Informe de Diseño del Plan de Pruebas

Proyecto de Pruebas: PFT-DZ-GRUPO06-2025
Plan de Pruebas: Sistema ASUR - Versión 1.0
Suite de Pruebas: Gestión Institucional de ASUR - API

Imprimido por TestLink el 21/08/2026

2012 © TestLink Community

Plan de Pruebas: Sistema ASUR - Versión 1.0

El presente plan de pruebas tiene como objetivo validar el correcto funcionamiento e integración de los diferentes componentes del sistema institucional ASUR, garantizando que los requerimientos funcionales definidos en la documentación del proyecto sean cumplidos.

El plan abarca todas las pruebas correspondientes a las suites principales del sistema:

- **Suite de Pruebas de la Aplicación Web:** enfocada en la verificación de la interfaz de usuario, la navegación, la validación de datos y los flujos funcionales principales del sistema.
- **Suite de Pruebas de los Servicios del Sistema (API/Backend):** orientada a comprobar la lógica de negocio, la correcta comunicación entre los módulos, el manejo de errores y los códigos de respuesta HTTP.

Suite de Pruebas : Gestión Institucional de ASUR - API

Casos destinados a validar los endpoints del backend en Spring Boot: autenticación, CRUD y manejo de errores.

Las pruebas se ejecutarán mediante colecciones organizadas en Postman, estructuradas por módulos funcionales (Login, Usuarios, Actividades, Espacios, Auditoría, Seguridad). Cada colección incluye los endpoints correspondientes con sus cuerpos JSON de request cuando aplique y casos de prueba para escenarios positivos y negativos.

Suite de Pruebas : Gestión de Usuarios

Conjunto de pruebas diseñado para validar el correcto funcionamiento de todos los endpoints del módulo de usuarios, garantizando la integridad de las operaciones CRUD, seguridad y rendimiento de las APIs.

Ejemplos de parámetros válidos para los campos de Gestión de Usuario:

- Primer Nombre: Juan
- Segundo Nombre: Andres
- Primer Apellido: Rodriguez
- Segundo Apellido: Garcia
- Tipo de Documento: opción “CI”
- Número de Documento: 65483687
- Fecha de Nacimiento: 1998-03-14
- Email: gabriel@gmail.com
- Contraseña: password123!
- Calle: 25 de agosto
- Numero de Puerta: 2
- Numero de Apartamento: 2B
- Bis: True
- Codigo Postal: 97000
- Id de Perfil: 1 (Admin) | 2 (Socio)
- Id de Ciudad: 1
- Telefonos: [092320240, 093256765]

Datos Adicionales de Socio

- Usa Lengua de Señas: True | False
- Tiene Dificultad Auditiva: False | True
- Tiene Subcomusión: True
- Descripción de Subcomisión: Organización de torneos y actividades deportivas

Ejemplos para parámetros inválidos:

Nombre y Apellidos:

- Primer Nombre: "Ju%n" | "" | "J" | "Juan123" | "A" * 100
- Segundo Nombre: "4ndres" | "Andrés@123" | "A" * 100
- Primer Apellido: "Rodrigue2" | "R0dr1gu3z" | "" | "A" * 100
- Segundo Apellido: "Garc|a" | "García*" | "A" * 100
- Tipo de Documento: opción “cédula” | "" | "ID"
- Número de Documento: 65487650 | 1 | 123456789012345 | "ABC123" | null
- Fecha de Nacimiento: "2025-03-14" | "2018-01-01" | "2005-03-14" | ""
- Email: "gabriel@.com" | "usuario" | "@dominio.com" | "a" * 100 + "@gmail.com"
- Contraseña: "123!" | "password" | "12345678" | "" | "a" * 7 | "A" * 8 | "1" * 8
- Calle: "" | "A" * 201 | "Calle%%%123"
- Numero de Puerta: "" | 0 | -1 | 99999 | "ABC"
- Numero de Apartamento: "2%" | "A" * 21 | "Apt-1#"
- Bis: 1 | "true" | "yes" | 2
- Codigo Postal: "code" | ""
- Id de Perfil: 6 | 0 | -1 | "perfil" | null
- Id de Ciudad: 10 | 0 | -5 | "montevideo" | null
- Telefonos: [09232, 093256] | [091] | [123456789012345] | ["abc"] | [] | null

Datos Adicionales de Socio

- Usa Lengua de Señas: "verdadero" | 1 | "si"
- Tiene Dificultad Auditiva: "falso" | 0 | "no"
- Tiene Subcomusión: True con Descripción: null | "" | " " | "A" * 501
- Descripción de Subcomisión: False con Descripción: "Alguna descripción"

Nota: Tiene Subcomusión: True y Descripción de Subcomisión: null daran error, ya que si se tiene subcomisión se debe describir la misma.

Suite de Pruebas : Registro de Usuario

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint registro de usuario.

Caso de Prueba PFTDZGR06-216: Registrar usuario administrador exitosamente

Resumen:

Registro de usuario Administrador con todos los campos del formulario completos con datos válidos

Precondiciones:

El usuario no debe estar registrado en el sistema.

Endpoint POST /api/usuarios disponible.

Nº:	Pasos:	Resultados Esperados:

1	Construir una solicitud **POST** al endpoint /api/usuarios con el siguiente **body JSON**:
	{
	"nroDocumento": "",
	"priNombre": "",
	"segNombre": "",
	"priApellido": "",
	"segApellido": "",
	"tipoDoc": "",
	"fecNacimiento": "",
	"mail": "",
	"contrasenaHash": "",
	"calle": "",
	"nroPuerta": "",
	"nroApartamento": "",
	"bis": "",
	"codPostal": "",
	"idPerfil": "",
	"idCiudad": "",
	"telefonos": [""]
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

2	Enviar la solicitud	El código de respuesta es **201 Created**.
		El cuerpo de la respuesta contiene un mensaje y el objeto del usuario recién creado:
		{
		"message": "Usuario registrado correctamente. Actualmente se encuentra pendiente de validación.",
		"data": {
		"idUsuario": <autogenerado>,
		"priNombre": "Ana",
		"priApellido": "Pérez",
		...
		}
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-217: Registrar usuario socio exitosamente

Resumen:

Registro de usuario Socio con todos los campos del formulario completos con datos válidos

Precondiciones:

El usuario no debe estar registrado en el sistema.

Endpoint POST /api/usuarios disponible.

Nº:	Pasos:	Resultados Esperados:

1	Construir una solicitud POST al endpoint /api/usuarios con el siguiente body JSON:
	{
	"nroDocumento": "",
	"priNombre": "",
	"segNombre": "",
	"priApellido": "",
	"segApellido": "",
	"tipoDoc": "",
	"fecNacimiento": "",
	"mail": "",
	"contrasenaHash": "",
	"calle": "",
	"nroPuerta": "",
	"nroApartamento": "",
	"bis": "",
	"codPostal": "",
	"idPerfil": "",
	"idCiudad": "",
	"telefonos": [""],
	"usaLenguaSenias": "",
	"tieneDificultadAuditiva": "",
	"tieneSubcomision": "",
	"descripcionSubcomision": ""
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

2	Enviar la solicitud	El código de respuesta es **201 Created**.
		El cuerpo de la respuesta contiene un mensaje y el objeto del usuario recién creado:
		{
		"message": "Usuario registrado correctamente. Actualmente se encuentra pendiente de validación.",
		"data": {
		"idUsuario": <autogenerado>,
		"priNombre": "Ana",
		"priApellido": "Pérez",
		...
		}
		}

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	2.00
Prioridad:	Alta

Caso de Prueba PFTDZGR06-218: Registrar usuario No Socio exitosamente

Resumen:

Registro de usuario No Socio con todos los campos del formulario completos con datos válidos

Precondiciones:

El usuario no debe estar registrado en el sistema.

Endpoint POST /api/usuarios disponible.

Nº:	Pasos:	Resultados Esperados:

1	Construir una solicitud POST al endpoint /api/usuarios con el siguiente body JSON:
	{
	"nroDocumento": "",
	"priNombre": "",
	"segNombre": "",
	"priApellido": "",
	"segApellido": "",
	"tipoDoc": "",
	"fecNacimiento": "",
	"mail": "",
	"contrasenaHash": "",
	"calle": "",
	"nroPuerta": "",
	"nroApartamento": "",
	"bis": "",
	"codPostal": "",
	"idPerfil": "",
	"idCiudad": "",
	"telefonos": [""],
	"usaLenguaSenias": "",
	"tieneDificultadAuditiva": "",
	"tieneSubcomision": "",
	"descripcionSubcomision": ""
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

2	Enviar la solicitud	El código de respuesta es **201 Created**.
		El cuerpo de la respuesta contiene un mensaje y el objeto del usuario recién creado:
		{
		"message": "Usuario registrado correctamente. Actualmente se encuentra pendiente de validación.",
		"data": {
		"idUsuario": <autogenerado>,
		"priNombre": "Ana",
		"priApellido": "Pérez",
		...
		}
		}

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	2.00
Prioridad:	Alta

Caso de Prueba PFTDZGR06-219: Registrar usuario con datos inválidos

Resumen:

Verificar que el endpoint /api/usuarios rechace correctamente solicitudes de registro cuando uno o más campos contienen **datos inválidos**, respetando las reglas de validación del sistema.

Precondiciones:

El endpoint /api/usuarios está activo y accesible.

El usuario no está previamente registrado

Nº:	Pasos:	Resultados Esperados:

1	Construir una solicitud **POST** al endpoint /api/usuarios	

2	Incluir en el **body JSON** los campos requeridos, reemplazando uno o varios con valores inválidos. Ejemplo:
	{
	"nroDocumento": "",
	"priNombre": "",
	"segNombre": "",
	"priApellido": "",
	"segApellido": "",
	"tipoDoc": "",
	"fecNacimiento": "",
	"mail": "",
	"contrasenaHash": "",
	"calle": "",
	"nroPuerta": "",
	"nroApartamento": "",
	"bis": "",
	"codPostal": "",
	"idPerfil": "",
	"idCiudad": "",
	"telefonos": [""]
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

3	Enviar la solicitud.	El código de respuesta es **400 Bad Request** La respuesta incluye un **mensaje de error** describiendo los campos inválidos. Ningún usuario es creado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	5.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-220: Registrar usuario con campos obligatorios faltantes

Resumen:

Verificar que el endpoint /api/usuarios rechace correctamente solicitudes de registro cuando uno o más **campos obligatorios** no son enviados o están vacíos.

Precondiciones:

El endpoint /api/usuarios está activo y accesible.

El usuario no está previamente registrado

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud **POST** al endpoint /api/usuarios	

2	Incluir en el **body JSON** los campos del usuario, dejando vacío uno o varios campos obligatorios. Ejemplo:
	{
	"nroDocumento": "",
	"priNombre": "",
	"segNombre": "",
	"priApellido": null,
	"segApellido": "",
	"tipoDoc": "",
	"fecNacimiento": null,
	"mail": null,
	"contrasenaHash": "",
	"calle": "",
	"nroPuerta": null,
	"nroApartamento": "",
	"bis": "",
	"codPostal": "",
	"idPerfil": null,
	"idCiudad": "",
	"telefonos": [""]
	}
	(null y "" cuentan como vacios)	

3	Enviar la solicitud	El código de respuesta es **400 Bad Request**. La respuesta contiene un mensaje claro describiendo los **campos obligatorios faltantes**. Ningún usuario es creado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-221: Registro de Usuario Duplicado

Resumen:

Verificar que el endpoint /api/usuarios **rechace solicitudes de registro** cuando los datos enviados ya existen en la base de datos, asegurando que **no se creen usuarios duplicados**.
Esto incluye campos únicos como **número de documento** o **correo electrónico**.

Precondiciones:

El endpoint /api/usuarios está activo y accesible.

El usuario no está previamente registrado

Existe previamente un usuario con alguno de los siguientes datos:

- nroDocumento = "65483687"
- mail = "adrian.apa@example.com"
- telefono = "09232248"

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud **POST** al endpoint /api/usuarios.	
2	Incluir en el **body JSON** los datos de un usuario que ya exista en la base de datos. Ejemplo: { "nroDocumento": "", "priNombre": "", "segNombre": "", "priApellido": "", "segApellido": "", "tipoDoc": "", "fecNacimiento": "", "mail": "", "contrasenaHash": "",	

	"calle": "", "nroPuerta": "", "nroApartamento": "", "bis": "", "codPostal": "", "idPerfil": "", "idCiudad": "", "telefonos": [""] } (Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas. Los datos duplicados se encuentran el los prerequisitos del caso de prueba)	
3	Enviar la solicitud.	Código de respuesta: **409 Conflict**. Mensaje de error en la respuesta indicando que el usuario ya existe. No se crea un nuevo usuario en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Suite de Pruebas : Listado de Usuarios

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint listado de usuarios.

Caso de Prueba PFTDZGR06-245: Acceso a listado de usuarios exitosamente

Resumen:

Verificar que el endpoint GET /api/usuarios devuelva correctamente la lista de usuarios registrados en el sistema, asegurando que el formato de respuesta y el código HTTP sean los esperados.

Precondiciones:

El endpoint GET /api/usuarios está disponible.

Existe al menos un usuario registrado en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud **GET** al endpoint /api/usuarios	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar la solicitud.	Código de respuesta: **200 OK**
		El cuerpo de la respuesta contiene un objeto con el listado de usuarios y un mensaje de exito, por ejemplo:
		{
		"mensaje": "Listado de usuarios obtenido correctamente.",
		"data": [
		{
		"idUsuario": 1,
		"priNombre": "Ana",
		"priApellido": "Pérez",
		"mail": "ana.perez@example.com",
		"idPerfil": 1,
		...
		},
		{
		"idUsuario": 2,
		"priNombre": "Luis",
		"priApellido": "Gómez",
		"mail": "luis.gomez@example.com",
		"idPerfil": 2
		}
		]
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-246: Acceso a listado de usuarios por Usuario Admin

Resumen:

Verificar que el endpoint GET /api/usuarios devuelva correctamente la lista de usuarios registrados si el usuarios es de tipo ADMIN

Precondiciones:

El endpoint GET /api/usuarios está disponible.

Existe al menos un usuario registrado en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado tiene perfil ADMIN

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/usuarios	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar la solicitud.	Código de respuesta: 200 OK
		El cuerpo de la respuesta contiene un objeto con el listado de usuarios y un mensaje de exito, por ejemplo:
		{
		"mensaje": "Listado de usuarios obtenido correctamente.",
		"data": [
		{
		"idUsuario": 1,
		"priNombre": "Ana",
		"priApellido": "Pérez",
		"mail": "ana.perez@example.com",
		"idPerfil": 1,
		...
		},
		{
		"idUsuario": 2,
		"priNombre": "Luis",
		"priApellido": "Gómez",
		"mail": "luis.gomez@example.com",
		"idPerfil": 2
		}
		]
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-247: Acceso a listado de usuarios por Usuario Socio/No Socio

Resumen:

Verificar que el endpoint GET /api/usuarios no devuelva la lista de usuarios registrados en el sistema si el usuario que intenta acceder es de tipo Socio o No Socio, asegurando que el formato de respuesta y el código HTTP sean los esperados.

