backend (.env.production)

# ============================================================================  
# DATABASE CONFIGURATION  
# ============================================================================  
DATABASE\_URL="postgresql://prod\_user:secure\_password@prod-host:5432/pepsico\_fleet\_prod?sslmode=require"  
  
# ============================================================================  
# SERVER CONFIGURATION  
# ============================================================================  
PORT=3000  
NODE\_ENV=production  
FRONTEND\_URL=https://fleet.pepsico.cl  
  
# ============================================================================  
# JWT CONFIGURATION  
# ============================================================================  
JWT\_SECRET="super-secret-production-key-minimum-64-characters-long"  
JWT\_EXPIRES\_IN=15m  
JWT\_REFRESH\_EXPIRES\_IN=7d  
  
# ============================================================================  
# CORS CONFIGURATION  
# ============================================================================  
CORS\_ORIGIN=https://fleet.pepsico.cl  
CORS\_CREDENTIALS=true  
  
# ============================================================================  
# RATE LIMITING  
# ============================================================================  
RATE\_LIMIT\_WINDOW\_MS=900000  
RATE\_LIMIT\_MAX\_REQUESTS=1000  
  
# ============================================================================  
# LOGGING  
# ============================================================================  
LOG\_LEVEL=warn  
LOG\_FILE\_PATH=/var/log/pepsico-fleet  
  
# ============================================================================  
# EMAIL CONFIGURATION  
# ============================================================================  
SMTP\_HOST=smtp.pepsico.cl  
SMTP\_PORT=587  
SMTP\_USER=noreply@pepsico.cl  
SMTP\_PASS=secure\_email\_password  
SMTP\_FROM=noreply@pepsico.cl  
  
# ============================================================================  
# FILE UPLOAD  
# ============================================================================  
MAX\_FILE\_SIZE=10485760  
UPLOAD\_PATH=/var/uploads  
  
# ============================================================================  
# CLOUDINARY  
# ============================================================================  
CLOUDINARY\_CLOUD\_NAME=pepsico-fleet  
CLOUDINARY\_API\_KEY=your\_api\_key  
CLOUDINARY\_API\_SECRET=your\_api\_secret

#### Frontend (.env.production)

# ============================================================================  
# API CONFIGURATION  
# ============================================================================  
VITE\_API\_URL=https://api.fleet.pepsico.cl/api  
  
# ============================================================================  
# APP CONFIGURATION  
# ============================================================================  
VITE\_APP\_NAME=Gestión de Flota PepsiCo  
VITE\_APP\_VERSION=1.0.0  
  
# ============================================================================  
# ENVIRONMENT  
# ============================================================================  
VITE\_NODE\_ENV=production  
  
# ============================================================================  
# FEATURES FLAGS  
# ============================================================================  
VITE\_ENABLE\_ANALYTICS=true  
VITE\_ENABLE\_DEBUG=false

### 8.2 Build para Producción

#### Backend

cd backend  
  
# Instalar dependencias de producción  
npm ci --only=production  
  
# Compilar TypeScript  
npm run build  
  
# Ejecutar migraciones  
npm run db:migrate:deploy  
  
# Iniciar aplicación  
npm start

#### Frontend

cd frontend  
  
# Instalar dependencias  
npm ci  
  
# Build para producción  
npm run build  
  
# Los archivos estarán en dist/

### 8.3 Configuración de Servidor

#### Nginx Configuration

server {  
 listen 80;  
 server\_name fleet.pepsico.cl;  
   
 # Redirect HTTP to HTTPS  
 return 301 https://$server\_name$request\_uri;  
}  
  
server {  
 listen 443 ssl http2;  
 server\_name fleet.pepsico.cl;  
   
 # SSL Configuration  
 ssl\_certificate /path/to/certificate.crt;  
 ssl\_certificate\_key /path/to/private.key;  
   
 # Frontend  
 location / {  
 root /var/www/pepsico-fleet/dist;  
 index index.html;  
 try\_files $uri $uri/ /index.html;  
 }  
   
