Informe de Ejecución del Plan de Pruebas

Proyecto de Pruebas: Pruebas manuales API REST
Plan de Pruebas: Plan de pruebas API REST

Imprimido por TestLink el 20/10/2023

Plan de Pruebas: Plan de pruebas API REST

1. Introducción

1.1 Propósito Garantizar la calidad en las funcionalidades de una API REST realizada con el lenguaje de programación Java y el framework Spring Boot.

1.2 Objetivos del plan

El siguiente plan comprende una serie de procesos que se llevarán a cabo para la correcta verificación y validación del aplicativo, con el objeto de:

• Hallar defectos y fallas en el producto

• Validar las necesidades del usuario

• Verificar los requerimientos

• Determinar y evaluar la calidad del producto

• Describir y especificar los tipos de pruebas que se implementaran

• Definir y describir tareas necesarias para el proceso de pruebas

• Indicar herramientas necesarias para las pruebas

1.3 Alcance del Testing

Este plan de pruebas cubre las principales funcionalidades de la API REST [ARTIST API], incluyendo operaciones CRUD (Crear, Leer Eliminar) considerando aspectos de seguridad, manejo de errores y rendimiento.

1.4 Criterios de Entrada

Para dar inicio a la fase de pruebas, es necesario que se cumplan los siguientes criterios:

• El desarrollador a cargo (Luis Fuentes), ha liberado el proyecto para realizar las pruebas.

• El desarrollador confirma que el entregable se encuentra funcional para el ambiente de pruebas.

• El desarrollador entrega la documentación relacionada con la funcionalidad de la versión del sistema liberado.

• Los requerimientos definidos cumplen con los criterios de salidas especificados

• El Analista QA tiene en su poder todas las herramientas necesarias para ejecutar el sistema a probar.

1.5 Criterios de Salida

Los siguientes criterios deben ser satisfechos para considerar los casos de prueba como pasados:

• No existen fallas, paradas o procesos culminados de forma inesperada.

• El probador ha ejecutado todas las pruebas planeadas para cada ambiente.

• El desarrollador ha resuelto todos los defectos que deben ser depurados.

• Se realizan pruebas de regresión y confirmación.

• El dueño del producto aprueba que el producto satisface las expectativas impuestas por el usuario.

2. Configuración de ambiente de pruebas

2.1 Tecnologías, herramientas y software

• Eclipse IDE for Java Developers v4.29.0

• Java Development Kit 21

• Postman v9.4

• H2 Database v2.2.224

• Spring Data JPA v3.1.4

• Lombok v 1.18.30

• Spring Web v5.3.27

• Spring Boot v3.1.4

2.2 Datos de entrada

• Nombre Artista

• Primer nombre

• Primer Apellido

• Fecha de nacimiento

• Edad actual

3. Pruebas del sistema

Módulos a probar:

• Registro de artistas

• Actualizar registro de artistas por ID

• Eliminar registro de artistas por ID

• Ver todos los registros de artistas

Cada caso de prueba creado debe ser documentado por el Analista QA (acciones a tomar, datos a usar y resultados esperados). La documentación debe ser lo suficientemente clara y concisa para que cualquier otro probador pueda ejecutarlo.

3.1 Estrategias de las pruebas del sistema

Este análisis identificó áreas importantes que están sujetas a fallas o son una prioridad para el negocio. Estas áreas requieren datos de prueba nuevos o actualizados, casos de pruebas, entre otras tareas.

• Preparar conjunto de datos para los requisitos de las pruebas de rendimiento.

• No solo escenarios ficticios sino datos reales que cumplen con la entrada esperada actual.

• Se crearán casos de prueba manuales para la aplicación web que cubrirán cualquier característica funcional.

• Se deben crear datos de prueba para enfatizar cualquier condición compleja requerida por la aplicación.