Precondiciones:

El endpoint GET /api/usuarios está disponible.

Existe al menos un usuario registrado en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado debe ser de tipo Socio o No Socio

Nº:	Pasos:	Resultados Esperados:

1	Construir una solicitud GET al endpoint /api/usuarios	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar la solicitud.	Código de respuesta: 403 Forbidden
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-248: Filtrado del listado de usuarios

Resumen:

Verificar que el endpoint GET /api/usuarios devuelva correctamente la lista de usuarios registrados en el sistema de acuerdo al filtro seleccionado, asegurando que el formato de respuesta y el código HTTP sean los esperados.

Precondiciones:

El endpoint GET /api/usuarios está disponible.

Existe al menos un usuario registrado en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado debe ser ADMIN

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/usuarios	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	(Opcional) Agregar parámetros de filtro válidos, por ejemplo: • /api/usuarios?priNombre=Juan • /api/usuarios?segNombre=Andres • /api/usuarios?priApellidos=Rodriguez • /api/usuarios?segApellidos=Garcia • /api/usuarios?nroDocumento=65481140 • /api/usuarios?tipoUsuario=2 • /api/usuarios?estado=ACTIVO Los filtros se pueden combinar usando &	

5	Enviar la solicitud	Código de respuesta: **200 OK**
		El cuerpo de la respuesta contiene un objeto con el listado de usuarios que coinciden con los filtros aplicados y un mensaje de exito, por ejemplo:
		{
		"mensaje": "Listado de usuarios obtenido correctamente.",
		"data": [
		{
		"idUsuario": 1,
		"priNombre": "Ana",
		"priApellido": "Pérez",
		"mail": "ana.perez@example.com",
		"idPerfil": 1,
		...
		},
		{
		"idUsuario": 2,
		"priNombre": "Luis",
		"priApellido": "Gómez",
		"mail": "luis.gomez@example.com",
		"idPerfil": 2
		}
		]
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-249: Acceder a listado sin usuarios registrados

Resumen:

Verificar que el endpoint GET /api/usuarios devuelva una lista vacía de usuarios y el código HTTP sea lo esperado.

Precondiciones:

El endpoint GET /api/usuarios está disponible.

No existen usuarios registrado en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado es de tipo ADMIN

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/usuarios	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar la solicitud.	Código de respuesta: **404 Not Found**
		El cuerpo de la respuesta contiene un objeto con el listado vacío de usuarios y un mensaje de informativo, por ejemplo:
		{
		"mensaje": "No se encontraron usuarios que coincidan con los filtros.",
		"data": [ ]
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-250: Acceder a listado sin token de autorización

Resumen:

Validar que el sistema no permita acceder al listado de usuarios cuando el token de autenticación está ausente o es inválido, devolviendo el código de estado y mensaje de error correspondiente.

Precondiciones:

El endpoint GET /api/usuarios requiere autenticación mediante token JWT

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/usuarios	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token inválido>	

3	Enviar la solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Suite de Pruebas : Modificación de Usuario

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint modificar datos de usuario por un administrador.

Caso de Prueba PFTDZGR06-253: Modificar de Socio a No Socio

Resumen:

Verifica que un administrador pueda modificar correctamente el perfil de un usuario, cambiando su rol de *socio* a *no socio*.

Precondiciones:

El endpoint PUT /api/usuarios/{id} está disponible y operativo.

Existe un usuario registrado con un id válido.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

El usuario a modificar tiene actualmente el perfil de *socio*.

El usuario logueado posee rol de *admin*.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/usuarios/{id}, reemplazando {id} por el identificador del usuario existente.	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	En el cuerpo (body) de la solicitud, enviar un objeto JSON con los campos a modificar, modificando tipo de usuario a no socio.
	Ejemplo:
	{
	"priNombre": "Carlos",
	"priApellido": "Gómez",
	"mail": "carlos.gomez@dominio.com",
	"idPerfil": 3 (id de perfil no socio),
	"idCiudad": 15
	}
	(Ver detalles de la suite para obtener datos válidos)	

4	Enviar la solicitud.	Código de respuesta: **200 OK**
		El cuerpo de la respuesta contiene un objeto con el usuario y sus datos modificados y un mensaje de exito, por ejemplo:
		{
		"mensaje": "El usuario ha sido actualizado correctamente.",
		"data": [
		{
		"idUsuario": 1,
		"priNombre": "Ana",
		"priApellido": "Pérez",
		"mail": "ana.perez@example.com",
		"idPerfil": 1,
		"usaLenguaSenias": false,
		"tieneDificultadAuditiva": false,
		"tieneSubcomision": false,
		"descripcionSubcomision": null
		}
		]
		}
		(los datos de socios deben ser false o null)

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-251: Modificar datos de usuario por administrador exitosamente

Resumen:

Modificar usuario por parte de un usuario administrador con todos los campos del formulario completos con datos válidos

Precondiciones:

El usuario no debe estar registrado en el sistema.

Endpoint POST /api/usuarios disponible.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/usuarios/{id}, reemplazando {id} por el identificador del usuario existente.	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	En el cuerpo (body) de la solicitud, enviar un objeto JSON con los campos a modificar.
	Ejemplo:
	{
	"priNombre": "Carlos",
	"priApellido": "Gómez",
	"mail": "carlos.gomez@dominio.com",
	"idPerfil": 2,
	"idCiudad": 15
	}
	(Ver detalles de la suite para obtener datos válidos)	

4	Enviar la solicitud.	Código de respuesta: 200 OK
		El cuerpo de la respuesta contiene un objeto con el usuario y sus datos modificados y un mensaje de exito, por ejemplo:
		{
		"mensaje": "El usuario ha sido actualizado correctamente.",
		"data": [
		{
		"idUsuario": 1,
		"priNombre": "Ana",
		"priApellido": "Pérez",
		"mail": "ana.perez@example.com",
		"idPerfil": 1,
		...
		}
		]
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-252: Modificar de No Socio a Socio

Resumen:

Verifica que un administrador pueda modificar correctamente el perfil de un usuario, cambiando su rol de *no socio* a *socio*.

Precondiciones:

El endpoint PUT /api/usuarios/{id} está disponible y operativo.

Existe un usuario registrado con un id válido.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

El usuario a modificar tiene actualmente el perfil de *no socio*.

El usuario logueado posee rol de *admin*.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/usuarios/{id}, reemplazando {id} por el identificador del usuario existente.	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	En el cuerpo (body) de la solicitud, enviar un objeto JSON con los campos a modificar, modificando el tipo de usuario de no socio a socio
	Ejemplo:
	{
	"priNombre": "Carlos",
	"priApellido": "Gómez",
	"mail": "carlos.gomez@dominio.com",
	"idPerfil": 2 (id de socio),
	"idCiudad": 15
	}
	(Ver detalles de la suite para obtener datos válidos)	

4	Enviar la solicitud.	Código de respuesta: 200 OK
		El cuerpo de la respuesta contiene un objeto con el usuario y sus datos modificados y un mensaje de exito, mostrando los nuevos datos de socio, incluido el número de socio, por ejemplo:
		{
		"mensaje": "El usuario ha sido actualizado correctamente.",
		"data": [
		{
		"idUsuario": 1,
		"priNombre": "Ana",
		"priApellido": "Pérez",
		"mail": "ana.perez@example.com",
		"idPerfil": 1,
		"nroSocio": SOC-2025-08,
		"usaLenguaSenias": false,
		"tieneDificultadAuditiva": true,
		"tieneSubcomision": false,
		"descripcionSubcomision": "Subcomisión deportiva"
		}
		]
		}

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	2.00
Prioridad:	Alta

Caso de Prueba PFTDZGR06-254: Modificación con datos obligatorios faltantes

Resumen:

Verifica que el sistema no permita actualizar un usuario si faltan campos obligatorios.

Precondiciones:

El endpoint PUT /api/usuarios/{id} está disponible y operativo.

Existe al menos un usuario registrado con un id válido.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/usuarios/{id}, reemplazando {id} por el identificador del usuario existente.	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos del usuario, dejando vacío uno o varios campos obligatorios. Ejemplo:
	{
	"nroDocumento": "",
	"priNombre": "",
	"segNombre": "",
	"priApellido": null,
	"segApellido": "",
	"tipoDoc": "",
	"fecNacimiento": null,
	"mail": null,
	"calle": "",
	"nroPuerta": null,
	"nroApartamento": "",
	"bis": "",
	"codPostal": "",
	"idPerfil": null,
	"idCiudad": "",
	"telefonos": [""]
	}
	(null y "" cuentan como vacios)	

4	Enviar solicitud	El código de respuesta: **400 Bad Request.** La respuesta contiene un mensaje claro describiendo los campos obligatorios faltantes. Ningún dato del usuario es modificado en la base de datos.

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

Caso de Prueba PFTDZGR06-255: Modificación con datos inválidos

Resumen:

Verifica que el sistema rechace modificaciones con datos en formato incorrecto o inválido.

Precondiciones:

El endpoint PUT /api/usuarios/{id} está disponible y operativo.

Existe al menos un usuario registrado con un id válido.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/usuarios/{id}, reemplazando {id} por el identificador del usuario existente.	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos requeridos, reemplazando uno o varios con valores inválidos. Ejemplo:
	{
	"nroDocumento": "",
	"priNombre": "",
	"segNombre": "",
	"priApellido": "",
	"segApellido": "",
	"tipoDoc": "",
	"fecNacimiento": "",
	"mail": "",
	"contrasenaHash": "",
	"calle": "",
	"nroPuerta": "",
	"nroApartamento": "",
	"bis": "",
	"codPostal": "",
	"idPerfil": "",
	"idCiudad": "",
	"telefonos": [""]
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	El código de respuesta: 400 Bad Request. La respuesta contiene un mensaje claro describiendo los datos inválidos y su formato correcto. Ningún dato del usuario es modificado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-256: Modificación por parte de un usuario no autorizado

Resumen:

Verifica que solo un usuario con rol *admin* pueda modificar usuarios.

Precondiciones:

El endpoint PUT /api/usuarios/{id} está disponible y operativo.

Se dispone de un **token JWT** (Se obtiene al iniciar sesión)

Usuario autenticado sin rol admin (por ejemplo, socio o no socio)

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/usuarios/{id}, reemplazando {id} por el identificador del usuario existente.	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos requeridos. Ejemplo:
	{
	"nroDocumento": "",
	"priNombre": "",
	"segNombre": "",
	"priApellido": "",
	"segApellido": "",
	"tipoDoc": "",
	"fecNacimiento": "",
	"mail": "",
	"contrasenaHash": "",
	"calle": "",
	"nroPuerta": "",
	"nroApartamento": "",
	"bis": "",
	"codPostal": "",
	"idPerfil": "",
	"idCiudad": "",
	"telefonos": [""]
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-257: Modificación con datos duplicados

Resumen:

Verifica que el sistema no permita modificar datos que ya están registrados en otro usuario (por ejemplo, documento o correo electrónico duplicado).

Precondiciones:

El endpoint PUT /api/usuarios/{id} está disponible y operativo.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

Los datos nuevos a modificar son válidos según las reglas del sistema.

Existen al menos dos usuarios en el sistema.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/usuarios/{id}, reemplazando {id} por el identificador del usuario existente.	
2	Incluir los siguientes encabezados:	

	• Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos requeridos, reemplazando uno o varios campos únicos con valores ya existentes en la base de datos. Ejemplo:
	{
	"nroDocumento": 65481140,
	"priNombre": "",
	"segNombre": "",
	"priApellido": "",
	"segApellido": "",
	"tipoDoc": "",
	"fecNacimiento": "",
	"mail": juan.rodriguez@gmail.com,
	"calle": "",
	"nroPuerta": "",
	"nroApartamento": "",
	"bis": "",
	"codPostal": "",
	"idPerfil": "",
	"idCiudad": "",
	"telefonos": ["092320240"]
	}
	(El resto de los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	Código de respuesta: **409 Conflict**. Mensaje de error en la respuesta indicando que ya existe un usuario con los datos ingresados. No se modifican los datos del usuario en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-272: Modificación de datos privados

Resumen:

Validar que un usuario administrador no pueda modificar los datos privados de un usuario (por ejemplo, contraseña).

Precondiciones:

El endpoint PUT /api/usuarios/{id} está disponible y operativo.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

El usuario logueado es de tipo Admin

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/usuarios/{id}, reemplazando {id} por el identificador del usuario existente.	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos del usuario, incluyendo contraseña. Ejemplo:
	{
	"nroDocumento": "",
	"priNombre": "",
	"segNombre": "",
	"priApellido": "",
	"segApellido": "",
	"tipoDoc": "",
	"fecNacimiento": "",
	"mail": "",
	"contrasenaHash": "Nueva contraseña",
	"calle": "",
	"nroPuerta": "",
	"nroApartamento": "",
	"bis": "",
	"codPostal": "",
	"idPerfil": "",
	"idCiudad": "",
	"telefonos": [""]
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Suite de Pruebas : Baja/Reactivación de Usuario

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint baja de usuario y endpoint reactivar de usuario.

Caso de Prueba PFTDZGR06-258: Baja exitosa de usuario

Resumen:

Verifica que un administrador pueda dar de baja correctamente a un usuario activo.

Precondiciones:

El usuario existe y está en estado **activo**.

El usuario logueado tiene rol admin.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/usuarios/baja/{id}, reemplazando {id} por el identificador del usuario existente.	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	Incluir en el body JSON el nuevo estado del usuarios. Ejemplo:	

	{
	"nuevoEstado": "INACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **200 OK**.
		La respuesta contiene un mensaje de exito y el objeto del usuario con su nuevo estado
		{
		"mensaje": "La cuenta del usuario ha sido desactivada exitosamente.",
		"data": [
		{
		"idUsuario": 1,
		"priNombre": "Ana",
		"priApellido": "Pérez",
		"mail": "ana.perez@example.com",
		"idPerfil": 1,
		"estado": "INACTIVO",
		"tieneDificultadAuditiva": false,
		"tieneSubcomision": false,
		"descripcionSubcomision": null
		}
		]
		}
		El usuario queda sin acceso al sistema.

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-259: Baja de usuario ya inactivo

Resumen:

Verifica que el sistema no permita dar de baja a un usuario que ya se encuentra inactivo.

