PROYECTO – EVALUACIÓN 03

- (Pariona Yauricasa,Erick)

IDAT -

Contenido

PARTE TÉCNICA

Competencia General de la Evaluación

Desarrolla aplicaciones web aplicando el patrón MVC con Spring Boot, JPA y Thymeleaf, integrando modelos de datos relacionales, lógica de negocio y vistas dinámicas, para construir soluciones funcionales y mantenibles según buenas prácticas de desarrollo.

Logro de Aprendizaje de la Evaluación

El estudiante implementa un módulo web funcional aplicando Spring Boot, JPA y Thymeleaf, modelando adecuadamente una relación 1 a muchos, desarrollando CRUD completos, validaciones de datos, vistas dinámicas, filtrado de datos, y aplicando el patrón MVC con separación clara de capas; demostrando dominio técnico, organización del proyecto y buenas prácticas.

Capacidades Específicas

- Modela e implementa entidades JPA con relación 1 a N.
- Construye CRUD completos (crear, listar, editar, eliminar lógico).
- Implementa validaciones con Bean Validation.
- Construye vistas Thymeleaf con formularios dinámicos.
- Aplica correctamente la arquitectura MVC.
- Utiliza servicios y repositorios de manera adecuada.
- Implementa filtros y navegación entre entidades.
- Usa Bootstrap 5 correctamente para una UI clara.
- Documenta y estructura el proyecto adecuadamente.
- Desplegar proyectos en Azure.

Dimensión	Indicador	Inicio (0–1 pts)	En Proceso (2–3 pts)	Logrado (4 pts)	Puntaje Máx.
A. Modelado y Base de Datos	Modela adecuadamente la relación 1–N	La relación está incompleta o no funcional; errores en FK o entidades.	Relación implementada pero con errores menores en nombres, mapeos o cascadas.	Implementa correctamente entidades JPA con relación 1–N, llaves foráneas, @OneToMany y @ManyToOne, totalmente funcional.	4 pts
B. Lógica de Negocio y Servicios	Implementa CRUD completos	CRUD incompleto o con fallos graves que impiden uso.	CRUD funcional pero con faltantes (sin eliminar lógico, sin buscar por ID, etc.).	Crea, edita, lista, busca y elimina lógicamente ambas entidades sin errores.	4 pts
C. Interfaz y Presentación (Thymeleaf + Bootstrap)	Construye vistas dinámicas completas	Vistas incompletas, estáticas o no funcionales; errores graves de estructura.	Vistas funcionales pero con fallas estéticas, diseño pobre o errores menores en Thymeleaf.	Vistas bien diseñadas con tablas, formularios, alertas, validaciones y navegación correcta, usando Thymeleaf adecuadamente.	4 pts
D. Validaciones y Manejo de Errores	Valida datos y gestiona errores	Sin validaciones o errores no manejados.	Validaciones parciales o mensajes de error incompletos.	Usa @Valid, BindingResult, mensajes por campo y manejo adecuado de excepciones.	4 pts
E. Calidad del Código y Arquitectura MVC	Organización del proyecto y aplicación del patrón MVC	Proyecto desordenado, mezcla de capas, malas prácticas.	Estructura medianamente adecuada pero con desorganización o duplicación de lógica.	Proyecto bien estructurado: controladores, servicios, repositorios, entidades y templates claramente separados; código limpio y organizado.	4 pts

PARTE DOCUMENTACIÓN

Elemento	Requisito
Tipo de letra	Arial o Calibri
Tamaño de letra	11 o 12 para contenido / 14 para títulos
Interlineado	1.5
Alineación	Justificado
Márgenes	Superior/Inferior 2.5 cm, Izquierdo 3 cm, Derecho 2.5 cm
Numeración	En pie de página
Título de capítulos	En negrita, tamaño 14
Subtítulos	Negrita, tamaño 12
Imágenes / capturas	Claras y legibles, con título debajo
Paginación	Desde la introducción (portada sin numerar)

1. Documento Técnico del Proyecto
Debe incluir:
1.1. Portada
- Nombre del estudiante
- Carrera y ciclo
- Curso y sección
- Docente
- Nombre del proyecto

1.2. Introducción del proyecto
- Breve resumen del sistema desarrollado
- Propósito
- Alcance
- Tecnologías usadas

1.3. Objetivos del proyecto
- Objetivo general
- 2–3 objetivos específicos

2. Modelo de Datos
2.1. Diagrama ER o Modelo Relacional
Debe mostrar:
- Tabla padre
- Tabla hija
- Llave primaria
- Llave foránea
- Cardinalidad 1–N

2.2. Definición de cada tabla
- Campos
- Tipo de dato

- Restricciones (NOT NULL, UNIQUE, FK)
- Estado (A/I)

3. Especificación Técnica del Desarrollo
3.1. Estructura de paquetes
Ejemplo:
controller/
service/
service.impl/
repository/
entity/
templates/
3.2. Descripción de entidades JPA
Incluyendo anotaciones:

- @Entity
- @Table
- @OneToMany
- @ManyToOne
- @Id, @GeneratedValue
- Validaciones (@NotBlank, @Size, etc.)