 # API Backend  
 location /api {  
 proxy\_pass http://localhost:3000;  
 proxy\_http\_version 1.1;  
 proxy\_set\_header Upgrade $http\_upgrade;  
 proxy\_set\_header Connection 'upgrade';  
 proxy\_set\_header Host $host;  
 proxy\_set\_header X-Real-IP $remote\_addr;  
 proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  
 proxy\_set\_header X-Forwarded-Proto $scheme;  
 proxy\_cache\_bypass $http\_upgrade;  
 }  
}

#### PM2 Configuration

// ecosystem.config.js  
module.exports = {  
 apps: [{  
 name: 'pepsico-fleet-api',  
 script: './dist/index.js',  
 instances: 'max',  
 exec\_mode: 'cluster',  
 env: {  
 NODE\_ENV: 'production',  
 PORT: 3000  
 },  
 error\_file: './logs/err.log',  
 out\_file: './logs/out.log',  
 log\_file: './logs/combined.log',  
 time: true  
 }]  
}

---

## 9. Troubleshooting

### 9.1 Problemas Comunes

#### Error: "Cannot connect to database"

# Verificar que PostgreSQL esté ejecutándose  
sudo systemctl status postgresql  
  
# Verificar conexión  
psql -U postgres -h localhost  
  
# Verificar variables de entorno  
echo $DATABASE\_URL

#### Error: "Port already in use"

# Windows  
netstat -ano | findstr :3000  
taskkill /PID <pid> /F  
  
# Linux/Mac  
lsof -ti:3000 | xargs kill

#### Error: "Module not found"

# Limpiar y reinstalar  
rm -rf node\_modules package-lock.json  
npm install  
  
# Verificar versiones  
node --version  
npm --version

#### Error: "Prisma client not generated"

cd backend  
npm run db:generate

#### Error: "Migration failed"

# Resetear base de datos (¡CUIDADO!)  
npm run db:reset  
  
# O aplicar migraciones manualmente  
npm run db:migrate:deploy

### 9.2 Logs y Debugging

#### Ver Logs del Backend

# Logs en tiempo real  
tail -f backend/logs/combined.log  
  
# Solo errores  
tail -f backend/logs/error.log  
  
# Buscar errores específicos  
grep "ERROR" backend/logs/combined.log

#### Debug del Frontend

# Abrir DevTools en el navegador  
# F12 o Ctrl+Shift+I  
  
# Ver requests de red  
# Tab Network  
  
# Ver estado de la aplicación  
# Tab Console

### 9.3 Performance Issues

#### Backend Lento

# Verificar uso de CPU y memoria  
top  
htop  
  
# Verificar conexiones de base de datos  
psql -c "SELECT \* FROM pg\_stat\_activity;"  
  
# Verificar índices  
psql -c "\d+ table\_name"

#### Frontend Lento

# Verificar bundle size  
npm run build  
ls -la dist/  
  
# Verificar dependencias  
npm ls --depth=0

---

## 10. Verificación de Instalación

### 10.1 Checklist de Verificación

#### Backend

* [ ] Node.js instalado y funcionando
* [ ] Dependencias instaladas
* [ ] Variables de entorno configuradas
* [ ] Base de datos conectada
* [ ] Migraciones ejecutadas
* [ ] Servidor iniciado en puerto 3000
* [ ] Health check responde OK
* [ ] API endpoints funcionando

#### Frontend

* [ ] Dependencias instaladas
* [ ] Variables de entorno configuradas
* [ ] Servidor de desarrollo iniciado en puerto 5173
* [ ] Aplicación carga en el navegador
* [ ] Login funciona
* [ ] Dashboards cargan correctamente

#### Base de Datos

* [ ] PostgreSQL instalado y funcionando
* [ ] Base de datos creada
* [ ] Usuario creado con permisos
* [ ] Tablas creadas correctamente
* [ ] Datos de prueba cargados (opcional)