Precondiciones:

El usuario está en estado **inactivo**.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado tiene rol admin

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/usuarios/baja/{id}, reemplazando {id} por el identificador del usuario existente.	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado del usuarios. Ejemplo:
	{
	"nuevoEstado": "INACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **409 Conflict.** La respuesta contiene un mensaje claro describiendo el error. Ningún dato del usuario es modificado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Media	

Caso de Prueba PFTDZGR06-260: Baja con usuario no existente

Resumen:

Verifica la respuesta del sistema al intentar dar de baja a un usuario inexistente.

Precondiciones:

ID de usuario no existente en la base de datos.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado tiene rol admin

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/usuarios/baja/{id}, reemplazando {id} por el identificador del usuario no existente.	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado del usuario. Ejemplo:
	{
	"nuevoEstado": "INACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **404 Not Found.** La respuesta contiene un mensaje claro describiendo que el usuario no existe.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-261: Baja por usuario no autorizado

Resumen:

Verifica que solo un usuario con rol *admin* pueda dar de baja usuarios.

Precondiciones:

Usuario autenticado con rol **no administrador (Socio o No Socio)**.

Usuario objetivo existe y está activo.

Se dispone de un **token JWT** (Se obtiene al iniciar sesión).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/usuarios/baja/{id}, reemplazando {id} por el identificador del usuario existente.	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado del usuario. Ejemplo:
	{
	"nuevoEstado": "INACTIVO"
	}	

4	Enviar solicitud	Código de respuesta: **403 Forbidden** Objeto de respuesta con mensaje de error: { "error": "Acceso denegado", "mensaje": "No tiene permiso para acceder a este recurso."

		}
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-262: Reactivación exitosa de usuario

Resumen:

Verifica que un administrador pueda reactivar correctamente un usuario dado de baja.

Precondiciones:

Usuario existente en estado **inactivo**.

El usuario logueado tiene rol admin.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/usuarios/reactivar/{id}, reemplazando {id} por el identificador del usuario existente.	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado del usuario. Ejemplo:
	{
	"nuevoEstado": "ACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **200 OK**.
		La respuesta contiene un mensaje de exito y el objeto del usuario con su nuevo estado
		{
		"mensaje": "La cuenta del usuario ha sido desactivada exitosamente.",
		"data": [
		{
		"idUsuario": 1,
		"priNombre": "Ana",
		"priApellido": "Pérez",
		"mail": "ana.perez@example.com",
		"idPerfil": 1,
		"estado": "ACTIVO",
		"tieneDificultadAuditiva": false,
		"tieneSubcomision": false,
		"descripcionSubcomision": null
		}
		]
		}
		El usuario queda sin acceso al sistema.

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-263: Reactivación de usuario ya activo

Resumen:

Verifica que el sistema no permita reactivar un usuario que ya se encuentra activo.

Precondiciones:

Usuario con estado **activo**.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado tiene rol admin

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/usuarios/reactivar/{id}, reemplazando {id} por el identificador del usuario existente.	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el estado del usuario. Ejemplo:
	{
	"nuevoEstado": "ACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **409 Conflict.** La respuesta contiene un mensaje claro describiendo el error. Ningún dato del usuario es modificado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Media	

Caso de Prueba PFTDZGR06-264: Reactivación de usuario inexistente

Resumen:

Verifica la respuesta al intentar reactivar un usuario que no existe.

Precondiciones:

ID de usuario no existente en la base de datos.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado tiene rol admin

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/usuarios/reactivar/{id}, reemplazando {id} por el identificador del usuario no existente.	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado del usuario. Ejemplo:
	{
	"nuevoEstado": "ACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **404 Not Found.** La respuesta contiene un mensaje claro describiendo que el usuario no existe.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-265: Reactivación por usuario no autorizado

Resumen:

Verifica que solo administradores puedan reactivar usuarios.

Precondiciones:

Usuario autenticado con rol **no administrador (Socio o No Socio)**.

Usuario objetivo existe y está activo.

Se dispone de un **token JWT** (Se obtiene al iniciar sesión).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/usuarios/reactivar/{id}, reemplazando {id} por el identificador del usuario existente.	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado del usuario. Ejemplo:
	{
	"nuevoEstado": "ACTIVO"
	}	

4	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Suite de Pruebas : Login de Usuario

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint login de usuario.

Caso de Prueba PFTDZGR06-461: Login de usuario Sin Validar

Resumen:

Verifica que el sistema no permita el inicio de sesión de un usuario sin validar

Precondiciones:

Usuario registrado en estado **sin validar**

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/auth/login	

2	Incluir en el body JSON las credenciales del usuario. Ejemplo:
	{
	"mail": "juan.rodriguez@gmail.com",
	"contrasena": "password123!"
	}	

3	Enviar solicitud	El código de respuesta: **401 Unauthorized** Muestra mensaje especificando que el usuario no tiene acceso al sistema
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-266: Inicio de sesión exitoso

Resumen:

Verifica que un usuario registrado y activo pueda iniciar sesión correctamente con credenciales válidas.

Precondiciones:

El usuario existe en la base de datos.

El usuario está **activo**.

Tiene correo y contraseña válidos registrados.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/auth/login	

2	Incluir en el body JSON las credenciales del usuario. Ejemplo:
	{
	"mail": "juan.rodriguez@gmail.com",
	"contrasena": "password123!"
	}	

3	Enviar solicitud	El código de respuesta: 200 OK. Devuelve el token de autorización
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-267: Login con credenciales inválidas

Resumen:

Verifica que el sistema rechace intentos de inicio de sesión con contraseña o correo incorrectos.

Precondiciones:

Usuario registrado y activo.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/auth/login	

2	Incluir en el body JSON las credenciales del usuarios incorrectos. Ejemplo:
	{
	"mail": "correo.incorrecto@gmail.com",
	"contrasena": "contraIncorrecto"
	}	

3	Enviar solicitud	El código de respuesta: **400 Bad Request.** La respuesta contiene un mensaje claro especificando que el correo y/o la contraseña son incorrectos. No se inicia sesión, ni se obtiene token de autorización
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-268: Login de usuario inactivo

Resumen:

Verifica que el sistema no permita el inicio de sesión de un usuario inactivo

Precondiciones:

Usuario registrado en estado **inactivo**

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/auth/login	

2	Incluir en el body JSON las credenciales del usuario. Ejemplo:
	{
	"mail": "juan.rodriguez@gmail.com",
	"contrasena": "password123!"
	}	

3	Enviar solicitud	El código de respuesta: **401 Unauthorized** Muestra mensaje especificando que el usuario no tiene acceso al sistema
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-270: Login con campos obligatorios faltantes

Resumen:

Verifica que el sistema valide la presencia de todos los campos obligatorios (email y contraseña).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/auth/login	

2	Incluir en el body JSON las credenciales del usuarios, dejando vacío uno de los campos. Ejemplo:
	{
	"mail": "",
	"contrasena": "password123!"
	}	

3	Enviar solicitud	El código de respuesta: **400 Bad Request.** La respuesta contiene un mensaje claro describiendo los campos obligatorios faltantes. No se inicia sesión, ni se obtiene token de autorización
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-271: Ataque de fuerza bruta o múltiples intentos fallidos

Resumen:

Verifica la respuesta del sistema ante múltiples intentos fallidos consecutivos de inicio de sesión.

Precondiciones:

Usuario existente y activo.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/auth/login	

2	Incluir en el body JSON las credenciales incorrectas del usuario. Ejemplo:
	{
	"mail": "juan.incorrectos@gmail.com",
	"contrasena": "badpassword"
	}	

3	Intentar el login al menos 6 veces	El sistema responde con mensaje de error y bloquea el acceso por 15 minutos
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	5.00	
Prioridad:	Alta	

Suite de Pruebas : Modificación de datos propios del Usuario

Conjunto de pruebas diseñadas para validar el correcto funcionamiento del endpoint modificar datos propios del usuario

Caso de Prueba PFTDZGR06-273: Modificación de datos personales exitosa

Resumen:

Validar que un usuario autenticado pueda modificar correctamente sus propios datos personales (por ejemplo, dirección, teléfono o correo electrónico).

Precondiciones:

El usuario debe estar autenticado y poseer un token válido.

El usuario debe existir en la base de datos.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/usuarios/me	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	En el cuerpo (body) de la solicitud, enviar un objeto JSON con los campos a modificar. Ejemplo:
	{
	"priNombre": "Carlos",
	"priApellido": "Gómez",
	"mail": "carlos.gomez@dominio.com",
	"idPerfil": 2,
	"idCiudad": 15,
	"contrasenaHash": "Nueva contraseña"
	}
	(Ver detalles de la suite para obtener datos válidos)	

4	Enviar solicitud	Código de respuesta: 200 OK El cuerpo de la respuesta contiene un objeto con el usuario y sus datos modificados y un mensaje de exito, por ejemplo: { "mensaje": "Tus datos han sido actualizados correctamente.", "data": [{ "idUsuario": 1,

		"priNombre": "Carlos", "priApellido": "Gómez", "mail": "carlos.gomez@dominio.com", "idPerfil": 2, ... }] }
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	5.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-274: Modificación con datos inválidos

Resumen:

Verificar que el sistema no acepte valores inválidos (por ejemplo, correo malformado o número de puerta con caracteres no numéricos).

Precondiciones:

Usuario autenticado con token válido.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/usuarios/me	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	En el cuerpo (body) de la solicitud, enviar un objeto JSON con los campos a modificar, ingresando datos invalidos.
	Ejemplo:
	{
	"priNombre": "",
	"priApellido": "",
	"mail": "",
	"idPerfil": "",
	"idCiudad": "",
	"contrasenaHash": ""
	}
	(Ver detalles de la suite para obtener campos y datos inválidos)	

4	Enviar la solicitud.	El código de respuesta: **400 Bad Request.** La respuesta contiene un mensaje claro describiendo los campos inválidos y su formato correcto. Ningún dato del usuario es modificado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	4.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-275: Modificación con campos obligatorios faltantes

Resumen:

Validar que el sistema no permita modificar los datos cuando faltan campos requeridos.

Precondiciones:

Usuario autenticado con token válido.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/usuarios/me	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	En el cuerpo (body) de la solicitud, enviar un objeto JSON con los campos a modificar, dejando campos obligatorios vacíos.
	Ejemplo:
	{
	"priNombre": null,
	"priApellido": "",
	"mail": "",
	"idPerfil": "",
	"idCiudad": null,
	"contrasenaHash": ""
	}
	(Ver detalles de la suite para obtener campos y datos inválidos)
	(null y "" cuentan como vacío)	

4	Enviar solicitud	El código de respuesta: **400 Bad Request.** La respuesta contiene un mensaje claro describiendo los campos obligatorios faltantes. Ningún dato del usuario es modificado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-276: Modificación con datos duplicados

Resumen:

Verificar que no se pueda cambiar un dato a uno que ya pertenece a otro usuario (por ejemplo, correo electrónico o documento).

Precondiciones:

Usuario autenticado.

Ya existe otro usuario con los datos ingresado.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/usuarios/me	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	En el cuerpo (body) de la solicitud, enviar un objeto JSON con los campos a modificar, ingresando datos duplicados, como correo o documento. Ejemplo: { "priNombre": "", "priApellido": "", "mail": "juan.rodriguez@gmail.com", "idPerfil": "",	

	"idCiudad": "", "contrasenaHash": "" } (Ver detalles de la suite para obtener campos y datos)	
4	Enviar solicitud	El código de respuesta: **409 Conflict.** La respuesta contiene un mensaje claro describiendo los campos duplicados, especificando que ya existente en la base de datos. Ningún dato del usuario es modificado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	4.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-277: Modificación sin token o con token inválido

Resumen:

Asegurar que un usuario no autenticado o con token inválido no pueda modificar datos.

Precondiciones:

Usuario no autenticado o token inválido.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/usuarios/me	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token inválido>	

3	En el cuerpo (body) de la solicitud, enviar un objeto JSON con los campos a modificar.
	Ejemplo:
	{
	"priNombre": "",
	"priApellido": "",
	"mail": "",
	"idPerfil": "",
	"idCiudad": "",
	"contrasenaHash": ""
	}
	(Ver detalles de la suite para obtener campos y datos válidos)	

4	Enviar solicitud	Código de respuesta: **401 Unauthorized**
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Suite de Pruebas : Gestión de Perfiles

Conjunto de pruebas diseñado para validar el correcto funcionamiento de todos los endpoints del módulo de perfiles, garantizando la integridad de las operaciones CRUD, seguridad y rendimiento de las APIs.

Ejemplos de parámetros válidos para los campos de Gestión de Perfil:

- Nombre: Administrador
- Descripción: Usuario con acceso completo al sistema
- Estado: ACTIVO | INACTIVO
- Rol: ADMIN | USER

Ejemplos para parámetros inválidos:

- Nombre: "" | "Adm1n%" | "A" * 51 | "A" | "Admin Admin Admin Admin Admin Admin" | null
- Descripción: "" | "A" * 501 | "Descripción #$%" | " " | null | "Descripción con <script>alert()</script>" (inyección código)
- Estado: DISPONIBLE | null | "" | activo
- Rol: administrador | null | "" | usuario

Suite de Pruebas : Registro de Perfil

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint registro de perfil

Caso de Prueba PFTDZGR06-303: Registrar Perfil exitosamente

Resumen:

Registro de nuevo perfil con todos los campos del formulario completos con datos válidos.

Precondiciones:

El endpoint POST /api/perfiles está disponible y operativo.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

El usuario logueado posee rol de *admin*.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/perfiles	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos requeridos. Ejemplo:
	{
	"nomPerfil": "",
	"desPerfil": "",
	"rol": "",
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	El código de respuesta: **201 Created.**
		La respuesta contiene un mensaje de exito y el objeto del nuevo perfil registrado
		{
		"mensaje": "El perfil ha sido registrado correctamente.",
		"data": [
		{
		"idPerfil": 1,
		"nomPerfil": "Nuevo Perfil",
		"desPerfil": "Nueva descripcion",
		"rol": "ADMIN",
		"estado": "INACTIVO",
		}
		]
		}
		por defecto el estado es INACTIVO

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	2.00
Prioridad:	Alta

Caso de Prueba PFTDZGR06-304: Registrar perfil con datos inválidos

Resumen:

Verificar que el endpoint POST /api/perfiles rechace correctamente solicitudes de registro cuando uno o más campos contienen **datos inválidos**, respetando las reglas de validación del sistema.

Precondiciones:

El endpoint POST /api/perfiles está disponible y operativo.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

El usuario logueado posee rol de *admin*

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/perfiles	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos requeridos, reemplazando uno o varios con valores inválidos. Ejemplo:
	{
	"nomPerfil": "",
	"desPerfil": "",
	"rol": "",
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	El código de respuesta: **400 Bad Request** La respuesta incluye un **mensaje de error** describiendo los campos inválidos. Ningún perfil es creado en la base de datos.

