Informe de Implementación: Servicio de Consulta de Citas por Paciente

Este documento describe la estructura, el flujo de trabajo y la justificación arquitectónica de los componentes desarrollados para el servicio de **Consulta de Citas de un Paciente** (GET /pacientes/:numeroDoc/cita), siguiendo la Arquitectura Hexagonal.

1. Fundamento Arquitectónico

La implementación sigue el patrón de diseño de **Arquitectura Hexagonal** (Puertos y Adaptadores), asegurando que la lógica de negocio (verificación de existencia) sea independiente de la tecnología de persistencia (PostgreSQL) y del framework web (Fastify).

2. Servicio de Consulta de Citas de un Paciente

Qué hace: Permite consultar todas las citas (pasadas y futuras) asociadas a un paciente, retornando datos completos de la cita, médico (incluida especialidad) y consultorio.

2.1. Flujo Requerido y Componentes Clave

Capa	Componente	Responsabilidad	Flujo Esperado (Punto 2)
Aplicación (Puerto)	IConsultaCitasPacienteCasosUso.ts	Define el contrato: ejecutarServicio(numeroDoc, limite?).	Desacopla la lógica de aplicación del controlador.
Aplicación (Caso de Uso)	ConsultaPacienteCasosUso.ts	**Regla de Negocio Central:** 1. Llama a repositorioPacientes.obtenerPacientePorId para **verificar la existencia del paciente**. 2. Si el paciente no existe, lanza PacienteNoExisteError. 3. Si existe, llama al repositorio de citas para obtener los datos.	1. Se verifica que el paciente exista. 2. Se retorna lista de citas (pasadas y/o futuras).
Infraestructura (Salida)	CitasRepositorio.ts	Adaptador PostgreSQL. Contiene la consulta SQL que une las tablas	Persistencia y retorno de datos estructurados.

		(citas_medicas, medicos, asignacion_medicos y consultorios) para obtener la lista de citas. **Se asegura de incluir la especialidad del médico.**	
Infraestructura (Entrada)	pacientesEnrutador.ts	Configura la ruta GET /pacientes/:numeroDoc/cita en Fastify.	Recepción de la solicitud.
Infraestructura (Controlador)	PacienteControlador.ts	Adaptador HTTP. 1. Extrae los parámetros (numeroDoc, limite). 2. Llama al Caso de Uso. 3. **Manejo de Errores:** Captura PacienteNoExisteError (Dominio) y lo traduce a **HTTP 404** (Paciente Inexistente). 4. Retorna **HTTP 200** con la lista de citas (incluyendo lista vacía).	Manejo de respuesta HTTP y traducción de errores de Dominio.

2.2. Casos a Evidenciar Cubiertos

1. **Paciente inexistente (HTTP 404):**
    - La verificación ocurre en ConsultaPacienteCasosUso.
    - Al no encontrar al paciente, lanza el PacienteNoExisteError.
    - El PacienteControlador captura este error de dominio y responde con un código **HTTP 404**, con un mensaje claro que indica que el paciente no existe.
2. **Paciente sin citas (lista vacía) (HTTP 200):**
    - El ConsultaPacienteCasosUso verifica que el paciente existe y procede a llamar a CitasRepositorio.
    - CitasRepositorio ejecuta la consulta SQL y retorna un arreglo vacío ([]).
    - El PacienteControlador responde con un código **HTTP 200** y el cuerpo JSON contendrá la propiedad citas como un arreglo vacío, cumpliendo con la definición de respuesta válida.