KPIs Rosti

Manual del Sistema de Indicadores

Área: Tecnología de la Información

Versión: 1.0

Fecha: Febrero 2026

Clasificación: Uso interno

📋 Tabla de Contenidos

01 Visión General

KPIs Rosti es una plataforma web interna para análisis de indicadores de desempeño de los restaurantes de la cadena Rosti Pollos Costa Rica. Consolida datos de ventas, presupuesto, personal, tiempos de cocina e incidencias en un solo lugar, accesible desde red corporativa o VPN.

Módulos disponibles

Módulo	Descripción
📊 Alcance Presupuesto	Análisis mensual, anual y tendencias del alcance vs presupuesto por restaurante y grupo
🕐 Tiempos de Cocina	Métricas de velocidad de servicio por restaurante y período
👥 Personal	Indicadores de planilla, asistencia y dotación
🎫 Incidencias (InvGate)	Tickets de soporte integrados desde la nube InvGate
📋 Formularios	Datos de Microsoft Forms sincronizados automáticamente vía Graph API
⚙️ Administración	Gestión de usuarios, perfiles, conexiones de BD e integraciones (solo Admin)

02 Arquitectura del Sistema

Componentes principales

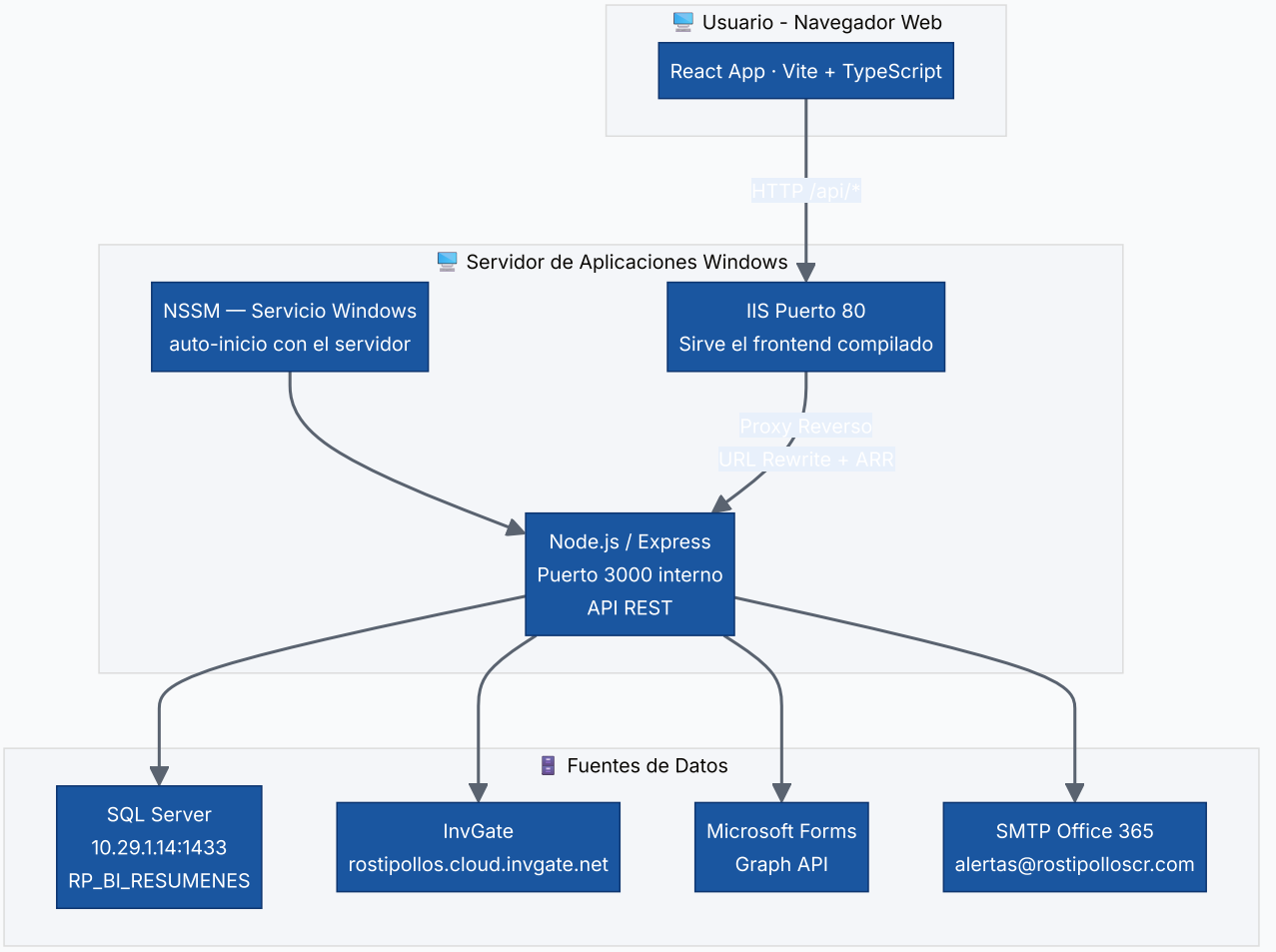


Diagrama 1 — Componentes del sistema

Flujo de una petición HTTP

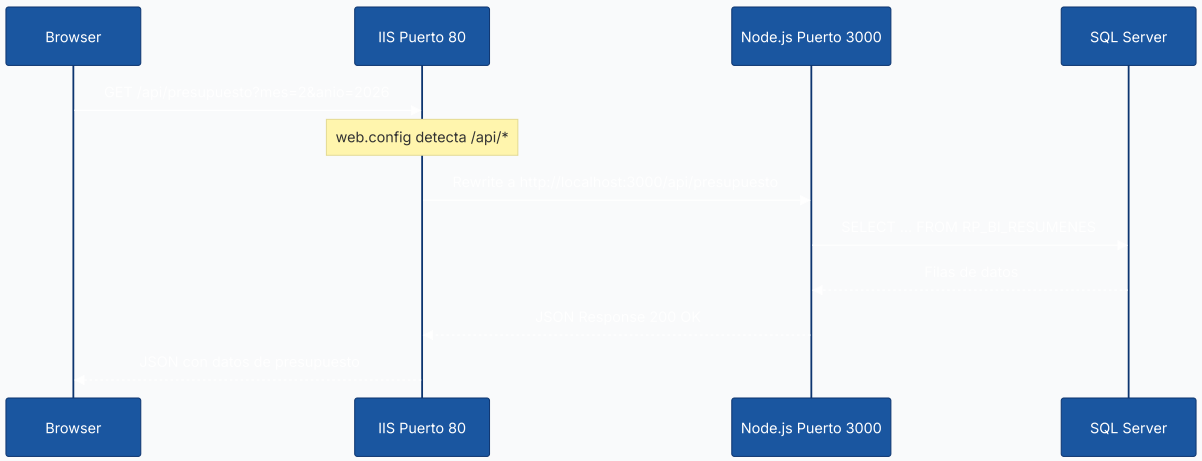


Diagrama 2 — Ciclo de vida de una petición

Proxy Reverso

El puerto 3000 de Node.js **nunca se expone** al exterior. Todo el tráfico pasa por IIS vía URL Rewrite + ARR.

03 Módulos del Frontend

Estructura de carpetas

```
web-app/src/
├── components/
│   ├── App.tsx ← Raíz: filtros globales, navegación
│   ├── AnnualCalendar.tsx ← Vista anual (12 meses)
│   ├── TendenciaAlcance.tsx ← Tendencias y grupos
│   ├── RangosView.tsx ← Análisis por rangos
│   ├── PersonalModule.tsx ← Módulo de personal
│   ├── TiemposCocina.tsx ← Tiempos de cocina
│   ├── InvGateModule.tsx ← Incidencias InvGate
│   ├── FormsModule.tsx ← Formularios MS Forms
│   ├── AdminPage.tsx ← Panel de administración
│   └── LoginPage.tsx ← Autenticación con JWT
├── context/
│   ├── UserPreferences.tsx ← Preferencias del usuario
│   └── AuthContext.tsx ← Estado de sesión global
└── hooks/
    └── useFormatCurrency.ts ← Formato ₡ full / miles / millones
```