3.3. Servicios implementados

- Métodos CRUD
- Lógica usada
- Eliminación lógica

3.4. Controladores

- Métodos
- Rutas
- Gestión de validaciones con @Valid y BindingResult

4. Diseño de Interfaces (Thymeleaf)
4.1. Mockups
Puede ser hechos en:

- Figma
- Canva
- Papel

4.2. Vistas finales
Debe incluir capturas de:

- Lista padre
- Lista hijo
- Form nuevo
- Form editar
- Vista detalle o filtrado
- Alertas de éxito/error

5. Manual de Usuario
Debe ser claro y simple:
5.1. ¿Cómo se ejecuta el proyecto?

- Versión de Java
- Comando de ejecución
- Ruta inicial del sistema

5.2. Manual paso a paso
Cómo registrar:

- Un elemento padre
- Un elemento hijo
- Cómo editar
- Cómo eliminar (lógico)

- Cómo ver los hijos por padre

6. Guía de Instalación o Configuración
Debe incluir:
6.1. Dependencias de Maven
Ej.
spring-boot-starter-web
spring-boot-starter-thymeleaf
spring-boot-starter-data-jpa
6.2. application.properties
Conexión H2 o MySQL
6.3. Script SQL (si usa MySQL)

- Creación de tablas
- Llaves primarias y foráneas
- Inserts opcionales

7. Pruebas del Sistema
7.1. Pruebas funcionales
Tabla como esta:

Caso	Acción	Resultado Esperado	Resultado Obtenido	Estado
1	Registrar categoría	Se guarda correctamente	OK	Correcto

7.2. Capturas de las pruebas

8. README Técnico en GITHUB.
Debe incluir:

- Nombre del proyecto
- Requisitos
- Pasos de instalación
- Ejecución
- Capturas del sistema
- Autores

9. Documento de Reflexión
El estudiante responde 3 preguntas:

1. ¿Qué aprendí en este proyecto?
2. ¿Qué dificultades tuve y cómo las resolví?
3. ¿Qué mejoraría si tuviera más tiempo?

RÚBRICA PARTE DOCUMENTACIÓN

Nº	Criterio	Inicio (0 pts)	En Proceso (1–2 pts)	Logrado (máx)	Puntaje Máximo
1	Documento Técnico	Documento incompleto o desordenado; falta coherencia.	Presenta el documento pero faltan 1–2 secciones o está poco claro.	Contiene portada, introducción, contexto, objetivos y tecnologías con claridad.	3 pts
2	Modelo de Datos (ER + descripción)	ER incorrecto o muy incompleto; falta la relación 1–N.	ER correcto pero con omisiones en tipos o descripciones.	ER completo, FK clara, tablas bien descritas con tipos y restricciones.	3 pts
3	Especificación Técnica (MVC)	Explicación mínima o incorrecta de entidades, servicios o controladores.	Estructura MVC documentada parcialmente, faltan detalles o métodos.	Documenta claramente entidades JPA, anotaciones, servicios, repositorios y controladores.	3 pts
4	Diseño de Interfaces (Thymeleaf + Bootstrap)	Sin capturas claras o vistas incompletas.	Vistas básicas pero faltan pantallas o presentan errores.	Interfaces completas, bien diseñadas, formularios y tablas funcionales.	2 pts
5	Manual de Usuario	No presenta manual o está incompleto.	Manual básico, sin capturas o con pasos muy generales.	Manual completo con pasos claros y capturas por pantalla.	2 pts
6	Guía de Instalación y Configuración	Falta de instrucciones o no permite ejecutar la app.	Guía parcial; faltan detalles de BD o ejecución.	Explica instalación, BD, dependencias y ejecución paso a paso.	2 pts
7	Pruebas del Sistema	No incluye pruebas o no son verificables.	Incluye algunas pruebas pero no completas o sin evidencia.	Tabla de casos de prueba + resultados + capturas claras.	2 pts
8	README Técnico	README mínimo o ausente.	README presente pero faltan pasos o rutas.	README profesional con instalación, ejecución, rutas y capturas.	2 pts
9	Capturas de Evidencia	Pocas o ninguna captura del sistema.	Capturas incompletas, faltan pantallas importantes.	Todas las pantallas: CRUD padre, CRUD hijo, validaciones, filtrado.	3 pts
				TOTAL	20 pts

Proyectos

Minimarket – Categorías y Productos
Requerimientos funcionales

RF1.1 Registrar categorías de productos (nombre, descripción, estado).
RF1.2 Listar todas las categorías con opción de filtrar por estado.
RF1.3 Editar una categoría existente.
RF1.4 Realizar eliminación lógica de una categoría (cambiar a inactiva).
RF1.5 Registrar productos asociados a una categoría (precio, stock, estado).
RF1.6 Listar productos por categoría seleccionada.
RF1.7 Editar datos de un producto (precio, stock, estado).
RF1.8 Realizar eliminación lógica de un producto.
RF1.9 Mostrar advertencia cuando el stock de un producto sea menor o igual al mínimo definido.
RF1.10 Buscar productos por nombre dentro de una categoría.

