Prueba Técnica – Desarrollador(a) Django (Nivel Intermedio)

Objetivo

Evaluar las habilidades del candidato en el desarrollo de aplicaciones web utilizando Python y Django REST Framework, incluyendo:

- Modelado de datos y uso del ORM.
- Construcción de API REST siguiendo buenas prácticas.
- Manejo de autenticación y permisos.
- Organización del código, claridad y mantenibilidad.
- Uso de herramientas complementarias como pruebas, documentación y optimización.

Esta prueba busca simular un reto técnico que podría encontrarse en un proyecto real de la empresa.

Tiempo y Entrega

Duración estimada:
3 a 4 horas.
Forma de entrega:
Repositorio en GitHub o GitLab (público o privado con acceso compartido).
Alternativamente, un archivo comprimido (.zip) con el proyecto.
Incluye:
Código fuente funcional.
Archivo README.md con instrucciones claras de instalación y ejecución.
Breve explicación de decisiones técnicas (ej. elección de librerías, estructura, etc.).

Contexto

Se requiere una aplicación de gestión de tareas colaborativa y sencilla.
Cada usuario podrá:

- Registrarse e iniciar sesión.
- Crear, modificar y eliminar sus propias tareas.
- Organizar tareas en categorías.
- Filtrar tareas por estado o categoría.

El propósito es evaluar cómo estructuras el proyecto, aplicas las buenas prácticas de Django, y construyes una API que sea clara, mantenible y segura.

Requerimientos Técnicos

Modelos

- Status
- Category
- Task

Funcionalidad obligatoria

- Autenticación de usuarios (registro/login con DRF).
- CRUD de categorías.
- CRUD de tareas (solo el dueño puede ver o modificar sus tareas).
- Filtrado de tareas por estado y categoría.
- API REST expuesta con Django REST Framework.

Extras opcionales

- Optimización de consultas.
- Pruebas automáticas (unitarias o con pytest).
- Uso de signals (ejemplo: log al crear una tarea, actualización de registros).
- Documentación de la API con Swagger o drf-spectacular.
- Uso de Docker para facilitar la instalación.

Criterios de Evaluación

El entregable será evaluado con base en los siguientes criterios:

Funcionalidad (40%)
Cumplimiento de los requerimientos principales.
Correcto manejo de autenticación y permisos.
API funcionando sin errores críticos.

Calidad del código (30%)
Organización del proyecto (estructura de carpetas, modularidad).
Claridad, legibilidad y buenas prácticas de Django/DRF.
Manejo adecuado de relaciones en el ORM.

Documentación e instrucciones (15%)
README claro y completo (instalación, ejecución, ejemplos de uso).
Explicación de decisiones técnicas.

Extras opcionales (15%)
Test automatizados.
Optimización y buenas prácticas avanzadas.
Uso de herramientas adicionales (Swagger, Docker, etc.).