Prueba de Desempeño Spring Boot

Nombre: _____

Clan: Meta

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 precisión y eficacia de la recopilación de datos para ingreso de nuevos coders a través de sus encuestas. Para abordar esta problemática, se requiere el desarrollo de un sistema que permita gestionar de manera eficiente las respuestas de los usuarios, las preguntas formuladas y las encuestas realizadas. Este sistema debe ser capaz de relacionar correctamente las respuestas con los usuarios, las preguntas y las opciones seleccionadas, garantizando así la integridad y coherencia de los datos recopilados.

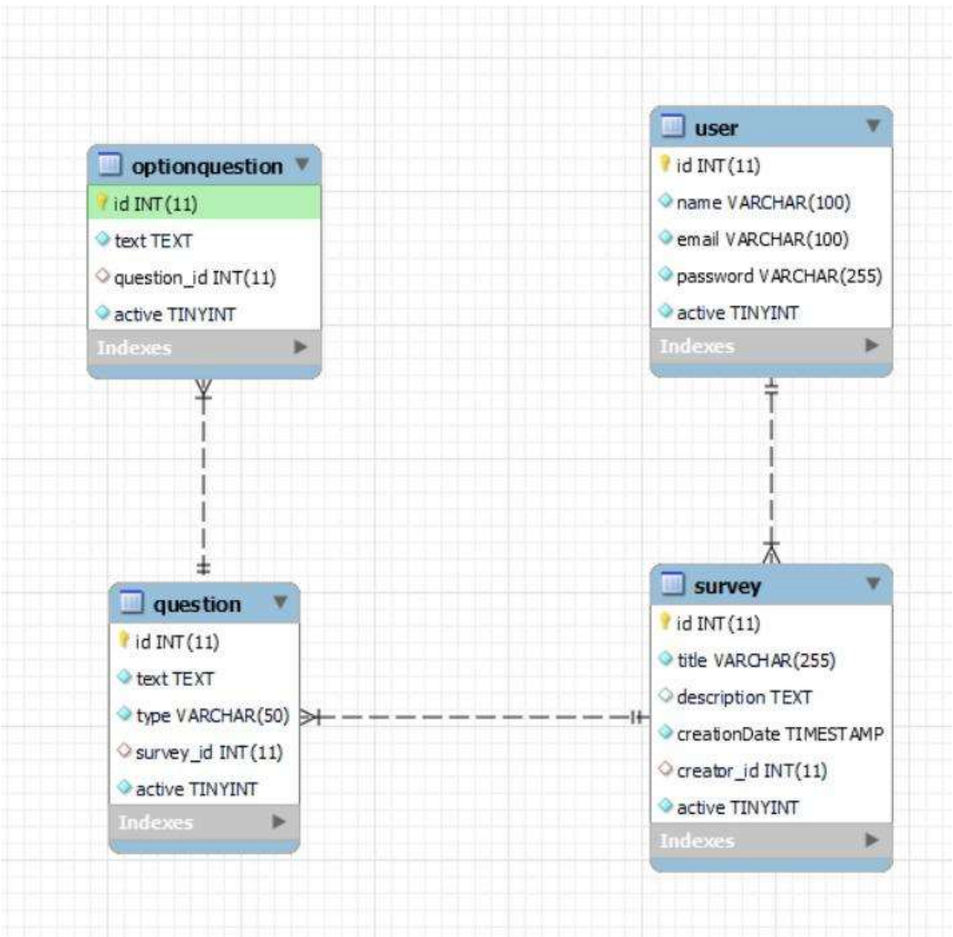
Objetivo:

Desarrollar un sistema de gestión de encuestas que permita a Riwi almacenar y gestionar de manera eficientemente el banco de preguntas de las encuestas, manteniendo la relación entre entre las encuentras, sus preguntas y opciones de respuesta.

Alcance:

En esta versión del sistema, solo se guardará y gestionará el banco de preguntas de las encuestas. Es importante destacar que esta versión no incluirá el módulo de respuestas, actuando únicamente como un repositorio de las encuestas, preguntas y opciones de respuesta.

Modelo Entidad Relación Propuesto:



Relaciones entre entidades

- **User-Survey:** Un usuario puede crear muchas encuestas (uno a muchos), y cada encuesta está asociada a un solo usuario (muchos a uno).
- **Survey-Question:** Una encuesta puede tener muchas preguntas (uno a muchos), y cada pregunta pertenece a una sola encuesta (muchos a uno).
- **Question-OptionQuestion:** Una pregunta puede tener muchas opciones, y cada opción pertenece a una sola pregunta.
- **Question-Answer:** Una pregunta puede tener muchas respuestas, y cada respuesta está asociada a una sola pregunta.
- **OptionQuestion-Answer:** Una opción puede estar seleccionada en muchas respuestas , y cada respuesta puede estar asociada a una sola opción.