Requerimientos no funcionales (validación)

RNF1.V1 Validar que el nombre de categoría no esté vacío y tenga longitud mínima.
RNF1.V2 Validar que el precio sea numérico y mayor que cero.
RNF1.V3 Validar que el stock sea entero y mayor o igual a cero.
RNF1.V4 No permitir registrar un producto sin una categoría válida.

Academia – Cursos y Temas
Requerimientos funcionales

RF2.1 Registrar cursos (nombre, descripción, estado).
RF2.2 Listar todos los cursos existentes.
RF2.3 Editar información de un curso.
RF2.4 Desactivar un curso sin eliminarlo físicamente.
RF2.5 Registrar temas/unidades asociados a un curso.
RF2.6 Listar los temas de un curso seleccionado.
RF2.7 Editar información de un tema.
RF2.8 Eliminar lógicamente un tema.
RF2.9 Permitir reordenar los temas según un número de secuencia.
RF2.10 Buscar cursos por nombre o parte del nombre.

Requerimientos no funcionales (validación)

RNF2.V1 Validar que el nombre del curso sea único y no vacío.
RNF2.V2 Validar que cada tema tenga título obligatorio.
RNF2.V3 Validar que la secuencia del tema sea un número entero positivo.
RNF2.V4 No permitir registrar temas si el curso está inactivo.

Clínica – Especialidades y Médicos
Requerimientos funcionales

RF3.1 Registrar especialidades médicas (nombre, estado).
RF3.2 Listar todas las especialidades.
RF3.3 Editar especialidades existentes.
RF3.4 Inactivar especialidades sin eliminarlas.
RF3.5 Registrar médicos asociados a una especialidad.

RF3.6 Listar médicos por especialidad.
RF3.7 Editar datos del médico (nombre, CMP, correo, estado).
RF3.8 Inactivar médicos sin eliminarlos.
RF3.9 Buscar médicos por nombre o CMP.
RF3.10 Mostrar en una vista el detalle de la especialidad con todos sus médicos.

Requerimientos no funcionales (validación)

RNF3.V1 Validar que el CMP sea único y obligatorio.
RNF3.V2 Validar formato de correo electrónico del médico.
RNF3.V3 No permitir registrar médico sin especialidad válida.
RNF3.V4 No permitir inactivar una especialidad si todavía tiene médicos activos (opcional según política).

Logística – Proveedores y Productos Suministrados
Requerimientos funcionales

RF4.1 Registrar proveedores (RUC, razón social, contacto, estado).
RF4.2 Listar todos los proveedores.
RF4.3 Editar datos de un proveedor.
RF4.4 Inactivar proveedores sin eliminarlos.
RF4.5 Registrar productos suministrados por un proveedor.
RF4.6 Listar los productos de un proveedor seleccionado.
RF4.7 Editar información de un producto suministrado.
RF4.8 Inactivar productos suministrados.
RF4.9 Buscar proveedores por RUC o razón social.
RF4.10 Filtrar productos por proveedor y estado.

Requerimientos no funcionales (validación)

RNF4.V1 Validar longitud y formato del RUC.
RNF4.V2 Validar que la razón social no esté vacía.
RNF4.V3 No permitir duplicidad de RUC.
RNF4.V4 No permitir registrar producto sin proveedor asociado.

Oficina de Proyectos – Proyectos y Tareas
Requerimientos funcionales

RF5.1 Registrar proyectos (nombre, descripción, fechas, estado).
RF5.2 Listar proyectos con su estado actual.
RF5.3 Editar información del proyecto.
RF5.4 Inactivar proyectos que ya no estén en ejecución.
RF5.5 Registrar tareas asociadas a un proyecto.
RF5.6 Listar tareas de un proyecto seleccionado.
RF5.7 Cambiar estado de una tarea (pendiente, en proceso, completada).
RF5.8 Editar descripción y fechas de las tareas.
RF5.9 Filtrar tareas por estado dentro de un proyecto.
RF5.10 Mostrar resumen de número de tareas por estado para un proyecto.

Requerimientos no funcionales (validación)

RNF5.V1 Validar que la fecha de fin del proyecto no sea anterior a la de inicio.
RNF5.V2 Validar que la tarea tenga nombre obligatorio.
RNF5.V3 Validar que el estado de la tarea sea uno de los valores permitidos.
RNF5.V4 No permitir tareas sin estar vinculadas a un proyecto.