3.2 Pruebas en Operación Normal

Módulos a probar:

1. Registro de artistas

2. Actualizar registro de artistas por ID

3. Eliminar registro de artistas por ID

4. Ver registro específico con ID

5. Ver todos los registros de artistas

Funcionalidad Módulo 1: Se debe ingresar la URL correspondiente al Endpoint proporcionado por el desarrollador. Al entregar los datos de entrada en el apartado de body (cuerpo) y enviar la solicitud “POST”, se espera que el sistema entregue una respuesta “201”, correspondiente a “Created” (Creado)”.

Funcionalidad Módulo 2: Se debe ingresar la URL correspondiente al Endpoint proporcionado por el desarrollador, sumándole el ID del registro que se desea actualizar. En el apartado body (cuerpo), se realizan los cambios deseados a los datos de entrada. Se procede a enviar la solicitud 5 “PUT” y esperar la respuesta “200” correspondiente a “OK”, es decir, una solicitud exitosa.

Funcionalidad Módulo 3: Se debe ingresar la URL correspondiente al Endpoint proporcionado por el desarrollador, sumándole el ID del registro que se desea eliminar. Se procede a enviar la solicitud “DELETE” y esperar la respuesta “200” correspondiente a “OK”, es decir, una solicitud exitosa.

Funcionalidad Módulo 4: Se debe ingresar la URL correspondiente al Endpoint proporcionado por el desarrollador, sumándole el ID del registro que se desea ver. Se procede a enviar la solicitud “GET” y esperar la respuesta “200” correspondiente a “OK”, es decir, una solicitud exitosa. En el apartado de Response de Postman, se podrá ver los datos del registo especificado. Funcionalidad Módulo 5: Se debe ingresar la URL correspondiente al Endpoint proporcionado por el desarrollador. Se procede a enviar la solicitud “GET” y esperar la respuesta “200” correspondiente a “OK”, es decir, una solicitud exitosa. En el apartado de Response de Postman, se podrá ver una lista con los datos de los registros almacenados.

3.1 Pruebas en Condiciones de Excepción:

Errores de entrada de datos: Enviar datos de entrada incorrectos o mal formateados y verificar cómo la API REST responde.

Plataforma y entorno: Ejecución de pruebas en diferentes entornos de prueba para observar el comportamiento de la API REST de manera consistente en diferentes sistemas operativos y configuraciones.

Seguridad: Observar cómo la API REST responde a diversos ataques simulados de seguridad SQL.

Red y comunicación: Realizar pruebas que simulen interrupciones de red o tiempos de espera agotados al intentar la comunicación con la API REST.

Integración de terceros: Probar la interacción con servicios externos a la API, observando el manejo de respuestas inesperadas o errores de esos servicios externos.

Rendimiento: Ejecutar pruebas de carga que evalúen la respuesta de la API a cargas inesperadas o picos de tráfico.

3.2 Criterios de Éxito de Pruebas:

Respuestas correctas: Los datos obtenidos deben coincidir con los datos esperados según el caso de prueba especificado. Un ejemplo es obtener la lista de registro de artistas al enviar una solicitud GET.

Códigos de estado HTTP: Al obtener las respuestas de las API, estas deben devolver los códigos de estado HTTP esperados para cada solicitud.

Tiempo de respuesta: Las respuestas que entrega la API deben tener un tiempo aceptable.

Cumplimiento de requisitos: La funcionalidad de la API REST se comporta según lo esperado y cumple requisitos funcionales.

Manejo de errores: Se manejan de forma correcta las excepciones y devuelve los mensajes de error descriptivos para casa problema.

3.3 Criterios de Fracaso de Pruebas:

Respuestas incorrectas: Las respuestas de la API no coinciden con los datos esperados o contienen errores.

Códigos de estado HTTP incorrectos: Se obtienen códigos HTTP incorrectos o incoherentes con lo esperado.

Tiempo de respuestas inaceptables: El tiempo de respuesta excede a los límites aceptables o es excesivamente lento.

Errores en casos de prueba: Casos de prueba propuestos fallan, indicando que la API no se comporta según lo esperado.