Navegación entre módulos

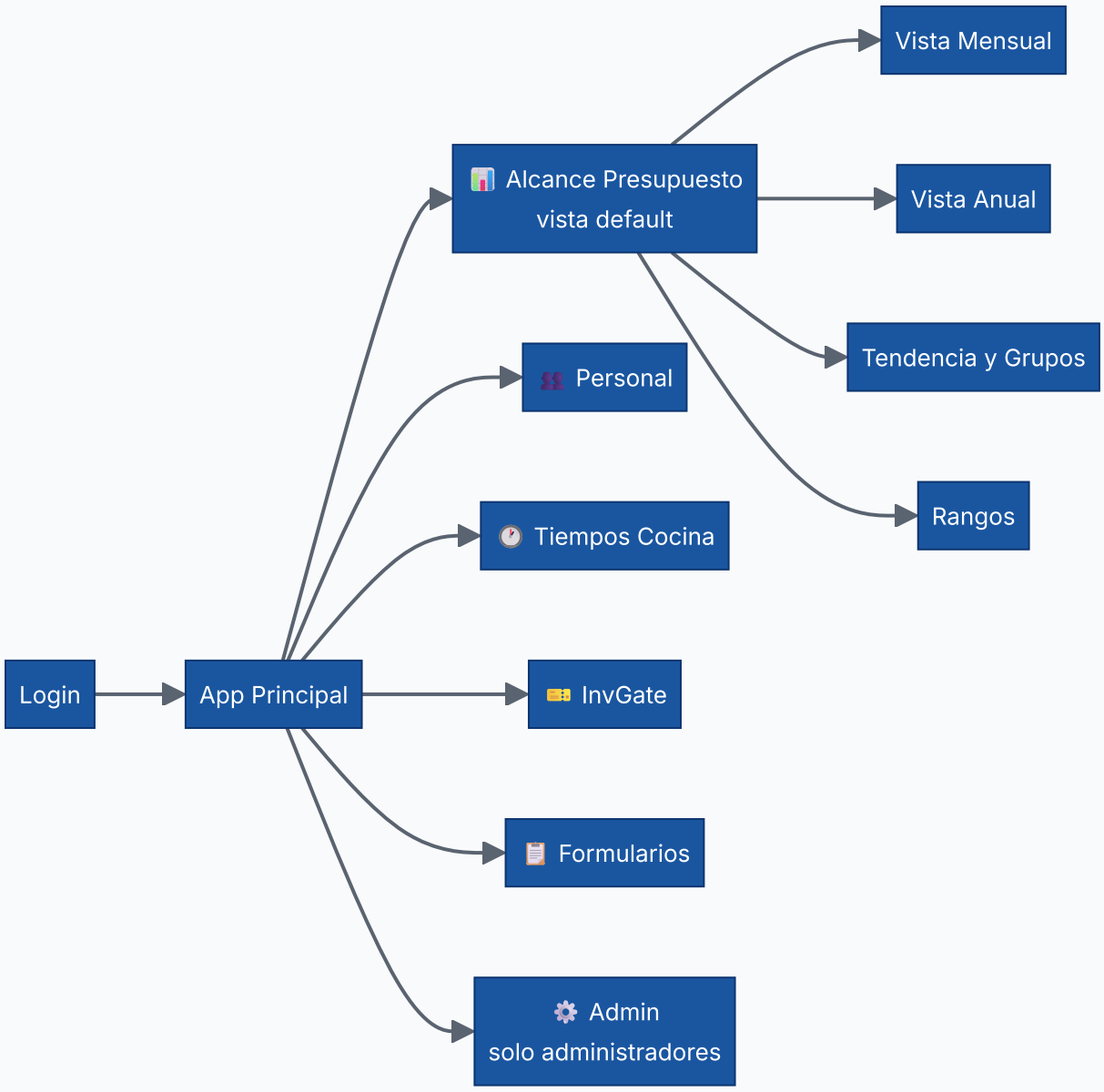


Diagrama 3 — Navegación de módulos

04 Backend — API y Servicios

Archivos principales

Archivo	Función
`server.js`	Punto de entrada Express, todas las rutas REST
`auth.js`	Autenticación JWT, usuarios, perfiles y permisos
`tendencia.js`	Endpoints de tendencia y alcance presupuestal
`rangos.js`	Endpoints para análisis por rangos de ventas
`personal.js`	Indicadores de personal y planilla
`forms_endpoints.js`	Sincronización con Microsoft Forms vía Graph API
`emailService.js`	Envío de correos SMTP Office 365
`dbConnectionManager.js`	Pool de conexiones SQL Server (mssql)
`services/invgateService.js`	Integración InvGate (tickets de soporte)

Variables de entorno (.env)

server/.env — NO versionar en Git

```
# Base de datos principal
DB_USER=sa
DB_PASSWORD=****
DB_SERVER=10.29.1.14
DB_DATABASE=RP_BI_RESUMENES

# Seguridad JWT
JWT_SECRET=clave-aleatoria-muy-larga

# Correo electrónico SMTP
SMTP_HOST=smtp.office365.com
SMTP_PORT=587
SMTP_USER=alertas@rostipolloscr.com
SMTP_PASS=****
```

Seguridad crítica

El archivo `.env` contiene credenciales de producción. Nunca se debe versionar en Git ni compartir por correo.

05 Base de Datos

Parámetro	Valor
Servidor	`10.29.1.14:1433`
Base de datos	`RP_BI_RESUMENES`
Motor	Microsoft SQL Server
Acceso requerido	Red corporativa o VPN

> ⚠️ **Requiere red corporativa**
>
> Sin VPN el backend no puede conectar al SQL Server y no mostrará datos.

