Prueba de Desempeño Spring Boot

Nombre: _____

Clan: LOVELACE

Reglas de la prueba

• **Comunicación:** Está prohibido hablar o comunicarse de cualquier forma con otros estudiantes durante el examen.

• **Integridad Académica:** Cualquier forma de trampa, incluido el plagio, copia o uso de material no autorizado, resultará en una calificación de cero en el examen y puede llevar a sanciones adicionales según las políticas de RIWI.

• **Material Permitido:** Solo se permite ver material de apoyo como lo son diapositivas o ejercicios realizados en clase, si se requiere ver estos materiales de apoyo se debe comunicar al Trainer.

• **Permanencia en el Aula:** Una vez iniciado el examen, no se permite salir del aula hasta haber entregado el examen y, de preferencia, hasta que haya transcurrido al menos la mitad del tiempo asignado.

• **Entrega:** Una vez finalizada la prueba se debe subir el entregable en moddle, donde debe estar el código comprimido, más la respectiva documentación y el link del repositorio de GitHub.

Introducción:

RIWI se enfrenta al desafío de mejorar la gestión de contenido multimedia para sus clases virtuales. Para abordar esta problemática, se requiere el desarrollo de un sistema que permita gestionar de manera eficiente las clases, lecciones, materiales multimedia y estudiantes. Este sistema debe ser capaz de relacionar correctamente las clases con las lecciones y materiales multimedia, garantizando así la integridad y coherencia de los datos recopilados.

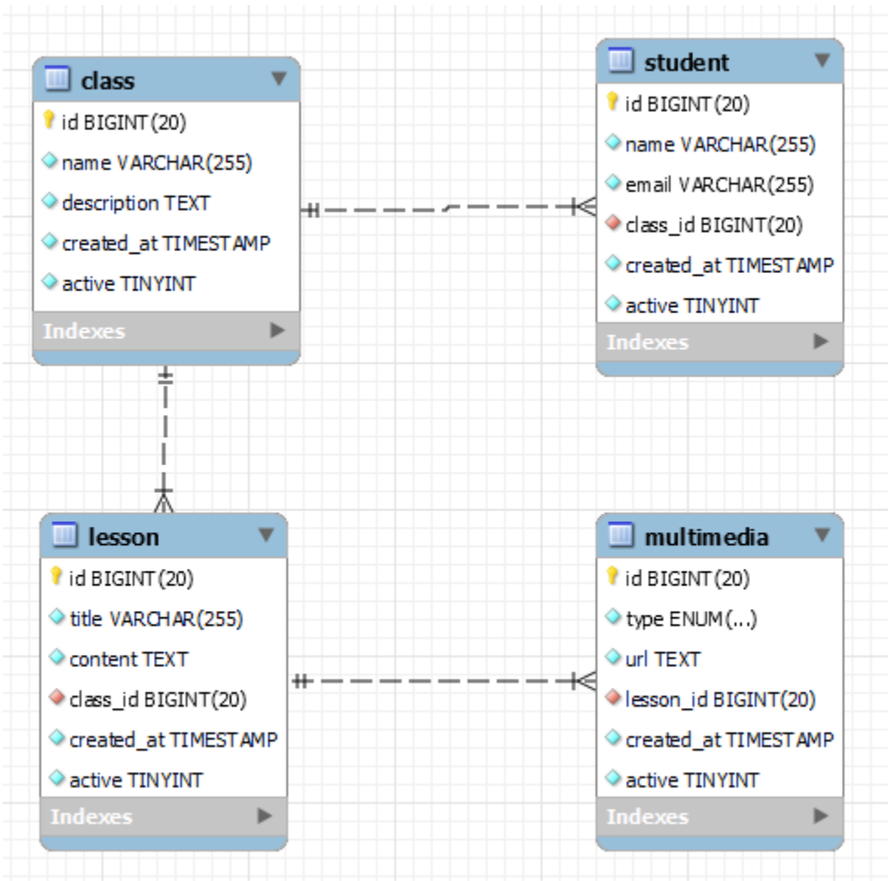
Objetivo:

Desarrollar un sistema de gestión de contenido multimedia para clases virtuales que permita a RIWI almacenar y gestionar de manera eficiente las clases, lecciones y materiales multimedia.

