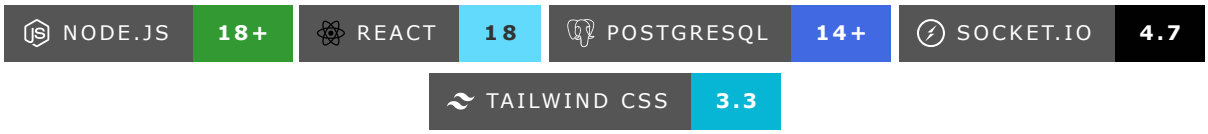
Emilia Café - Sistema de Gestión de Restaurante



Plataforma web responsiva para la gestión operativa y comercial de restaurantes

Sincronización en tiempo real entre inventario, menú y órdenes

Tabla de Contenidos

Características

Sincronización en Tiempo Real

- Disponibilidad de menú calculada dinámicamente según inventario
- Notificaciones instantáneas de nuevas órdenes vía WebSocket
- Alertas de stock bajo/agotado en tiempo real
- Actualización automática de estados de órdenes
- Indicador de conexión en la interfaz

Gestión de Inventario

- Control de ingredientes con stock mínimo configurable
- Registro de movimientos (entradas, salidas, ajustes, mermas)
- Alertas automáticas de reabastecimiento
- Historial completo de movimientos con usuario y fecha
- Ubicación física de ingredientes en almacén

🍽️ Gestión de Menú

- Categorías con iconos personalizados (Bebidas, Panadería, etc.)
- Platillos con múltiples ingredientes y porciones requeridas
- Disponibilidad automática basada en stock de ingredientes
- Indicador visual de porciones restantes
- Precios con formato MXN

📝 Sistema de Órdenes

- Creación rápida con carrito interactivo
- Selección de mesas disponibles (opcional)
- Flujo de estados: `pendiente` → `preparando` → `listo` → `entregado`
- Cancelación con restauración automática de inventario
- Notas por ítem y por orden
- Cálculo automático de subtotal, IVA (16%) y total

📊 Dashboard Analítico (Solo Gerente)

- KPIs en tiempo real: ventas del día, ticket promedio, mesas activas, alertas
- Gráfico de ventas por hora (barras)
- Tendencia de ventas últimos 14 días (líneas)
- Distribución de ventas por categoría (pie chart)
- Top 5 productos más vendidos
- Desglose de órdenes por estado

👥 Gestión de Usuarios (Solo Gerente)

- CRUD completo de usuarios
- Asignación de roles (Gerente/Empleado)
- Activar/desactivar usuarios
- Reset de contraseña
- Historial de último acceso

