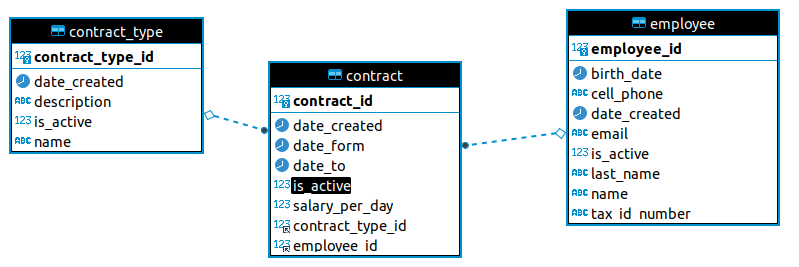
EJERCICIO TÉCNICO DESARROLLADOR JAVA

Grupo Salinas está en un proceso para el desarrollo de una solución de software de Recursos Humanos y está solicitando un primer entregable que tenga como alcance el desarrollo de cinco servicios para la gestión de sus empleados y la generación de sus contratos.

Considere el siguiente schema de base de datos para realizar la implementación y la API correspondiente.



**REQUERIMIENTO**

1. Utilizar java 1.8 de preferencia
2. Crear un proyecto con Spring Framework Spring Boot para generar la API correspondiente al requerimiento expuesto.
3. Utilizar JDBC/JPA para crear la conexión entre base de datos, mapeo del schema y las operaciones hacia cualquiera de los siguientes motores: MySQL u Oracle (a su elección, elegir solo uno).
4. Utilizar Repositorios/Scripts/SPs para realizar las consultas a la base de datos como al igual las inserciones y actualizaciones
5. Generar únicamente los siguientes cinco endpoints:
   1. Obtener el listado de empleados activos extrayendo solo los siguientes campos:
      * Name y LastName (campo concatenado).
      * axIdNumber (RFC).
      * Email.
      * Nombre del tipo de contrato (ContractType.Name) en el que se encuentre actualmente activo. Sino cuenta con contrato activo enviar valor NULL.
      * Fecha inicio del contrato en el que se encuentre actualmente activo (Contract.DateFrom). Si no cuenta con contrato activo enviar valor NULL.
      * Fecha final del contrato en el que se encuentre actualmente vigente (Contract.DateTo), Si no cuenta con contrato activo enviar valor NULL.
      * Salario diario (Contract.SalaryPerDay), si no cuenta con contrato activo enviar valor NULL.
   2. Agregar un contrato nuevo.
      * Si el empleado no tiene un contrato lo agregará con los datos enviados.
      * Si el empleado tiene un contrato actual vigente en la base de datos, deberá dar de baja el contrato actual actualizando la fecha DateTo a la de la operación y posteriormente ingresar el nuevo contrato con los datos enviados. Un empleado no debe tener dos contratos activos.
   3. Agregar un nuevo empleado.
      * Deberá validar que el formato del TaxIdNumber (RFC) sea correcto a través de una regex. (18 posiciones, contenga solo números y letras, contenga fecha de nacimiento 6 posiciones contiguas)

En caso de ser incorrecto no guardará el registro enviando un error.

* + - Debe validar que el TaxIdNumber (RFC) no esté registrado, en caso de existir deberá mandar un error indicando que ya existe el empleado .
  1. Actualizar los datos de un empleado existente a través de su folio.
     + Si el TaxIdNumber (RFC) se modifica, debe validar previamente que el nuevo tenga el formato correcto y que de igual manera no exista.
  2. Eliminar un contrato vigente de un empleado seleccionado.
     + Deberá desactivar un contrato actualizando la fecha DateTo a la actual.
     + Si el empleado no cuenta con un contrato vigente o el proceso no encuentra ninguno deberá mandar un error.

Nota: No te detengas, como mínimo consigue codificar tres endpoint (se considerará valor agregado la generación de los cinco)

1. Documentar a través de OpenAPI (swagger o documento de Word manual) como es que se deben acceder a los endpoints por parte de un cliente (aplicación web, móvil, etc) y las respuestas que arrojará.
   1. Especificar el controller/method-name, verbo a utilizar, header y body.
   2. Especificar todas las respuestas posibles (Ok, Created, Bad Request, Server Error, etc) de cada endpoint con su estructura Json y su código de error
2. Simulación de microservicios,
   1. Generar un Servicio/API (bypass) para ejecutar alguno(s) de los cinco enpoints desarrollados (Valoración: Conocimientos de comunicación entre servicios)

**GENERALES**:

1. Deberá utilizar buenas convenciones del lenguaje Java.
   1. Respetar notaciones camel case, pascal case, etc. para clases y métodos.
   2. Modular el proyecto de tal forma que el código sea legible.
2. Proveer todas las instrucciones para le ejecución correcta de la solución, incluyendo recursos de tablas población de datos (scripts, instalación de paquetes, etc.).
3. Deberá utilizarse una arquitectura N-Layers, a 3 capas como mínimo (Controladores, Negocio y Persistencia de datos) agregar capas de manera justificada dará valor a la prueba.
4. Generar Pruebas Unitarias dará valor a la prueba.
5. Contenerizar la solución dará valor a la prueba

REPOSITORIO:

1. Debe usar un repositorio personal público (GitHub, Bitbucket, Gitlab) o similar, subir el proyecto y enviarnos la URL del repositorio junto con la documentación.
2. Una breve descripción de cómo se estructuró el proyecto, y comentarios de ejecución.
3. En caso de utilizar Docker, generará valor a la prueba. Realizar un “all in one” (Docker Compose)

**Tiempo:**

Duración: 24 horas naturales a partir de su entrega. Si se termina el tiempo entrega lo que se haya finalizado hasta ese momento, no te sientas frustrado si no lograste terminar.