Tienda Online – Clientes y Pedidos
Requerimientos funcionales

RF6.1 Registrar clientes (nombre, documento, correo, estado).
RF6.2 Listar todos los clientes.
RF6.3 Editar información de clientes.
RF6.4 Inactivar clientes sin eliminarlos.
RF6.5 Registrar pedidos asociados a un cliente.
RF6.6 Listar pedidos de un cliente específico.
RF6.7 Asignar estado a los pedidos (registrado, en proceso, entregado, cancelado).
RF6.8 Consultar historial de pedidos de un cliente.
RF6.9 Buscar clientes por nombre o documento.
RF6.10 Mostrar detalle del pedido (fecha, monto, estado, cliente).

Requerimientos no funcionales (validación)

RNF6.V1 Validar formato y longitud del documento de identidad.
RNF6.V2 Validar formato de correo electrónico del cliente.
RNF6.V3 No permitir registrar pedido sin cliente válido.
RNF6.V4 Validar que la fecha del pedido no esté en el futuro (si aplica).

Blog Interno – Autores y Publicaciones
Requerimientos funcionales

RF7.1 Registrar autores (nombre, correo, estado).
RF7.2 Listar todos los autores.
RF7.3 Editar información de autores.
RF7.4 Inactivar autores sin eliminarlos.
RF7.5 Registrar publicaciones asociadas a un autor.
RF7.6 Listar publicaciones por autor.
RF7.7 Cambiar estado de la publicación (borrador, publicada, archivada).
RF7.8 Editar el contenido de una publicación.
RF7.9 Buscar publicaciones por título.
RF7.10 Mostrar detalle de la publicación con su autor.

Requerimientos no funcionales (validación)

RNF7.V1 Validar que el título de la publicación tenga longitud mínima y máxima.
RNF7.V2 Validar formato del correo del autor.
RNF7.V3 No permitir registrar publicación sin autor.
RNF7.V4 Validar que el estado de la publicación sea uno de los valores permitidos.

Biblioteca – Autores y Libros
Requerimientos funcionales

RF8.1 Registrar autores (nombre, nacionalidad opcional, estado).
RF8.2 Listar autores registrados.
RF8.3 Editar datos de autores.
RF8.4 Inactivar autores.
RF8.5 Registrar libros asociados a un autor.
RF8.6 Listar libros por autor.
RF8.7 Editar datos de un libro (título, año, estado).
RF8.8 Buscar libros por título.
RF8.9 Filtrar libros por estado (disponible, prestado, fuera de catálogo).

RF8.10 Mostrar detalle de un libro con información de su autor.

Requerimientos no funcionales (validación)

RNF8.V1 Validar que el nombre del autor no esté vacío.
RNF8.V2 Validar que el año de publicación sea numérico y razonable.
RNF8.V3 No permitir registrar libro sin autor.
RNF8.V4 Validar que el título del libro tenga longitud mínima.

Empresa – Áreas y Empleados
Requerimientos funcionales

RF9.1 Registrar áreas de la empresa (nombre, descripción, estado).
RF9.2 Listar todas las áreas.
RF9.3 Editar información de áreas.
RF9.4 Inactivar áreas.
RF9.5 Registrar empleados asociados a un área.
RF9.6 Listar empleados de un área específica.
RF9.7 Editar datos de empleados (nombre, cargo, estado).
RF9.8 Cambiar de área a un empleado.
RF9.9 Buscar empleados por nombre o documento.
RF9.10 Mostrar resumen de cantidad de empleados por área.

Requerimientos no funcionales (validación)

RNF9.V1 Validar que el nombre del área sea obligatorio y único.
RNF9.V2 Validar que el empleado tenga nombre y documento obligatorios.
RNF9.V3 No permitir registrar empleado sin área.
RNF9.V4 Validar que el documento del empleado cumpla longitud y formato.

Agencia Turística – Destinos y Actividades
Requerimientos funcionales

RF10.1 Registrar destinos turísticos (nombre, descripción, estado).
RF10.2 Listar todos los destinos.
RF10.3 Editar información de destinos.
RF10.4 Inactivar destinos.
RF10.5 Registrar actividades asociadas a un destino.
RF10.6 Listar actividades por destino.
RF10.7 Editar datos de una actividad (nombre, precio, duración, estado).
RF10.8 Inactivar actividades sin eliminarlas.
RF10.9 Buscar destinos por nombre.
RF10.10 Mostrar detalle de un destino con sus actividades.

Requerimientos no funcionales (validación)

RNF10.V1 Validar que el nombre del destino sea obligatorio.
RNF10.V2 Validar que el precio de la actividad sea numérico y mayor que cero.
RNF10.V3 No permitir actividad sin destino.
RNF10.V4 Validar que la duración de la actividad sea un número positivo (horas/minutos).