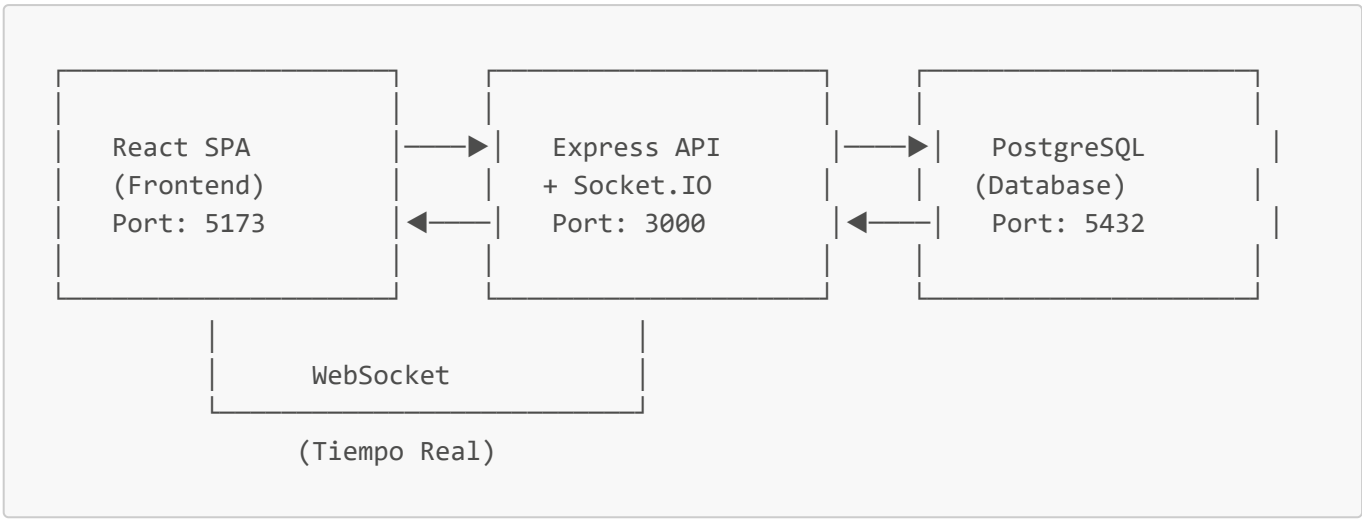
⚙️ Configuración Personal

- Edición de perfil (nombre, email, teléfono)
- Cambio de contraseña seguro
- Preferencias de notificaciones
- Selección de tema e idioma

🔐 Seguridad

- Autenticación JWT con access y refresh tokens
- Control de acceso basado en roles (RBAC)
- Contraseñas hasheadas con bcrypt
- Rate limiting para prevenir ataques
- Auditoría de acciones importantes

Arquitectura



Flujo de Disponibilidad del Menú

Tecnologías

Backend

Tecnología	Versión	Descripción

Tecnología	Versión	Descripción
Node.js	18+	Runtime de JavaScript
Express	4.18	Framework web HTTP
Socket.IO	4.7	WebSockets bidireccionales
PostgreSQL	14+	Base de datos relacional
JWT	-	Tokens de autenticación
bcryptjs	-	Hash seguro de contraseñas
Winston	-	Sistema de logging
express-validator	-	Validación de datos
helmet	-	Seguridad HTTP headers
cors	-	Cross-Origin Resource Sharing

Frontend

Tecnología	Versión	Descripción
React	18	Biblioteca de UI
Vite	5	Build tool y dev server
Tailwind CSS	3.3	Framework de estilos utility-first
React Router	6	Enrutamiento SPA
Recharts	2.10	Gráficos y visualizaciones
Lucide React	-	Iconos SVG
Axios	-	Cliente HTTP
Socket.IO Client	4.7	Cliente WebSocket
react-hot-toast	-	Notificaciones toast
clsx	-	Clases CSS condicionales

📋 Requisitos Previos

- **Node.js** >= 18.0.0
- **npm** >= 9.0.0
- **PostgreSQL** >= 14.0

Verificar instalación

```
node --version    # v18.x.x o superior
npm --version     # 9.x.x o superior
psql --version    # psql (PostgreSQL) 14.x o superior
```

🚀 Instalación Rápida

El proyecto incluye scripts `.bat` para automatizar todo en Windows:

Paso 1: Instalar dependencias

```
.\install.bat
```

Paso 2: Configurar proyecto

Edita `backend\.env` con tu contraseña de PostgreSQL, luego:

```
.\setup.bat
```

Paso 3: Iniciar desarrollo

```
.\dev.bat
```

Scripts disponibles

Script	Descripción
`install.bat`	Instala dependencias de backend y frontend
`setup.bat`	Crea archivos .env, ejecuta migraciones y seed
`dev.bat`	Inicia backend y frontend en ventanas separadas
`db-reset.bat`	Resetea la base de datos (con confirmación)
`start-backend.bat`	Solo inicia el backend
`start-frontend.bat`	Solo inicia el frontend

📖 Instalación Manual

1. Clonar el repositorio

```
```

```
git clone <url-del-repositorio>
cd restaurante
```

2. Instalar dependencias

```
# Backend
cd backend
npm install

# Frontend
cd ../frontend
npm install
```

3. Configurar variables de entorno

```
# Copiar archivos de ejemplo (o crearlos manualmente)
cp backend/.env.example backend/.env
cp frontend/.env.example frontend/.env
```

4. Crear base de datos

```
# Conectar a PostgreSQL
psql -U postgres

# En el prompt de psql:
CREATE DATABASE emilia_cafe;
\q
```

5. Ejecutar migraciones y seed

```
cd backend
npm run db:migrate    # Crear tablas
npm run db:seed       # Insertar datos de prueba
```