Alcance:

En esta versión del sistema, se guardará y gestionará el catálogo de clases, lecciones y materiales multimedia. También se incluirá un módulo de gestión de estudiantes inscritos en las clases.

Modelo Entidad Relación Propuesto:



Modelo Entidad-Relación Propuesto

Class: Representa una clase en el sistema de gestión de contenido multimedia para clases virtuales.

Lesson: Representa una lección dentro de una clase.

Multimedia: Representa material multimedia asociado a una lección.

Student: Representa a un estudiante registrado en el sistema.

Relaciones entre entidades:

Class y Lesson: Una clase puede tener muchas lecciones (relación uno a muchos), y cada lección pertenece a una sola clase.

Lesson y Multimedia: Una lección puede tener muchos materiales multimedia (relación uno a muchos), y cada material multimedia pertenece a una sola lección.

Class y Student: Una clase puede tener muchos estudiantes (relación uno a muchos), y cada estudiante está inscrito en una sola clase.

Criterios de aceptación generales.

1. Se debe realizar la api con **SpringBoot, Spring JPA y Hibernate.**
2. Todos los endpoints deben estar documentados en **Swagger**.
3. Se debe manejar DTOs para la respuesta y entrada de datos.
4. La información debe venir de la base de datos.
5. Todas las entradas deben tener validaciones para que no afecte a la base de datos.
6. Si no se encuentra algún registro, se debe manejar el error y responde con un mensaje y su respectivo estatus de error, evitando los errores 500.
7. Las relaciones con JPA deben ser bidireccionales.
8. Cada tarea debe se debe realizar en una rama del repositorio de GitHub y todas las ramas deben **unirse** a una rama principal **(main)**, se deben unir mediante pull request.

   Ejemplo de nombre de rama: feature/task-dev-01-endpoint-add-class

   Entregables:

   - Link del repositorio de gitHub (público) donde debe estar el código de la prueba con las respectivas ramas y pull request.
   - Zip comprimido, donde debe estar el código finalizado y funcional, con su respectiva documentación.