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

Caso de Prueba PFTDZGR06-305: Registrar perfil con datos obligatorios faltantes

Resumen:

Verifica que el sistema no permita registrar un perfil si faltan campos obligatorios.

Precondiciones:

El endpoint POST /api/perfiles está disponible y operativo.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

El usuario logueado posee rol de *admin*.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/perfiles	
2	Incluir los siguientes encabezados:	

	• Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos requeridos, dejando vacío uno o varios campos obligatorios. Ejemplo:
	{
	"nomPerfil": "",
	"desPerfil": "",
	"rol": "",
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	El código de respuesta es **400 Bad Request**. La respuesta contiene un mensaje claro describiendo los **campos obligatorios faltantes**. Ningún usuario es creado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-306: Registrar perfil con datos duplicados

Resumen:

Verifica que el sistema no permita registrar un nuevo perfil si contiene datos registrados en otro perfil (por ejemplo, nombre).

Precondiciones:

El endpoint POST /api/perfiles está disponible y operativo.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

El usuario logueado posee rol de *admin*.

Debe existir al menos un perfil en la base de datos

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/perfiles	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos requeridos, reemplazando nombre con nombre ya existente. Ejemplo:
	{
	"nomPerfil": "Administrador",
	"desPerfil": "",
	"rol": "",
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	Código de respuesta: **409 Conflict**. Mensaje de error en la respuesta indicando que el perfil ya existe. No se crea un nuevo perfil en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-307: Registrar perfil por parte de usuario no autorizado

Resumen:

Verifica que solo un usuario con rol *admin* pueda registrar nuevos perfiles.

Precondiciones:

El endpoint POST /api/perfiles está disponible y operativo.

Se dispone de un **token JWT** (Se obtiene al iniciar sesión)

Usuario autenticado sin rol admin (por ejemplo, socio o no socio)

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/perfiles	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos requeridos. Ejemplo:
	{
	"nomPerfil": "",
	"desPerfil": "",
	"rol": "",
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	4.00	
Prioridad:	Alta	

Suite de Pruebas : Listado de Perfiles

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint listado de perfiles

Caso de Prueba PFTDZGR06-313: Acceder a listado sin token de autorización

Resumen:

Validar que el sistema no permita acceder al listado de perfiles cuando el token de autenticación está ausente o es inválido, devolviendo el código de estado y mensaje de error correspondiente.

Precondiciones:

El endpoint GET /api/perfiles requiere autenticación mediante token JWT

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/perfiles	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token inválido>	
3	Enviar solicitud	Código de respuesta: **403 Forbidden**

		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-312: Acceder a listado sin perfiles registrados

Resumen:

Verificar que el endpoint GET /api/perfiles devuelva una lista vacía de perfiles y el código HTTP sea lo esperado.

Precondiciones:

El endpoint GET /api/perfiles está disponible.

No existen perfiles registrado en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado es de tipo ADMIN

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/perfiles	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar solicitud	Código de respuesta: **404 Not Found**
		El cuerpo de la respuesta contiene un objeto con el listado vacío de perfiles y un mensaje de informativo, por ejemplo:
		{
		"mensaje": "No se encontraron perfiles que coincidan con los filtros.",
		"data": [ ]
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-311: Filtrado del listado de perfiles

Resumen:

Verificar que el endpoint GET /api/perfiles devuelva correctamente la lista de perfiles registrados en el sistema de acuerdo al filtro seleccionado, asegurando que el formato de respuesta y el código HTTP sean los esperados.

Precondiciones:

El endpoint GET /api/perfiles está disponible.

Existe al menos un perfil registrado en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado debe ser ADMIN

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/perfiles	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	(Opcional) Agregar parámetros de filtro válidos, por ejemplo: • /api/perfiles?nombre=Socio • /api/perfiles?estado=ACTIVO Los filtros se pueden combinar usando &	

4	Enviar solicitud	Código de respuesta: **200 OK**
		El cuerpo de la respuesta contiene un objeto con el listado de perfiles de acuerdo a los filtros y un mensaje de exito, por ejemplo:
		{
		"mensaje": "Listado de perfiles obtenido correctamente.",
		"data": [
		{
		"idPerfil": 1,
		"nomPerfil": "Administrador",
		"desPerfil": "Usuario con total acceso",
		"rol": "ADMIN",
		"estado": "ACTIVO",
		...
		},
		{
		"idPerfil": 2,
		"nomPerfil": "Socio",
		"desPerfil": "Usuario con acceso parcial",
		"rol": "USER",
		"estado": "INACTIVO",
		}
		]
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-310: Acceso a listado de perfiles por Usuario Socio/No Socio

Resumen:

Verificar que el endpoint GET /api/perfiles no devuelva la lista de perfiles registrados en el sistema si el usuario que intenta acceder es de tipo Socio o No Socio, asegurando que el formato de respuesta y el código HTTP sean los esperados.

Precondiciones:

El endpoint GET /api/perfiles está disponible.

Existe al menos un perfil registrado en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado debe ser de tipo Socio o No Socio

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/perfiles	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-309: Acceso a listado de perfiles por Usuario Admin

Resumen:

Verificar que el endpoint GET /api/perfiles devuelva correctamente la lista de perfiles registrados si el usuarios es de tipo ADMIN

Precondiciones:

El endpoint GET /api/perfiles está disponible.

Existe al menos un perfil registrado en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado tiene perfil ADMIN

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/perfiles	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar solicitud	Código de respuesta: **200 OK**
		El cuerpo de la respuesta contiene un objeto con el listado de perfiles y un mensaje de exito, por ejemplo:
		{
		"mensaje": "Listado de perfiles obtenido correctamente.",
		"data": [
		{
		"idPerfil": 1,
		"nomPerfil": "Administrador",
		"desPerfil": "Usuario con total acceso",
		"rol": "ADMIN",
		"estado": "ACTIVO",
		...
		},
		{
		"idPerfil": 2,
		"nomPerfil": "Socio",
		"desPerfil": "Usuario con acceso parcial",
		"rol": "USER",
		"estado": "INACTIVO",
		}
		]
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-308: Acceso a listado de perfiles exitosamente

Resumen:

Verificar que el endpoint GET /api/perfiles devuelva correctamente la lista de perfiles registrados en el sistema, asegurando que el formato de respuesta y el código HTTP sean los esperados.

Precondiciones:

El endpoint GET /api/perfiles está disponible.

Existe al menos un perfil registrado en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado debe ser de tipo *Admin*

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/perfiles	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar la solicitud.	Código de respuesta: **200 OK**
		El cuerpo de la respuesta contiene un objeto con el listado de perfiles y un mensaje de exito, por ejemplo:
		{
		"mensaje": "Listado de perfiles obtenido correctamente.",
		"data": [
		{
		"idPerfil": 1,
		"nomPerfil": "Administrador",
		"desPerfil": "Usuario con total acceso",
		"rol": "ADMIN",
		"estado": "ACTIVO",
		...
		},
		{
		"idPerfil": 2,
		"nomPerfil": "Socio",
		"desPerfil": "Usuario con acceso parcial",
		"rol": "USER",
		"estado": "INACTIVO",
		}
		]
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Suite de Pruebas : Modificación de perfil

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint modificar de perfil

Caso de Prueba PFTDZGR06-318: Modificar perfil por parte de un usuario no autorizado

Resumen:

Verificar que los usuarios que no poseen el rol administrador no puedan modificar perfiles.

Precondiciones:

El endpoint PUT /api/perfiles/{id} está disponible y operativo.

Existe un perfil en la base de datos

Se dispone de un **token JWT** (Se obtiene al iniciar sesión)

Usuario autenticado sin rol admin (por ejemplo, socio o no socio)

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/perfiles/{id}, con un id de un perfil existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos requeridos. Ejemplo:
	{
	"desPerfil": "",
	"estado": "ACTIVO"
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	

Prioridad:	Alta

Caso de Prueba PFTDZGR06-316: Modificar perfil con datos inválidos

Resumen:

Verifica que el sistema rechace modificaciones con datos en formato incorrecto o inválido.

Precondiciones:

El endpoint PUT /api/perfiles/{id} está disponible y operativo.

Existe al menos un perfil registrado en la base de datos

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado es de tipo *Admin*

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/perfiles/{id}, con un id de un perfil existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos requeridos con formatos inválidos. Ejemplo:
	{
	"desPerfil": "",
	"estado": "ACTIVO"
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	El código de respuesta: **400 Bad Request** La respuesta incluye un **mensaje de error** describiendo los campos inválidos. Ningún perfil es modificado en la base de datos.

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

Caso de Prueba PFTDZGR06-315: Modificar perfil con datos obligatorios faltantes

Resumen:

Verifica que el sistema no permita actualizar un perfil si faltan campos obligatorios.

Precondiciones:

El endpoint PUT /api/perfiles/{id} está disponible y operativo.

Existe al menos un perfil registrado en la base de datos

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado es de tipo *Admin*

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/perfiles/{id}, con un id de un perfil existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos requeridos, dejando campos obligatorios vacíos. Ejemplo:
	{
	"desPerfil": "",
	"estado": null
	}
	(null y "" cuentan como vacío.)	

4	Enviar solicitud	El código de respuesta: **400 Bad Request** La respuesta incluye un **mensaje de error** describiendo los campos faltantes. Ningún perfil es modificado en la base de datos.

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

Caso de Prueba PFTDZGR06-314: Modificar perfil exitosamente

Resumen:

Verificar que el sistema permita modificar correctamente los datos de un perfil existente cuando la solicitud es válida y proviene de un usuario autorizado.

Precondiciones:

El endpoint PUT /api/perfiles/{id} está disponible y operativo.

Existe al menos un perfil en la base de datos

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

El usuario logueado posee rol de *admin*.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/perfiles/{id}, con un id de un perfil existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos requeridos. Ejemplo:
	{
	"descripcion": "",
	"estado": "",
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar la solicitud	El código de respuesta es **200 OK**.
		El cuerpo de la respuesta contiene un mensaje y el objeto del perfil modificado:
		{
		"message": "La funcionalidad ha sido actualizada correctamente.",
		"data": {
		"idPerfil": 1,
		"nomPerfil": "Socio",
		"desPerfil": "Usuario con acceso parcial",
		"estado": "ACTIVO",
		"rol": "ADMIN",
		...
		}
		}

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	2.00
Prioridad:	Alta

Caso de Prueba PFTDZGR06-319: Modificar perfil no existente		
Resumen: Verificar que el sistema retorne un error cuando se intenta modificar un perfil que no existe.		
Precondiciones: El endpoint PUT /api/perfiles/{id} está disponible y operativo. Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión) El usuario logueado es de tipo *Admin*		
Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/perfiles/{id}, con un id de un perfil no existente	

2	Incluir en el body JSON los campos requeridos. Ejemplo:
	{
	"desPerfil": "",
	"estado": "ACTIVO"
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

3	Enviar solicitud	El código de respuesta: **404 Not Found.** La respuesta contiene un mensaje claro describiendo que el perfil no existe.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Suite de Pruebas : Baja/Reactivación de Perfil

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint baja de perfil y reactivación de perfil

Caso de Prueba PFTDZGR06-327: Reactivación de perfil por usuario no autorizado		
Resumen: Verifica que solo administradores puedan reactivar perfiles.		
Precondiciones: Usuario autenticado con rol **no administrador (Socio o No Socio)**. Perfil objetivo existe y está activo. Se dispone de un **token JWT** (Se obtiene al iniciar sesión).		
Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/perfiles/reactivar/{id}, con un id de un perfil existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado del perfil. Ejemplo:
	{
	"nuevoEstado": "ACTIVO"
	}	

4	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-326: Reactivación de perfil no existente		
Resumen: Verifica la respuesta al intentar reactivar un perfil que no existe.		
Precondiciones: ID de perfil no existente en la base de datos. Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión) El usuario logueado tiene rol admin		
Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/perfiles/reactivar/{id}, con un id de un perfil no existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado del perfil. Ejemplo:
	{
	"nuevoEstado": "ACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **404 Not Found.** La respuesta contiene un mensaje claro describiendo que el perfil no existe.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-325: Reactivación de perfil ya activo		
Resumen: Verifica que el sistema no permita reactivar un perfil que ya se encuentra activo.		
Precondiciones: Perfil con estado **activo**. Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión) El usuario logueado tiene rol admin		
Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/perfiles/reactivar/{id}, con un id de un perfil existente con estado ACTIVO	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	Incluir en el body JSON el estado del perfil. Ejemplo: {	

	"nuevoEstado": "ACTIVO" }	
4	Enviar solicitud	El código de respuesta: **409 Conflict.** La respuesta contiene un mensaje claro describiendo el error. Ningún dato del perfil es modificado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Media	

Caso de Prueba PFTDZGR06-324: Reactivación exitosa de perfil

Resumen:

Verifica que un administrador pueda reactivar correctamente un perfil dado de baja.

Precondiciones:

Perfil existente en estado **inactivo**.

El usuario logueado tiene rol admin.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/perfiles/reactivar/{id}, con un id de un perfil existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado del perfil. Ejemplo:
	{
	"nuevoEstado": "ACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **200 OK**.
		La respuesta contiene un mensaje de exito y el objeto del perfil con su nuevo estado
		{
		"mensaje": "El perfil ha sido activado exitosamente.",
		"data": [
		{
		"idPerfil": 1,
		"nomPerfil": "Socio",
		"desPerfil": "Usuario con acceso parcial",
		"rol": "USER,
		"estado": "ACTIVO",
		}
		]
		}
		El perfil queda activo y puede ser vinculado a otra entidad.

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-323: Baja de perfil por usuario no autorizado

Resumen:

Verifica que solo un usuario con rol *admin* pueda dar de baja perfiles.

Precondiciones:

Usuario autenticado con rol **no administrador (Socio o No Socio)**.

Perfil objetivo existe y está activo.

Se dispone de un **token JWT** (Se obtiene al iniciar sesión).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/perfiles/baja/{id}, con un id de un perfil existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado del perfil. Ejemplo:
	{
	"nuevoEstado": "INACTIVO"
	}	

4	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-322: Baja con perfil no existente

Resumen:

Verifica la respuesta del sistema al intentar dar de baja a un perfil inexistente.

Precondiciones:

ID de perfil no existente en la base de datos.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado tiene rol admin

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/perfiles/baja/{id}, con un id de un perfil no existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado del perfil. Ejemplo:
	{
	"nuevoEstado": "INACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **404 Not Found.** La respuesta contiene un mensaje claro describiendo que el perfil no existe.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	

Prioridad:	Alta

Caso de Prueba PFTDZGR06-321: Baja de perfil ya inactivo

Resumen:

Verifica que el sistema no permita dar de baja a un perfil que ya se encuentra inactivo.