Tablas de la aplicación

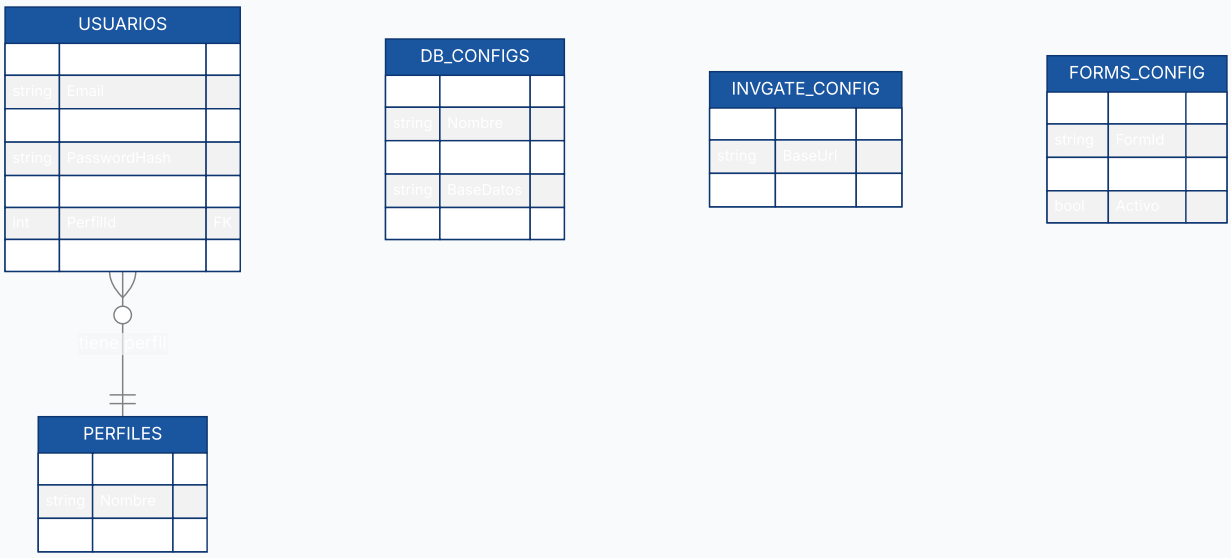


Diagrama 4 — Tablas propias de la aplicación

06 Autenticación y Usuarios

Flujo de login

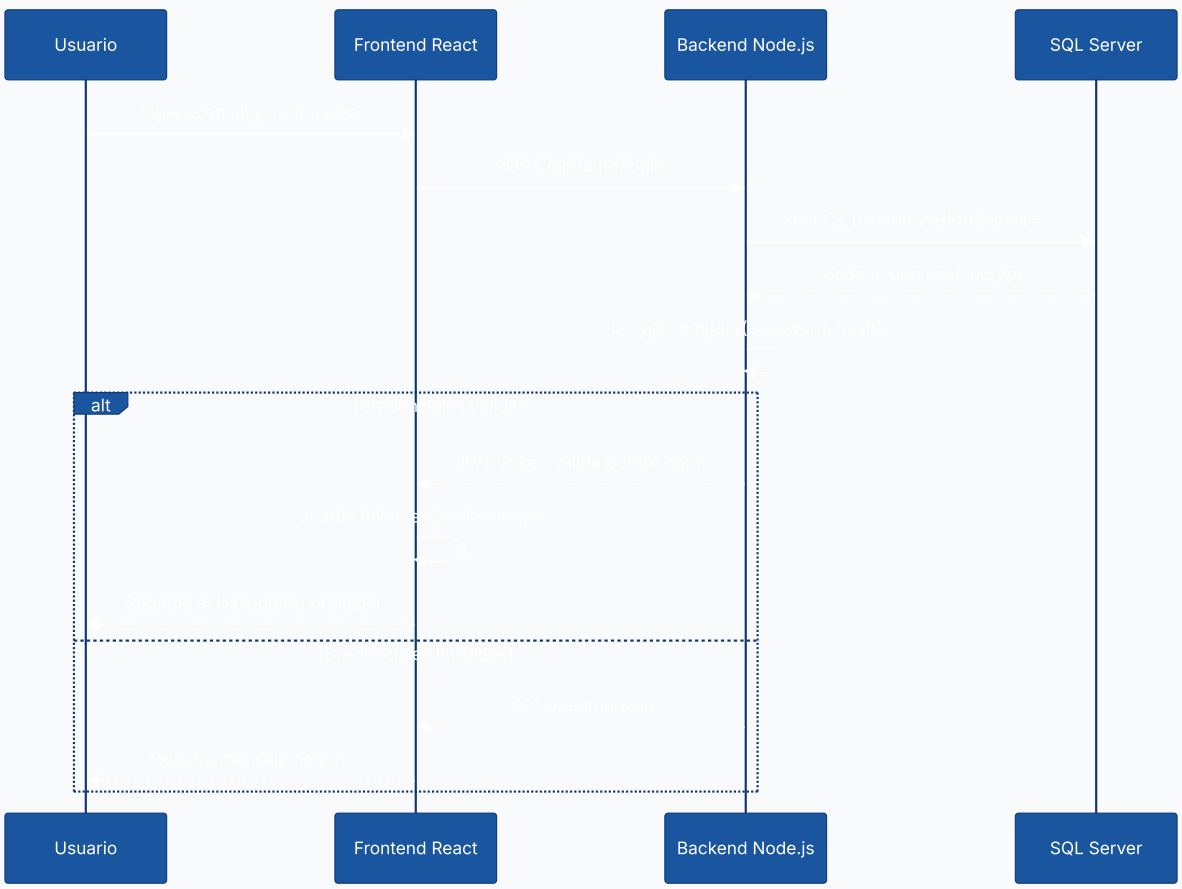


Diagrama 5 — Flujo de autenticación JWT

Niveles de acceso

Perfil	Módulos disponibles
⭐ **Administrador**	Todo + Panel Admin (usuarios, BD, InvGate, Forms)
📊 **Gerencia**	Presupuesto, Personal, Tiempos, InvGate, Forms
👁 **Operativo**	Presupuesto (solo lectura)
🔧 **Personalizado**	Módulos configurados individualmente por el Admin