Tareas a realizar

Tarea	Entidad	Historia de usuario	Descripción	Path	Método Http	Criterios de aceptación
task-dev-01	Student	Listar dos los estudiantes	Quiero ver la lista de estudiantes registrados en el sistema de forma paginada, y poder buscar por nombre	/api/v1/students	**GET**	Toda la información debe estar paginada, y se debe retornar una lista con solo la información de las clases. La paginación debe ser dinámica,y en el mismo endpoint se debe poder buscar por nombre o descripcion, recibiendo los valores a buscar como request params, **SOLO SE deben RETORNAR DATOS QUE TENGAN EN SU VALOR DE ACTIVE EN TRUE**
task-dev-02	Student	Obtener un estudiante en especifico	Quiero obtener la información de un estudiante por su identificador	/api/v1/students/{id}	**GET**	Se debe retornar una lista con la información de los estudiantes y la información de la clase a la que pertenecen
task-dev-03	Student	Agregar un estudiante	Quiero poder registrar un nuevo estudiante	/api/v1/students	**POST**	Se debe validar que el email sea válido, además de validar los campos que tengan excepciones en la base de datos, si no se cumple estas validaciones se debe responder con el respectivo status, el atributo CreateAt, debe llenarse automaticamente con la fecha y hora de la creación, se debe recibir el identificador de la clase a la que pertenece y validar que está clase si exista.
task-dev-04	Student	Deshabilitar un estudiante	Quiero deshabilitar un estudiante	/api/v1/students/{id}/disable	**PATCH**	Se debe deshabilitar el estudiante (cambiar su atributo active = false) que coincida con el identificador dinámico que se encuentra en la URL. Se debe retornar toda la información del estudiante deshabilitado.
task-dev-05	Student	Actualizar un Estudiante	Quiero poder actualizar un estudiante	/api/v1/students	**PUT**	Se debe validar que el email sea válido, además de validar los campos que tengan excepciones en la base de datos, si no se cumple estas validaciones se debe responder con el respectivo status y mensaje de error.
task-dev-06	Class	Listar todas las clases	Quiero ver la lista de clases registradas en el sistema de forma paginada,poder buscar por nombre y descripción	/api/v1/class	**GET**	Toda la información debe estar paginada, y se debe retornar una lista con solo la información de las clases. La paginación debe ser dinámica,y en el mismo endpoint se debe poder buscar por nombre o descripcion, recibiendo los valores a buscar como request params, **SOLO SE DEBEN RETORNAR DATOS QUE TENGAN EN SU VALOR DE ACTIVE EN TRUE**
task-dev-07	Class	Obtener una clase en especifico	Quiero obtener la información de una clase por su identificador	/api/v1/class/{id}	**GET**	Se debe retornar una lista con la información de la clase que coincida con el identificador, además de todos los estudiante que pertenecen a esta clase
task-dev-08	Class	Agregar una nueva clase	Quiero poder agregar una nueva clase	/api/v1/class	**POST**	Se debe validar la entrada de datos, guardar en la base de datos y responder con la información de la clase guardada
task-dev-09	Lesson	Guardar una lección	Quiero poder guardar una lección con su respectivo contenido multimedia	/api/v1/lessons	**POST**	Se debe pedir como entrada toda la información la lección y validarla, además se debe pedir una lista de objetos que contenga la información de los contenidos multimedias de esa lección, se debe validar que el tipo del contenido multimedia sea **VIDEO, AUDIO, DOCUMENT**, se debe guardar la lección y asociar todo el contenido multimedia a esta lección.
task-dev-10	Lesson	Deshabilitar una lección	Quiero poder deshabilitar una lección, y su contenido multimedia	/api/v1/lessons/{id}/disable	**PATCH**	Se debe deshabilitar la lección que coincida con el identificador que va en la url, y **todo su contenido multimedia** , en caso de que tenga contenido asociado
task-dev-11	Lesson	Obtener una lección en especifico	Quero poder obtener una lección con su respectivo contenido multimedia	/api/v1/lessons/{id}/multimedia	**GET**	Se debe retornar la información de la lección encontrada, la clase a la que pertenece y la lista de contenido multimedia asociada, en caso de que no se encuentre se debe responder con el respectivo error y estatus.
task-dev-12	Lesson	Guardar una lección	Quiero poder guardar una lección con su respectivo contenido multimedia	/api/v1/lessons	**POST**	Se debe pedir como entrada toda la información la lección y validarla, además se debe pedir una lista de objetos que contenga la información de los contenidos multimedias de esa lección, se debe validar que el tipo del contenido multimedia sea **VIDEO, AUDIO, DOCUMENT**, se debe guardar la lección y asociar todo el contenido multimedia a esta lección.

Nombre actividad	**Prueba de desempeño ruta avanzada**
Referencias	N/A
Links	
Descripción	Desarrollar una API funcional que implemente acciones para una entidad específica en un dominio genérico
Fecha de entrega	lunes, 27 de mayo

Alcance	Desarrollar una API RESTful funcional para gestionar una entidad específica en un dominio genérico, aplicando las mejores prácticas de desarrollo de software y asegurando una correcta implementación técnica, manejo de errores, y documentación adecuada.
Evidencias a evaluar	1. Cumplimiento de los criterios de aceptación (Listados uno a uno en el enunciado de la prueba)
	2. Respuestas y Códigos de Estado
	3. Funcionamiento de API
	4. Persistencia de datos
	5. Buenas practicas y código limpio
	6. Sustentación técnica