Precondiciones:

El perfil está en estado **inactivo**.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado tiene rol admin

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/perfiles/baja/{id}, con un id de un perfil existente con estado INACTIVO	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el estado del perfil. Ejemplo:
	{
	"nuevoEstado": "INACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **409 Conflict.** La respuesta contiene un mensaje claro describiendo el error. Ningún dato del perfil es modificado en la base de datos.

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	2.00
Prioridad:	Media

Caso de Prueba PFTDZGR06-320: Baja exitosa de perfil

Resumen:

Verifica que un administrador pueda dar de baja correctamente a un perfil activo.

Precondiciones:

El perfil existe y está en estado **activo**.

El usuario logueado tiene rol admin.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/perfiles/baja/{id}, con un id de un perfil existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado del perfil. Ejemplo:
	{
	"nuevoEstado": "INACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **200 OK**.
		La respuesta contiene un mensaje de exito y el objeto del perfil con su nuevo estado
		
		{
		"mensaje": "El perfil ha sido dado de baja exitosamente.",
		"data": [
		{
		"idPerfil": 1,
		"nomPerfil": "Socio",
		"desPerfil": "Usuario con acceso parcial",
		"rol": "USER,
		"estado": "INACTIVO",
		}
		]
		}
		
		El perfil queda inactivo y no puede ser vinculado a otra entidad.

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	2.00
Prioridad:	Alta

Caso de Prueba PFTDZGR06-328: Validar restricción de uso de perfiles dados de baja

Resumen:

Verificar que el sistema **no permita asociar ni relacionar** un perfil que se encuentra dado de baja (inactivo) con otros registros, como usuarios u otras entidades dependientes.

Precondiciones:

Existe un perfil previamente dado de baja (estado inactivo).

Endpoint de creación o modificación de usuario disponible.

Usuario autenticado con rol administrador.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/usuarios, con un id de un perfil existente con estado INACTIVO	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	construir el siguiente **body JSON** incluyendo el id del perfil con estado **INACTIVO**:
	{
	"nroDocumento": "",
	"priNombre": "",
	"segNombre": "",
	"priApellido": "",
	"segApellido": "",
	"tipoDoc": "",
	"fecNacimiento": "",
	"mail": "",
	"contrasenaHash": "",
	"calle": "",
	"nroPuerta": "",
	"nroApartamento": "",
	"bis": "",
	"codPostal": "",
	"idPerfil": 3 (perfil con estado INACTIVO),
	"idCiudad": "",
	"telefonos": [""]
	}	

	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	Código de respuesta: **422 Unprocessable Entity**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No es posible asignar un perfil inactivo al usuario."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Suite de Pruebas : Gestión de Funcionalidades

Conjunto de pruebas diseñado para validar el correcto funcionamiento de todos los endpoints del módulo de funcionalidades, garantizando la integridad de las operaciones CRUD, seguridad y rendimiento de las APIs.

Ejemplos de parámetros válidos para los campos de Gestión de Funcionalidad:

- Nombre: Listado de Usuarios
- Descripción: Acceso a funcionalidad de listado
- Estado: ACTIVO | INACTIVO

Ejemplos para parámetros inválidos:

- Nombre: "" | "Gest1ón%" | "A" * 101 | "G" |"Func Func Func Func Func Func Func" | " " | null | "Listado<script>"
- Descripción: "" | "A" * 501 | "Descripción #$%" | " " | null | "Descripción con <script>alert()</script>"
- Estado: DISPONIBLE | null | "" | "activo"

Suite de Pruebas : Registro de Funcionalidad

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint registro de funcionalidad.

Caso de Prueba PFTDZGR06-459: Registrar funcionalidad con datos duplicados

Resumen:

Verifica que el sistema no permita registrar una nueva funcionalidad si contiene datos registrados en otro funcionalidad (por ejemplo, nombre).

Precondiciones:

El endpoint POST /api/funcionalidades está disponible y operativo.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

El usuario logueado posee rol de *admin*.

Debe existir al menos una funcionalidad en la base de datos

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/funcionalidades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos requeridos, reemplazando nombre con nombre ya existente. Ejemplo:
	{
	"nombre": "Listado de usuarios",
	"descripcion": "",
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	Código de respuesta: **409 Conflict**. Mensaje de error en la respuesta indicando que la funcionalidad ya existe. No se crea una nueva funcionalidad en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-419: Registrar funcionalidad exitosamente

Resumen:

Registro de nueva funcionalidad con todos los campos del formulario completos con datos válidos.

Precondiciones:

El endpoint POST /api/funcionalidades está disponible y operativo.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

El usuario logueado posee rol de *admin*.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/funcionalidades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos requeridos. Ejemplo:
	{
	"nombre": "",
	"descripcion": "",
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	El código de respuesta: **201 Created.**
		La respuesta contiene un mensaje de exito y el objeto del nueva funcionalidad registrada
		{
		"mensaje": "La funcionalidad ha sida registrada correctamente.",
		"data": [
		{
		"idFuncionalidad": 1,
		"nombre": "Nueva Funcionalidad",
		"descripcion": "Nueva descripcion",
		"estado": "INACTIVO",
		}
		]
		}
		por defecto el estado es INACTIVO

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-420: Registrar funcionalidad con datos inválidos

Resumen:

Verificar que el endpoint POST /api/funcionalidades rechace correctamente solicitudes de registro cuando uno o más campos contienen **datos inválidos**, respetando las reglas de validación del sistema.

Precondiciones:

El endpoint POST /api/funcionalidades está disponible y operativo.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

El usuario logueado posee rol de *admin*

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/funcionalidades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	Incluir en el body JSON los campos requeridos, reemplazando uno o varios con valores inválidos. Ejemplo:	

	{
	"nombre": "",
	"descripcion": ""
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	El código de respuesta: **400 Bad Request** La respuesta incluye un **mensaje de error** describiendo los campos inválidos. Ninguna funcionalidad es creada en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-421: Registrar funcionalidad con datos obligatorios faltantes

Resumen:

Verifica que el sistema no permita registrar una funcionalidad si faltan campos obligatorios.

Precondiciones:

El endpoint POST /api/funcionalidades está disponible y operativo.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

El usuario logueado posee rol de *admin*.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/funcionalidades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos requeridos, dejando vacío uno o varios campos obligatorios. Ejemplo:
	{
	"nombre": "",
	"descripcion": ""
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	El código de respuesta es **400 Bad Request**. La respuesta contiene un mensaje claro describiendo los **campos obligatorios faltantes**. Ninguna funcionalidad es creada en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-423: Registrar funcionalidad por parte de usuario no autorizado

Resumen:

Verifica que solo un usuario con rol *admin* pueda registrar nuevas funcionalidades.

Precondiciones:

El endpoint POST /api/funcionalidades está disponible y operativo.

Se dispone de un **token JWT** (Se obtiene al iniciar sesión)

Usuario autenticado sin rol admin (por ejemplo, socio o no socio)

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/funcionalidades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos requeridos. Ejemplo:
	{
	"nombre": "",
	"descripcion": ""
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	4.00	
Prioridad:	Alta	

Suite de Pruebas : Listado de Funcionalidades

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint listado de funcionalidades

Caso de Prueba PFTDZGR06-424: Acceder a listado sin token de autorización

Resumen:

Validar que el sistema no permita acceder al listado de funcionalidades cuando el token de autenticación está ausente o es inválido, devolviendo el código de estado y mensaje de error correspondiente.

Precondiciones:

El endpoint GET /api/funcionalidades requiere autenticación mediante token JWT

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/perfiles	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token inválido>	

3	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-425: Acceder a listado sin funcionalidades registradas

Resumen:

Verificar que el endpoint GET /api/funcionalidades devuelva una lista vacía de funcionalidades y el código HTTP sea lo esperado.

Precondiciones:

El endpoint GET /api/funcionalidades está disponible.

No existen funcionalidades registradas en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado es de tipo ADMIN

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/funcionalidades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar solicitud	Código de respuesta: **404 Not Found**
		El cuerpo de la respuesta contiene un objeto con el listado vacío de funcionalidades y un mensaje de informativo, por ejemplo:
		{
		"mensaje": "No se encontraron funcionalidades que coincidan con los filtros.",
		"data": [ ]
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-426: Filtrado del listado de funcionalidades

Resumen:

Verificar que el endpoint GET /api/funcionalidades devuelva correctamente la lista de funcionalidades registradas en el sistema de acuerdo al filtro seleccionado, asegurando que el formato de respuesta y el código HTTP sean los esperados.

Precondiciones:

El endpoint GET /api/funcionalidades está disponible.

Existe al menos una funcionalidad registrada en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado debe ser ADMIN

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/funcionalidades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	(Opcional) Agregar parámetros de filtro válidos, por ejemplo: • /api/funcionalidades?nombre=Listado • /api/funcionalidades?estado=ACTIVO Los filtros se pueden combinar usando &	

4	Enviar solicitud	Código de respuesta: **200 OK**
		El cuerpo de la respuesta contiene un objeto con el listado de funcionalidades de acuerdo a los filtros y un mensaje de exito, por ejemplo:
		{
		"mensaje": "Listado de funcionalidades obtenido correctamente.",
		"data": [
		{
		"idFuncionalidad": 1,
		"nombre": "Listado de usuario",
		"descripcion": "Acceder al listado de usuarios",
		"estado": "ACTIVO",
		...
		},
		{
		"idFuncionalidad": 2,
		"nombre": "Crear actividad",
		"descripcion": "Crear nuevas actividades en el sistema",
		"estado": "INACTIVO",
		}
		]
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-427: Acceso a listado de funcionalidades por Usuario Socio/No Socio

Resumen:

Verificar que el endpoint GET /api/funcionalidades no devuelva la lista de funcionalidades registradas en el sistema si el usuario que intenta acceder es de tipo Socio o No Socio, asegurando que el formato de respuesta y el código HTTP sean los esperados.

Precondiciones:

El endpoint GET /api/funcionalidades está disponible.

Existe al menos una funcionalidad registrada en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado debe ser de tipo Socio o No Socio

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/funcionalidades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-428: Acceso a listado de funcionalidades por Usuario Admin

Resumen:

Verificar que el endpoint GET /api/funcionalidades devuelva correctamente la lista de funcionalidades registradas si el usuarios es de tipo ADMIN

Precondiciones:

El endpoint GET /api/funcionalidades está disponible.

Existe al menos una funcionalidad registrada en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado tiene perfil ADMIN

Nº:	Pasos:	Resultados Esperados:

1	Construir una solicitud GET al endpoint /api/funcionalidades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar solicitud	Código de respuesta: **200 OK**
		El cuerpo de la respuesta contiene un objeto con el listado de funcionalidades y un mensaje de exito, por ejemplo:
		{
		"mensaje": "Listado de funcionalidades obtenido correctamente.",
		"data": [
		{
		"idFuncionalidad": 1,
		"nombre": "Listado de usuario",
		"descripcion": "Acceder al listado de usuarios",
		"estado": "ACTIVO",
		...
		},
		{
		"idFuncionalidad": 2,
		"nombre": "Crear actividade",
		"descripcion": "Crear nuevas actividades en el sistema",
		"estado": "INACTIVO",
		}
		]
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-429: Acceso a listado de funcionalidades exitosamente

Resumen:

Verificar que el endpoint GET /api/funcionalidades devuelva correctamente la lista de funcionalidades registradas en el sistema, asegurando que el formato de respuesta y el código HTTP sean los esperados.

Precondiciones:

El endpoint GET /api/funcionalidades está disponible.

Existe al menos una funcionalidad registrada en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado debe ser de tipo *Admin*

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/funcionalidades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar la solicitud.	Código de respuesta: **200 OK**
		El cuerpo de la respuesta contiene un objeto con el listado de funcionalidades y un mensaje de exito, por ejemplo:
		{
		"mensaje": "Listado de funcionalidades obtenido correctamente.",
		"data": [
		{
		"idFuncionalidad": 1,
		"nombre": "Listado de usuarios",
		"descripcion": "Acceso a listado de usuarios",
		"estado": "ACTIVO",
		...
		},
		{
		"idFuncionalidad": 2,
		"nombre": "Crear actividad",
		"descripcion": "Craer nueva actividad en el sistema",
		"estado": "INACTIVO",
		}
		]
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Suite de Pruebas : Modificación de Funcionalidad

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint modificación de funcionalidad

Caso de Prueba PFTDZGR06-430: Modificar funcionalidad por parte de un usuario no autorizado

Resumen:

Verificar que los usuarios que no poseen el rol administrador no puedan modificar funcionalidades.

Precondiciones:

El endpoint PUT /api/funcionalidades/{id} está disponible y operativo.

Existe una funcionalidad en la base de datos

Se dispone de un **token JWT** (Se obtiene al iniciar sesión)

Usuario autenticado sin rol admin (por ejemplo, socio o no socio)

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/funcionalidades/{id}, con un id de una funcionalidad existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos requeridos. Ejemplo:
	{
	"descripcion": "",
	"estado": "ACTIVO"
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-431: Modificar funcionalidad con datos inválidos

Resumen:

Verifica que el sistema rechace modificaciones con datos en formato incorrecto o inválido.

Precondiciones:

El endpoint PUT /api/funcionalidades/{id} está disponible y operativo.

Existe al menos una funcionalidad registrada en la base de datos

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado es de tipo *Admin*

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/funcionalidades/{id}, con un id de una funcionalidad existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos requeridos con formatos inválidos. Ejemplo:
	{
	"descripcion": "",
	"estado": "ACTIVO"
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	El código de respuesta: **400 Bad Request** La respuesta incluye un **mensaje de error** describiendo los campos inválidos. Ninguna funcionalidad es modificado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-432: Modificar funcionalidad con datos obligatorios faltantes

Resumen:

Verifica que el sistema no permita actualizar una funcionalidad si faltan campos obligatorios.

Precondiciones:

El endpoint PUT /api/funcionalidades/{id} está disponible y operativo.

Existe al menos una funcionalidad registrada en la base de datos

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado es de tipo *Admin*

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/funcionalidades/{id}, con un id de una funcionalidad existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos requeridos, dejando campos obligatorios vacíos. Ejemplo:
	{
	"descripcion": "",
	"estado": null
	}
	(null y "" cuentan como vacío.)	

4	Enviar solicitud	El código de respuesta: **400 Bad Request** La respuesta incluye un **mensaje de error** describiendo los campos faltantes. Ninguna funcionalidad es modificada en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-433: Modificar funcionalidad exitosamente

Resumen:

Verificar que el sistema permita modificar correctamente los datos de una funcionalidad existente cuando la solicitud es válida y proviene de un usuario autorizado.

Precondiciones:

El endpoint PUT /api/funcionalidades/{id} está disponible y operativo.