3.3 Entregables

El documento para observar los resultados de la fase de pruebas, contendrá toda la información recabada al ejecutar las pruebas, considerando posibles arreglos, oportunidades de mejoras a futuro, informes de defectos, áreas críticas del sistema, etc.

4. Tareas

4.1 Actividades

La secuencia de actividades para probar el sistema es:

1. Planificación: Se definen los objetivos de prueba según las expectativas del usuario, considerando los posibles riesgos que se obtengan al probar la API.

2. Monitoreo y control: Ocurre durante todo el cronograma de pruebas, principalmente, para establecer posibles acciones correctivas a la API REST.

3. Análisis de prueba: De acuerdo al nivel de prueba en que se encuentre el aplicativo, se definen las bases de prueba de acuerdo a la especificación de requisitos.

4. Diseño de pruebas: Se priorizan casos de prueba y conjunto de pruebas, se definen los datos necesarios atingentes a la API, se especifica el entorno de prueba que se utilizará y las herramientas necesarias.

5. Implementación de pruebas: Se crean y organizan las suites de prueba de acuerdo a las funcionalidades de la API REST.

6. Ejecución de pruebas: Se ejecutan las suites de prueba especificadas, se registran resultados del funcionamiento de la API, se evalúa si es necesario realizar más pruebas, comparación de resultados, informar defectos y repetir pruebas para verificar correcciones.

7. Compleción de pruebas: Recopilación de datos, evaluando si los resultados cumplen los objetivos propuestos. Evaluar si se requieren más pruebas. Documentar sobre la aceptación del producto.

4.2 Responsabilidades

Responsabilidades del Grupo de Desarrollo

• Ejecutar las pruebas unitarias

• Ejecutar y probar la integración de bajo nivel

• Depuración del código

• Entregar documentación con especificaciones del producto

• Responder inquietudes a probador

Responsabilidades del Grupo de Testing

• Planificar las pruebas del sistema

• Configurar el ambiente de prueba

• Ejecutar las pruebas del sistema

• Reporte de defectos

• Redacción de informes de resultados y mejoras

Suite de Pruebas : Método de petición HTTP POST

Caso de Prueba TC-1: Registro exitoso de artista y álbum [Versión : 1]				
Autor:	franmadariaga			
Resumen: Al ejecutar este caso de prueba, se espera que la solicitud POST registre de forma exitosa la información referente a un artista y su álbum.				
Precondiciones: • Ejecutar proyecto Java/SpringBoot • Establecer conexión a base de datos H2 • Verificar conexión entre backend y base de datos • Probar conexión mediante software "Postman"				
Nº:	Pasos:	Resultados Esperados:	Notas de la ejecución:	Estado de la ejecución:
1	Especificar Endpoint en Postman con path "api/artist" para hacer efectivo el registro			Pasado
2	Establecer método POST en dropdown de Postman			Pasado
3	Configurar Body con formato "JSON"			Pasado

4	Ingresar datos de entrada especificados tanto para artista como para álbum
	
	{ "artistName": "Shakira",
	
	"firstName": "Shakira",
	
	"lastName": "Mebarak",
	
	"birthDate": "1997-02-02",
	
	"age": 46,
	
	"albums": [{
	
	"nameAlbum": " Pies descalzos",
	
	"totalSongs":11,
	
	"albumDuration": 41 }, {
	
	"nameAlbum":
	
	"Grandes éxitos",
	
	"totalSongs": 15,
	
	"albumDuration": 52 }] }
				Pasado

5	Dar click a botón "SEND" para enviar solicitud	Código de status "201 Created". Lo que se traduce	Se obtiene el código especificado para un registro éxitoso.	Pasado

		en que el registro de artista fue exitoso	En el body se puede identificar la asignación de una ID única para el registro.	

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	3.00
Prioridad:	Alta
Detalles de la ejecución	
Build	V 1.0
Tester	franmadariaga
Resultado de la Ejecución:	**Pasado**
Modo de Ejecución:	**Manual**
Duración de le ejecución (min):	
Notas de la Ejecución	El plan de pruebas se lleva a cabo según los resultados esperados. No existen observaciones relevantes.