07 Deployment (Publicación)

Infraestructura de producción

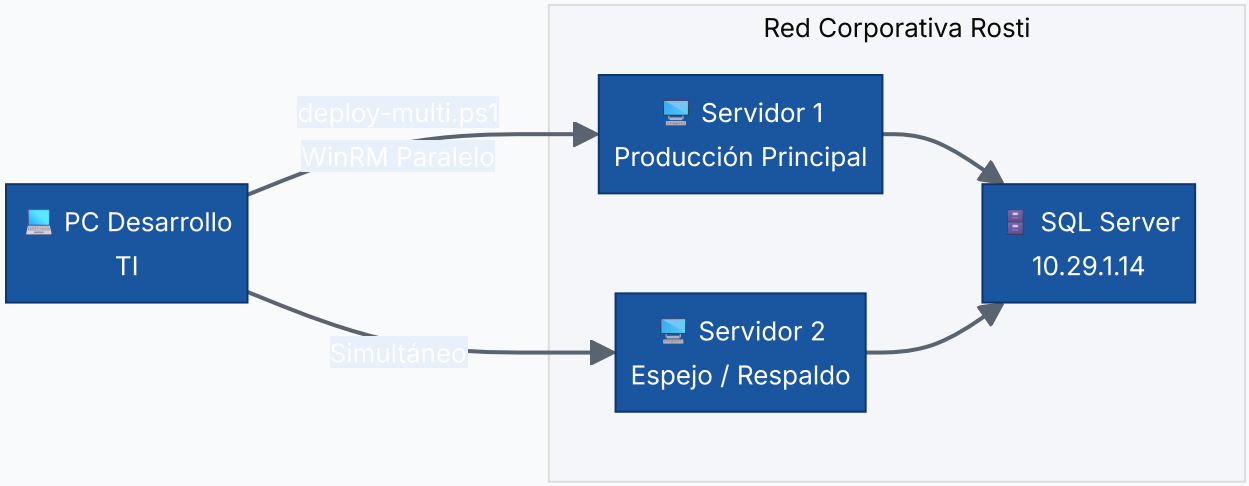
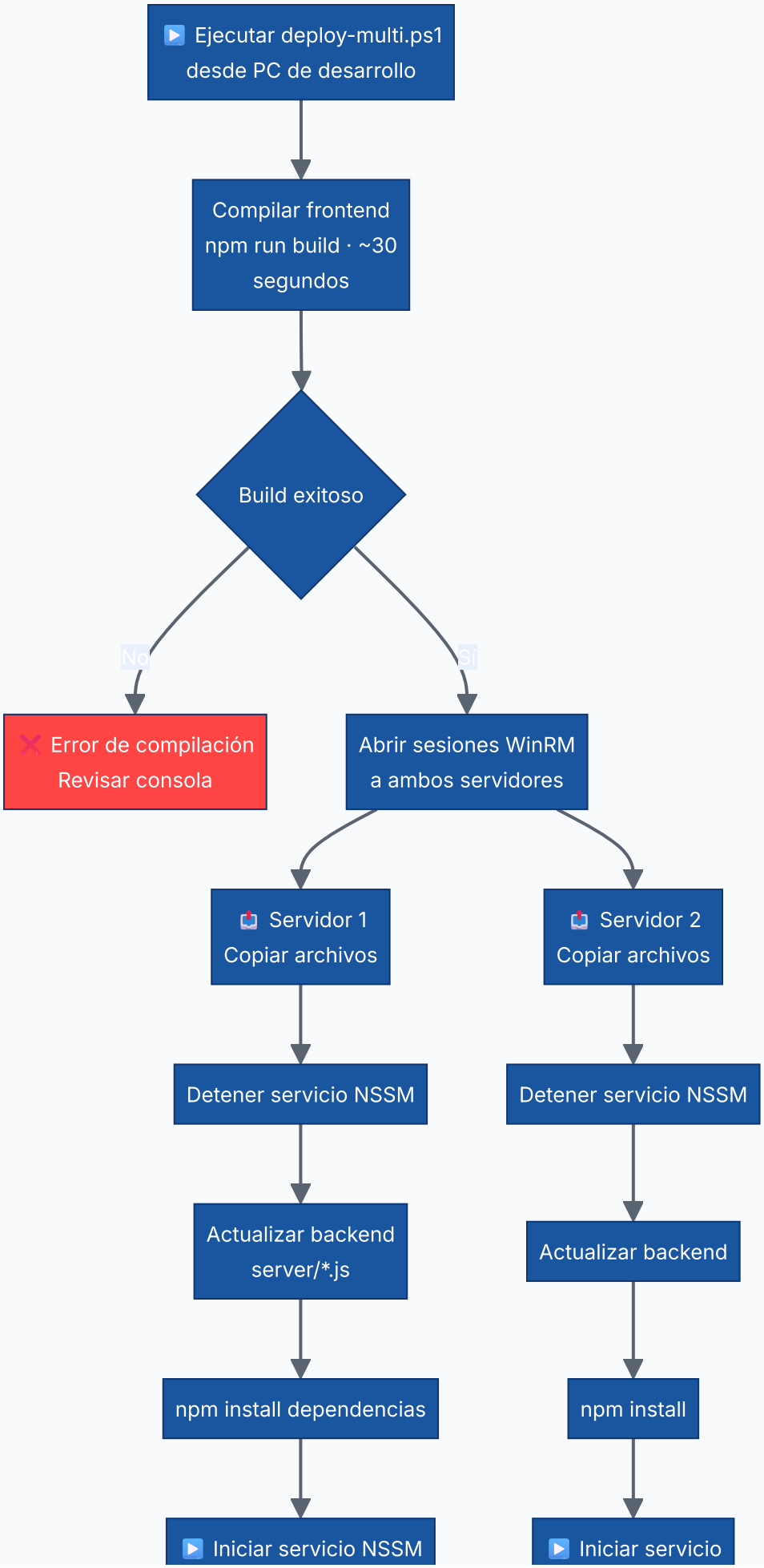