Existe al menos una funcionalidad en la base de datos

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

El usuario logueado posee rol de *admin*.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/funcionalidades/{id}, con un id de una funcionalidad existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON los campos requeridos. Ejemplo:
	{
	"descripcion": "",
	"estado": "",
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar la solicitud	El código de respuesta es **200 OK**.
		El cuerpo de la respuesta contiene un mensaje y el objeto de la funcionalidad modificado:
		{
		"message": "La funcionalidad ha sido actualizada correctamente.",
		"data": {
		"idFuncionalidad": 1,
		"nombre": "Listado de usuarios",
		"descripcion": "Descripcion modificada",
		"estado": "ACTIVO",
		}
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-434: Modificar funcionalidad no existente

Resumen:

Verificar que el sistema retorne un error cuando se intenta modificar una funcionalidad que no existe.

Precondiciones:

El endpoint PUT /api/funcionalidades/{id} está disponible y operativo.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado es de tipo *Admin*

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/funcionalidades/{id}, con un id de una funcionalidad no existente	
2	Incluir en el body JSON los campos requeridos. Ejemplo:	

	{
	"descripcion": "",
	"estado": "ACTIVO"
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

3	Enviar solicitud	El código de respuesta: **404 Not Found.** La respuesta contiene un mensaje claro describiendo que la funcionalidad no existe.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Suite de Pruebas : Baja/Reactivación de Funcionalidad

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint baja de funcionalidad y reactivación de funcionalidad

Caso de Prueba PFTDZGR06-435: Reactivación de funcionalidad por usuario no autorizado

Resumen:

Verifica que solo administradores puedan reactivar funcionalidades.

Precondiciones:

Usuario autenticado con rol **no administrador (Socio o No Socio)**.

Funcionalidad objetivo existe y está activa.

Se dispone de un **token JWT** (Se obtiene al iniciar sesión).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/funcionalidades/reactivar/{id}, con un id de una funcionalidad existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado de la funcionalidad. Ejemplo:
	{
	"nuevoEstado": "ACTIVO"
	}	

4	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-436: Reactivación de funcionalidad no existente

Resumen:

Verifica la respuesta al intentar reactivar una funcionalidad que no existe.

Precondiciones:

ID de funcionalidad no existente en la base de datos.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado tiene rol admin

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/funcionalidades/reactivar/{id}, con un id de una funcionalidad no existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado de la funcionalidad. Ejemplo:
	{
	"nuevoEstado": "ACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **404 Not Found.** La respuesta contiene un mensaje claro describiendo que la funcionalidad no existe.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-437: Reactivación de funcionalidad ya activo

Resumen:

Verifica que el sistema no permita reactivar una funcionalidad que ya se encuentra activa.

Precondiciones:

Funcionalidad con estado **activo**.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado tiene rol admin

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/funcionalidades/reactivar/{id}, con un id de una funcionalidad existente con estado ACTIVO	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el estado de la funcionalidad . Ejemplo:
	{
	"nuevoEstado": "ACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **409 Conflict.** La respuesta contiene un mensaje claro describiendo el error. Ningún dato de la funcionalidad es modificado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Media	

Caso de Prueba PFTDZGR06-438: Reactivación exitosa de funcionalidad

Resumen:

Verifica que un administrador pueda reactivar correctamente una funcionalidad dado de baja.

Precondiciones:

Funcionalidad existente en estado **inactivo**.

El usuario logueado tiene rol admin.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/funcionalidades/reactivar/{id}, con un id de una funcionalidad existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado de la funcionalidad. Ejemplo:
	{
	"nuevoEstado": "ACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **200 OK**.
		La respuesta contiene un mensaje de exito y el objeto de la funcionalidad con su nuevo estado
		{
		"mensaje": "La funcionalidad ha sido activado exitosamente.",
		"data": [
		{
		"idFuncionalidad ": 1,
		"nombre": "Listado de funcionalidades",
		"descripcion": "Acceder al listado de usuarios",
		"estado": "ACTIVO",
		}
		]
		}
		La funcionalidad queda activa y puede ser vinculada a otra entidad.

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-439: Baja de funcionalidad por usuario no autorizado

Resumen:

Verifica que solo un usuario con rol *admin* pueda dar de baja funcionalidades.

Precondiciones:

Usuario autenticado con rol **no administrador (Socio o No Socio)**.

Funcionalidad objetivo existe y está activo.

Se dispone de un **token JWT** (Se obtiene al iniciar sesión).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/funcionalidades/baja/{id}, con un id de una funcionalidad existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado de la funcionalidad. Ejemplo:
	{
	"nuevoEstado": "INACTIVO"
	}	

4	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-440: Baja con funcionalidad no existente

Resumen:

Verifica la respuesta del sistema al intentar dar de baja a una funcionalidad inexistente.

Precondiciones:

ID de funcionalidad no existente en la base de datos.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado tiene rol admin

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/funcionalidades/baja/{id}, con un id de una funcionalidad no existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado de la funcionalidad. Ejemplo:
	{
	"nuevoEstado": "INACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **404 Not Found.** La respuesta contiene un mensaje claro describiendo que la funcionalidad no existe.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-441: Baja de funcionalidad ya inactivo

Resumen:

Verifica que el sistema no permita dar de baja a una funcionalidad que ya se encuentra inactiva.

Precondiciones:

La funcionalidad está en estado **inactiva**.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado tiene rol admin

Nº:	Pasos:	Resultados Esperados:

1	Construir una solicitud PUT al endpoint /api/funcionalidades/baja/{id}, con un id de una funcionalidad existente con estado INACTIVO	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el estado de la funcionalidad. Ejemplo:
	{
	"nuevoEstado": "INACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **409 Conflict.** La respuesta contiene un mensaje claro describiendo el error. Ningún dato de la funcionalidad es modificado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Media	

Caso de Prueba PFTDZGR06-442: Baja exitosa de funcionalidad

Resumen:

Verifica que un administrador pueda dar de baja correctamente a una funcionalidad activa.

Precondiciones:

La funcionalidad existe y está en estado **activa**.

El usuario logueado tiene rol admin.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/funcionalidades/baja/{id}, con un id de una funcionalidad existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado de la funcionalidad. Ejemplo:
	{
	"nuevoEstado": "INACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **200 OK**.
		La respuesta contiene un mensaje de exito y el objeto de la funcionalidad con su nuevo estado
		{
		"mensaje": "La funcionalidad ha sido dado de baja exitosamente.",
		"data": [
		{
		"idFuncionalidad ": 1,
		"nombre": "Listado de usuarios",
		"descripcion": "Acceso a listado de usuarios",
		"estado": "INACTIVO",
		}
		]
		}
		La funcionalidad queda inactiva y no puede ser vinculada a otra entidad.

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-443: Validar restricción de uso de funcionalidades dadas de baja

Resumen:

Verificar que el sistema **no permita asociar ni relacionar** una funcionalidad que se encuentra dada de baja (inactiva) con otros registros, como usuarios u otras entidades dependientes.

Precondiciones:

Existe una funcionalidad previamente dada de baja (estado inactiva).

Endpoint de vinculación de funcionalidad a perfiles disponible.

Usuario autenticado con rol administrador.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/permisos/acceso-funcionalidad, con un id de una funcionalidad existente con estado INACTIVO	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	construir el siguiente **body JSON** incluyendo el id de la funcionalidad con estado **INACTIVO**:
	{
	"idPerfil": 1,
	"funcionalidadesIds": [4, 5, 6 (funcionalidad inactiva)]
	}	

4	Enviar solicitud	Código de respuesta: **422 Unprocessable Entity**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No es posible asignar una funcionalidad inactiva al perfil."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Suite de Pruebas : Acceso a Funcionalidades

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint acceso de funcionalidad

Caso de Prueba PFTDZGR06-444: Vincular funcionalidades a un perfil exitosamente

Resumen:

Verificar que el sistema permita vincular correctamente una o varias funcionalidades activas a un perfil existente.

Precondiciones:

El perfil indicado existe y está activo.

Las funcionalidades enviadas existen y están activas.

El usuario autenticado tiene rol de administrador.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/permisos/acceso-funcionalidad	

2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	construir el siguiente **body JSON.**
	Ejemplo:
	{
	"idPerfil": 1,
	"funcionalidadesIds": [4, 5, 6]
	}	

4	Enviar solicitud	código de respuesta: **200 OK** mensaje de exito: "Funcionalidades vinculadas correctamente al perfil"
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-445: Vincular funcionalidades a un perfil no existente

Resumen:

Verificar que el sistema no permita vincular funcionalidades a un perfil que no existe.

Precondiciones:

El ID de perfil no existe en el sistema.

Estar logueado con un usuario de tipo administrador

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/permisos/acceso-funcionalidad, asignando el id de un perfil no existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	construir el siguiente **body JSON** incluyendo el id de un perfil no existente:
	{
	"idPerfil": 10 (no existe),
	"funcionalidadesIds": [4, 5, 6]
	}	

4	Enviar solicitud	El código de respuesta: **404 Not Found.** La respuesta contiene un mensaje claro describiendo que el perfil no existe.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-446: Vincular funcionalidades no existentes a un perfil

Resumen:

Verificar que el sistema rechace la vinculación cuando alguna funcionalidad no exista.

Precondiciones:

El perfil existe y está activo.

Alguna funcionalidad del body no existe.

El usuario autenticado tiene rol de administrador.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/permisos/acceso-funcionalidad, asignando el id de una funcionalidad no existente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	construir el siguiente **body JSON** incluyendo el id de la funcionalidad que no existe:
	{
	"idPerfil": 1,
	"funcionalidadesIds": [4, 5, 6 (funcionalidad inexistente)]
	}	

4	Enviar solicitud	El código de respuesta: **404 Not Found.** La respuesta contiene un mensaje claro describiendo que la funcionalidad no existe.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-447: Vincular funcionalidades inactivas a un perfil

Resumen:

Verificar que el sistema rechace la vinculación cuando alguna funcionalidad esté inactiva.

Precondiciones:

El perfil existe y está activo.

Alguna funcionalidad del body está inactiva.

El usuario autenticado tiene rol de administrador.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/permisos/acceso-funcionalidad, asignando el id de una funcionalidad inactiva	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	construir el siguiente **body JSON** incluyendo el id de la funcionalidad con estado **INACTIVO**:
	{
	"idPerfil": 1,
	"funcionalidadesIds": [4, 5, 6 (funcionalidad inactiva)]
	}	

4	Enviar solicitud	Código de respuesta: **422 Unprocessable Entity**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No es posible asignar una funcionalidad inactiva al perfil."
		}

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	2.00
Prioridad:	Alta

Caso de Prueba PFTDZGR06-448: Vincular funcionalidades a un perfil inactivo

Resumen:

Verificar que el sistema rechace la vinculación cuando el perfil seleccionado esté inactivo.

Precondiciones:

El perfil existe y está inactivo.

Existen funcionalidades activas.

El usuario autenticado tiene rol de administrador.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/permisos/acceso-funcionalidad, asignando el id de un perfil con estado inactivo	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	construir el siguiente **body JSON** incluyendo el id de perfil con estado **INACTIVO**:
	{
	"idPerfil": 1 (perfil inactivo),
	"funcionalidadesIds": [4, 5, 6]
	}	

4	Enviar solicitud	Código de respuesta: **422 Unprocessable Entity**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No es posible vincular un perfil inactivo a las funcionalidades."
		}

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

Caso de Prueba PFTDZGR06-449: Vincular funcionalidad por usuario no autorizado

Resumen:

Verificar que solo usuarios administradores puedan realizar la vinculación.

Precondiciones:

El usuario autenticado no tiene permisos de administrador.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/permisos/acceso-funcionalidad	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	construir el siguiente **body JSON.**
	Ejemplo:
	{
	"idPerfil": 1,
	"funcionalidadesIds": [4, 5, 6]
	}	

4	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	5.00
Prioridad:	Alta

Caso de Prueba PFTDZGR06-450: Vincular funcionalidades a un perfil con token inválido o ausente

Resumen:

Verificar que el sistema rechace solicitudes sin autenticación válida.

Precondiciones:

Token JWT ausente o inválido.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/permisos/acceso-funcionalidad	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token inválido>	

3	construir el siguiente **body JSON.**
	Ejemplo:
	{
	"idPerfil": 1,
	"funcionalidadesIds": [4, 5, 6]
	}	

4	Enviar solicitud	Código de respuesta: **401 Unauthorized**

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

Caso de Prueba PFTDZGR06-451: Vincualr funcionalidades ya vinculadas previamente

Resumen:

Verificar que el sistema maneje correctamente la solicitud cuando las funcionalidades ya están asociadas al perfil.

Precondiciones:

Las funcionalidades ya están vinculadas al perfil.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/permisos/acceso-funcionalidad	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	construir el siguiente **body JSON.**
	Ejemplo:
	{
	"idPerfil": 1,
	"funcionalidadesIds": [4, 5, 6 (funcionalidad ya vinculada)]
	}	

4	Enviar solicitud	El código de respuesta: **409 Conflict.** La respuesta contiene un mensaje claro describiendo el error. Ningún dato es modificado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Media	

Suite de Pruebas : Modificación de Acceso a Funcionalidades

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint modificación de acceso a funcionalidad

Caso de Prueba PFTDZGR06-458: Modificar acceso a funcionalidades a un perfil con token inválido o ausente

Resumen:

Verificar que el sistema rechace solicitudes sin autenticación válida.

Precondiciones:

Token JWT ausente o inválido.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/permisos/remover-acceso-funcionalidad	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token inválido>	

3	construir el siguiente **body JSON.**
	Ejemplo:
	{
	"idPerfil": 1,
	"funcionalidadesIds": [4, 5, 6]
	}	

4	Enviar solicitud	Código de respuesta: **401 Unauthorized**
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-457: Modificar acceso a funcionalidad por usuario no autorizado

Resumen:

Verificar que solo usuarios administradores puedan realizar la vinculación.

Precondiciones:

El usuario autenticado no tiene permisos de administrador.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/permisos/remover-acceso-funcionalidad	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	construir el siguiente **body JSON.**
	Ejemplo:
	{
	"idPerfil": 1,
	"funcionalidadesIds": [4, 5, 6]
	}	

4	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	5.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-452: Modificar acceso de una funcionalidad en un perfil exitosamente

Resumen:

Verificar que el sistema permita modificar correctamente el acceso (activar o desactivar) de una funcionalidad para un perfil existente.

Precondiciones:

El perfil y la funcionalidad existen y están activos.

El usuario autenticado tiene rol **administrador**

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/permisos/remover-acceso-funcionalidad	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	construir el siguiente **body JSON.**
	Ejemplo:
	{
	"idPerfil": 1,
	"funcionalidadesIds": [4, 5, 6]
	}	

4	Enviar solicitud	código de respuesta: **200 OK** mensaje de exito: "Funcionalidades desvinculadas correctamente del perfil"
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-453: Modificar acceso de una funcionalidad en un perfil inexistente

Resumen:

Verificar que el sistema no permita modificar el acceso si el perfil no existe.

