Plan Rework Sistema Solución Contable – Módulos (Auth y Visor CFDIS)

Este documento describe el plan de trabajo para la reescritura de un aplicativo de solución contable, modernizando su arquitectura y tecnologías, manteniendo una arquitectura monolítica modular orientada a una futura migración a microservicios.

🎯 Objetivos principales

- Reescribir el sistema con tecnologías modernas, manteniendo compatibilidad funcional.
- Implementar una arquitectura **hexagonal (Ports & Adapters)** con comunicación **event-driven**.
- Mantenemos MySql como motor de BD.
- Facilitar futura migración a microservicios.
- Mejorar la seguridad y mantenibilidad.

🛠️ Tecnologías seleccionadas

Backend

- **Node.js + TypeScript**: Entorno robusto y tipado.
- **Express.js**: Framework flexible.
- **TypeORM**: ORM para MySQL.
- **Passport.js**: Autenticación local y OAuth.
- **bcrypt**: Hashing de contraseñas.
- **Multer**: Gestión de archivos.
- **Bull + Redis**: Procesamiento asíncrono de CFDIs.
- **Zod**: Validación de datos.
- **Winston / Pino**: Logging estructurado.

Frontend

- **Vue 3 + TypeScript + Vite**: UI moderna y eficiente.
- **TailwindCSS**: Estilos rápidos y consistentes.
- **Pinia**: Manejo de estado.

Infraestructura

- **Docker + docker-compose**: Contenerización.
- **NGINX**: Reverse proxy (opcional).
- **Swagger / OpenAPI**: Documentación de API.
- **GitHub Actions / GitLab CI**: CI/CD.

🚀 **Roadmap de trabajo**

1 Planeación

- Definir modelo de datos en MySQL.
- Diseñar contratos de API con OpenAPI.
- Estructura inicial del monolito modular por dominio.
- Definir eventos y esquema de comunicación interna (EventBus interno).

2 Configuración del entorno

- Inicialización de proyecto (Node.js + TypeScript + Express + TypeORM).
- Configuración de Redis + Bull.
- Configuración de Docker.

3 Desarrollo backend

Módulo Autenticación y usuarios

- Registro / Login / Logout local y OAuth.
- Recuperación de contraseña.
- Gestión de roles y permisos.
- JWT para sesiones.

Módulo Importador CFDIs

- Subida de XML (con Multer).
- Procesamiento masivo de CFDIs con Bull.
- Parsing y persistencia en MySQL.
- Listado de archivos pendientes.

4 Desarrollo frontend

- Login / registro.
- Gestión de usuarios / roles.
- Carga de XML y monitoreo de procesamiento.
- Consultas y reportes.

5 Pruebas

- Unitarias (Jest / Vitest).
- Integración.
- Validaciones estrictas.

6 Despliegue y documentación

🔎 **Notas finales**

- La arquitectura hexagonal facilitará la transición a microservicios.
- El event-driven interno permitirá desacoplar casos de uso y adaptadores.

🕑 Estimación de horas por fase del roadmap

Fase	Actividades clave	Horas
1. Planeación	• Modelo de datos MySQL • Contratos OpenAPI • Esqueleto hexagonal • Definir eventos internos	**12 h**
2. Configuración entorno	• Repo Node+TS+Express+TypeORM • Docker: MySQL + Redis • BullMQ básico	**10 h**
3. Backend	**Auth(15 h)** – Registro/Login/JWT (6 h) – OAuth (3 h) – Roles/Permisos (6 h) **Importador CFDI (20 h)** – Upload XML (3 h) – Procesamiento Bull (7 h) – Parsing + inserciones (7 h) – Listado pendientes (3 h)	**35 h**
4. Frontend	• Login/registro • Gestión usuarios/roles • Carga XML + progreso • Consultas/reportes básicos	**20 h**
5. Pruebas	• Unitarias/Jest • Integración • Validaciones	**10 h**
6. Despliegue & Docs	• Swagger/OpenAPI • Docker Compose final • Pipeline CI/CD	**8 h**
Total (estimado)		**95 – 100 h**

🚀 ¿Qué se puede lograr?