⚙ Configuración

Backend (`backend/.env`)

```
# Servidor
PORT=3000
```

```
NODE_ENV=development

# Base de datos PostgreSQL
DB_HOST=localhost
DB_PORT=5432
DB_NAME=emilia_cafe
DB_USER=postgres
DB_PASSWORD=tu_password_aqui

# JWT (cambiar en producción)
JWT_SECRET=emilia_cafe_super_secret_key_2024_muy_segura
JWT_EXPIRES_IN=15m
JWT_REFRESH_SECRET=emilia_cafe_refresh_secret_key_2024
JWT_REFRESH_EXPIRES_IN=7d

# CORS
CORS_ORIGIN=http://localhost:5173
```

Frontend (`frontend/.env`)

```
VITE_API_URL=http://localhost:3000/api
VITE_SOCKET_URL=http://localhost:3000
```

▶ Ejecución

Desarrollo

Terminal 1 - Backend:

```
cd backend
npm run dev
```

Terminal 2 - Frontend:

```
cd frontend
npm run dev
```

URLs de Acceso

Servicio	URL
💻 Frontend	http://localhost:5173
🔌 API Backend	http://localhost:3000/api

Servicio	URL
Health Check	http://localhost:3000/api/health

Scripts de Base de Datos

```
cd backend

npm run db:migrate    # Ejecutar migraciones (crear tablas)
npm run db:seed       # Insertar datos de prueba
npm run db:reset      # Eliminar y recrear todo
```

Estructura del Proyecto

```
restaurante/
│
├── backend/
│   ├── src/
│   │   ├── config/
│   │   │   ├── database.js         # Pool de conexiones PostgreSQL
│   │   │   └── index.js            # Variables de configuración
│   │   │
│   │   ├── controllers/
│   │   │   ├── authController.js   # Login, logout, refresh, perfil
│   │   │   ├── userController.js   # CRUD de usuarios
│   │   │   ├── menuController.js   # Menú, categorías, disponibilidad
│   │   │   ├── inventoryController.js  # Ingredientes, stock, movimientos
│   │   │   ├── orderController.js  # Órdenes, estados, mesas
│   │   │   └── dashboardController.js  # KPIs, gráficos, reportes
│   │   │
│   │   ├── middleware/
│   │   │   ├── auth.js             # Verificación de JWT
│   │   │   ├── authorize.js        # Control de roles RBAC
│   │   │   ├── validation.js       # Validación de request body
│   │   │   └── errorHandler.js     # Manejo global de errores
│   │   │
│   │   ├── routes/
│   │   │   ├── index.js            # Router principal
│   │   │   ├── auth.js             # /api/auth/*
│   │   │   ├── users.js            # /api/users/*
│   │   │   ├── menu.js             # /api/menu/*
│   │   │   ├── inventory.js        # /api/inventory/*
│   │   │   ├── orders.js           # /api/orders/*
│   │   │   └── dashboard.js        # /api/dashboard/*
│   │   │
│   │   ├── sockets/
│   │   │   └── index.js            # Handlers de WebSocket
│   │   │
│   │   ├── database/
```

```
│   │   │   ├── migrate.js          # Esquema de tablas SQL
│   │   │   └── seed.js             # Datos iniciales de prueba
│   │   │
│   │   ├── 🗀 utils/
│   │   │   ├── logger.js           # Winston logger
│   │   │   ├── helpers.js          # Funciones utilitarias
│   │   │   └── permissions.js      # Definición de permisos RBAC
│   │   │
│   │   └── index.js                # Entry point del servidor
│   │
│   ├── package.json
│   └── .env
│
├── 🗀 frontend/
│   ├── 🗀 src/
│   │   ├── 🗀 components/
│   │   │   ├── 🗀 layouts/
│   │   │   │   ├── MainLayout.jsx  # Layout con sidebar y header
│   │   │   │   └── AuthLayout.jsx  # Layout de login
│   │   │   │
│   │   │   └── 🗀 ui/              # Componentes reutilizables
│   │   │       ├── Alert.jsx
│   │   │       ├── Badge.jsx
│   │   │       ├── Button.jsx
│   │   │       ├── Card.jsx
│   │   │       ├── Input.jsx
│   │   │       ├── Modal.jsx
│   │   │       ├── Spinner.jsx
│   │   │       └── index.js
│   │   │
│   │   ├── 🗀 contexts/
│   │   │   ├── AuthContext.jsx     # Estado de autenticación
│   │   │   └── SocketContext.jsx   # Conexión WebSocket
│   │   │
│   │   ├── 🗀 pages/
│   │   │   ├── Login.jsx           # Página de inicio de sesión
│   │   │   ├── Dashboard.jsx       # Dashboard con KPIs (gerente)
│   │   │   ├── Menu.jsx            # Visualización del menú
│   │   │   ├── Orders.jsx          # Lista de órdenes
│   │   │   ├── NewOrder.jsx        # Crear nueva orden
│   │   │   ├── Inventory.jsx       # Gestión de inventario (gerente)
│   │   │   ├── Users.jsx           # Gestión de usuarios (gerente)
│   │   │   ├── Settings.jsx        # Configuración personal
│   │   │   └── NotFound.jsx        # Página 404
│   │   │
│   │   ├── 🗀 services/
│   │   │   └── api.js              # Cliente Axios configurado
│   │   │
│   │   ├── App.jsx                 # Rutas de la aplicación
│   │   ├── main.jsx                # Entry point React
│   │   └── index.css               # Estilos Tailwind
│   │
│   ├── package.json
│   ├── vite.config.js
```

```
│   ├── tailwind.config.js
│   └── .env
│
├── install.bat                 # Script de instalación
├── setup.bat                   # Script de configuración
├── dev.bat                     # Script de desarrollo
├── db-reset.bat                # Script de reset BD
├── start-backend.bat           # Iniciar solo backend
├── start-frontend.bat          # Iniciar solo frontend
├── README.md                   # Esta documentación
└── .gitignore
```