Transporte – Conductores y Vehículos
Requerimientos funcionales

RF11.1 Registrar conductores (nombre, documento, licencia, estado).
RF11.2 Listar todos los conductores registrados.
RF11.3 Editar la información de un conductor.
RF11.4 Inactivar conductores sin eliminarlos físicamente.
RF11.5 Registrar vehículos asociados a un conductor (placa, modelo, año, estado).
RF11.6 Listar los vehículos de un conductor seleccionado.
RF11.7 Editar los datos de un vehículo (modelo, año, estado, conductor asignado).
RF11.8 Cambiar el conductor asignado a un vehículo.
RF11.9 Buscar conductores por nombre o documento.
RF11.10 Mostrar un resumen del número de vehículos por conductor en una vista.

Requerimientos no funcionales (validación)

RNF11.V1 Validar que el número de licencia del conductor tenga formato y longitud correctos.
RNF11.V2 Validar que la placa del vehículo tenga el formato requerido y sea única.
RNF11.V3 No permitir registrar vehículo sin un conductor válido asignado.
RNF11.V4 Validar que el año del vehículo sea numérico y dentro de un rango razonable (por ejemplo, 1990–año actual).

Colegio – Padres/Apoderados y Estudiantes
Requerimientos funcionales

RF12.1 Registrar apoderados (nombre, documento, teléfono, correo, estado).
RF12.2 Listar todos los apoderados.
RF12.3 Editar la información de un apoderado.
RF12.4 Inactivar apoderados sin eliminarlos.
RF12.5 Registrar estudiantes asociados a un apoderado.
RF12.6 Listar estudiantes de un apoderado seleccionado.
RF12.7 Editar información de estudiantes (nombre, grado, sección, estado).
RF12.8 Cambiar el apoderado principal de un estudiante.
RF12.9 Buscar estudiantes por nombre o código.
RF12.10 Mostrar ficha del estudiante con los datos de su apoderado.

Requerimientos no funcionales (validación)

RNF12.V1 Validar formato y longitud del documento de identidad del apoderado.
RNF12.V2 Validar formato del correo electrónico del apoderado.
RNF12.V3 No permitir registrar estudiante sin apoderado asociado.
RNF12.V4 Validar que el nombre del estudiante y apoderado no estén vacíos.

Restaurante – Recetas y Pasos
Requerimientos funcionales

RF13.1 Registrar recetas (nombre, descripción, tipo de plato, estado).
RF13.2 Listar todas las recetas registradas.
RF13.3 Editar la información de una receta.
RF13.4 Inactivar recetas sin eliminarlas físicamente.
RF13.5 Registrar pasos de preparación asociados a una receta (número de paso, descripción).
RF13.6 Listar los pasos de una receta en orden de secuencia.
RF13.7 Editar la descripción de un paso.

RF13.8 Reordenar los pasos de una receta (cambiar número de paso).
RF13.9 Buscar recetas por nombre o tipo de plato.
RF13.10 Mostrar el detalle de la receta con todos sus pasos ordenados.

Requerimientos no funcionales (validación)

RNF13.V1 Validar que el nombre de la receta sea obligatorio y tenga longitud mínima.
RNF13.V2 Validar que el número de paso sea un entero positivo y no se repita dentro de la misma receta.
RNF13.V3 No permitir registrar pasos sin asociarlos a una receta válida.
RNF13.V4 Validar que la descripción del paso no esté vacía.

Oficina Pública – Tipos de Documentos y Documentos
Requerimientos funcionales

RF14.1 Registrar tipos de documentos (nombre del tipo, descripción, estado).
RF14.2 Listar todos los tipos de documentos.
RF14.3 Editar la información de un tipo de documento.
RF14.4 Inactivar tipos de documentos sin eliminarlos.
RF14.5 Registrar documentos asociados a un tipo de documento (número, asunto, fecha, estado).
RF14.6 Listar documentos por tipo de documento.
RF14.7 Editar datos de un documento (asunto, fecha, estado).
RF14.8 Buscar documentos por número o asunto.
RF14.9 Filtrar documentos por estado (emitido, recibido, archivado).
RF14.10 Mostrar detalle del documento con su tipo correspondiente.

Requerimientos no funcionales (validación)

RNF14.V1 Validar que el nombre del tipo de documento sea obligatorio y único.
RNF14.V2 Validar que el número de documento no esté vacío y pueda ser único según política.
RNF14.V3 No permitir registrar documento sin tipo de documento asociado.
RNF14.V4 Validar formato de fecha de emisión del documento.

Área TI – Categorías de Incidentes e Incidentes
Requerimientos funcionales

RF15.1 Registrar categorías de incidentes (hardware, software, red, etc.).
RF15.2 Listar todas las categorías de incidentes.
RF15.3 Editar una categoría de incidente.
RF15.4 Inactivar categorías que ya no se utilicen.
RF15.5 Registrar incidentes asociados a una categoría (descripción, solicitante, fecha, estado).
RF15.6 Listar incidentes por categoría.
RF15.7 Cambiar el estado de un incidente (registrado, en atención, resuelto, cerrado).
RF15.8 Editar la descripción o datos de un incidente.
RF15.9 Buscar incidentes por palabra clave en la descripción.
RF15.10 Mostrar un resumen del número de incidentes por estado y categoría.