Caso de Prueba TC-2: Registro fallido por error en path [Versión : 1]

Autor:	franmadariaga

Resumen:

Preparamos la prueba con pasos similares para un registro exitoso, pero en esta ocasión se escribe de forma incorrecta el path especificado para la solicitud POST.

Precondiciones:

- Ejecutar proyecto Java/SpringBoot
- Establecer conexión a base de datos H2
- Verificar conexión entre backend y base de datos
- Probar conexión mediante software "Postman"

Nº:	Pasos:	Resultados Esperados:	Notas de la ejecución:	Estado de la ejecución:
1	Para un registro exitoso se espera el path "api/artist" Se escribe un path distinto al especificado para observar comportamiento de la solicitud			Pasado
2	Establecer método POST en dropdown de Postman			Pasado
3	Configurar Body con formato "JSON"			Pasado
4	Ingresar datos de entrada especificados tanto para artista como para álbum { "artistName": "Shakira", "firstName": "Shakira", "lastName": "Mebarak", "birthDate": "1997-02-02", "age": 46, "albums": [{ "nameAlbum": " Pies descalzos", "totalSongs":11,			Pasado

	"albumDuration": 41 }, { "nameAlbum": "Grandes éxitos", "totalSongs": 15, "albumDuration": 52 }] }			
5	Dar click a botón "SEND" para enviar solicitud	Se espera un código de status "404 Not Found" lo que corresponde a que no existe el recurso solicitado.		Pasado

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	
Prioridad:	Alta
Detalles de la ejecución	
Build	V 1.0
Tester	franmadariaga
Resultado de la Ejecución:	**Pasado**
Modo de Ejecución:	**Manual**
Duración de le ejecución (min):	
Notas de la Ejecución	El comportamiento de la API respecto a ese caso responde a lo esperado

Caso de Prueba TC-3: Registro fallido por ausencia de atributo firstName [Versión : 1]

Autor: franmadariaga

Resumen:

Preparamos la prueba con pasos similares para un registro exitoso, pero en esta ocasión se deja un string vacío en el atributo "firstName"

Precondiciones:

- Ejecutar proyecto Java/SpringBoot
- Establecer conexión a base de datos H2
- Verificar conexión entre backend y base de datos
- Probar conexión mediante software "Postman"

Nº:	Pasos:	Resultados Esperados:	Notas de la ejecución:	Estado de la ejecución:
1	Se especifica el path "api/artist" para el endpoint correspondiente a la solicitud POST			Pasado
2	Establecer método POST en dropdown de Postman			Pasado
3	Configurar Body con formato "JSON"			Pasado
4	Ingresar datos de entrada especificados tanto para artista como para álbum, para el atributo firstName, se asigna un string vacio. { "artistName": "Shakira", "firstName": "", "lastName": "Mebarak",			Pasado

	"birthDate": "1997-02-02",
	"age": 46,
	"albums": [{
	"nameAlbum": " Pies descalzos",
	"totalSongs":11,
	"albumDuration": 41 }, {
	"nameAlbum":
	"Grandes éxitos",
	"totalSongs": 15,
	"albumDuration": 52 }] }			

5	Dar click a botón "SEND" para enviar solicitud	Se espera un código de status "400 Bad Request" lo que corresponde a recepción de datos incorrectos o mal formateados en la solicitud.	La solicitud responde con un código de status "201 Created", lo que significa que el registro es creado incluso si se le entrega un String vacío	Fallado

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	
Prioridad:	Alta
Detalles de la ejecución	
Build	V 1.0
Tester	franmadariaga
Resultado de la Ejecución:	**Fallado**
Modo de Ejecución:	**Manual**
Duración de le ejecución (min):	
Notas de la Ejecución	No existe validación de datos para el atributo "firstName", por lo cual se permite que se registren datos nulos o vacíos.