API REST

Autenticación

Método	Endpoint	Descripción	Auth
POST	`/api/auth/login`	Iniciar sesión	❌
POST	`/api/auth/logout`	Cerrar sesión	☑
POST	`/api/auth/refresh`	Renovar access token	❌
GET	`/api/auth/profile`	Obtener perfil actual	☑
POST	`/api/auth/change-password`	Cambiar contraseña	☑

Usuarios (Solo Gerente)

Método	Endpoint	Descripción
GET	`/api/users`	Listar usuarios (paginado)
GET	`/api/users/:id`	Obtener usuario por ID
POST	`/api/users`	Crear nuevo usuario
PUT	`/api/users/:id`	Actualizar usuario
DELETE	`/api/users/:id`	Eliminar usuario
PUT	`/api/users/:id/password`	Resetear contraseña

Menú

Método	Endpoint	Descripción	Auth
GET	`/api/menu`	Listar menú con disponibilidad	☑
GET	`/api/menu/categories`	Listar categorías	☑
GET	`/api/menu/:id`	Obtener platillo	☑

Método	Endpoint	Descripción	Auth
POST	`/api/menu`	Crear platillo	Gerente
PUT	`/api/menu/:id`	Actualizar platillo	Gerente
DELETE	`/api/menu/:id`	Eliminar platillo	Gerente

Inventario (Solo Gerente)

Método	Endpoint	Descripción
GET	`/api/inventory`	Listar ingredientes
GET	`/api/inventory/alerts`	Obtener alertas de stock
GET	`/api/inventory/movements`	Historial de movimientos
GET	`/api/inventory/:id`	Obtener ingrediente
POST	`/api/inventory`	Crear ingrediente
PUT	`/api/inventory/:id`	Actualizar ingrediente
POST	`/api/inventory/:id/adjust`	Ajustar stock

Órdenes

Método	Endpoint	Descripción	Auth
GET	`/api/orders`	Listar órdenes	☑
GET	`/api/orders/tables`	Listar mesas	☑
GET	`/api/orders/:id`	Obtener orden	☑
POST	`/api/orders`	Crear orden	☑
PUT	`/api/orders/:id/status`	Cambiar estado	☑
DELETE	`/api/orders/:id`	Cancelar orden	☑

Dashboard (Solo Gerente)

Método	Endpoint	Descripción
GET	`/api/dashboard/kpis`	KPIs del día
GET	`/api/dashboard/sales-by-hour`	Ventas por hora
GET	`/api/dashboard/sales-by-day`	Ventas últimos 14 días
GET	`/api/dashboard/top-products`	Productos más vendidos
GET	`/api/dashboard/sales-by-category`	Ventas por categoría

WebSockets

Conexión

```
import { io } from 'socket.io-client';

const socket = io('http://localhost:3000', {
  auth: { token: 'jwt_access_token' }
});
```