Precondiciones:

ID de perfil inexistente.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/permisos/remover-acceso-funcionalidad	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	construir el siguiente **body JSON**, incluyendo el id de un perfil inexistente.
	Ejemplo:
	{
	"idPerfil": 1 (no existe),
	"funcionalidadesIds": [4, 5, 6]
	}	

4	Enviar solicitud	El código de respuesta: **404 Not Found.** La respuesta contiene un mensaje claro describiendo que el perfil no existe.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-454: Modificar acceso de una funcionalidad inexistente en un perfil

Resumen:

Verificar que el sistema devuelva error si la funcionalidad no existe.

Precondiciones:

El perfil existe.

ID de funcionalidad inexistente.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/permisos/remover-acceso-funcionalidad	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	construir el siguiente **body JSON**, incluyendo el id de una funcionalidad no existente.
	Ejemplo:
	{
	"idPerfil": 1,
	"funcionalidadesIds": [4, 5, 6 (funcionalidad no existente)]
	}	

4	Enviar solicitud	El código de respuesta: **404 Not Found.** La respuesta contiene un mensaje claro describiendo que la funcionalidad no existe.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-455: Modificar acceso de una funcionalidad en un perfil inactivo

Resumen:

Verificar que el sistema impida modificar accesos si el perfil está dado de baja.

Precondiciones:

El perfil existe pero está inactivo.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/permisos/remover-acceso-funcionalidad	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	construir el siguiente **body JSON**, incluyendo el id de un perfil con estado INACTIVO.
	Ejemplo:
	{
	"idPerfil": 1 (perfil inactivo),
	"funcionalidadesIds": [4, 5, 6]
	}	

4	Enviar solicitud	Código de respuesta: **422 Unprocessable Entity**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No es posible realizar acciones sobre un perfil inactivo."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-456: Modificar acceso de una funcionalidad inactiva en un perfil

Resumen:

Verificar que el sistema impida modificar accesos si la funcionalidad está inactiva.

Precondiciones:

La funcionalidad existe pero está inactiva.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/permisos/remover-acceso-funcionalidad	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	construir el siguiente **body JSON**, incluyendo el id de una funcionalidad con estado inactiva.
	Ejemplo:
	{
	"idPerfil": 1,
	"funcionalidadesIds": [4, 5, 6 (funcionalidad inactiva)]
	}	

4	Enviar solicitud	Código de respuesta: **422 Unprocessable Entity**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No es posible realizar acciones sobre una funcionalidad inactiva."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Suite de Pruebas : Gestión de Actividades

Conjunto de pruebas diseñado para validar el correcto funcionamiento de todos los endpoints del módulo de actividades, garantizando la integridad de las operaciones CRUD, seguridad y rendimiento de las APIs.

Ejemplos de parámetros válidos para los campos de Gestión de Actividades:

- Nombre de Actividad: Taller de Arte
- Descripcion de Actividad: Taller práctico de técnicas de pintura al óleo
- Objetivo: Desarrollar habilidades artísticas en adultos
- Fecha de Actividad: 2025-12-15
- Hora de Actividad: 15:00:00
- Duración: 3
- Inscripción: true
- Costo: 250
- Fecha de Inscripción: 2025-11-20
- Tipo de Pago: EFECTIVO
- Observaciones: Traer materiales básicos
- Id de Tipo de Actividad: 2
- Id de Espacio: 3
- Estado: ACTIVO
- Estado de Actividad: PROGRAMADA

Ejemplos para parámetros inválidos:

- Nombre de Actividad: "" | "A" * 201 | "Taller@2024"
- Descripcion de Actividad: "" | "A" * 1001
- Objetivo: "" | "A" * 1001
- Fecha de Actividad: 2023-01-01 (fecha pasada) | 2025-13-45 (fecha inexistente)
- Hora de Actividad: 25:00:00 (hora inválida) | 14:60:00 (minutos inválidos)
- Duración: -2 | 25 | dos
- Inscripción: si | ""
- Costo: -100 | 1000000
- Fecha de Inscripción: 2025-12-11 (posterior a fecha de actividad)
- Tipo de Pago: "BITCOIN" | "CRIPTO" | ""
- Observaciones: "" | "A" * 1001
- Id de Tipo de Actividad: 999 | 0 | -1
- Id de Espacio: 999 | 0 | -1
- Estado: activo | "" | 0
- Estado de Actividad: prog | "" | 0

Suite de Pruebas : Registro de Actividad

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint registro de actividad

Caso de Prueba PFTDZGR06-333: Registrar actividad por parte de usuario no autorizado

Resumen:

Verifica que solo un usuario con rol *admin* pueda registrar nuevas actividades.

Precondiciones:

El endpoint POST /api/actividades está disponible y operativo.

Se dispone de un **token JWT** (Se obtiene al iniciar sesión)

Usuario autenticado sin rol admin (por ejemplo, socio o no socio)

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/actividades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Construir una solicitud con el siguiente **body JSON.**
	Ejemplo:
	{
	"actividadNom": "",
	"actividadDescripcion": "",
	"objetivo": "",
	"fecActividad": "",
	"horaActividad": "00:00:00",
	"duracion": 0,
	"inscripcion": true,
	"costo": 0,
	"fecInscripcion": "",
	"tipoPago": "",
	"actObservaciones": "",
	"idTipoActividad": 1,
	"idEspacio": 1
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	4.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-332: Registrar actividad con datos duplicado

Resumen:

Verificar que el endpoint /api/actividades **rechace solicitudes de registro** cuando los datos enviados ya existen en la base de datos, asegurando que **no se creen actividades duplicados**.
Esto incluye campos únicos como **nombre**.

Precondiciones:

El endpoint /api/actividades está activo y accesible.

La actividad no está previamente registrada

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/actividades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	Incluir en el **body JSON** los datos de un usuario que ya exista en la base de datos. Ejemplo: Construir una solicitud con el siguiente **body JSON.** Ejemplo: { "actividadNom": "", "actividadDescripcion": "",	

	"objetivo": "", "fecActividad": "", "horaActividad": "00:00:00", "duracion": 0, "inscripcion": true, "costo": 0, "fecInscripcion": "", "tipoPago": "", "actObservaciones": "", "idTipoActividad": 1, "idEspacio": 1 } (Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	
4	Enviar la solicitud.	Código de respuesta: **409 Conflict**. Mensaje de error en la respuesta indicando que la actividad ya existe. No se crea una nueva actividad en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-331: Registrar actividad con campos obligatorios faltantes

Resumen:

Verificar que el endpoint /api/actividades rechace correctamente solicitudes de registro cuando uno o más **campos obligatorios** no son enviados o están vacíos.

Precondiciones:

El endpoint /api/actividades está activo y accesible.

La actividad no está previamente registrado

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/actividades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el **body JSON** los campos del usuario, dejando vacío uno o varios campos obligatorios. Ejemplo:
	Construir una solicitud con el siguiente **body JSON.**
	Ejemplo:
	{
	"actividadNom": "",
	"actividadDescripcion": "",
	"objetivo": "",
	"fecActividad": "",
	"horaActividad": "00:00:00",
	"duracion": null,
	"inscripcion": null,
	"costo": null,
	"fecInscripcion": "",
	"tipoPago": "",
	"actObservaciones": "",
	"idTipoActividad": null,
	"idEspacio": null
	}
	(null y "" cuentan como vacios)	

4	Enviar la solicitud	El código de respuesta es **400 Bad Request**. La respuesta contiene un mensaje claro describiendo los **campos obligatorios faltantes**. Ninguna actividad es creada en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-330: Registrar actividad con datos inválidos

Resumen:

Verificar que el endpoint /api/actividades rechace correctamente solicitudes de registro cuando uno o más campos contienen **datos inválidos**, respetando las reglas de validación del sistema.

Precondiciones:

El endpoint /api/actividades está activo y accesible.

La actividad no está previamente registrada

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/actividades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el **body JSON** los campos requeridos, reemplazando uno o varios con valores inválidos. Ejemplo:
	Construir una solicitud con el siguiente **body JSON.**
	Ejemplo:
	{
	"actividadNom": "",
	"actividadDescripcion": "",
	"objetivo": "",
	"fecActividad": "",
	"horaActividad": "00:00:00",
	"duracion": 0,
	"inscripcion": true,
	"costo": 0,
	"fecInscripcion": "",
	"tipoPago": "",
	"actObservaciones": "",
	"idTipoActividad": 1,
	"idEspacio": 1
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar la solicitud.	El código de respuesta es **400 Bad Request** La respuesta incluye un **mensaje de error** describiendo los campos inválidos.

		Ninguna actividad es creada en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	5.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-329: Registrar actividad exitosamente

Resumen:

Registro de actividad con todos los campos del formulario completos con datos válidos

Precondiciones:

La actividad no debe estar registrada en el sistema.

Endpoint POST /api/actividades disponible.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/actividades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Construir una solicitud con el siguiente **body JSON.**
	Ejemplo:
	{
	"actividadNom": "",
	"actividadDescripcion": "",
	"objetivo": "",
	"fecActividad": "",
	"horaActividad": "00:00:00",
	"duracion": 0,
	"inscripcion": true,
	"costo": 0,
	"fecInscripcion": "",
	"tipoPago": "",
	"actObservaciones": "",
	"idTipoActividad": 1,
	"idEspacio": 1
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar la solicitud	El código de respuesta es **201 Created**.
		El cuerpo de la respuesta contiene un mensaje y el objeto de la actividad recién creada:
		{
		"message": "La actividad ha sido registrada correctamente y se encuentra disponible en la lista de actividades.",
		"data": {
		"idActividad": <autogenerado>,
		"actividadNom": "Clase de danza",
		"actividadDescripcion": "CLase con maestro especializado",
		...
		}
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-370: Registrar actividad con solapamiento de horarios

Resumen:

Verificar que el sistema impida el registro de una actividad cuyo horario coincide parcial o totalmente con otra ya registrada.

Precondiciones:

Existe otra actividad con horario que se solapa total o parcialmente.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/actividades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Construir una solicitud con el siguiente **body JSON.**
	Ejemplo:
	{
	"actividadNom": "",
	"actividadDescripcion": "",
	"objetivo": "",
	"fecActividad": "2025-12-22",
	"horaActividad": "14:30:00",
	"duracion": 0,
	"inscripcion": true,
	"costo": 0,
	"fecInscripcion": "",
	"tipoPago": "",
	"actObservaciones": "",
	"idTipoActividad": 1,
	"idEspacio": 1
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)
	Ya existe una actividad el 2025-12-22 a las 14:30	

4	Enviar solicitud	Código de respuesta: **409 Conflict**. Mensaje de error en la respuesta indicando que el espacio ya está reservado para ese momento. mensaje: El espacio no está disponible en el horario seleccionado No se crea una nueva actividad en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	5.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-371: Registrar actividades consecutivas sin solapamiento

Resumen:

Verificar que el sistema permita el registro de una actividad cuando las actividades terminan e inician en horarios contiguos (sin superposición).

Precondiciones:

Existe otra actividad que inicia cuando termina la nueva actividad (con margen de 30 minutos).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/actividades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Construir una solicitud con el siguiente **body JSON.**
	Ejemplo:
	{
	"actividadNom": "",
	"actividadDescripcion": "",
	"objetivo": "",
	"fecActividad": "2025-12-10",
	"horaActividad": "16:00:00",
	"duracion": 0,
	"inscripcion": true,
	"costo": 0,
	"fecInscripcion": "",
	"tipoPago": "",
	"actObservaciones": "",
	"idTipoActividad": 1,
	"idEspacio": 1
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)
	Ya existe una actividad el 2025-12-10 que termina a las 15:30	

4	Enviar solicitud	El código de respuesta es **201 Created**.
		El cuerpo de la respuesta contiene un mensaje y el objeto de la actividad recién creada:
		{
		"message": "La actividad ha sido registrada correctamente y se encuentra disponible en la lista de actividades.",
		"data": {
		"idActividad": <autogenerado>,
		"actividadNom": "Clase de danza",
		"actividadDescripcion": "CLase con maestro especializado",
		...
		}
		}

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

Suite de Pruebas : Listado de Actividades

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint listado de actividades

Caso de Prueba PFTDZGR06-339: Acceso a listado de actividades exitosamente

Resumen:

Verificar que el endpoint GET /api/actividades devuelva correctamente la lista de actividades registradas en el sistema, asegurando que el formato de respuesta y el código HTTP sean los esperados.

Precondiciones:

El endpoint GET /api/actividades está disponible.

Existe al menos una actividad registrada en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado debe ser de tipo *Admin*

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/actividades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar la solicitud.	Código de respuesta: **200 OK**
		El cuerpo de la respuesta contiene un objeto con el listado de actividades y un mensaje de exito, por ejemplo:
		{
		"mensaje": "Listado de actividades obtenido correctamente.",
		"data": [
		{
		"idActividad": 1,
		"actividadNom": "Clase de danza",
		"actividadDescripcion": "Clase con profesor especializado",
		"estado": "PROGRAMADA",
		...
		},
		{
		"idActividad": 2,
		"actividadNom": "Clase de Arte",
		"actividadDescripcion": "Arte abstracto",
		"estado": "CANCELADA",
		...
		}
		]
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-338: Acceso a listado de actividades por Usuario Admin

Resumen:

Verificar que el endpoint GET /api/actividades devuelva correctamente la lista de actividades registradas si el usuarios es de tipo ADMIN

Precondiciones:

El endpoint GET /api/actividades está disponible.

Existe al menos una actividad registrada en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado tiene perfil ADMIN

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/actividades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	Enviar solicitud	Código de respuesta: **200 OK** El cuerpo de la respuesta contiene un objeto con el listado de actividades y un mensaje de exito, por ejemplo: { "mensaje": "Listado de actividades obtenido correctamente.", "data": [{ "idActividad": 1, "actividadNom": "Clase de danza", "actividadDescripcion": "Clase con profesor especializado",

		"estado": "PROGRAMADA",
		...
		},
		{
		"idActividad": 2,
		"actividadNom": "Clase de Arte",
		"actividadDescripcion": "Arte abstracto",
		"estado": "CANCELADA",
		...
		}
		]
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-337: Acceso a listado de actividades por Usuario Socio/No Socio

Resumen:

Verificar que el endpoint GET /api/actividades no devuelva la lista de actividades registradas en el sistema si el usuario que intenta acceder es de tipo Socio o No Socio, asegurando que el formato de respuesta y el código HTTP sean los esperados.

Precondiciones:

El endpoint GET /api/actividades está disponible.

Existe al menos una actividad registrada en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado debe ser de tipo Socio o No Socio

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/actividades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-336: Filtrado del listado de actividades

Resumen:

Verificar que el endpoint GET /api/actividades devuelva correctamente la lista de actividades registradas en el sistema de acuerdo al filtro seleccionado, asegurando que el formato de respuesta y el código HTTP sean los esperados.