Requerimientos no funcionales (validación)

RNF15.V1 Validar que el nombre de la categoría no esté vacío.
RNF15.V2 Validar que la descripción del incidente tenga longitud mínima.
RNF15.V3 Validar que el estado del incidente sea uno de los permitidos.

RNF15.V4 No permitir registrar incidente sin categoría asociada.

Escuela de Música – Bandas y Canciones
Requerimientos funcionales

RF16.1 Registrar bandas (nombre, género, estado).
RF16.2 Listar todas las bandas.
RF16.3 Editar datos de una banda.
RF16.4 Inactivar bandas sin eliminarlas.
RF16.5 Registrar canciones asociadas a una banda (título, duración, estado).
RF16.6 Listar canciones por banda seleccionada.
RF16.7 Editar la información de una canción (título, duración, estado).
RF16.8 Buscar canciones por título.
RF16.9 Filtrar canciones por estado (activa, inactiva).
RF16.10 Mostrar una vista de detalle de banda con todas sus canciones.

Requerimientos no funcionales (validación)

RNF16.V1 Validar que el nombre de la banda sea obligatorio y único.
RNF16.V2 Validar que el título de la canción no esté vacío.
RNF16.V3 Validar que la duración de la canción sea un número positivo (minutos o segundos).
RNF16.V4 No permitir registrar canción sin banda asociada.

Recursos Digitales – Categorías y Recursos
Requerimientos funcionales

RF17.1 Registrar categorías de recursos digitales (ej: PDF, video, podcast, guía).
RF17.2 Listar todas las categorías de recursos.
RF17.3 Editar datos de una categoría.
RF17.4 Inactivar categorías.
RF17.5 Registrar recursos digitales asociados a una categoría (título, URL o ruta, descripción, estado).
RF17.6 Listar recursos digitales por categoría.
RF17.7 Editar datos de un recurso (título, enlace, descripción).
RF17.8 Buscar recursos por título.
RF17.9 Filtrar recursos por estado (disponible, no disponible).
RF17.10 Mostrar detalle del recurso con su categoría.

Requerimientos no funcionales (validación)

RNF17.V1 Validar que el nombre de la categoría sea obligatorio.
RNF17.V2 Validar que la URL o ruta del recurso tenga un formato válido (http(s):// o ruta interna).
RNF17.V3 Validar que el título del recurso tenga longitud mínima.
RNF17.V4 No permitir registrar recurso sin categoría asociada.

Empresa – Eventos y Participantes
Requerimientos funcionales

RF18.1 Registrar eventos (nombre, descripción, fecha, lugar, estado).
RF18.2 Listar todos los eventos.
RF18.3 Editar la información de un evento.
RF18.4 Inactivar eventos ya realizados o cancelados.

RF18.5 Registrar participantes asociados a un evento (nombre, correo, cargo u organización).
RF18.6 Listar participantes de un evento seleccionado.
RF18.7 Editar información de un participante.
RF18.8 Buscar participantes por nombre dentro de un evento.
RF18.9 Mostrar un resumen del número de participantes por evento.
RF18.10 Permitir marcar la asistencia de participantes (asistió/no asistió).

Requerimientos no funcionales (validación)

RNF18.V1 Validar que el nombre del evento no esté vacío.
RNF18.V2 Validar que la fecha del evento tenga formato correcto y no sea nula.
RNF18.V3 Validar que el nombre y correo del participante sean obligatorios, con formato correcto en el correo.
RNF18.V4 No permitir registrar participante sin evento asociado.

Tienda de Ropa – Colecciones y Prendas
Requerimientos funcionales

RF19.1 Registrar colecciones de ropa (nombre, temporada, año, estado).
RF19.2 Listar todas las colecciones.
RF19.3 Editar información de una colección.
RF19.4 Inactivar colecciones antiguas sin eliminarlas.
RF19.5 Registrar prendas asociadas a una colección (nombre, talla, color, precio, estado).
RF19.6 Listar prendas por colección.
RF19.7 Editar información de una prenda (talla, color, precio, estado).
RF19.8 Buscar prendas por nombre o talla.
RF19.9 Filtrar prendas por estado (disponible, agotada, descontinuada).
RF19.10 Mostrar detalle de una colección con todas sus prendas.

Requerimientos no funcionales (validación)

RNF19.V1 Validar que el nombre de la colección sea obligatorio.
RNF19.V2 Validar que el año de la colección sea numérico y razonable.
RNF19.V3 Validar que el precio de la prenda sea un número mayor que cero.
RNF19.V4 No permitir registrar prenda sin colección asociada.