Eventos del Servidor → Cliente

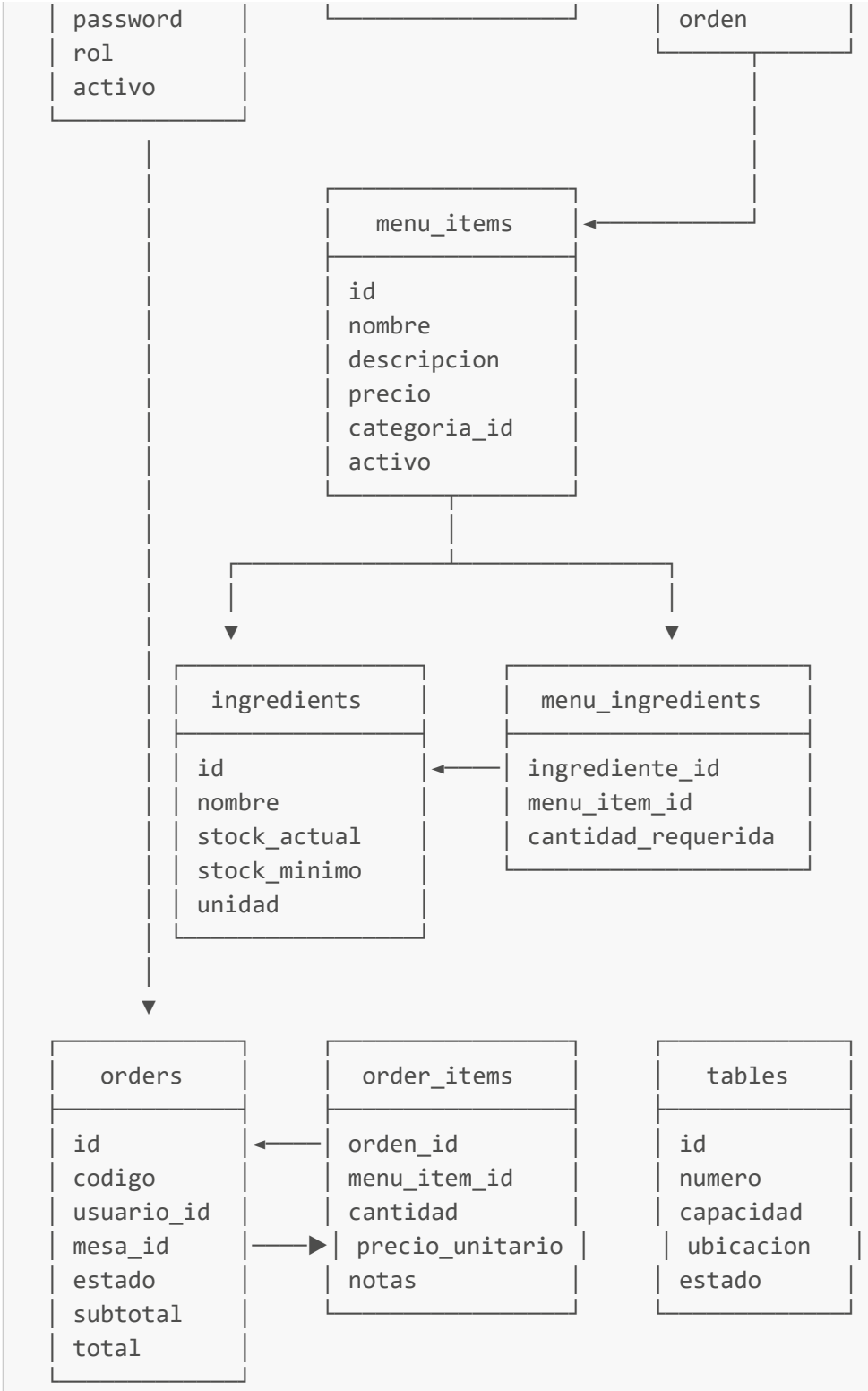
Evento	Payload	Descripción
`inventory:update`	`{ ingredientId, nombre, stock_actual, alerta }`	Stock actualizado
`menu:availability`	`{ items: [{ id, disponible, max_porciones }] }`	Disponibilidad recalculada
`order:new`	`{ order }`	Nueva orden creada
`order:status`	`{ orderId, status, updatedBy }`	Estado de orden cambió
`alert:new`	`{ type, message, severity }`	Nueva alerta del sistema

Salas (Rooms)

