

Prueba Técnica BackEnd

Contexto

Eres el Desarrollador BackEnd de Inlaze, una compañía de marketing de afiliados especializada en apuestas deportivas en Colombia. Tu responsabilidad principal es garantizar el desarrollo y mantenimiento eficiente de los sistemas backend, asegurando el cumplimiento de las normativas de seguridad y optimizando los recursos relacionados con el departamento de TI. Actualmente, Inlaze se está posicionando como el servicio web de apuestas deportivas más grande de Latinoamérica.

IMPORTANTE ANTES DE EMPEZAR!

No buscamos contratar a Chatgpt o Gemini! 🤖

Objetivo de la prueba

Evaluar tu conocimiento en el diseño e implementación de un sistema de gestión de tareas utilizando NestJS, adoptando una arquitectura de microservicios, y aplicando las mejores prácticas en desarrollo backend.

1. Descripción del Proyecto

Desarrolla un sistema de gestión de tareas llamado "MicroTask Manager" utilizando NestJS y adoptando una arquitectura de microservicios. El sistema debe permitir a los usuarios crear, actualizar, asignar y visualizar tareas en un entorno colaborativo distribuido.

2. Funcionalidades Requeridas

Gestión de Tareas:

1. Crear nuevas tareas:

- Los usuarios deben poder crear nuevas tareas especificando un título, una descripción, una fecha límite y un estado (por hacer, en progreso, completada).

2. Actualizar tareas existentes:

- Los usuarios deben poder actualizar la información de una tarea existente, incluyendo el título, la descripción, la fecha límite y el estado.

3. Asignar tareas:

- Los usuarios deben poder asignar una tarea a un usuario o equipo específico.

4. **Marcar tareas como completadas:**

- Los usuarios deben poder marcar una tarea como completada y restablecerla al estado anterior si es necesario.

5. **Eliminar tareas:**

- Los usuarios deben poder eliminar una tarea existente.

Arquitectura de Microservicios:

1. **División de funcionalidades:**

- Divide la funcionalidad de gestión de tareas en microservicios independientes, cada uno manejando una parte específica de la lógica de negocio.

2. **Comunicación entre microservicios:**

- Cada microservicio debe ser independiente y comunicarse con los otros a través de una interfaz bien definida (por ejemplo, API RESTful o mensajería asincrónica).

3. **Patrones de diseño:**

- Considera utilizar patrones de diseño como Event Sourcing o CQRS para la gestión de eventos y la separación de comandos y consultas.

Escalabilidad y Tolerancia a Fallos:

1. **Escalabilidad horizontal:**

- Diseña los microservicios de manera que sean escalables horizontalmente y puedan manejar grandes volúmenes de solicitudes de manera eficiente.

2. **Tolerancia a fallos y recuperación:**

- Implementa estrategias de tolerancia a fallos y recuperación para garantizar la disponibilidad y fiabilidad del sistema.

Seguridad:

1. **Autenticación y autorización:**

- Implementa un sistema de autenticación y autorización para proteger los microservicios y restringir el acceso a usuarios autenticados.

2. **Prácticas de seguridad:**

- Utiliza prácticas de seguridad recomendadas para proteger los datos sensibles y prevenir ataques como la inyección de código malicioso.

3. Requisitos Técnicos:

1. **Framework:**

- Utiliza NestJS para desarrollar los microservicios y asegúrate de que cada microservicio sea independiente y autocontenido.

2. Bases de Datos:

- Utiliza PostgreSQL y MongoDB para almacenar y gestionar los datos de las tareas y usuarios.

3. Contenedores:

- Utiliza Docker y Docker Compose para facilitar la implementación y la ejecución de los microservicios en entornos de desarrollo y producción (opcional, punto extra).

4. Pruebas:

- Implementa pruebas unitarias y pruebas de integración para garantizar la calidad y fiabilidad del código (opcional, punto extra).

4. ¿Qué se espera de ti en esta prueba? (IMPORTANTE)

