# **Justificación de la Arquitectura de 3 Capas para el Sistema de Gestión de Proyectos Contratistas**

## **Arquitectura Seleccionada**

Para el proyecto "Gestión integral de proyectos contratistas" se implementa una **arquitectura de 3 capas (Three-tier architecture)** que divide el sistema en:

1. **Capa de Presentación**: React.js + Vite (Frontend web, móvil y desktop) Hosting en Firebase
2. **Capa de Lógica de Negocio**: Node.js + Express (Backend y APIs) Despliegue en CloudFlare/Render
3. **Capa de Datos**: PostgreSQL con Supabase (Base de datos)

## **Justificación de la Selección**

### **Simplicidad y Eficiencia**

La arquitectura de 3 capas elimina complejidades innecesarias, permitiendo un desarrollo más directo y eficiente. Cada capa tiene responsabilidades claras: presentación de datos, procesamiento de lógica empresarial y almacenamiento de información.

### **Separación de Responsabilidades**

* **Frontend**: Gestión de interfaces, formularios de clientes, proyectos, presupuestos y reportes
* **Backend**: Autenticación JWT, validaciones, cálculos financieros y lógica de negocio
* **Base de datos**: Almacenamiento relacional de clientes, proyectos, materiales y rendiciones

### **Escalabilidad Focalizada**

Cada capa puede escalar independientemente según las necesidades del negocio. El backend puede manejar más usuarios simultáneos, el frontend puede adaptarse a diferentes dispositivos, y la base de datos puede optimizarse para consultas complejas.

### **Desarrollo en Equipo Optimizado**

La estructura permite trabajo paralelo eficiente: Nicolás Cárcamo desarrolla backend y base de datos, mientras David Coo se enfoca en frontend y despliegue, maximizando productividad en el cronograma de 12 semanas.

## **Alineación con el Proyecto**

### **Cumplimiento de Competencias**

* **Construir modelos de datos**: Diseño relacional en PostgreSQL
* **Desarrollar soluciones sistematizadas**: Metodología por capas estructurada
* **Gestionar proyectos informáticos**: Control y planificación por componentes
* **Realizar pruebas de certificación**: Testing específico por capa

### **Soporte a Objetivos del Proyecto**

La arquitectura permite centralizar la gestión de clientes, proyectos, presupuestos, rendiciones y materiales, asegurando trazabilidad y control financiero a través del flujo completo entre las 3 capas.

## **Factibilidad Técnica**

**Recursos disponibles**: Entorno de desarrollo React.js/Node.js, acceso a PostgreSQL/Supabase **Despliegue**: Firebase Hosting (frontend), servicios cloud (backend), Supabase (base de datos) **Tiempo**: 12 semanas permiten desarrollo eficiente sin complejidades arquitectónicas excesivas

### **Seguridad del Sistema**

Para asegurar la protección de datos y la disponibilidad del sistema, se implementan las siguientes medidas:

#### **Cloudflare como Gestor de Seguridad**

Se integra **Cloudflare** como solución de seguridad y rendimiento para el sistema. Esto permite:

* **Protección contra ataques DDoS:** Cloudflare mitiga solicitudes maliciosas que puedan sobrecargar el backend.
* **Firewall de aplicaciones web (WAF):** Reglas que ayudan a bloquear el tráfico sospechoso o malintencionado que intenta dañar o entrar al sistema.
* **Cifrado SSL/TLS automático:** Todo lo que se envía o recibe desde el sistema está cifrado (protegido), lo que evita que otras personas puedan ver esa información.
* **Optimización de rendimiento:** Mejora los tiempos de carga del frontend gracias al CDN distribuido globalmente.

#### **Otras Medidas de Seguridad**

* **Inicio de sesión seguro:** El sistema usa un método de identificación que asegura que solo los usuarios autorizados puedan acceder.  
  **Permisos por usuario:** Dependiendo del tipo de usuario, solo se permite ver o modificar cierta información.
* **Revisión de datos:** Toda la información que los usuarios envían se revisa para evitar errores o acciones maliciosas.
* Etc. se puede agregar más todavía.