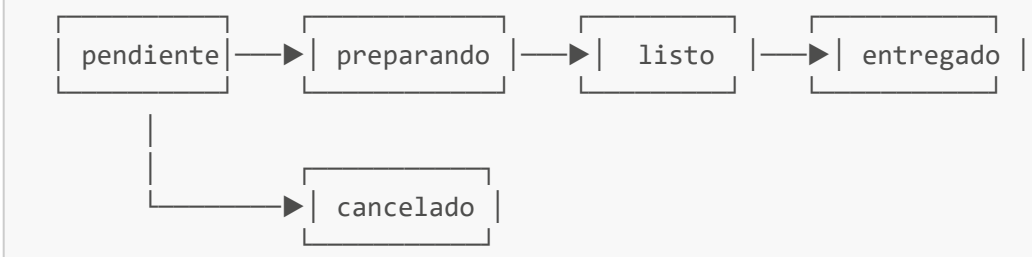
Sala	Descripción
`role:gerente`	Todos los gerentes conectados
`role:empleado`	Todos los empleados conectados
`user:{id}`	Usuario específico
`table:{id}`	Observadores de una mesa
`order:{id}`	Observadores de una orden

Base de Datos

Diagrama de Tablas



Estados de Órdenes



Usuarios de Prueba

Después de ejecutar `npm run db:seed`:

Rol	Nombre	Email	Contraseña
Gerente	Gerente Admin	gerente@emiliacafe.com	password123
Empleado	María García	maria@emiliacafe.com	password123

Permisos por Rol

Gerente

Módulo	Permisos
Dashboard	☑ Ver KPIs y gráficos
Menú	☑ Ver, Crear, Editar, Eliminar
Inventario	☑ Ver, Ajustar stock, Ver movimientos
Órdenes	☑ Ver todas, Crear, Cambiar estado, Cancelar
Usuarios	☑ Ver, Crear, Editar, Eliminar, Reset password
Configuración	☑ Sistema y personal

Empleado

Módulo	Permisos
Dashboard	✗ Sin acceso
Menú	☑ Solo ver
Inventario	✗ Sin acceso
Órdenes	☑ Ver propias, Crear, Cambiar estado
Usuarios	✗ Sin acceso
Configuración	☑ Solo personal

Solución de Problemas

Error: ECONNREFUSED PostgreSQL

```
Error: connect ECONNREFUSED 127.0.0.1:5432
```

Solución:

1. Verificar que PostgreSQL esté corriendo
    - Windows: `services.msc` → buscar "postgresql" → Iniciar
2. Verificar credenciales en `backend/.env`

Error: CORS

```
Access to XMLHttpRequest blocked by CORS policy
```

Solución: Verificar que `CORS_ORIGIN` en `backend/.env` coincida con la URL del frontend:

```
CORS_ORIGIN=http://localhost:5173
```

Error: Puerto en uso

```
Error: listen EADDRINUSE :::3000
```

Solución:

```
# Windows - Cerrar procesos Node
taskkill /f /im node.exe

# O cambiar puerto en backend/.env
PORT=3001
```

Error: Base de datos no existe

```
error: database "emilia_cafe" does not exist
```

Solución:

```
psql -U postgres -c "CREATE DATABASE emilia_cafe;"
```

Pantalla en blanco en alguna página

Solución:

1. Abrir DevTools (F12) → Console
2. Buscar el error específico

3. Generalmente es un problema de estructura de datos de la API

Datos de Prueba Incluidos

El seed incluye:

- **2 usuarios** (gerente y empleado)
- **6 categorías** (Bebidas Calientes, Bebidas Frías, Panadería, Desayunos, Snacks, Postres)
- **25 ingredientes** con stock inicial
- **23 platillos** del menú con sus ingredientes
- **10 mesas** disponibles
- **Configuraciones** del sistema

Actualizaciones Futuras

- [ ] Modo offline con sincronización
- [ ] Exportación de reportes (PDF, Excel)
- [ ] Notificaciones push
- [ ] Tema oscuro
- [ ] Multi-idioma completo
- [ ] Integración con impresoras de tickets
- [ ] Sistema de reservaciones

Licencia

Este proyecto es privado y de uso exclusivo para **Emilia Café**.

Desarrollo

Desarrollado con ♥ para Emilia Café

Stack: Node.js + Express + PostgreSQL + Socket.IO + React + Vite + Tailwind CSS

Última actualización: Diciembre 2025