Criterios de aceptación generales.

1. Se debe realizar la prueba en **Spring Boot, Spring JPA y Hibernate.**
2. Todos los endpoints deben estar documentados utilizando **Swagger.**
3. Se debe implementar **DTOs** para la respuesta y entrada de datos.
4. La información debe venir de la base de datos **MySQL**, por lo tanto debe existir una conexión.
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 responder con su 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-user

   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	User	Listar todos los usuarios	Quiero ver la lista de usuarios registrados en el sistema de forma paginada	/api/v1/class	GET	Toda la información debe estar paginada, y se debe retornar una lista con la información de las clases menos la contraseña, paginación debe ser dinamica
task-dev-02	User	Obtener un usuario	Quiero obtener la información de un usuario por su identificador	/api/v1/users/{id}	GET	Se debe retornar una lista con la información de los usuarios menos la contraseña, paginación debe ser dinamica
task-dev-03	User	Agregar un usuario	Quiero poder registrar un nuevo usuario	/api/v1/users	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
task-dev-04	User	Actualizar un Usuario	Quiero poder actualizar un nuevo usuario	/api/v1/users	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
task-dev-05	Questions	Agregar una nueva pregunta con sus opciones de respuesta	Quiero poder agregar una nueva pregunta con sus respectivas opciones de respuesta	/api/v1/questions	POST	Se debe pedir la información de la pregunta, asi como una lista con las opciones de respuesta, al momento de guardar la pregunta, se debe guardar primero las opciones de la pregunta, asociarlas y asociar la pregunta con el formulario al que pertenece esto solamente si el tipo de la pregunta es CLOSED, si el tipo es OPEN, entonces no se debe pedir las opciones de pregunta
task-dev-06	Questions	Listar todos las preguntas	Quiero ver toda la lista de preguntas con sus respectivas opciones de respuesta	/api/v1/questions	GET	Toda la información debe estar paginada, y se debe retornar una lista con la información de las preguntas, cada pregunta debe tener un atributos llamado options, y debe tener una lista con sus opciones de respuesta
task-dev-07	Questions	Actualizar una pregunta y sus opciones de respuesta	Quiero ver toda la lista de preguntas con sus respectivas opciones de respuesta	/api/v1/questions/{id}/options	PUT	Se debe pedir la información de la pregunta, asi como una lista con las opciones de respuesta, al momento de guardar la pregunta, se debe guardar primero las opciones de la pregunta, asociarlas y asociar la pregunta con el formulario al que pertenece esto solamente si el tipo de la pregunta es CLOSED, si el tipo es OPEN, entonces no se debe pedir las opciones de pregunta
task-dev-08	Questions	Actualizar una pregunta	Quiero actualizar la pregunta, sin cambiar sus opciones de respuesta	/api/v1/questions/{id}	PATCH	Se debe pedir la información de la pregunta a actualizar, y actualizar la descripción de la pregunta, más no sus opciones de respuesta.
task-dev-09	Questions	Eliminar una pregunta	Quiero eliminar una pregunta y sus opciones	/api/v1/questions/{id}	DELETE	Se debe eliminar la pregunta que coincida con el id que va en la url, ademas se debe eliminar en cascada opciones de pregunta que pertenecian a esta pregunta.
task-dev-10	Survey	Obtener todas las encuestas	Quiero poder obtener todas las encuestas con su informacion de forma paginada y poder filtrar por rango de fechas.	/api/v1/surveys	GET	Toda la información debe estar paginada, y se debe retornar una lista con la información de las encuestas sin las preguntas, paginación debe ser dinamica, Ademas este endpoint debe permitir filtrar entre un rago de fechas.
task-dev-11	Survey	Obtener una encuesta con sus preguntas y opciones de respuesta	Quiero poder obtener una encuesta por el idenetificador con sus preguntas y opciones	/api/v1/surveys/{id}	GET	Se debe poder obtener una encuesta por id, con todas sus preguntas asociadas, ademas de sus respectivas opciones de preguntas si corresponde.

task-dev-12	Survey	Agregar una encuesta	Quiero poder agregar una nueva encuesta	/api/v1/surveys	POST	Se debe poder agregar una nueva encuesta, se deben validar los atributos de entrada, ademas que no pueden existir encuestas con el mismo nombre, ademas de esto se debe **enviar un correo electronico** al usuario que creo la encuesta, confirmando que la creacion fue exitosa.

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.	