Rúbrica

Criterios	Código	Valor	Escala de evaluación						Otros parámetros
			0	1	2	3	4	5	
1. Cumplimiento de los criterios de aceptación (Listados uno a uno en el enunciado de la prueba)	C1	15%	El estudiante no cumple ninguno de los criterios de aceptacion generales de la prueba	Cumple algunos criterios básicos, pero no la mayoría.	Cumple la mayoría de los criterios de aceptación , pero hay áreas significativas de incumplimiento.	Cumple la mayoría de los criterios de aceptación especificados en el enunciado.	Cumple todos los criterios de aceptación especificados en el enunciado con algunos detalles menores que pueden mejorarse.	Cumple todos los requisitos técnicos y metodológicos especificados en el enunciado de manera excepcional, demostrando un alto grado de comprensión y aplicación.	n
2. Respuestas y Códigos de Estado	C2	10%	El código no tiene manejo de errores.	El código maneja solo una parte de los errores, y no muestra descripción detallada	El código maneja errores, pero no responde con los estatus de error correspondientes, y sus mensajes	El manejo de errores está presente y cubre la mayoría de los casos, pero los códigos de estado de error no se devuelven correctamente (por ejemplo, no se utilizan los códigos HTTP adecuados). Los mensajes de error son más detallados, pero aún pueden mejorar en claridad y utilidad.	El manejo de errores es robusto y cubre casi todos los casos posibles. Los códigos de estado de error se devuelven correctamente y de acuerdo con las mejores prácticas (por ejemplo, códigos HTTP apropiados para aplicaciones web). Los mensajes de error son claros y útiles, pero pueden carecer de algunos detalles específicos que ayuden en la depuración.	El manejo de errores es exhaustivo y cubre todos los posibles puntos de fallo. Los códigos de estado de error se devuelven correctamente y son consistentes con las mejores prácticas. Los mensajes de error son muy detallados, claros y proporcionan información específica que facilita la depuración y resolución de problemas.	
3. Funcionamiento de API	C3	15%	Los endpoints no responden correctamente o no están implementados. Las respuestas no coinciden con el formato esperado y faltan datos importantes. No hay documentación ni ejemplos de uso.	La mayoría de los endpoints no responden correctamente o no están implementados. Las respuestas no coinciden con el formato esperado y faltan datos importantes. No hay documentación ni ejemplos de uso.	Solo unos pocos endpoints funcionan como se espera, pero muchos presentan errores o no devuelven la información correcta. La documentación es insuficiente y no hay ejemplos claros de uso. Las respuestas a menudo no tienen el formato correcto o carecen de datos cruciales.	La mayoría de los endpoints están implementados y responden correctamente, pero algunos aún presentan problemas menores. Las respuestas están en el formato correcto y contienen los datos necesarios, aunque podrían ser más consistentes. La documentación está presente, pero podría ser más detallada y clara	Todos los endpoints están correctamente implementados y responden conforme a los criterios de aceptación. Las respuestas son consistentes, están bien formateadas y contienen todos los datos necesarios. La documentación es completa y clara, con ejemplos útiles, aunque podría beneficiarse de algunas mejoras menores en detalles y organización.	Cada endpoint está implementado y funciona perfectamente conforme a los criterios de aceptación. Las respuestas son consistentes, bien formateadas y contienen todos los datos necesarios y adicionales que pueden ser útiles. La documentación es exhaustiva, clara y contiene ejemplos detallados para cada endpoint, lo que facilita enormemente su uso por parte de los	
4. Persistencia de datos	C4	10%	No hay ninguna configuración de mapeo para el modelo en el ORM. No existen clases o anotaciones que indiquen la relación con la base de datos.	El modelo tiene alguna configuración básica, pero está incompleto. Algunas propiedades no están mapeadas correctamente y faltan relaciones importantes. Hay problemas significativos que impiden el correcto funcionamiento de la base de datos.	La mayoría de las propiedades están mapeadas, pero hay errores en algunas configuraciones. Las relaciones entre tablas pueden no estar bien definidas. La configuración puede llevar a problemas de integridad de datos y dificultades en consultas complejas.	El modelo está mapeado y funciona para la mayoría de las operaciones, pero hay algunas áreas que necesitan ajustes. Las relaciones y las propiedades están mayormente correctas, pero puede haber inconsistencias menores. La configuración es funcional, pero podría ser más robusta y optimizada.	El modelo está correctamente mapeado y funciona de manera eficiente. Las propiedades y relaciones están bien definidas y soportan la mayoría de las operaciones sin problemas. La configuración es clara y sigue las mejores prácticas, aunque hay espacio para optimizaciones menores.	El modelo está completamente mapeado y optimizado en el ORM. Todas las propiedades y relaciones están correctamente definidas y la configuración es robusta. El modelo soporta todas las operaciones de base de datos de manera eficiente y sigue las mejores prácticas de mapeo. La configuración es clara, bien documentada y no presenta errores.	