Caso de Prueba TC-4: Registro fallido por tipo de dato erróneo en atributo firstName [Versión : 1]

Autor:	franmadariaga

Resumen:

Preparamos la prueba con pasos similares para un registro exitoso, pero en esta ocasión se especifica un tipo de dato erróneo (number) al asignado al atributo firstName (String).

Precondiciones:

- Ejecutar proyecto Java/SpringBoot
- Establecer conexión a base de datos H2
- Verificar conexión entre backend y base de datos
- Probar conexión mediante software "Postman"

Nº:	Pasos:	Resultados Esperados:	Notas de la ejecución:	Estado de la ejecución:
1	Especificar Endpoint en Postman con path "api/artist" para hacer efectivo el registro			Pasado
2	Establecer método POST en dropdown de Postman			Pasado

3	Configurar Body con formato "JSON"			Pasado

4	Ingresar datos de entrada especificados tanto para artista como para álbum, para el atributo firstName, se asigna valor de tipo numérico (number).
	{ "artistName": "Shakira",
	"firstName": 2000,
	"lastName": "Mebarak",
	"birthDate": "1997-02-02",
	"age": 46,
	"albums": [{
	"nameAlbum": " Pies descalzos",
	"totalSongs":11,
	"albumDuration": 41 }, {
	"nameAlbum":
	"Grandes éxitos",
	"totalSongs": 15,
	"albumDuration": 52 }] }
				Pasado

5	Dar click a botón "SEND" para enviar solicitud	Se espera un código de status "400 Bad Request" lo que corresponde a recepción de datos incorrectos o mal formateados en la solicitud.	Se obtiene un código de status "201 Created", por ende, se creó el registro aunque el tipo de dato sea erróneo para el campo.	Fallado

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	3.00
Prioridad:	Alta
Detalles de la ejecución	
Build	V 1.0
Tester	franmadariaga
Resultado de la Ejecución:	**Fallado**
Modo de Ejecución:	**Manual**
Duración de le ejecución (min):	
Notas de la Ejecución	No existe validación de datos para el atributo "firstName", por lo cual se registran datos de tipo Number para un campo que requiere String.

Caso de Prueba TC-5: Registro fallido por número excesivo de caracteres en atributo firstName [Versión : 1]

Autor: franmadariaga

Resumen:

Preparamos la prueba con pasos similares para un registro exitoso, pero en esta ocasión el valor asignado al atributo firstName (String) excede con creces el número de caracteres esperado.

Precondiciones:

- Ejecutar proyecto Java/SpringBoot
- Establecer conexión a base de datos H2
- Verificar conexión entre backend y base de datos

- Probar conexión mediante software "Postman"

Nº:	Pasos:	Resultados Esperados:	Notas de la ejecución:	Estado de la ejecución:
1	Se especifica el path "api/artist" para el endpoint correspondiente a la solicitud POST			Pasado
2	Establecer método POST en dropdown de Postman			Pasado
3	Configurar Body con formato "JSON"			Pasado

4	Ingresar datos de entrada especificados tanto para artista como para álbum, para el atributo firstName se asigna un valor string con caracteres excesivos.
	{ "artistName": "Shakira",
	"firstName": "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA",
	"lastName": "Mebarak",
	"birthDate": "1997-02-02",
	"age": 46,
	"albums": [{
	"nameAlbum": " Pies descalzos",
	"totalSongs":11,
	"albumDuration": 41 }, {
	"nameAlbum":
	"Grandes éxitos",
	"totalSongs": 15,
	"albumDuration": 52 }] }
				Pasado

5	Dar click a botón "SEND" para enviar solicitud	Se espera un código de status "400 Bad Request" lo que corresponde a recepción de datos incorrectos o mal formateados en la solicitud.	Se obtiene un código de status "201 Created", por ende, se creó el registro aunque los caracteres ingresados sean excesivos para el campo "firstName"	Fallado

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	3.00
Prioridad:	Alta
Detalles de la ejecución	

Build	V 1.0
Tester	franmadariaga
Resultado de la Ejecución:	**Fallado**
Modo de Ejecución:	**Manual**
Duración de le ejecución (min):	
Notas de la Ejecución	El atributo no cuenta con la validación de datos correspondiente para limitar el número de caracteres que puede ser ingresado.

