PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA
INGENIERÍA DE LAS TELECOMUNICACIONES

SÉPTIMO LABORATORIO

CURSO: GESTIÓN DE SERVICIOS DE TICS [TEL137-H791]

SEMESTRE: 2025-1

Tema: Creación de Web Services RESTful con Spring Boot - Validaciones, Manejo de Errores y Documentación

- Duración: 1 hora y 40 minutos.
- El laboratorio se realizará de manera individual.
- El entregable será el enlace del repositorio remoto del proyecto en GitHub, junto con el script de creación de Base de Datos y un documento que contenga la documentación completa del servicio web desarrollado.
- Está terminantemente prohibido el uso de cualquier tipo de herramienta de inteligencia artificial para el desarrollo del laboratorio.

Recomendación: Leer todo el documento antes de empezar con el desarrollo del laboratorio.

Recientemente, una universidad ha decidido construir un ecosistema de aplicaciones modernas que faciliten su gestión académica. Uno de los primeros pasos es implementar un servicio web centralizado que permita registrar, consultar y administrar los datos de los estudiantes, ya que esta información será consumida por otros sistemas como aplicaciones móviles, etc.

Su misión es desarrollar un servicio web RESTful usando Spring Boot, que permita realizar un CRUD sobre la entidad Estudiante, aplicando buenas prácticas de desarrollo como:

- Validación exhaustiva de los datos de entrada
- Manejo estandarizado de errores
- Devolución de mensajes claros
- Documentación completa de los endpoints expuestos

1. Requerimientos Funcionales

- Creación, edición, consulta y eliminación de estudiantes. Cada estudiante debe contar con los siguientes campos:
  - Nombres (Alfabético)
  - Apellidos (Alfabético)
  - DNI (Numérico, exactamente 8 dígitos, único)
  - Código PUCP (Numérico, exactamente 8 dígitos, único)
  - Fecha de Nacimiento (Fecha válida, debe ser mayor a 16 años)
  - Sexo (‘M’ o ‘F’)

    - Correo Institucional (Formato válido, dominio @pucp.edu.pe)
    - Correo Personal (Formato válido, no debe repetirse con el institucional)
    - Teléfono (Formato de celular nacional, 9 dígitos y empieza con 9)
    - Dirección (Texto libre, máximo 100 caracteres)
    - Departamento (Alfabético)
    - Provincia (Alfabético)
    - Carrera (Alfabético)
    - Fecha de Registro (LocalDateTime, generado automáticamente)
    - Última Actualización (LocalDateTime, generado automáticamente)
    - Estado (Boolean, activo/inactivo)
- Listado de Estudiantes
    - Endpoint que devuelva todos los estudiantes registrados.
        i. Excluir los campos:Fecha de Registro y Última Actualización.
        ii. Traducir el campo Estado: true -> “Activo”, false -> “Inactivo”.
    - Endpoint que permite consultar un estudiante por su DNI.
        i. Traducir el campo Estado: true -> “Activo”, false -> “Inactivo”.
- Registro de Estudiantes
    - Endpoint que permite registrar un estudiante.
        i. Todos los campos deben validarse según su tipo y restricciones.
        ii. No se deben enviar los campos: Fecha de Registro, Última Actualización y Estado.
        iii. El campo Estado se establece como “Activo” por defecto.
        iv. El campo Fecha de Registro se genera automáticamente.
        v. Se deberá asignar null al campo “Última Actualización”.
- Edición de Estudiantes
    - Endpoint para editar los datos de un estudiante.
        i. Sólo los campos que se envíen se deberán de actualizar.
        ii. Queda a criterio del alumno qué campos se pueden modificar.
        iii. No se deben enviar los campos: Fecha de Registro y Última Actualización.
        iv. Se deben aplicar las mismas validaciones que en la creación.
        v. La actualización del campo “Última Actualización” se genera automáticamente.
- Eliminación de Estudiantes
    - Endpoint que permita hacer un borrado lógico de los estudiantes.

2. Documentación del Servicio Web

Todos los endpoints desarrollados deberán estar debidamente documentos, debiendo incluir lo siguiente:

- Descripción clara de cada endpoint.
- Ejemplos de requests, responses y errores comunes. (Adjuntar capturas)
- Detalles sobre los parámetros de entrada y salida.

3. Manejo de excepciones

El servicio debe implementar un manejo de errores que garantice una experiencia consistente y clara para el usuario. Se deberá considerar:

- Mensajes de error personalizados, claros y orientados al usuario.
- Manejo adecuado de excepciones no controladas para evitar errores genéricos del servicio.
- Gestión de errores en la validación de los campos de la entidad.

4. Consideraciones

- Se evaluará el uso adecuado de los códigos de estado HTTP (200 OK, 201 Created, 400 Bad Request, etc.) según el contexto de cada respuesta.
- El servicio debe estar desarrollado siguiendo los principios de un Web Service RESTful, utilizando correctamente los métodos HTTP (GET, POST, PUT, DELETE).
- Las respuestas del servicio deben incluir mensajes claros y estructurados, facilitando su interpretación por el usuario.
- Se debe manejar correctamente las excepciones personalizadas y los errores no controlados
- Su repositorio en GitHub debe contar con 2 ramas: “main” y “develop”.
    - Todo el desarrollo del laboratorio deberá realizarse exclusivamente en la rama “develop”.
    - Por cada funcionalidad o endpoint implementado, deberá realizar un commit.
    - Al finalizar el laboratorio, se deberá generar un “Pull Request” desde la rama “develop” hacia “main”.

4. Entrega:

- Repositorio GitHub: Incluye el enlace a tu repositorio. Este deberá ser nombrado de la siguiente manera: LAB7_H791_20251_XXXXXX, donde ‘XXXXX’ es su código PUCP.
- Script de creación de Base de Datos.
- Archivo PDF con la documentación de su Web Service. Este deberá ser nombrado de la siguiente manera: LAB7_H791_20251_XXXXXX_WS, donde ‘XXXXX’ es su código PUCP.