Prueba practica Java Spring OLSoftware

Requerimiento:

Se requiere crear un aplicativo REST ful en lenguaje Java (1.8 o superior) con comunicación a una base de datos, esta aplicación debe permitir realizar consumos y peticiones.

Se requiere un software de inventario, el cual permita el almacenamiento de equipos de cómputo, monitores y otros dispositivos como teclados, mouses, impresoras y bases para monitor o pantallas; debe permitirse la gestión y parametrización de características y datos técnicos del dispositivo.

- El software debe permitir la gestión de inventario para dispositivos (llámese dispositivos a computadores, servidores, monitores, teclados, mouses, etc), este módulo debe permitir consultar, crear, actualizar y eliminar registros de inventario; se debe tener en cuenta los siguientes campos: Nombre (hace referencia a como se identificará el equipo en el inventario, Ejemplo: Equipo_01), Nombre del usuario al cual ha sido asignado (este debe ser usuario creado en el sistema), grupo o área al que pertenece el usuario (ejemplo: infraestructura), Estado del dispositivo (en bodega, en producción, en reparación, etc.), Tipo de dispositivo (si corresponde a un portátil, equipo de mesa, un servidor, etc), fabricante (fabricante del dispositivo: Lenovo, Dell, etc), Modelo del dispositivo (Modelo del dispositivo, ejemplo: MT09ET), número de serie del dispositivo (ejemplo: SN-2997ekJSKAss), número de inventario (número consecutivo que se le asignará al equipo (puede ser un número manual o una secuencia de la base de datos, preferiblemente la segunda opción) y por ultimo un campo de comentarios.
- El software debe permitir la gestión de usuarios, los usuarios deberán tener un identificador, un tipo y número de identificación, primer nombre, segundo nombre (opcional), primer apellido, segundo apellido (opcional), correo electrónico (debe ser único), teléfono, usuario (debe ser único) y contraseña, debe tener asociado un rol (administrador o usuario)
- El software debe permitir la gestión de áreas o grupos a las cuales pertenecen los usuarios, por ejemplo: contabilidad, soporte, etc.
- El software debe permitir la gestión de Estados de los dispositivos y los estados generales del sistema, a estos últimos, solo se permitirá los datos ACTIVO e INACTIVO, para las tablas con las que tenga relación, se asociará A e I respectivamente.
- El software debe permitir la gestión de tipos de dispositivos, aquí se indicarán si son portátiles, servidores, etc.
- El software debe permitir la gestión de Fabricantes de los dispositivos.
- El software debe permitir la gestión de los modelos de los dispositivos, cada modelo puede estar asociado a uno o muchos fabricantes.
- El sistema debe permitir la generación de reportes, los archivos a generar podrán ser CSV y PDF.
- El software debe permitir la gestión de roles, se requiere principalmente un rol administrativo y otro de supervisión, el administrador del sistema podrá realizar la gestión de cada uno de los puntos anteriores, el supervisor solamente podrá consultar la información del inventario (los dispositivos), la demás información será restringida para este.
- Los servicios NO pueden ser consumidos sin autorización, para esto se requiere un implementar un nivel de seguridad en el api.

- Si cree necesaria la creación de entidades adicionales, puede realizar la entrega de estas conservando la misma estructura.
- El sistema deberá integrarse con un servicio externo, el cual traerá datos de consulta, podrá crear y actualizar de datos.

  Para el consumo del sistema externo, deberá primero realizar una petición de login mediante un usuario y contraseña que le será proporcionado al presentar la prueba, este servicio retornará un token bearer jwt con el cual realizará el consumo de los servicios siguientes.
- Se debe implementar pruebas unitarias al proyecto, deberá escoger uno de los CRUDs creados para realizar este punto.

  - https://dev.olsoftware.com:5670/api/aspirantes/login

    Request
    {
     "user": "",
     "password": ""
    }

    Response
    {
     "string authorization token"
    }

  - https://dev.olsoftware.com:5670/api/aspirantes/getAttributes

    Headers
      - Bearer Token

    Request
    {
     "attribute": "" (0,1,2,3,4,5,6,7,8,9)
    }

    Response
    {
     "Object": "",
     "codError":"", (0: Éxito, 1: Error)
     "msjError":"" (La transacción ha sido satisfactoria, Ha ocurrido un error + [traza])
    }

- https://dev.olsoftware.com:5670/api/aspirantes/createAttributes

  Headers

  - Bearer Token

  Request

  {

  “Object”: ”” (Objeto devuelto en el servicio getAttributes)

  }

  Response

  {

  “identidicator”:”” (id identificador de la prueba, copiar este ID en la entrega de la prueba)

  “codError”:””, (0: Éxito, 1: Error)

  “msjError”:”” (La creación del objecto ha sido satisfactoria, Ha ocurrido un error + [traza])

  }

- https://dev.olsoftware.com:5670/api/aspirantes/updateAttributes

  Request

  {

  “Object”: ”” (Objeto devuelto en el servicio getAttributes o creado en el servicio createAttribute)