Suite de Pruebas : Método de petición HTTP GET ALL

Caso de Prueba TC-6: Obtener lista con todos los artistas registrados [Versión : 1]				
Autor:	franmadariaga			
Resumen: Al ejecutar este caso de prueba, se espera que la solicitud GET devuelva una lista de todos los registros de artistas y álbumes almacenados con anterioridad.				
Precondiciones: • Ejecutar proyecto Java/SpringBoot • Establecer conexión a base de datos H2 • Verificar conexión entre backend y base de datos • Probar conexión mediante software "Postman" • Haber creado un registro de artista y álbum previamente				
Nº:	Pasos:	Resultados Esperados:	Notas de la ejecución:	Estado de la ejecución:
1	Especificar Endpoint en Postman con path "api/artist" para la obtención de los registros			Pasado
3	Establecer método GET en dropdown de Postman			Pasado
4	Dar click a botón "SEND" para enviar solicitud	Código de status "200 OK". Lo que se traduce en que la solicitud fue recepcionada con éxito y se observa una lista con los registros en el apartado de "Body".	Se obtiene un código de status"200 OK" ya que la solicitud fue exitosa y se traen los recursos correspondientes como lista	Pasado
5	Observar lista de registros en apartado "Body"		Los datos son traídos correctamente	Pasado
Tipo de ejecución:	Manual			
Duración estimada de la ejec. (min):	3.00			
Prioridad:	Alta			
Detalles de la ejecución				
Build	V 1.0			
Tester	franmadariaga			
Resultado de la Ejecución:	**Pasado**			
Modo de Ejecución:	**Manual**			
Duración de le ejecución (min):				
Notas de la Ejecución	La ejecución corresponde a lo esperado			

Caso de Prueba TC-7: Obtención fallida de registros por error en path [Versión : 1]

Autor:	franmadariaga

Resumen:

Se prepara la prueba con pasos similares para una obtención de registros exitosa, pero en esta ocasión se escribe de forma incorrecta el path especificado para la solicitud GET.

Precondiciones:

- Ejecutar proyecto Java/SpringBoot
- Establecer conexión a base de datos H2
- Verificar conexión entre backend y base de datos
- Probar conexión mediante software "Postman"

Nº:	Pasos:	Resultados Esperados:	Notas de la ejecución:	Estado de la ejecución:
1	El endpoint esperado para el path del método GET es "api/artist" Se modifica agregándolo de la siguiente manera "api/artists"			Pasado
3	Establecer método GET en dropdown de Postman			Pasado
4	Dar click a botón "SEND" para enviar solicitud	Se espera un código de status "404 Not Found" lo que corresponde a que no existe el recurso solicitado.	Se obtiene el código de status "404 Not Found" debido a que el path se encuentra erróneo	Pasado
5	Observar resultados obtenidos en apartado "Body"	No se obtienen resultados en el apartado "Body".		Pasado

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	3.00
Prioridad:	Alta

Detalles de la ejecución	
Build	V 1.0
Tester	franmadariaga
Resultado de la Ejecución:	**Pasado**
Modo de Ejecución:	**Manual**
Duración de le ejecución (min):	
Notas de la Ejecución	La ejecución es llevada a cabo según lo esperado

