PRUEBA TÉCNICA

Gestión de notas: CRUD con manejo de fechas y tests

AVISO LEGAL

1. Objetivo del Programa

Crear una aplicación que permita a los usuarios gestionar notas de forma eficiente. Cada nota debe incluir un **ID único**, un **título descriptivo**, el **contenido de la nota** y la capacidad de registrar la **fecha de creación o última modificación**.

2. Enunciado

Usando el lenguaje de programación con el que te **sientas más cómodo** (Python, C#, Java, JavaScript, GoLang, etc..), diseña un programa sencillo que permita gestionar notas con las siguientes funcionalidades:

1. **Crear, Leer, Actualizar y Eliminar (CRUD)** notas.

- Cada nota debe tener:
  - Un ID único.
  - Un título.
  - Un contenido.
  - **Una fecha de creación y/o última modificación.**

2. **Almacenamiento de datos**:

Puedes utilizar una base de datos, ya sea **relacional** (por ejemplo, SQLite, MySQL) o **no relacional** (por ejemplo, MongoDB, Firebase).

Sin embargo, si decides no usar una base de datos, puedes **simular** su funcionalidad utilizando **estructuras de datos en memoria**, como arreglos o mapas, para almacenar y gestionar la información temporalmente.

3. **Tests unitarios**:

- Implementa al menos 2 tests unitarios que validen la funcionalidad del CRUD.
  - Por ejemplo: creación de notas, actualización de notas y eliminación de notas.

4. **Extra (Opcional)**:

- Graba un video explicando tu solución (máximo 5 minutos):
  - Qué hiciste.
  - Cómo implementaste el CRUD.
  - Cómo manejaste las fechas.
  - Cómo escribiste los tests.

3. Requisitos

Es importante cumplir con los requisitos principales para la evaluación de tu trabajo.

1. **CRUD con fechas**:

- Implementa las operaciones básicas de un CRUD:
  - **Crear**: Permite agregar una nueva nota con la fecha actual.
  - **Leer**: Lista todas las notas, mostrando:
    - Título.
    - Contenido.
    - Fecha de creación y/o última modificación.
  - **Actualizar**: Modifica una nota existente y actualiza su fecha de última modificación.
  - **Eliminar**: Borra una nota por su ID.

2. **Entrada y salida**:

- El programa no tendrá una interfaz de usuario, sino que funcionará como un **web service**.
- Los usuarios podrán interactuar con el programa enviando **solicitudes HTTP** utilizando herramientas como **Postman** o cualquier cliente HTTP.
- Endpoints sugeridos:
  - **POST /notes**: Crear una nueva nota.
  - **GET /notes**: Listar todas las notas.
  - **GET /notes/{id}**: Leer una nota específica.
  - **PUT /notes/{id}**: Actualizar una nota existente.
  - **DELETE /notes/{id}**: Eliminar una nota.
- Cada endpoint permitirá realizar las operaciones CRUD de manera estructurada y eficiente.

3. **Manejo de fechas**:

- Utiliza un formato claro para las fechas. (dd/MM/yyyy HH:mm).
- Al crear una nota, asigna automáticamente la fecha actual.
- Al actualizar una nota, actualiza la fecha de última modificación.

4. **Tests unitarios**:

- Usa herramientas de pruebas disponibles en el lenguaje.
- Las pruebas pueden verificar:
  - La creación correcta de notas con fecha.
  - La actualización de notas y de la fecha de última modificación.
  - La eliminación de notas.

Importante: Siempre puedes añadir mejoras, nuevas funcionalidades o incorporar algo propio. Valoramos tu **creatividad**, el uso de **buenas prácticas**, técnicas de desarrollo y conocimientos de patrones de diseño. Además, apreciamos tu habilidad para resolver problemas de manera eficiente con **cualquier tecnología** y tu enfoque profesional hacia el trabajo.

4. Extras

Dificultades relacionadas con API y lógica

- **Paginación**: Añadir soporte para la paginación al obtener la lista de notas.
- **Búsqueda y filtrado**: Implementar una búsqueda de notas basada en palabras clave dentro del título o contenido.
- **Ordenación**: Añadir la opción de ordenar las notas por fecha de creación, título u otros parámetros.
- **Encabezados personalizados**: Usar encabezados HTTP para transmitir datos, como el idioma para manejar fechas (localización).
- **Operaciones masivas**: Implementar la capacidad de eliminar o actualizar varias notas en una sola solicitud.
- **Gestión de conflictos**: Implementar un sistema de bloqueo para evitar que varias personas editen la misma nota al mismo tiempo.

Dificultades relacionadas con la seguridad

- **Cifrado de datos**: Almacenar las notas en la base de datos en un formato cifrado.

Dificultades relacionadas con datos y bases de datos

- **Copia de seguridad**: Crear copias de seguridad automáticas de las notas (por ejemplo, exportarlas a un archivo JSON).
- **Migraciones de base de datos**: Añadir soporte para migraciones que faciliten la actualización del esquema de la base de datos.

Dificultades relacionadas con el rendimiento

- **Procesamiento asíncrono/concurrencia**: Implementar la ejecución asíncrona/diferentes hilos de tareas, como la creación de notas mediante una cola de mensajes.

Dificultades relacionadas con la integración

- **Integración con APIs externas**: Añadir la posibilidad de guardar las notas directamente en servicios en la nube, como Google Drive.