  }

  Response

  {

  “identidicator”:”” (id identificador de la prueba, copiar este ID en la entrega de la prueba)

  “codError”:””, (0: Éxito, 1: Error)

  “msjError”:”” (La actualización del objeto ha sido satisfactoria, Ha ocurrido un error + [traza])

  }

- https://dev.olsoftware.com:5670/api/aspirantes/getAttribute?identidicator= identidicator (identificador generado al consumir el servicio createAttribute)

  Headers

  - Bearer Token

  Response

  {

  “Object”: “”,

  “codError”:””, (0: Éxito, 1: Error)

  “msjError”:”” (La transacción ha sido satisfactoria, Ha ocurrido un error + [traza])

  }

- https://dev.olsoftware.com:5670/api/aspirantes/deleteAttributes

  Headers

  - Bearer Token

  Request

  {

  “identidicator”:”” (id identificador de la prueba, copiar este ID en la entrega de la prueba)

  }

  Response

  {

  “codError”:””, (0: Éxito, 1: Error)

  “msjError”:”” (La transacción ha sido satisfactoria, Ha ocurrido un error + [traza])

  }

El sistema deberá realizar validaciones a los tipos de datos, los campos numéricos solo podrán recibir datos numéricos, Los campos con datos especiales, tales como correos, teléfonos, identificadores, deberán tener validaciones de datos, es decir, un campo teléfono no podrá guardar caracteres alfanuméricos, solamente numéricos y se deberá realizar validaciones para datos obligatorios.

La arquitectura puede ser planteada por el aspirante, se calificará el uso del framework Spring o Spring boot. el uso de otros frameworks es opcional, también se sugiere de forma opcional el uso de hibernate o Spring boot JPA para el manejo de transacciones a la base de datos.

No hay restricción en cuanto al motor de base de datos, pero se evaluarán los siguientes puntos:

- Debe existir una relación entre las tablas de acuerdo con la necesidad.
- Toda tabla debe tener un identificador único.
- Toda tabla que tenga relación con otra deberá tener su identificador foráneo.
- Obligatoriedad de datos.
- Buen manejo en los tipos de datos.

Criterios de aceptación:

Los siguientes serán los puntos para evaluar y con los cuales se determinará las capacidades técnicas, conceptuales y aptitudinales del aspirante, deben ser entregados los siguientes ítems:

- Carpeta Documentación.
  - Modelo entidad relación.
  - Archivos Json correspondientes a los Request de cada uno de los servicios separados en carpetas por cada Request Maping (este entregable es opcional siempre y cuando se realice la implementación de swagger para documentar las Apis del proyecto).
    - Usuarios
    - Roles
    - Dispositivos
    - Fabricantes
    - Modelos
    - Áreas
    - Estados dispositivos
    - Estados generales
    - Tipos de dispositivos
    - Autorización

    Si cree necesaria la creación de otras entidades adicionales, puede realizar la entrega conservando la misma estructura.

    Los end-points para cada servicio deberá ser llamado /api/inventario/nombre_entidad/nombre_servicio, los servicios principales para evaluar serán:
    - getEntidad (Object)
    - getEntidad (List)
    - createEntidad (Object)
    - updateEntidad (Object)

- deleteEntidad (Object)

Ejemplo:

- /api/inventario/users/getUser
- /api/inventario/users/getUsers
- /api/inventario/users/createUser
- /api/inventario/users/updateUser
- /api/inventario/users/deleteUser

- Carpeta Scripts DB
  - Script de creación de base de datos.
  - Script de inicialización de la base de datos.
  - Script de creación de triggers (opcionales).
- Carpeta Código Fuente.
  - Fuentes Back-end
  - Fuentes Front-end (opcionales).
- Desplegables.
  - Desplegable WAR o JAR (debe tener configurado el pool o Datasource de conexión a la base de datos, si es pool, la conexión debe apuntar a localhost, puerto por defecto de conexión a la base de datos, credenciales de la base de datos creada anteriormente; si es Datasource, debe ser llamado apiTestXXXXXXX, donde las XXXXXXX corresponderá a el nombre del motor de base de datos en Camel Case, Oracle, Postgres, Mysql, etc., para ambos tipos de conexión, el usuario y la contraseña de la base de datos será el mismo nombre del motor de base de datos en minúscula, ejemplo: oracle:oracle).
- Versionamiento de código.

  En la entrega de la prueba, deberá adjuntar el link donde se versionó las fuentes del proyecto, este puede ser GitHub, GitLab o el versionador de preferencia.

Los patrones de diseño, estándares de programación, frameworks adicionales y documentación completa bajo los estándares solicitados, el uso de swagger, serán tenidos en cuenta como puntos adicionales que pueden ayudar al éxito de la prueba.

Como ultima recomendación, validar que las fuentes desplegables funcionen de manera correcta al desplegarse como WAR o JAR, si la aplicación presenta errores en el despliegue, los scripts de base de datos no compilan o generan errores, se perderán puntos en la calificación de la prueba.