### 10.2 Comandos de Verificación

#### Verificar Backend

# Health check  
curl http://localhost:3000/health  
  
# API endpoints  
curl -X POST http://localhost:3000/api/auth/login \  
 -H "Content-Type: application/json" \  
 -d '{"email":"admin@pepsico.cl","password":"admin123"}'  
  
# Base de datos  
cd backend  
npm run db:studio

#### Verificar Frontend

# Abrir navegador  
open http://localhost:5173  
  
# Verificar en DevTools  
# Console: No errores  
# Network: Requests exitosos  
# Application: LocalStorage con tokens

#### Verificar Base de Datos

# Conectar a PostgreSQL  
psql -U pepsico\_user -d pepsico\_fleet  
  
# Verificar tablas  
\dt  
  
# Verificar datos  
SELECT COUNT(\*) FROM users;  
SELECT COUNT(\*) FROM vehicles;  
SELECT COUNT(\*) FROM work\_orders;  
  
# Salir  
\q

### 10.3 Pruebas de Funcionalidad

#### Prueba de Login

1. Abrir http://localhost:5173
2. Ingresar credenciales de prueba
3. Verificar que redirige al dashboard
4. Verificar que el token se guarda en localStorage

#### Prueba de API

1. Hacer login y obtener token
2. Probar endpoint de usuarios
3. Probar endpoint de vehículos
4. Verificar que las respuestas son correctas

#### Prueba de Base de Datos

1. Crear un nuevo usuario
2. Verificar que se guarda en la base de datos
3. Actualizar el usuario
4. Verificar que los cambios se reflejan

---

## 11. Comandos Útiles

### 11.1 Desarrollo

# Instalar dependencias  
npm install  
  
# Ejecutar en desarrollo  
npm run dev  
  
# Build para producción  
npm run build  
  
# Ejecutar tests  
npm test  
  
# Linter  
npm run lint  
  
# Formatear código  
npm run format

### 11.2 Base de Datos

# Generar cliente Prisma  
npm run db:generate  
  
# Crear migración  
npm run db:migrate  
  
# Aplicar migraciones  
npm run db:migrate:deploy  
  
# Resetear base de datos  
npm run db:reset  
  
# Poblar con datos de prueba  
npm run db:seed  
  
# Abrir Prisma Studio  
npm run db:studio

### 11.3 Git

# Clonar repositorio  
git clone <url>  
  
# Crear branch  
git checkout -b feature/nueva-funcionalidad  
  
# Commit cambios  
git add .  
git commit -m "feat: agregar nueva funcionalidad"  
  
# Push cambios  
git push origin feature/nueva-funcionalidad

---

## 12. Recursos Adicionales

### 12.1 Documentación Oficial

* [Node.js](https://nodejs.org/docs)
* [Express.js](https://expressjs.com)
* [Prisma](https://prisma.io/docs)
* [React](https://react.dev)
* [TypeScript](https://www.typescriptlang.org/docs)
* [PostgreSQL](https://www.postgresql.org/docs)

### 12.2 Herramientas Recomendadas

* \*\*Postman/Insomnia\*\*: Testing de API
* \*\*TablePlus\*\*: GUI para PostgreSQL
* \*\*Prisma Studio\*\*: GUI incluida con Prisma
* \*\*VS Code\*\*: Editor de código
* \*\*Git\*\*: Control de versiones

### 12.3 Comunidad y Soporte

* \*\*GitHub Issues\*\*: Para reportar bugs
* \*\*Discord/Slack\*\*: Para soporte en tiempo real
* \*\*Stack Overflow\*\*: Para preguntas técnicas
* \*\*Documentación\*\*: Para consultas específicas

---

**Última actualización:** Octubre 15, 2024

**Versión:** 1.0.0

**Mantenido por:** Joaquín Marín & Benjamin Vilches