Prueba Técnica de .NET para Desarrollador Junior Full Stack

Objetivo

Evaluar las habilidades técnicas de un desarrollador junior para implementar una solución de procesamiento de documentos PDF, integrando backend/servicios en .NET, base de datos SQL Server, y una interfaz web sencilla utilizando HTML/CSS/JS/Bootstrap.

Requisitos

- Fundamentos de desarrollo en C# y ASP.NET.
- comprensión de bases de datos SQL SERVER.
- Comprensión general de servicios web y arquitectura de microservicios.
- Habilidades para resolver problemas y pensar de forma crítica

Duración

La prueba técnica debe completarse en aproximadamente 2 días.

Descripción del Proyecto

Este proyecto implementa una solución modular para el procesamiento automatizado de archivos PDF, organizada en tres componentes principales:

- **Frontend (HTML, CSS, JS, Bootstrap):**
  Permite al usuario subir archivos PDF, gestionar palabras clave y visualizar los registros de procesamiento, mediante una interfaz web simple y clara.
- **Backend (ASP.NET Core API):**
  Maneja las operaciones CRUD sobre las claves y logs, recibe archivos desde el frontend y expone endpoints que también son utilizados por el servicio para registrar las acciones realizadas.
- **Servicio o aplicación Windows form de procesamiento (.NET):**
  Analiza los archivos PDF de una carpeta local, los renombra y clasifica según coincidencias con palabras clave, y envía los resultados al backend mediante API para registrar el log.

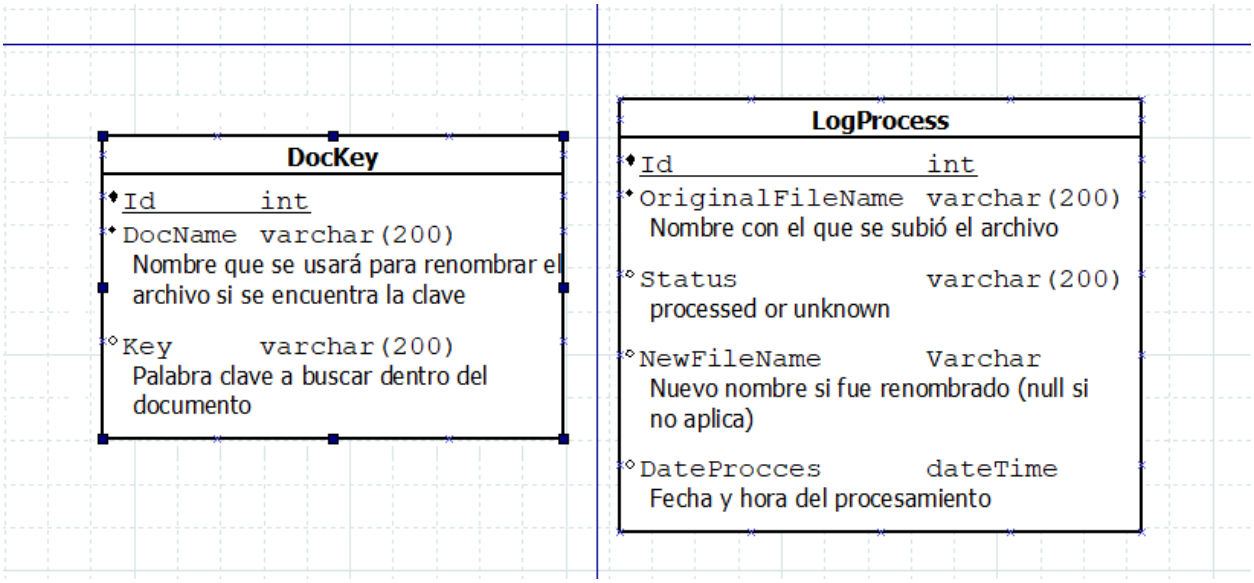
Requisitos Técnicos

1. Base de datos (SQL Server)

- **Crear dos tablas:**
    - Tabla de búsqueda DocKey
        - **Propósito:** Almacena las claves que deben buscarse en los documentos PDF, junto con el nombre que se asignará al archivo si se encuentra una coincidencia.

- **Uso:** Utilizada durante el procesamiento para identificar archivos que contengan palabras clave específicas y renombrarlos en consecuencia.

- Tabla de log de procesamiento:
  - **Uso:** Esta tabla servirá para visualizar el historial desde el frontend.
  - **Propósito:** Registrar un log de cada archivo procesado, incluyendo si fue reconocido, renombrado, y la fecha del proceso.



2. Backend (.NET C#)

a. Crear un servicio o aplicación Windows Form que:

- Se ejecute periódicamente para:
- Leer archivos PDF desde C:\PruebaEQ
- Comparar el contenido con las claves en la tabla DocKey
  - Si hay coincidencia:
    - Renombrar el archivo usando DocName
    - Moverlo a C:\PruebaEQ\OCR
    - Registrar en la tabla LogProcesamiento como "Processed"
  - Si no hay coincidencia:
    - Moverlo a C:\PruebaEQ\UNKNOWN
    - Registrar en la tabla LogProcesamiento como "Unknown"

El sistema debe crear las carpetas necesarias si no existen.