- **Dominio de las tecnologías:** NestJS (Backend), PostgreSQL, MongoDB (Database), Docker (Container)
- **Habilidades de documentación:**
 - **Redacción clara y concisa:** Redacta procedimientos operativos estándar (SOP) de manera clara, concisa y fácil de seguir, utilizando un lenguaje sencillo y evitando tecnicismos innecesarios.
 - **Estructura lógica:** Organiza la información de manera lógica y secuencial, siguiendo un flujo de trabajo claro y evitando saltos bruscos o información incompleta.
 - **Modelado de la arquitectura:** Diagramado del planteamiento de la solución.
- **Enfoque en la mejora continua:**
 - **Identificar oportunidades de mejora:** Analiza los procesos actuales y detecta áreas donde se pueden aplicar mejoras para optimizar la comunicación, prevenir errores y agilizar el proceso de desarrollo.
 - **Proponer soluciones efectivas:** Desarrolla e implementa soluciones prácticas y escalables que aborden las necesidades identificadas y contribuyan a la mejora continua de los procesos.
- **Comunicación efectiva de las propuestas:** Presenta las propuestas de manera clara y convincente, destacando los beneficios y el impacto positivo que tendrán en la organización.
 - La funcionalidad
 - La claridad y calidad del código
 - La implementación de la arquitectura de microservicios
 - La eficiencia del rendimiento y la escalabilidad del sistema

AVISO IMPORTANTE

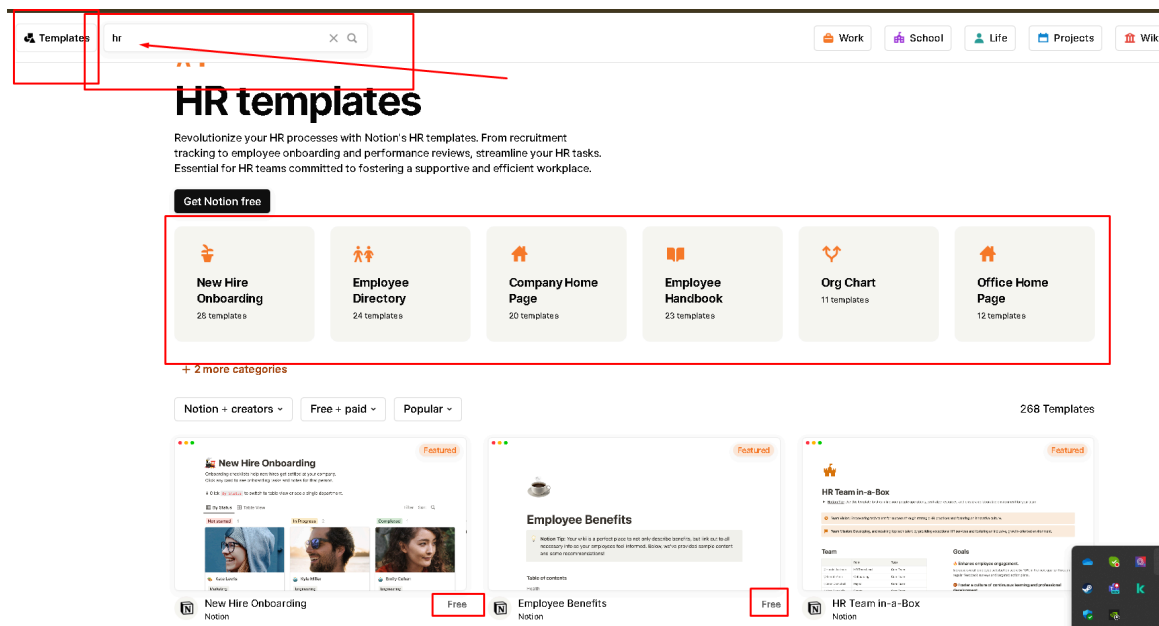
Apreciado candidato

Al postularte a este proceso de selección, te indicamos que las pruebas que realices tienen como único objetivo evaluar tus competencias profesionales y personales. Es importante destacar que estas pruebas NO garantizan automáticamente el acceso al empleo que desees.

Queremos que tengas la plena seguridad de que la empresa se compromete de manera estricta y transparente a no utilizar el material que desarrolles para ningún fin fuera del proceso de selección. Este material será utilizado exclusivamente para evaluar tus habilidades en el contexto de este proceso. Además, renuncias a cualquier reclamación de derechos de autor o propiedad intelectual sobre dicho producto, ya que su uso estará limitado únicamente a este proceso de selección

5. Recursos para hacer la prueba (EJEMPLO)

- Puedes usar repositorios base que te ayudaran a agilizar la prueba técnica, simplemente debes hacer un fork del original y trabajar sobre ese:
 - [GitHub - sport-enlace-sas/backend-challenge-base](#)
- **PLANTILLAS EJEMPLO DE DOCUMENTACION,** 🧠 Cree cualquier cosa con miles de plantillas, o incluso los templates disponibles en Notion hay MILES! 😁 (puedes buscar cualquiera gratis, con un tema puntual) Ejemplo, HR.



RECURSOS DE APOYO 🧠 (EJEMPLOS)

Plantilla Wiki de Desarrollo 2024 | Notion ...