✅ Completar **planeación y setup completo del entorno**.
✅ Desarrollar **backend funcional con autenticación, roles y permisos, y el importador CFDIs básico**.
✅ Tener un **frontend básico** que permita login, carga de XML y visualizar reportes básicos.
✅ Contar con **pruebas unitarias mínimas** y documentación básica (Swagger).
✅ **Docker-compose funcional** para despliegue inicial en un entorno controlado.

⚠️ **Limitación:**
👉 Las vistas frontend pueden ser simples (sin detalles finos de UI/UX).
👉 La parte de reportes avanzados, filtros complejos y optimización puede quedar pendiente para una fase 2.
👉 La cobertura de pruebas podría ser mínima y habría que reforzarla.

📌 Condiciones del plan

✅ **Inicio: Lunes 22 de julio de 2025**
✅ **Horas de trabajo al proyecto:**

- **Lunes a viernes: 3 h/día → 15 h / semana**
- **Sábado: 2 h**
- **Domingo: 3 h**
  👉 **Total semanal: ~20 h**

✅ **Objetivo: terminar el proyecto en 5 semanas aprox**
✅ **Total a lograr: 100 h**

🗓️ **Plan semanal detallado**

Aquí está el calendario ajustado:

Semana 1 (22 jun - 28 jun)

- 🔹 **Planeación (12 h)**
  - **Modelo de datos MySQL(3 h)**
  - **Contratos API OpenAPI (3 h)**
  - **Estructura hexagonal (3 h)**
  - **Definir eventos + comunicación event-driven (3 h)**
- 🔹 **Config. entorno (empezar)**
  - **Node + TS + Express + TypeORM (3 h)**

🕒 **Total semana: 15 h**

Semana 2 (29 jun - 5 jul)

- 🔹 **Config. entorno (7 h)**
  - **Redis + Bull + Docker base (7 h)**
- 🔹 **Backend Auth (completo 15 h)**
  - **Registro, login, JWT (6 h)**
  - **OAuth (3 h)**
  - **Roles y permisos (6 h)**
- 🔹 **Inicio Backend CFDI (3 h)**
  - **Subida XML + directorios (3 h)**

🕒 **Total semana: 25 h**

Semana 3 (6 jul - 12 jul)

- 🔹 **Backend CFDI (17 h)**
  - **Procesamiento masivo + Bull (6 h)**
  - **Parsing + persistencia MySQL (6 h)**
  - **Listado archivos pendientes (5 h)**
- 🔹 **Frontend (3 h)**
  - **Login / registro (3 h)**

🕒 **Total semana: 20 h**

Semana 4 (13 jul - 19 jul)

- 🔹 **Frontend (17 h)**
  - **Gestión usuarios / roles (4 h)**
  - **Carga XML y monitoreo (4 h)**
  - **Consultas básicas (4 h)**
  - **Reportes simples (5 h)**
- 🔹 **Pruebas (3 h)**
  - **Unitarias (3 h)**

🕒 **Total semana: 20 h**

Semana 5 (20 jul - 26 jul)

- 🔹 **Pruebas + despliegue (15 h)**
  - **Integración + validaciones (4 h)**
  - **Docker Compose final + CI/CD (4 h)**
  - **Swagger / OpenAPI (4 h)**

- **Ajustes finales / buffer (3 h)**

🔹 **Mejoras frontend o backend (5 h)**

- **Lo que identifiquemos como necesario durante las pruebas.**

🕒 **Total semana: 20 h**

🎯 **Resultados esperados**

👉 **Con este plan de 5 semanas:**
✅ **Se cubren las 100 h**
✅ **El sistema quedaría en un estado funcional y desplegable**
✅ **El monolito modular quedaría listo para extender o migrar a microservicios**
✅ **El frontend cubriría lo esencial (UI básica)**