A continuación, se provee una función base que el desarrollador puede adaptar dentro de un servicio para cumplir con el requerimiento. Esta lógica deberá modificarse para recorrer todos los archivos de la carpeta C:\PruebaEQ, consultar las claves desde la base de datos y aplicar la lógica de procesamiento:

```
using iTextSharp.text.pdf; using iTextSharp.text.pdf.parser;
public void OCR()
{
   string nameArchivo = "nombre_archivo_original.pdf";
   string pdfPath = @"C:\PruebaEQ\" + nameArchivo;
   string keyword = "palabra_clave_deDocKey";
   string outputFolder = @"C:\PruebaEQ\OCR\";

   string extractedText = ExtractTextFromPdf(pdfPath);

   if (extractedText.IndexOf(keyword, StringComparison.OrdinalIgnoreCase) >= 0)
   {
      // Palabra clave encontrada, renombrar y mover el archivo
      string newFileName = Path.Combine(outputFolder,
$"Encontrado_{Path.GetFileName(pdfPath)}");
      File.Move(pdfPath, newFileName);

      // Aquí se debe insertar el registro en LogProcess(estado: Processed)
   }
   else
   {
      // Palabra no encontrada, mover archivo a carpeta UKNOW
      string unknownFolder = @"C:\PruebaEQ\ UNKNOWN \";
      File.Move(pdfPath, Path.Combine(unknownFolder, Path.GetFileName(pdfPath)));

      // Aquí se debe insertar el registro en LogProcess(estado: unknown)
   }
}

private static string ExtractTextFromPdf(string pdfPath)
{
   using (PdfReader reader = new PdfReader(pdfPath))
   {
      string text = string.Empty;

      for (int i = 1; i <= reader.NumberOfPages; i++)
      {
         text += PdfTextExtractor.GetTextFromPage(reader, i);
      }

      return text;
```

```
    }
}
```

B. API REST en ASP.NET Core

Exponer endpoints para:

- Subir archivos PDF desde el frontend (guardarlos en C:\PruebaEQ)
- CRUD para claves (DocKey)
- Consulta del log (LogProcesamiento) para ver el estado de cada documento procesado en el front
- Insert de LogProcess , este lo llamara el servicio de procesamiento de pdf para inserta a base de datos.

3. Frontend (HTML, JS, CSS o Bootstrap)

Desarrollar una interfaz amigable que permita al usuario interactuar con el sistema. Funcionalidades clave:

Requisitos funcionales de front-end:

- **Carga de documentos PDF:**
    - Debe incluir un formulario para seleccionar y subir archivos PDF.
    - El sistema solo debe permitir la carga de archivos con extensión .pdf y tamaño máximo de 10 MB.
    - La interfaz debe mostrar mensajes claros de éxito o error ante problemas como archivos no válidos, tamaño excedido, o fallos al guardar.

Administración de claves (DocNamey Clave de la tabla DocKey):

- Formulario para agregar/editar claves (campos: Clave, DocName)
- Listado de claves existentes
- Permitir eliminar claves
- Validar campos obligatorios

Visualización de logs de procesamiento:

- Incluir un botón que permita actualizar y mostrar la información actual de la tabla de logs (LogProcess).
- Presentar los datos en una tabla con columnas como: Nombre original, Estado, Nuevo nombre, Fecha de procesamiento.

Deseable (No obligatorio)

- Implementar autenticación básica con **JWT hacia el API**
- Usar **Stored Procedures** para operaciones de crud con DocKey y LogProcesamiento
- Código organizado y bien documentado

Nota: solamente el proyecto API REST se puede comunicar con la BD.

Evaluación:

- **Correctitud técnica**: Cumplimiento de todos los requisitos funcionales y técnicos
- **Organización del Código:** Claridad, estructura y buenas prácticas de programación
- **Manejo de archivos:** Validación de extensión y tamaño, movimiento entre carpetas
- **Integración frontend-backend** Fluidez entre la interfaz y el servicio
- **CRUD en base de datos:** Creación, edición, eliminación y lectura de claves
- **Registro de logs:** Correcta inserción en la tabla LogProcesamiento
- **Interfaz de usuario:** Claridad, usabilidad y simplicidad del frontend
- **Uso de librerías:** Uso correcto de iTextSharp para OCR de PDFs

Entrega:

Opción 1:

Enviar el código fuente completo del proyecto comprimido en un archivo .zip (o formato similar), incluyendo:

- Proyecto backend (.NET), Proyecto servicio procesamiento de documentos, Proyecto frontend (HTML, JS, CSS/Bootstrap)
- Script de la creación de las tablas y de los Sp de ser necesarios
- Instrucciones claras sobre cómo compilar y ejecutar la aplicación.

Opción 2:

Publicar el proyecto en un repositorio privado o público en GitHub y compartir el enlace de acceso.

El repositorio debe incluir:

- Código fuente completo de frontend y backend y servicio procesamiento de documentos.
- Script de la creación de las tablas y de los Sp de ser necesarios
- Instrucciones claras en el README sobre cómo descargar, compilar y ejecutar el proyecto.