PRUEBA TÉCNICA LITE THINKING - 2026
DESARROLLADOR PYTHON, DJANGO, REACT

En el presente documento encontrará los detalles de la prueba técnica. La prueba deberá ser resuelta y entregada en la fecha enviada a tu correo. Toda prueba entregada posterior a dicha fecha no será válida.

Objetivo de la prueba:

Validar tus conocimientos y técnicas en el desarrollo de aplicaciones utilizando las tecnologías **Python**, **Django**, **React** y **PostgreSQL**. Aunque no está descrito en la prueba, ten en cuenta las buenas prácticas de código limpio. Aplica **Atomic Design** en el proyecto y se evaluará la calidad del código y la estructura de carpetas. Utiliza librerías de **IA**, **Blockchain** y otras librerías novedosas para proponer una funcionalidad adicional en este proyecto.

Descripción de la prueba:

Construir una aplicación que exponga las siguientes vistas:

a) Vista **Empresa** con un formulario que capture la siguiente información:

- NIT (Llave primaria).
- Nombre de la empresa.
- Dirección.
- Teléfono.

b) Vista de **Productos** con un formulario que captura la siguiente información:

- Código.
- Nombre del producto.
- Características.
- Precio en varias monedas.
- Empresa.

c) Vista de **Inicio de Sesión** con un formulario que capture la información del usuario: **correo y contraseña**.

d) Vista de **Inventario** con un formulario que permita la descargar de un PDF con la información de esa tabla y adicional utilizar alguna API REST o SOAP para poder enviar ese PDF a un correo deseado.

e) Deben existir dos tipos de usuarios:
- **Administrador:** Tiene acceso a las funciones de eliminación, registro y/o edición de una Empresa. Adicionalmente, este usuario podrá registrar productos por empresa y guardarlos en una tabla inventario, donde se vean los productos por empresa.
- **Externo:** Puede visualizar las empresas como visitante.

f) La contraseña utilizada debe estar encriptada para autenticación del Usuario **Administrador**.

g) Funcionalidad para este proyecto utilizando IA, Blockchain y otras librerías novedosas para proponer una vista o requerimiento adicional en este proyecto.

h) **Arquitectura:** Los modelos o entidades del negocio deben estar ubicados en una capa de dominio independiente del Backend, desarrollada en Python y Django, siguiendo principios de Arquitectura Limpia. La capa de dominio será **exclusiva para los modelos o entidades del negocio y sus reglas**, y deberá mantenerse **desacoplada de las capas de presentación, API e infraestructura**, evitando dependencias directas con:

- Vistas.
- Serializers.
- Controladores.
- Lógica HTTP.

El Backend, desarrollado con **Django**, actuará como capa de aplicación e infraestructura, encargándose de:

- Exposición de APIs.
- Autenticación.
- Persistencia de datos.
- Integraciones externas.

i) **Gestión de dependencias:** La gestión de dependencias y entornos del proyecto de la **capa de dominio, donde estarán ubicados los modelos o entidades**, debe realizarse utilizando la tecnología **Poetry**. Se deberá incluir el archivo **pyproject.toml** correctamente configurado como parte de la entrega del proyecto. **Dicho paquete de dominio deberá ser consumido desde el proyecto Backend.**

j) **Nota final de evaluación:**
- Organización del proyecto y estructura de carpetas
- Separación de responsabilidades
- Calidad del código
- Correcta implementación de los requerimientos
- Buenas prácticas de desarrollo
- Propuesta técnica de la funcionalidad adicional