Diagrama 6 — Infraestructura de deployment

Flujo completo de publicación


Ejecutar deploy-multi.ps1 desde PC de desarrollo
Compilar frontend npm run build · ~30 segundos
Build exitoso
No
Sí
Error de compilación Revisar consola
Abrir sesiones WinRM a ambos servidores
Servidor 1 Copiar archivos
Servidor 2 Copiar archivos
Detener servicio NSSM
Detener servicio NSSM
Actualizar backend server/*.js
Actualizar backend
npm install dependencias
npm install
Iniciar servicio NSSM
Iniciar servicio

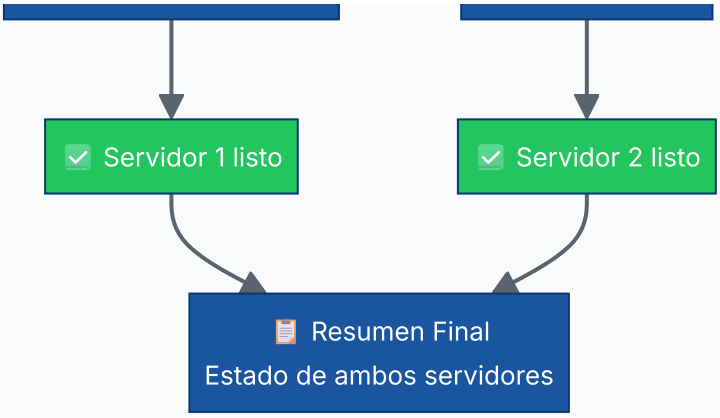


Diagrama 7 — Flujo de publicación paralela

Comandos de publicación

PowerShell — desde la PC de desarrollo

```
# Ir a la carpeta de deploy
cd C:\AntiGravityDev\CalendarioPresupuesto\deploy

# Actualizar TODO en ambos servidores (uso más frecuente)
.\deploy-multi.ps1 -Update

# Solo cambios de pantallas (no reinicia el backend)
.\deploy-multi.ps1 -Update -SkipBackend

# Solo cambios de lógica o API
.\deploy-multi.ps1 -Update -SkipFrontend

# Primera instalación completa en servidores nuevos
.\deploy-multi.ps1
```

Configurar servidores destino (una sola vez)

deploy-multi.ps1 — parámetro $Servers

```
$Servers = @(
    @{
        Name       = "Servidor-1"
        Host       = "10.29.1.XX"   # IP del servidor 1
        User       = "Administrador"
        Password   = ""             # Vacío = pide seguro al ejecutar
        InstallDir = "C:\Apps\CalendarioPresupuesto"
    },
    @{
        Name       = "Servidor-2"
        Host       = "10.29.1.YY"   # IP del servidor 2
        User       = "Administrador"
        Password   = ""
        InstallDir = "C:\Apps\CalendarioPresupuesto"
    }
)
```

Pre-requisito WinRM (solo una vez por servidor)

PowerShell como Administrador — en el servidor destino

```
winrm quickconfig -q
Set-Item WSMan:\localhost\Client\TrustedHosts -Value "*" -Force
Test-WSMan
```

08 Administración y Monitoreo

Estructura de archivos en producción

```
C:\Apps\CalendarioPresupuesto\
├── web-app\ ← Frontend (IIS sirve estos archivos)
│   ├── index.html
│   ├── assets\
│   └── web.config ← Proxy reverso /api/* → Node
├── server\ ← Backend Node.js
│   ├── server.js
│   ├── .env ← ⚠️ Credenciales de producción
│   └── logs\
│   ├── stdout.log ← Log de operación normal
│   └── stderr.log ← Log de errores
└── tools\
    └── nssm.exe ← Gestor del servicio de Windows
```

Comandos de administración en el servidor

PowerShell — en el servidor de producción

```powershell
# Ver estado del servicio
Get-Service CalendarioPresupuesto-API

# Reiniciar el backend (después de cambios en .env)
nssm restart CalendarioPresupuesto-API

# Ver logs en tiempo real
Get-Content C:\Apps\CalendarioPresupuesto\server\logs\stdout.log -Tail 50 -Wait

# Ver errores recientes
Get-Content C:\Apps\CalendarioPresupuesto\server\logs\stderr.log -Tail 30

# Verificar que la API responde
Invoke-WebRequest http://localhost:3000/api/columns -UseBasicParsing
```

09 Troubleshooting

Síntoma	Causa probable	Solución
App no carga / pantalla en blanco	IIS detenido	`Start-Website -Name CalendarioPresupuesto`
Error "No se pudo conectar al servidor"	Backend detenido	`nssm restart CalendarioPresupuesto-API`
Datos desactualizados	Caché del navegador	Ctrl + Shift + R
Login no funciona	JWT_SECRET cambió o sesión expiró	Verificar `.env` y reiniciar servicio
Error de base de datos	Sin VPN o BD caída	Verificar VPN activa hacia `10.29.1.14`
Deploy falla "WinRM refused"	WinRM no habilitado en servidor	`winrm quickconfig -q` en el servidor
Deploy falla "Access Denied"	Credenciales incorrectas	Verificar User/Password en `deploy-multi.ps1`
Frontend viejo después de deploy	dist/ no actualizado	Borrar `web-app/dist/` y repetir
InvGate no conecta	Credenciales o token expirado	Admin → Configuración → InvGate → Probar
Forms no sincroniza	Token de Graph API expirado	Admin → Forms → Forzar Sync