# Modelo C4 en ServiTex

## Diagrama de Contexto (System Context)

• Responde: ¿Quién usa el sistema y con qué otros sistemas se relaciona?

• Actor: Persona u organización que interactúa con el sistema (ej: Cliente, Técnico).

• Sistema principal: El sistema bajo análisis (ej: Sistema de Gestión ServiTex).

• Sistemas externos: Otros sistemas con los que se comunica (ej: Facturación, Notificaciones).

• Relaciones: Conexiones entre actores, el sistema y otros sistemas (ej: Cliente consulta estado → ServiTex).

• Uso: Ideal para mostrar una visión macro a interesados no técnicos.

## Diagrama de Contenedores (Container Diagram)

• Responde: ¿De qué está hecho el sistema?

• Contenedor: Aplicación o servicio ejecutable dentro del sistema. Cada contenedor es independiente y cumple un rol específico. Ej: Web Application, Mobile App, API Application, Database, Notification Service.

• Base de datos (Database): Contenedor especializado para almacenamiento de información.

• Interfaz de usuario (Frontend): Apps o páginas web que permiten interacción con usuarios.

• Backend (API/Application): Contenedor donde reside la lógica de negocio.

• Conexiones: Líneas que muestran cómo se comunican los contenedores y con sistemas externos.

• Uso: Muestra cómo el sistema está dividido en partes tecnológicas principales.

## Diagrama de Componentes (Component Diagram)

• Responde: ¿Cómo está organizado internamente un contenedor clave?

• Componente: Módulo lógico dentro de un contenedor, encargado de una responsabilidad específica. Ej: Diagnosis Component, Quotation Component, Notification Component.

• Controller: Puerta de entrada que recibe solicitudes y delega a componentes.

• Servicios de negocio: Implementan reglas del sistema (ej: Diagnosis, Maintenance).

• Integraciones: Conexiones con sistemas externos o módulos auxiliares (ej: Billing Integration).

• Relaciones: Flujo entre controladores, componentes, repositorios y servicios.

• Uso: Permite a arquitectos y desarrolladores entender la estructura interna del backend o de una parte crítica del sistema.

## Diagrama de Código (Code/Class Diagram)

• Responde: ¿Cómo está implementado un componente específico?

• Clase: Unidad de código que encapsula atributos y métodos (ej: DiagnosisService).

• Entidad (Entity): Clase que representa datos de negocio (ej: Equipment, Failure).

• DTO (Data Transfer Object): Clases que transportan datos entre capas (ej: DiagnosisRequest, DiagnosisResponse).

• Repositorio (Repository): Clase que maneja persistencia en la base de datos.

• Excepción (Exception): Clase para manejar errores de ejecución.

• Relaciones: Dependencias y uso entre clases (ej: DiagnosisService usa DiagnosisRepository).

• Uso: Brinda detalle técnico a nivel de implementación, pensado para programadores.