Tabla de Asignación

Nº	Estudiante	Proyecto Asignado
1	Bazalar Reyes, Julio Cesar	1. Minimarket – Categorías y Productos
2	Callupe Monteblanco, Benjamin Daniel	2. Academia – Cursos y Temas
3	Cancheros Rimache, Blanca Nazareth	3. Clínica – Especialidades y Médicos
4	Carrillo Millones, Alberto Laurencio	4. Logística – Proveedores y Productos Suministrados
5	Castillo Vega, Yefferson Carlos	5. Oficina de Proyectos – Proyectos y Tareas
6	Cossi Cerrañez, Marcelo Fabian	6. Tienda Online – Clientes y Pedidos
7	Enriquez Quispe, Josias Rafael	7. Blog Interno – Autores y Publicaciones
8	Galvez Luyo, Ismael	8. Biblioteca – Autores y Libros
9	Gutierrez Rodriguez de Souza, Marjorie	9. Empresa – Áreas y Empleados
10	Leon Macalapu, Piero Harryson	10. Agencia Turística – Destinos y Actividades
11	Montero Calderon, Anderson Junior	11. Transporte – Conductores y Vehículos
12	Ore Quispe, Angel Ismael	12. Colegio – Padres y Estudiantes
13	Quico Lavado, Hector Paolo	13. Restaurante – Recetas y Pasos
14	Ramirez Landeo, Junior Estuar	14. Oficina Pública – Tipos de Documentos y Documentos
15	Rodriguez Estacio, Daniel Junior	15. Área TI – Categorías de Incidentes e Incidentes
16	Rojas Agipe, Angelo Fabrizio	16. Escuela de Música – Bandas y Canciones
17	Roncal Ruiz, Gabriel Anibal	17. Recursos Digitales – Categorías y Recursos
18	Sanchez Quezada, Miguel Angel	18. Empresa – Eventos y Participantes
19	Santamaria Olivos, Wilder Joseph	19. Tienda de Ropa – Colecciones y Prendas

Modelo de evaluación: Sistema de Gestión de Categorías y Productos (Spring Boot + Thymeleaf, relación 1 a N)

1. Título de la evaluación

"Desarrollo de un módulo web de Gestión de Categorías y Productos usando Spring Boot, JPA y Thymeleaf (Relación 1 a Muchos)"

2. Contexto

Una pequeña tienda necesita registrar sus categorías de productos (por ejemplo: Bebidas, Lácteos, Aseo, etc.) y los productos que pertenecen a cada categoría.
Te piden implementar un módulo web usando Spring Boot + Spring Data JPA + Thymeleaf, donde exista una relación 1 a muchos entre:

- Categoría (1)
- Producto (N)

El sistema debe permitir administrar categorías y productos desde páginas web (no API REST) usando Thymeleaf.

3. Tecnologías obligatorias

- Java 17+
- Spring Boot 3.x
    - spring-boot-starter-web
    - spring-boot-starter-thymeleaf
    - spring-boot-starter-data-jpa
    - spring-boot-starter-validation
- Base de datos: H2 o MySQL (a elección del estudiante)
- Thymeleaf como motor de vistas
- Bootstrap 5 para el diseño básico (CDN es suficiente)
- Maven o Gradle

4. Modelo de datos (tablas y relación 1 a N)

4.1. Tabla categoria (lado 1)
Campos mínimos:

- id_categoria (PK, numérico, autoincremental)
- nombre (varchar(100), no nulo, único)
- descripcion (varchar(255), opcional)
- estado (char(1), no nulo, valores: 'A' = Activo, 'I' = Inactivo)

4.2. Tabla producto (lado N)
Campos mínimos:

- id_producto (PK, numérico, autoincremental)
- nombre (varchar(120), no nulo)
- precio (decimal(10,2), no nulo, > 0)
- stock (int, no nulo, ≥ 0)
- estado (char(1), no nulo, 'A' = Activo, 'I' = Inactivo)
- id_categoria (FK a categoria.id_categoria, no nulo)

4.3. Relación

- Una categoría puede tener muchos productos.
- Cada producto pertenece a una sola categoría.

En JPA debe estar mapeado como:

- Categoria con @OneToMany (lista de productos)
- Producto con @ManyToOne y @JoinColumn(name = "id_categoria")

5. Requerimientos funcionales

5.1. Gestión de Categorías
Debe existir un módulo para:

1. Listar categorías

- URL sugerida: GET /categorias
- Mostrar en una tabla:
  - ID
  - Nombre
  - Estado (mostrar texto “Activo” / “Inactivo”)
  - Botones: Editar, Eliminar (lógico) y Ver Productos

2. Registrar nueva categoría
   - URL sugerida: GET /categorias/nuevo (formulario)
   - URL sugerida: POST /categorias (guardar)
   - Validaciones:
     - nombre: obligatorio, mínimo 3 caracteres
   - Por defecto, estado = 'A'.
3. Editar categoría
   - URL sugerida: GET /categorias/editar/{id}
   - Cargar datos actuales en el formulario.
   - Guardar cambios con POST o PUT (a elección).
4. Eliminar categoría (eliminación lógica)
   - No se debe borrar el registro físicamente.
   - Cambiar estado de 'A' a 'I'.
   - URL sugerida: POST /categorias/{id}/eliminar o GET /categorias/eliminar/{id}.
5. Ver detalle de una categoría con sus productos
   - URL: GET /categorias/{id}
   - Mostrar:
     - Datos de la categoría (nombre, descripción, estado)
     - Tabla con todos los productos asociados (nombre, precio, stock, estado).

