NEXUS SOLUCIONES S.A.S

PRUEBA TÉCNICA

Desarrollador Full Stack

Next.js + NestJS + PostgreSQL + Prisma + AWS S3

1. DESCRIPCIÓN GENERAL

Desarrollar un sistema de gestión de clientes y documentos de identificación que permita almacenar información de clientes (personas naturales y empresas), sus documentos de identificación (Cédula y RUC ecuatorianos), y cargar las imágenes de estos documentos a un bucket S3 compatible.

Duración: 4 horas

2. STACK TECNOLÓGICO REQUERIDO

Backend (NestJS)

- NestJS (framework principal)
- Prisma ORM (conexión y migraciones de base de datos)
- PostgreSQL (base de datos proporcionada)
- AWS SDK o S3 client compatible (almacenamiento de archivos)

Frontend (Next.js)

- Next.js con App Router
- TypeScript
- Tailwind CSS
- Shadcn (opcional)
- React Hook Form y zod o similar (manejo de formularios)

3. CONFIGURACIÓN DE INFRAESTRUCTURA

3.1 Base de Datos PostgreSQL

URL de Conexión:

```
postgresql://
postgres:OvLsNmkFJviAGXjpmxEbLpfISTinBWEc@crossover.proxy.rlwy.net:13475/
railway
```

3.2 Bucket S3 Compatible (Almacenamiento de Archivos)

El bucket proporcionado es compatible con S3 (t3.storage). Utilice las siguientes credenciales:

Endpoint URL	`https://t3.storageapi.dev`
Region	`auto`
Bucket Name	`efficient-tupperware-zvahyp`
Access Key ID	`tid_PKjSLkigqbpxwjwpWFSuxMkESYpHalGAzILPxPVoCSAzbCzuMK`
Secret Access Key	`tsec_o3nMCvEuxJ_vG4NFI6K1sLRIMI9+EXYX6j2bUBKBCNQ0iHZI7PpGiT86sOHkQo6KezH3Xy`

4. MODELO DE DATOS (PRISMA SCHEMA)

Debe crear los modelos de datos según su criterio para el registro de clientes guardando cédula y otros documentos

5. REQUERIMIENTOS DEL BACKEND (NestJS)

5.1 Endpoints API REST

`POST /api/clients`

`GET /api/clients`

`GET /api/clients/:id`

5.2 Validaciones

Se tomará en cuenta las validaciones que se haga y los métodos que se utilice, además de la arquitectura y de las buenas prácticas

6. REQUERIMIENTOS DEL FRONTEND (Next.js)

6.1 Páginas y Componentes

Página: Listado de Clientes (/clients)

Página: Crear Cliente (/clients/new)

Página: Detalle de Cliente (/clients/[id])

6.2 Funcionalidades Requeridas

- Implementar llamadas al backend usando fetch o axios
- Se tomará en cuenta las validaciones y lo visual

7. CRITERIOS DE EVALUACIÓN

Criterio	Puntaje	Descripción
Funcionalidad Completa	35%	Todos los endpoints funcionan correctamente. CRUD completo. Subida de archivos a S3 operativa.
Validaciones	20%	
Calidad de Código	20%	Estructura clara, separación de responsabilidades, nomenclatura consistente, código limpio.
Integración Prisma + PostgreSQL	10%	Schema correcto, migraciones exitosas, relaciones bien definidas.
Integración S3 + Visualización	10%	Archivos se suben al bucket. URLs funcionan. Imágenes visibles en frontend.
TypeScript y Tipado	5%	Uso apropiado de interfaces y tipos. Código sin errores de tipo.
TOTAL	**100%**	**Puntaje mínimo aprobatorio: 70%**

8. PUNTOS BONUS (Opcional)

- Implementar búsqueda/filtrado en el listado de clientes
- Agregar tests unitarios básicos (Jest)
- Documentación OpenAPI/Swagger del backend
- Drag & drop para subir archivos
- Vista previa de imágenes antes de subir
- Gestión de estados con zustand (algo simple como modo claro, modo oscuro)
- Endpoint simulando un servicio de validación de cédula que aveces falle

9. ENTREGABLES

5. Repositorio Git con código fuente (backend y frontend)
6. README.md con instrucciones de instalación y ejecución
7. Archivo .env.example con las variables de entorno necesarias
8. Schema de Prisma con las migraciones
9. Aplicación funcionando en local (demostración en vivo)

NEXUS SOLUCIONES S.A.S

¡Éxito en la Prueba Técnica!

Nexus Soluciones S.A.S - Equipo de Desarrollo