Suite de Pruebas : Método de petición HTTP GET por ID

Caso de Prueba TC-8: Obtener registro específico proporcionando ID [Versión : 1]				
Autor:	franmadariaga			
Resumen: Al ejecutar este caso de prueba, se espera que la solicitud GET devuelva un registro en específico al proporcionar el ID en el path.				
Precondiciones: • Ejecutar proyecto Java/SpringBoot • Establecer conexión a base de datos H2 • Verificar conexión entre backend y base de datos • Probar conexión mediante software "Postman" • Haber creado un registro de artista y álbum previamente				
Nº:	Pasos:	Resultados Esperados:	Notas de la ejecución:	Estado de la ejecución:
1	Especificar Endpoint en Postman con path "api/artist/{id}" para la obtención del registro solicitado			Pasado
2	Establecer método GET en dropdown de Postman			Pasado
3	Dar click a botón "SEND" para enviar solicitud	Código de status "200 OK". Lo que se traduce en que la solicitud fue recepcionada con éxito y se observa el registro esperado.	Se obtiene un código de status "200 OK", ya que el recurso solicitado fue encontrado y traído en el body	Pasado
4	Observar registro en apartado "Body" y verificar que corresponda al ID especificado			Pasado
Tipo de ejecución:	Manual			
Duración estimada de la ejec. (min):	3.00			
Prioridad:	Alta			
Detalles de la ejecución				
Build	V 1.0			
Tester	franmadariaga			
Resultado de la Ejecución:	**Pasado**			
Modo de Ejecución:	**Manual**			
Duración de le ejecución (min):				
Notas de la Ejecución	La ejecución se lleva a cabo según lo esperado			

Caso de Prueba TC-9: Obtención fallida de registro con ID por path erróneo [Versión : 1]	
Autor:	franmadariaga
Resumen: Al ejecutar este caso de prueba, se espera que la solicitud sea denegada debido a un error en la escritura del path correspondiente a GET con ID.	

Precondiciones:

- Ejecutar proyecto Java/SpringBoot
- Establecer conexión a base de datos H2
- Verificar conexión entre backend y base de datos
- Probar conexión mediante software "Postman"

Nº:	Pasos:	Resultados Esperados:	Notas de la ejecución:	Estado de la ejecución:
1	Especificar Endpoint en Postman con path erróneo "api/artists/{id}" para la obtención de un registro fallido			Pasado
2	Establecer método GET en dropdown de Postman			Pasado
3	Dar click a botón "SEND" para enviar solicitud	Se espera un código de status "404 Not Found" lo que corresponde a que no existe el recurso solicitado	Se obtiene un código de status "404 Not Found" debido a que el path especificado es erróneo	Pasado
4	Observar registro en apartado "Body" y verificar que no traiga el registro especificado			Pasado

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	3.00
Prioridad:	Alta

Detalles de la ejecución	
Build	V 1.0
Tester	franmadariaga
Resultado de la Ejecución:	**Pasado**
Modo de Ejecución:	**Manual**
Duración de le ejecución (min):	
Notas de la Ejecución	La ejecución procede según los resultados esperados

Caso de Prueba TC-12: Verificar eliminación de registro por ID [Versión : 1]

Autor:	franmadariaga

Resumen:

Se espera que al haber eliminado un registro con el método POST, este no sea encontrado al obtener su ID con el método GET.

Precondiciones:

- Ejecutar proyecto Java/SpringBoot
- Establecer conexión a base de datos H2
- Verificar conexión entre backend y base de datos
- Probar conexión mediante software "Postman"
- Haber eliminado un registro de artista y álbum previamente

Nº:	Pasos:	Resultados Esperados:	Notas de la ejecución:	Estado de la ejecución:
1	Especificar Endpoint en Postman con path "api/artist/{id}" del registro eliminado			Pasado

2	Especificar Endpoint en Postman con path "api/artist/{id}" del registro eliminado			Pasado
3	Establecer método GET en dropdown de Postman			Pasado
4	Dar click a botón "SEND" para enviar solicitud	Código de status "404 Not Found". Lo que se traduce en que el recurso solicitado no existe	Se obtiene un código de status "500 Internal Server Error", acompañado de un mensaje que especifica que el valor no está presente	Pasado