5. Buenas practicas y código limpio	C5	10%	El código es desordenado, con mala indentación y falta de separación de responsabilidades. Los nombres de variables y funciones son confusos, no hay manejo adecuado de excepciones, y la lógica de negocio se mezcla con el código de acceso a datos. La documentación es inexistente o inadecuada.	Hay intentos de seguir buenas prácticas, pero prevalecen malas prácticas como nombres de variables y funciones poco claros, indentación inconsistente, y falta de separación de capas.	Se observan algunas buenas prácticas, como nombres de variables y funciones más claros y mejor indentación. Sin embargo, hay inconsistencias en la separación de responsabilidades y la lógica de negocio. El manejo de excepciones y la documentación son limitados. Algunas partes del código aún son confusas y difíciles de seguir.	El código es mayormente claro y bien estructurado, con nombres descriptivos para variables y funciones, buena indentación y comentarios útiles. La separación de responsabilidades es evidente, pero puede haber áreas con lógica compleja o acoplamiento innecesario. El manejo de excepciones es adecuado, pero podría ser más robusto.	El código es claro, bien estructurado y fácil de entender. Los nombres de variables y funciones son descriptivos, la indentación es consistente y los comentarios son claros y útiles. La separación de responsabilidades está bien definida y la lógica de negocio es sencilla y bien organizada.	El código es excepcionalmente claro, bien estructurado y fácil de leer. Todos los nombres de variables y funciones son altamente descriptivos, la indentación es perfecta y los comentarios son completos y útiles. La separación de responsabilidades es impecable. La lógica de negocio es simple y directa, sin redundancias ni complejidades innecesarias. El manejo de excepciones es excelente.El código sigue todas las mejores prácticas de programación backend, facilitando el mantenimiento y la colaboración.	
6. Sustentación técnica	C6	25%	El desarrollador no puede explicar su código. Sus explicaciones son confusas o inexistentes, no puede justificar las decisiones de diseño, y no utiliza terminología técnica correcta. Hay una falta total de claridad y comprensión en la explicación.	El desarrollador tiene dificultades para explicar su código y las decisiones de diseño. La terminología técnica es incorrecta o inadecuada, y las explicaciones son fragmentadas y poco comprensibles. Hay una falta de estructura lógica en la presentación.	El desarrollador puede explicar su código de manera básica, pero las explicaciones son superficiales y carecen de detalles importantes. La terminología técnica es utilizada parcialmente correcta, pero hay inconsistencias. La justificación de las decisiones de diseño es insuficiente.	El desarrollador explica su código de manera clara y lógica, aunque algunos aspectos pueden estar poco detallados. Utiliza la terminología técnica correctamente en su mayoría y puede justificar sus decisiones de diseño, aunque algunas explicaciones pueden ser vagas o incompletas.	El desarrollador explica su código de manera clara y detallada, utilizando correctamente la terminología técnica. Las explicaciones son lógicas y bien estructuradas, y puede justificar la mayoría de sus decisiones de diseño con argumentos sólidos. Hay pocos detalles que podrían mejorarse para una mayor precisión y claridad.	El desarrollador ofrece una explicación excepcionalmente clara y detallada de su código, utilizando la terminología técnica correcta en todo momento. Las explicaciones son lógicas, bien estructuradas y completas, y puede justificar todas sus decisiones con argumentos sólidos y bien fundamentados. La expresión es fluida y profesional, facilitando la	
7. Respuesta a preguntas	C7	15%	El desarrollador no puede proporcionar respuestas relevantes o coherentes a las preguntas. Demuestra una falta total de comprensión del código y de los conceptos subyacentes.	El desarrollador puede responder a algunas preguntas básicas, pero sus respuestas son incompletas, confusas y contienen errores significativos. No demuestra una comprensión adecuada de su propio código.	El desarrollador puede responder a las preguntas básicas de manera adecuada, pero se confunde con preguntas más complejas. Sus respuestas son a menudo superficiales y carecen de profundidad, y comete varios errores al explicar conceptos o justificaciones.	El desarrollador responde correctamente a la mayoría de las preguntas, demostrando una comprensión adecuada de su código. Sin embargo, puede tener dificultades con preguntas más detalladas o avanzadas y algunas respuestas pueden ser incompletas.	El desarrollador responde a casi todas las preguntas de manera precisa y detallada, demostrando una buena comprensión de su código. Puede justificar sus decisiones de diseño y explicar la lógica implementada. Las respuestas son claras y bien fundamentadas, con pocos detalles menores a mejorar.	El desarrollador responde a todas las preguntas de manera precisa, clara y detallada. Demuestra una comprensión profunda y completa de su código, justificando todas las decisiones de diseño y explicando la lógica de manera clara y concisa. Las respuestas son bien estructuradas y muestran un dominio total de los conceptos y prácticas de programación.	