Arquitectura y Tecnologías del Backend - SpeadLink

1. Visión General

Este documento describe las tecnologías, arquitectura y estructura del backend del sistema SpeadLink CRM, Facturación y Automatización.

2. Arquitectura General

Se utilizará una arquitectura MONOLITO MODULAR preparada para escalar a microservicios.

Componentes:
- API Backend
- Worker de Jobs
- PostgreSQL (datos críticos)
- MongoDB (logs e historial)
- Redis (colas y crons)

3. Tecnologías Principales

- Node.js + TypeScript
- NestJS
- PostgreSQL + Prisma ORM
- MongoDB
- Redis
- BullMQ
- Docker
- JWT Auth
- Nodemailer
- Puppeteer
- Winston

4. Uso de Bases de Datos

PostgreSQL:
- Usuarios
- Clientes
- Facturación
- Pagos

MongoDB:
- Logs por usuario
- Historial de acciones
- Correos enviados

- Notificaciones
- Errores del sistema

5. Jobs y Automatizaciones

Se usarán jobs con BullMQ para:
- Facturación mensual
- Envío de correos
- Recordatorios
- Limpieza de logs

Los schedules se administran desde el frontend mediante API.

6. Seguridad

- JWT + Refresh Tokens
- Roles y permisos
- Rate limit
- Logs de auditoría

7. Estructura del Proyecto

```
src/
├── modules/
│   ├── auth/
│   ├── users/
│   ├── clients/
│   ├── invoices/
│   ├── payments/
│   ├── jobs/
│   ├── logs/
│   └── notifications/
├── database/
│   ├── postgres/
│   └── mongo/
├── jobs/
├── common/
└── main.ts
```

8. Deploy

El sistema se desplegará usando Docker y docker-compose.
Servicios:
- API
- Worker
- PostgreSQL
- MongoDB
- Redis

9. Escalabilidad

El sistema está diseñado para escalar a microservicios en el futuro sin refactor grande.

10. Sistema de Notificaciones en Tiempo Real (WebSockets)

El sistema de notificaciones se implementará usando WebSockets para comunicación en tiempo real, combinado con persistencia en base de datos para historial y auditoría.

Arquitectura de Notificaciones

Flujo:

1. Ocurre un evento (factura generada, pago recibido, job ejecutado)
2. Se guarda la notificación en MongoDB
3. Se emite el evento por Socket.io
4. El frontend recibe la notificación en tiempo real
5. El usuario puede marcarla como leída

Tecnologías usadas

- Socket.io (NestJS Gateway)
- MongoDB (colección notifications)
- JWT para autenticación de sockets
- Redis (opcional para escalar sockets)

Tipos de notificaciones

- Facturación (invoices generadas, vencidas)
- Pagos recibidos
- Correos enviados
- Jobs completados o fallidos
- Alertas del sistema

Modelo de Notificación (MongoDB)

```
{
  userId,
  type,
  title,
  message,
  read,
  meta,
  createdAt
}
```

Ventajas del uso de WebSockets

- Notificaciones instantáneas
- Mejor experiencia de usuario

- No depende de polling
- Compatible con usuarios offline mediante persistencia