PRUEBA TÉCNICA – DESARROLLADOR FULLSTACK (LARAVEL + NEXT.JS / JWT / TELECOMUNICACIONES)

1. Premisa General

La presente prueba técnica tiene como finalidad evaluar la capacidad del candidato para **diseñar, desarrollar e integrar un sistema fullstack moderno** basado en **Laravel (Backend API)** y **Next.js (Frontend React)**.

El contexto simula una empresa de telecomunicaciones **—TelecomPlus S.A.S.—** que ofrece servicios de **Internet y Televisión**. El sistema debe gestionar usuarios, contratos y servicios mediante un flujo de autenticación segura con **JSON Web Tokens (JWT)**.

---

2. Descripción del Caso

La empresa desea implementar una aplicación interna que permita registrar clientes, gestionar sus contratos y los servicios asociados.
El sistema deberá incluir:

- **Autenticación JWT (Login y Registro)**
- **Gestión jerárquica de entidades:** Usuario → Contrato → Servicio
- **Interfaz web en Next.js** que permita interactuar con la API Laravel (registro, login y consulta de contratos y servicios)

---

3. Requerimientos Técnicos

3.1. Backend – Laravel API

1. Crear un proyecto con **Laravel 10 o superior**.
2. Instalar y configurar **tymon/jwt-auth** para el manejo de tokens JWT.
3. Implementar los siguientes **modelos y migraciones**:

User

Campo	Tipo	Descripción
id	integer	Clave primaria
name	string	Nombre del usuario
email	string	Correo único
password	string	Contraseña hasheada
role	string	Rol del usuario (por defecto "customer")

Contract

Campo	Tipo	Descripción
id	integer	Clave primaria
user_id	foreignId	Relación con usuario
contract_number	string	Número único del contrato
start_date	date	Fecha de inicio
status	string	Estado del contrato (“active”, “suspended”, etc.)

Service

Campo	Tipo	Descripción
id	integer	Clave primaria
contract_id	foreignId	Relación con contrato
type	enum	“internet” o “tv”
plan_name	string	Nombre del plan
price	decimal	Precio mensual

4. Relaciones:
   - **User** → tiene muchos **Contracts**.
   - **Contract** → pertenece a un **User** y tiene muchos **Services**.
   - **Service** → pertenece a un **Contract**.
5. Endpoints requeridos (protegidos con middleware `auth:api`):
   - `POST /api/register` – Registro de usuario.
   - `POST /api/login` – Inicio de sesión y generación del JWT.
   - `GET /api/me` – Obtener información del usuario autenticado.
   - `GET /api/contracts` – Listar contratos del usuario autenticado.
   - `POST /api/contracts` – Crear contrato.
   - `POST /api/services` – Crear servicio asociado a un contrato.
6. Validar datos y retornar códigos de estado HTTP apropiados (201, 400, 401, 404).

3.2. Frontend – Next.js

1. Crear un proyecto **Next.js 14 o superior** con el App Router habilitado.
2. Utilizar **React**, **TypeScript** y una librería de componentes moderna (ej. **ShadCN UI**, **Chakra UI** o **MUI**).

3. El frontend deberá consumir la API de Laravel y permitir las siguientes acciones:

a) Registro de usuario

- Formulario con campos `name`, `email`, `password`.
- Al enviar, registrar el usuario en la API.

b) Inicio de sesión

- Formulario con `email` y `password`.
- Si el login es exitoso, almacenar el token JWT (en localStorage o cookies).
- Redirigir al panel principal.

c) Panel principal

- Mostrar la información del usuario autenticado (`/api/me`).
- Listar los contratos asociados (`/api/contracts`).
- Mostrar los servicios de cada contrato.
- Permitir crear nuevos contratos y servicios mediante formularios modales simples.

---

4. Condiciones de la Prueba

1. **Duración máxima:** 72 horas desde la recepción del documento.
2. **Entrega:**
   - Enlace a un **repositorio público (GitHub/GitLab)** con dos carpetas:
     - `/backend` → proyecto Laravel.
     - `/frontend` → proyecto Next.js.
   - Archivo `README.md` en la raíz con:
     - Instrucciones de instalación y ejecución de ambos entornos.
     - Variables de entorno necesarias.
     - Endpoints principales y breve descripción del flujo.
3. **Evaluación:**

Criterio	Ponderación
Correcta implementación de JWT y flujo de autenticación	25%
Estructura, limpieza y organización del código	20%
Uso de relaciones entre modelos y migraciones correctas	15%
Integración funcional entre Laravel y Next.js	20%
Interfaz y experiencia de usuario (flujo básico login/registro/panel)	10%
Documentación clara y reproducibilidad	10%

4. **Requisitos adicionales:**
   - El backend debe correr en `http://localhost:8000`.
   - El frontend debe consumir la API desde `http://localhost:8000/api`.
   - Se deberá configurar CORS correctamente en Laravel.
5. **Prohibiciones:**
   - No usar plantillas ni boilerplates externos que oculten la lógica central.
   - No usar autenticación Laravel Sanctum o Passport (debe ser JWT puro).
   - No usar tokens falsos o mockeados.
6. **Entrega opcional (plus):**
   - Seeder para crear datos de ejemplo.
   - Documentación con Swagger o Postman Collection.
   - Deploy funcional (Render, Vercel o similar).

---

5. Resultado Esperado

El candidato deberá demostrar con su entrega que el sistema:

1. Permite registrar y autenticar usuarios mediante JWT.
2. Permite al usuario autenticado crear contratos y servicios.
3. Presenta una interfaz funcional en Next.js para:
   - Registro
   - Login
   - Visualización de contratos y servicios
4. Mantiene consistencia entre frontend y backend con buenas prácticas REST.

---

6. Cláusulas Finales

- La prueba tiene un propósito exclusivamente evaluativo y no implica relación laboral inmediata.
- El código entregado será revisado con fines de análisis técnico y buenas prácticas.
- TelecomPlus S.A.S. podrá solicitar una presentación breve del proyecto para verificar comprensión y autoría.