Usa esta plantilla para dar a los equipos de desarrollo una única fuente de información para procesos,...

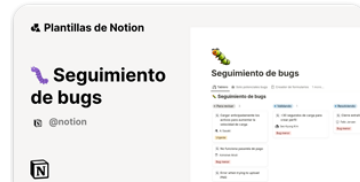
<https://www.notion.so/es-la/templates/engineering-wiki>




Plantilla Seguimiento de bugs 2024 | ...

Permite a las personas que no son parte de la organización informar incidencias, bugs y sugerir mejora...

<https://www.notion.so/es-la/templates/bug-tracker>

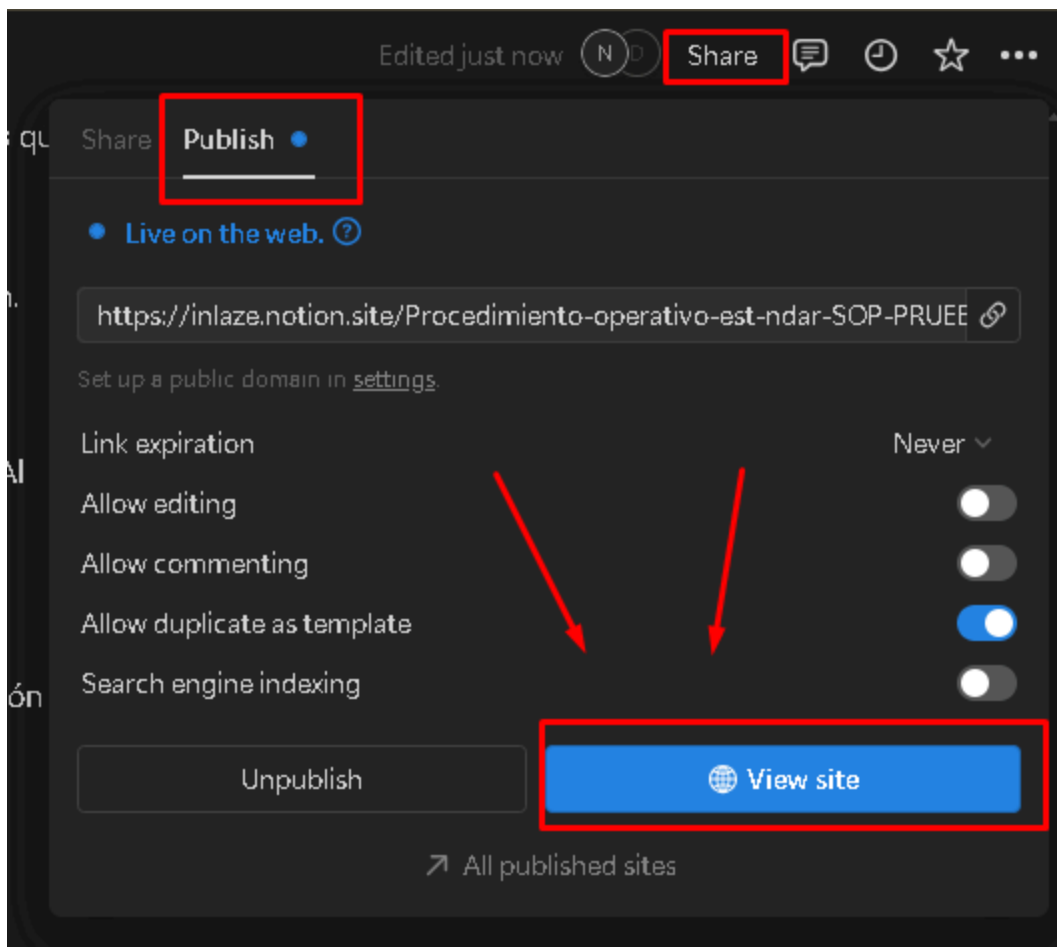


6. Respóndenlos vía email, formato View Site, mediante link live en la nube.

Tienes 24 horas desde la recepción de este email para elaborar y enviarnos de vuelta tu prueba. Responde vía email con un enlace en la nube (por ejemplo, un link live de Notion). 

¡Éxito en tu prueba!

(No archivos locales) 



Repositorio en GitHub:

- Proporciona el código fuente completo de los microservicios, incluyendo los archivos de configuración necesarios.
- Incluir un archivo README con instrucciones claras sobre cómo configurar y ejecutar los microservicios localmente utilizando Docker Compose.
- Puedes utilizar cualquier herramienta de control de versiones y alojar el código en un repositorio público.
- Entregar el código fuente del repositorio (git).