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	3.00
Prioridad:	Alta

Detalles de la ejecución	
Build	V 1.0
Tester	franmadariaga
Resultado de la Ejecución:	**Pasado**
Modo de Ejecución:	**Manual**
Duración de le ejecución (min):	
Notas de la Ejecución	Aunque el código esperado no sea el mismo, el recurso especificado no se encuentra disponible después de su eliminación

Suite de Pruebas : Método de petición HTTP PUT

Caso de Prueba TC-10: Actualización exitosa de información de registro [Versión : 1]				
Autor:	franmadariaga			
Resumen: Al ejecutar este caso de prueba, se espera que la solicitud PUT actualice de forma exitosa la información de un registro previamente creado.				
Precondiciones: • Ejecutar proyecto Java/SpringBoot • Establecer conexión a base de datos H2 • Verificar conexión entre backend y base de datos • Probar conexión mediante software "Postman" • Haber creado un registro de artista y álbum previamente				
Nº:	Pasos:	Resultados Esperados:	Notas de la ejecución:	Estado de la ejecución:
1	Especificar Endpoint en Postman con path "api/artist/{id}" para hacer efectiva la actualización de información			Pasado
2	Establecer método PUT en dropdown de Postman			Pasado
3	Configurar Body con formato "JSON"			Pasado

4	Ingresar datos de entrada especificados tanto para artista como para álbum, se actualiza atributos nameAlbum y totalSongs
	
	{ "artistName": "Shakira",
	
	"firstName": "Shakira",
	
	"lastName": "Mebarak",
	
	"birthDate": "1997-02-02",
	
	"age": 46,
	
	"albums": [{
	
	"nameAlbum": "La Tortura",
	
	"totalSongs":8,
	
	"albumDuration": 36 }, {
	
	"nameAlbum":
	
	"Grandes éxitos",
	
	"totalSongs": 12,
	
	"albumDuration": 55 }] }
				Pasado

5	Dar click a botón "SEND" para enviar solicitud	Código de status "200 OK". Lo que se traduce en que la solicitud fue exitosa	Se obtiene un código de status "405 Method Not Allowed", acompañado de un mensaje que refiere que el método PUT no está siendo soportado	Fallado
Tipo de ejecución:	Manual			
Duración estimada de la ejec. (min):	3.00			
Prioridad:	Alta			
Detalles de la ejecución				
Build	V 1.0			
Tester	franmadariaga			
Resultado de la Ejecución:	**Fallado**			
Modo de Ejecución:	**Manual**			
Duración de le ejecución (min):				
Notas de la Ejecución	No es posible actualizar los datos del registro al especificar el ID			

Suite de Pruebas : Método de petición HTTP DELETE

Caso de Prueba TC-11: Eliminar registro por ID [Versión : 1]				
Autor:	franmadariaga			
Resumen: Al ejecutar este caso de prueba, se espera que la solicitud DELETE elimine de forma exitosa un registro especificado por ID.				
Precondiciones: • Ejecutar proyecto Java/SpringBoot • Establecer conexión a base de datos H2 • Verificar conexión entre backend y base de datos • Probar conexión mediante software "Postman" • Haber creado un registro de artista y álbum previamente				
Nº:	Pasos:	Resultados Esperados:	Notas de la ejecución:	Estado de la ejecución:
1	Especificar Endpoint en Postman con path "api/artist/{id}" para su eliminación			Pasado
2	Establecer método DELETE en dropdown de Postman			Pasado
3	Dar click a botón "SEND" para enviar solicitud	Código de status "200 OK". Lo que se traduce en que la solicitud fue recepcionada con éxito y el registro fue eliminado	El código de status obtenido es "200 OK", puesto que el recurso se elimina correctamente.	Pasado
Tipo de ejecución:	Manual			
Duración estimada de la ejec. (min):	3.00			
Prioridad:	Alta			
Detalles de la ejecución				
Build	V 1.0			
Tester	franmadariaga			
Resultado de la Ejecución:	**Pasado**			
Modo de Ejecución:	**Manual**			
Duración de le ejecución (min):				
Notas de la Ejecución	La ejecución se logra según los resultados esperados.			