5.2. Gestión de Productos

Debe existir un módulo para:

1. Listar productos
   - URL sugerida: GET /productos
   - Mostrar:
     - Nombre
     - Categoría (nombre)
     - Precio
     - Stock
     - Estado
     - Botones: Editar, Eliminar
2. Registrar nuevo producto
   - URL sugerida: GET /productos/nuevo
   - Formulario con:
     - Nombre
     - Precio
     - Stock
     - Categoría (combo <select> que cargue todas las categorías activas desde la BD)
   - URL para guardar: POST /productos
   - Validaciones:
     - nombre: obligatorio
     - precio: obligatorio, numérico, > 0
     - stock: obligatorio, entero ≥ 0
     - categoria: obligatoria
   - Por defecto, estado = 'A'.

3. Editar producto
    - URL sugerida: GET /productos/editar/{id}
    - Permitir cambiar nombre, precio, stock, categoría, estado.
    - Guardar cambios con POST o PUT.
4. Eliminar producto (lógico)
    - Cambiar estado a 'I' sin borrar físicamente.
    - URL: POST /productos/{id}/eliminar o GET /productos/eliminar/{id}.
5. Listar productos filtrados por categoría
    - Desde la lista de categorías, botón “Ver productos” que lleve a:
        - GET /categorias/{id}/productos
    - Mostrar solo los productos de esa categoría.

6. Requerimientos de interfaz (Thymeleaf + Bootstrap)

6.1. Layout general

- Crear una plantilla base layout.html o similar que contenga:
    - <header> con el título del sistema: “Gestión de Categorías y Productos”.
    - Menú de navegación con enlaces:
        - Inicio
        - Categorías
        - Productos
    - <footer> simple con tu nombre y ciclo.
- Usar Thymeleaf Layout (opcional) o fragmentos con th:replace para header y footer.

6.2. Vistas obligatorias

Mínimo:

1. categorias/lista.html
2. categorias/form.html (para nuevo/editar)
3. categorias/detalle.html (categoría + productos)
4. productos/lista.html
5. productos/form.html

En las vistas se debe usar:

- Tablas con clases de Bootstrap (table, table-striped, etc.).
- Botones con clases (btn, btn-primary, btn-danger, etc.).
- Mensajes de éxito o error usando , renderizados desde el controlador (por ejemplo, con RedirectAttributes).

7. Requerimientos técnicos de la aplicación

7.1. Estructura de paquetes sugerida

- com.tuapellido.tienda (root package)
    - controller
        - CategoriaController
        - ProductoController
    - model (o entity)
        - Categoria
        - Producto
    - repository
        - CategoriaRepository
        - ProductoRepository
    - service
        - CategoriaService
        - ProductoService
    - service.impl
        - CategoriaServiceImpl
        - ProductoServiceImpl

        - config (opcional para configuración especial)
        - dto (opcional)

7.2. Capa Repository

- Usar Spring Data JPA:
    - interface CategoriaRepository extends JpaRepository<Categoria, Long>
    - interface ProductoRepository extends JpaRepository<Producto, Long>
- Agregar al menos un método custom:
    - List<Producto> findByCategoriaIdCategoria(Long idCategoria);
    - List<Categoria> findByEstado(String estado);

7.3. Capa Service

- Exponer métodos como:
    - List<Categoria> listarTodas();
    - Categoria guardar(Categoria categoria);
    - Optional<Categoria> buscarPorId(Long id);
    - void eliminarLogico(Long id);
- Similar para Producto.

7.4. Controladores

- Retornar vistas Thymeleaf (no JSON).
- Manejar:
    - Formularios con @ModelAttribute.
    - Validaciones con @Valid y BindingResult.
    - Redirecciones con return "redirect:/categorias";.

8. Validaciones y mensajes

- Utilizar anotaciones de validación en las entidades o en DTOs:
    - @NotBlank, @Size, @Positive, @Min(0), etc.
- Mostrar mensajes de error de validación debajo de cada campo de formulario usando:
    - th:if="${#fields.hasErrors('nombre')}"
    - th:errors="*{nombre}"
- Mostrar mensajes de éxito después de crear/editar/eliminar registros mediante RedirectAttributes:
    - redirectAttributes.addFlashAttribute("success", "Categoría creada correctamente");