# Don Diego App – Sistema de Gestión de Requisiciones

Este proyecto es un MVP de una aplicación administrativa para gestionar el flujo de solicitudes de compra dentro de una empresa. La aplicación está construida con **Next.js 14 (App Router)**, **Prisma** y **PostgreSQL**, y utiliza **Tailwind CSS** para la interfaz de usuario.

## Contexto del problema

En la fase inicial de análisis se definió un flujo de trabajo que involucra a múltiples actores:

| Rol | Descripción |
| --- | --- |
| **Solicitante** | Inicia la *Solicitud de Requisición (RQ)* indicando proyecto, ítems y especificaciones. |
| **Centro Logístico** | Asocia la RQ a un centro de costo. |
| **Compras** | Recibe cotizaciones de proveedores, carga documentos (RUT, certificados, condiciones) y genera un cuadro comparativo. |
| **Proveedor** | Envía cotizaciones y documentos (fuera del sistema). |
| **Autorizadores** | Revisan y aprueban o rechazan la opción elegida por el solicitante. |
| **Tesorería** | Programa y valida el pago al proveedor después de emitida la Orden de Compra (OC). |
| **Alertas** | Notifica a los actores sobre la emisión de OC, cierres de recepción y otros hitos. |

La aplicación implementa el flujo descrito en el **diagrama de secuencia del problema 1**, permitiendo crear RQ, gestionar cotizaciones, elegir ofertas, solicitar aprobaciones, emitir órdenes de compra y registrar la recepción conforme o no conforme.

## Requisitos previos

Para ejecutar la aplicación necesitarás:

* Node.js (v18 o superior).
* Un gestor de paquetes (npm o pnpm).
* Docker (opcional, para iniciar un contenedor de PostgreSQL en desarrollo).
* Una instancia de PostgreSQL local o remota.

## Instalación y puesta en marcha

1. **Clonar el repositorio** o copiar los archivos en tu máquina local.
2. **Instalar dependencias** con tu gestor preferido:

pnpm install  
# o con npm  
# npm install

1. **Configurar las variables de entorno**. Crea un archivo .env en la raíz del proyecto con un contenido similar a este:

DATABASE\_URL="postgresql://postgres:postgres@127.0.0.1:5433/dondiego?schema=public&sslmode=disable"  
AUTH\_SECRET="cambia-esto-por-uno-fuerte"  
AUTH\_URL="http://localhost:3000"

Ajusta la URL de la base de datos según tu entorno. Si usas Docker (ver paso 4), el puerto externo suele ser **5433**.

1. **Iniciar PostgreSQL con Docker (opcional)**. Si no tienes una instancia de PostgreSQL, puedes levantar un contenedor con:

docker run --name don-diego-pg \  
 -e POSTGRES\_USER=postgres \  
 -e POSTGRES\_PASSWORD=postgres \  
 -e POSTGRES\_DB=dondiego \  
 -p 5433:5432 \  
 -d postgres:16

Este comando inicia PostgreSQL mapeando el puerto interno 5432 al puerto **5433** en tu host.

1. **Ejecutar las migraciones de Prisma** para generar las tablas:

pnpm prisma migrate dev --name init  
# o  
# npx prisma migrate dev --name init

1. **Poblar la base de datos con datos de prueba** utilizando el script de semilla:

pnpm ts-node prisma/seed.ts  
# o  
# npx ts-node prisma/seed.ts

Este script crea usuarios de ejemplo para cada rol, algunos proyectos, centros de costo y una RQ inicial.

1. **Levantar el servidor de desarrollo**:

pnpm dev  
# o  
# npm run dev

1. **Abrir la aplicación en el navegador**. Accede a http://localhost:3000 para interactuar con la app.

## Funcionalidades implementadas

* **Creación de requisiciones (RQ)** con lista de ítems, especificaciones, proyecto y centro de costo.
* **Carga de cotizaciones y documentos** de proveedores (a través de la interfaz de Compras).
* **Cuadro comparativo y checklist** para evaluar ofertas.
* **Elección de oferta** por el solicitante y **flujo de aprobaciones** por autorizadores.
* **Generación de Orden de Compra (OC)** en PDF con numeración correlativa.
* **Registro de recepción de bienes** en el Centro Logístico, con manejo de No Conforme y reaperturas parciales.
* **Notificaciones básicas** por correo electrónico (con integración opcional a servicios como Resend).

## Estructura del proyecto

don-diego-app/  
├── prisma/  
│ ├── schema.prisma # Esquema de la base de datos  
│ └── seed.ts # Script de semilla con datos de ejemplo  
├── src/  
│ ├── app/  
│ │ ├── api/  
│ │ │ └── catalogs/route.ts # Ruta API para proyectos y centros de costo  
│ │ ├── dashboard/page.tsx # Vista de bandejas por rol  
│ │ ├── rq/  
│ │ │ ├── actions.ts # Acciones de servidor para RQ  
│ │ │ └── new/page.tsx # Formulario para crear nueva RQ  
│ │ ├── layout.tsx # Layout principal  
│ │ └── page.tsx # Home / landing  
│ ├── lib/  
│ │ ├── prisma.ts # Cliente Prisma (singleton)  
│ │ └── roles.ts # Utilidades y tipos de roles  
│ └── app/globals.css # Estilos globales (Tailwind)  
├── .env.example # Ejemplo de configuración  
├── package.json # Dependencias y scripts npm/pnpm  
└── README.md # Este documento

## Próximos pasos y mejoras

Este MVP cubre el flujo básico de requisiciones, pero puede ampliarse con:

* Autenticación y control de acceso basado en roles.
* Reportes y paneles de KPIs.
* Integración con ERP o sistemas contables.
* Flujos de reautorización por cambios de precios o especificaciones.
* Notificaciones por WhatsApp y log de auditoría.

¡Gracias por utilizar **Don Diego App**! Para cualquier duda o sugerencia, no dudes en abrir un *issue* o contacto directo.