Precondiciones:

El endpoint GET /api/actividades está disponible.

Existe al menos una actividad registrada en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado debe ser ADMIN

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/actividades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	(Opcional) Agregar parámetros de filtro válidos, por ejemplo: • /api/actividades?nombre=Taller • /api/actividades?tipoActividad=3 • /api/actividades?fecha=2025-12-05 • /api/actividades?estado=ACTIVO • /api/actividades?costo=200 Los filtros se pueden combinar usando &	

4	Enviar solicitud	Código de respuesta: **200 OK**
		El cuerpo de la respuesta contiene un objeto con el listado de actividades que coinciden con los filtros aplicados y un mensaje de exito, por ejemplo:
		{
		"mensaje": "Listado de actividades obtenido correctamente.",
		"data": [
		{
		"idActividad": 1,
		"actividadNom": "Clase de danza",
		"actividadDescripcion": "Clase con profesor especializado",
		"estado": "PROGRAMADA",
		...
		},
		{
		"idActividad": 2,
		"actividadNom": "Clase de Arte",
		"actividadDescripcion": "Arte abstracto",
		"estado": "CANCELADA",
		...
		}
		]
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-335: Acceder a listado sin actividades registradas

Resumen:

Verificar que el endpoint GET /api/actividades devuelva una lista vacía de actividades y el código HTTP sea lo esperado.

Precondiciones:

El endpoint GET /api/actividades está disponible.

No existen actividades registradas en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado es de tipo ADMIN

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/actividades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviaer solicitud	Código de respuesta: **404 Not Found**
		El cuerpo de la respuesta contiene un objeto con el listado vacío de actividades y un mensaje de informativo, por ejemplo:
		{
		"mensaje": "No se encontraron actividades que coincidan con los filtros.",
		"data": [ ]
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-334: Acceder a listado de actividad sin token de autorización

Resumen:

Validar que el sistema no permita acceder al listado de actividades cuando el token de autenticación está ausente o es inválido, devolviendo el código de estado y mensaje de error correspondiente.

Precondiciones:

El endpoint GET /api/actividades requiere autenticación mediante token JWT

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/actividades	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token inválido>	
3	Enviar solicitud	Código de respuesta: **401 Unauthorized**
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Suite de Pruebas : Modificación de Actividades

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint modificación de actividades

Caso de Prueba PFTDZGR06-345: Modificar actividad no existente

Resumen:

Verificar que el sistema retorne un error cuando se intenta modificar una actividad que no existe.

Precondiciones:

El endpoint PUT /api/actividades/{id} está disponible y operativo.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado es de tipo *Admin*

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/actividades/{id}, incluyendo un id de actividad inexistente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Construir una solicitud con el siguiente **body JSON.**
	Ejemplo:
	{
	"actividadDescripcion": "",
	"objetivo": "",
	"fecActividad": "",
	"horaActividad": "00:00:00",
	"duracion": 0,
	"inscripcion": true,
	"costo": 0,
	"fecInscripcion": "0",
	"tipoPago": "",
	"actObservaciones": "",
	"estado": "",
	"estadoActividad": "",
	"idTipoActividad": 1,
	"idEspacio": 1
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	El código de respuesta: **404 Not Found.** La respuesta contiene un mensaje claro describiendo que la actividad no existe.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-344: Modificar actividad exitosamente

Resumen:

Verificar que el sistema permita modificar correctamente los datos de una actividad existente cuando la solicitud es válida y proviene de un usuario autorizado.

Precondiciones:

El endpoint PUT /api/actividades/{id} está disponible y operativo.

Existe al menos una actividad en la base de datos

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

El usuario logueado posee rol de *admin*.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/actividades/{id}	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Construir una solicitud con el siguiente **body JSON.**
	Ejemplo:
	{
	"actividadDescripcion": "",
	"objetivo": "",
	"fecActividad": "",
	"horaActividad": "00:00:00",
	"duracion": 0,
	"inscripcion": true,
	"costo": 0,
	"fecInscripcion": "0",
	"tipoPago": "",
	"actObservaciones": "",
	"estado": "",
	"estadoActividad": "",
	"idTipoActividad": 1,
	"idEspacio": 1
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	El código de respuesta es **200 OK**.
		El cuerpo de la respuesta contiene un mensaje y el objeto de la actividad modificada:
		{
		"message": "La actividad ha sido actualizada correctamente.",
		"data": {
		"idActividad": 1,
		"actividadNom": "Clase de danza",
		"actividadDescripcion": "Clase grupal",
		...
		}
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-343: Modificar actividad con datos obligatorios faltantes

Resumen:

Verifica que el sistema no permita actualizar una actividad si faltan campos obligatorios.

Precondiciones:

El endpoint PUT /api/actividades/{id} está disponible y operativo.

Existe al menos una actividad registrada en la base de datos

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado es de tipo *Admin*

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/actividades/{id}	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Construir una solicitud con el siguiente **body JSON,** dejando vacío campos obligatorios.
	Ejemplo:
	{
	"actividadDescripcion": "",
	"objetivo": "",
	"fecActividad": "",
	"horaActividad": "00:00:00",
	"duracion": null,
	"inscripcion": true,
	"costo": 0,
	"fecInscripcion": "0",
	"tipoPago": "",
	"actObservaciones": null,
	"estado": "",
	"estadoActividad": "",
	"idTipoActividad": 1,
	"idEspacio": 1
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)
	(null y "" cuentan como vacío)	

4	Enviar solicitud	El código de respuesta: **400 Bad Request** La respuesta incluye un **mensaje de error** describiendo los campos faltantes. Ninguna actividad es modificada en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-342: Modificar actividad con datos inválidos

Resumen:

Verifica que el sistema rechace modificaciones con datos en formato incorrecto o inválido.

Precondiciones:

El endpoint PUT /api/actividades/{id} está disponible y operativo.

Existe al menos una actividad registrada en la base de datos

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado es de tipo *Admin*

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/actividades/{id}	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Construir una solicitud con el siguiente **body JSON,** ingresando datos inválidos en los campos.
	Ejemplo:
	{
	"actividadDescripcion": "",
	"objetivo": "",
	"fecActividad": "",
	"horaActividad": "00:00:00",
	"duracion": 0,
	"inscripcion": true,
	"costo": 0,
	"fecInscripcion": "0",
	"tipoPago": "",
	"actObservaciones": "",
	"estado": "",
	"estadoActividad": "",
	"idTipoActividad": 1,
	"idEspacio": 1
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	El código de respuesta: **400 Bad Request** La respuesta incluye un **mensaje de error** describiendo los campos inválidos y su formato correcto. Ninguna actividad es modificada en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-340: Modificar actividad por parte de un usuario no autorizado

Resumen:

Verificar que los usuarios que no poseen el rol administrador no puedan modificar actividades.

Precondiciones:

El endpoint PUT /api/actividades/{id} está disponible y operativo.

Existe una actividad en la base de datos

Se dispone de un **token JWT** (Se obtiene al iniciar sesión)

Usuario autenticado sin rol admin (por ejemplo, socio o no socio)

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/actividades/{id}	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Construir una solicitud con el siguiente **body JSON.**
	Ejemplo:
	{
	"actividadDescripcion": "",
	"objetivo": "",
	"fecActividad": "",
	"horaActividad": "00:00:00",
	"duracion": 0,
	"inscripcion": true,
	"costo": 0,
	"fecInscripcion": "0",
	"tipoPago": "",
	"actObservaciones": "",
	"estado": "",
	"estadoActividad": "",
	"idTipoActividad": 1,
	"idEspacio": 1
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	2.00
Prioridad:	Alta

Caso de Prueba PFTDZGR06-372: Modificación de horario de una actividad que genera solapamiento

Resumen:

Verificar que, al modificar el horario de una actividad, el sistema detecte si genera conflictos de horario con actividades existentes y activas.

Precondiciones:

Existen una actividad A.

Se intenta modificar el horario de la actividad A para que se superponga con la actividad B, en la cual también está activa en el horario seleccionado.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/actividades/{id}	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Construir una solicitud con el siguiente **body JSON.**
	Ejemplo:
	{
	"actividadDescripcion": "",
	"objetivo": "",
	"fecActividad": "2025-12-08",
	"horaActividad": "10:00:00",
	"duracion": 0,
	"inscripcion": true,
	"costo": 0,
	"fecInscripcion": "0",
	"tipoPago": "",
	"actObservaciones": "",
	"estado": "",
	"estadoActividad": "",
	"idTipoActividad": 1,
	"idEspacio": 1
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	Código de respuesta: **409 Conflict**. Mensaje de error en la respuesta indicando que el espacio ya está reservado para ese momento. mensaje: El espacio no está disponible en el horario seleccionado No se actualiza la información de la actividad en la base de datos.

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

Suite de Pruebas : Baja/Reactivación de Actividad

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint baja de actividad y reactivación de actividades

Caso de Prueba PFTDZGR06-354: Validar restricción de uso de actividades dados de baja

Resumen:

Verificar que el sistema **no permita asociar ni relacionar** una actividad que se encuentra dado de baja (inactivo) con otros registros, como usuarios u otras entidades dependientes.

Precondiciones:

Existe una actividad previamente dada de baja (estado inactivo).

Usuario autenticado con rol administrador.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/inscripciones/{idActividad}, incluyendo un id de actividad con estado inactivo	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar solicitud	Código de respuesta: **422 Unprocessable Entity**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "La actividad seleccionada está inactiva y no acepta inscripciones"
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-353: Baja exitosa de actividad

Resumen:

Verifica que un administrador pueda dar de baja correctamente a una actividad activo.

Precondiciones:

La actividad existe y está en estado **activo**.

El usuario logueado tiene rol admin.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/actividades/baja/{id}	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado de la actividad. Ejemplo:
	{
	"nuevoEstado": "INACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **200 OK**.
		La respuesta contiene un mensaje de exito y el objeto de la actividad con su nuevo estado
		{
		"mensaje": "La actividad ha sido dado de baja exitosamente.",
		"data": [
		{
		"idActividad": 1,
		"actividadNom": "Clase de Arte",
		"actividadDescripcion": "Arte abstracto",
		"estado": "INACTIVO",
		}
		]
		}
		La actividad queda inactiva y no puede ser vinculada a otra entidad o realizar más acciones sobre ella.

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-352: Baja de actividad ya inactivo

Resumen:

Verifica que el sistema no permita dar de baja a una actividad que ya se encuentra inactiva.

Precondiciones:

La actividad está en estado **inactiva**.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado tiene rol admin

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/actividades/baja/{id}, incluyendo un id de actividad con estado inactivo	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el estado de la actividad. Ejemplo:
	{
	"nuevoEstado": "INACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **409 Conflict.** La respuesta contiene un mensaje claro describiendo el error. Ningún dato de la actividad es modificado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Media	

Caso de Prueba PFTDZGR06-351: Baja con actividad no existente

Resumen:

Verifica la respuesta del sistema al intentar dar de baja a una actividad inexistente.

Precondiciones:

ID de actividad no existente en la base de datos.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado tiene rol admin

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/actividades/baja/{id}, incluyendo un id de actividad que no existe	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	Incluir en el body JSON el nuevo estado de la actividad. Ejemplo: { "nuevoEstado": "INACTIVO"	

	}	
4	Enviar solicitud	El código de respuesta: **404 Not Found.** La respuesta contiene un mensaje claro describiendo que la actividad no existe.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-350: Baja de actividad por usuario no autorizado

Resumen:

Verifica que solo un usuario con rol *admin* pueda dar de baja actividades.

Precondiciones:

Usuario autenticado con rol **no administrador (Socio o No Socio)**.

Actividad objetivo existe y está activo.

Se dispone de un **token JWT** (Se obtiene al iniciar sesión).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/actividades/baja/{id}	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado de la actividad. Ejemplo:
	{
	"nuevoEstado": "INACTIVO"
	}	

4	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-348: Reactivar actividad ya activo

Resumen:

Verifica que el sistema no permita reactivar una actividad que ya se encuentra activo.

Precondiciones:

Actividad con estado **activo**.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado tiene rol admin

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/actividades/reactivar/{id}, incluyendo un id de actividad con estado activo	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el estado de la actividad. Ejemplo:
	{
	"nuevoEstado": "ACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **409 Conflict.** La respuesta contiene un mensaje claro describiendo el error. Ningún dato de la actividad es modificado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Media	

Caso de Prueba PFTDZGR06-349: Reactivar actividad exitosamente

Resumen:

Verifica que un administrador pueda reactivar correctamente una actividad dada de baja.

Precondiciones:

Actividad existente en estado **inactivo**.

El usuario logueado tiene rol admin.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/actividades/reactivar/{id}	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado de la actividad. Ejemplo:
	{
	"nuevoEstado": "ACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **200 OK**.
		La respuesta contiene un mensaje de exito y el objeto de la actividad con su nuevo estado
		{
		"mensaje": "La actividad ha sido activada exitosamente.",
		"data": [
		{
		"idActividad": 1,
		"actividadNom": "Clase de Arte",
		"actividadDescripcion": "Arte abstracto",
		"estado": "ACTIVO",
		}
		]
		}
		La actividad queda activa y puede ser vinculada a otra entidad o realizar acciones sobre ella.

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-347: Reactivar actividad no existente

Resumen:

Verifica la respuesta al intentar reactivar una actividad que no existe.

Precondiciones:

ID de actividad no existente en la base de datos.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado tiene rol admin

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/actividades/reactivar/{id}, incluyendo un id de actividad que no existe	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado de la actividad. Ejemplo:
	{
	"nuevoEstado": "ACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **404 Not Found.** La respuesta contiene un mensaje claro describiendo que la actividad no existe.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-346: Reactivar actividad por usuario no autorizado

Resumen:

Verifica que solo administradores puedan reactivar actividades.

Precondiciones:

Usuario autenticado con rol **no administrador (Socio o No Socio)**.

Actividad objetivo existe y está activo.

Se dispone de un **token JWT** (Se obtiene al iniciar sesión).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/actividades/reactivar/{id}	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado de la actividad. Ejemplo:
	{
	"nuevoEstado": "ACTIVO"
	}	

4	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-418: Baja de actividad con periodo de inscripción actual

Resumen:

Verificar que el sistema no permita la baja una actividad si el periodo de la inscripción está acurriendo

Precondiciones:

Actividad con periodo de inscripción actual

Usuario autenticado (token válido)

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud **PUT** al endpoint /api/actividades/baja/{id}. Incluir un id de una actividad que este en periodo de inscripción	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado de la actividad. Ejemplo:
	{
	"nuevoEstado": "INACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **409 Conflict.** La respuesta contiene un mensaje claro describiendo el error. Ningún dato de la actividad es modificado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Suite de Pruebas : Inscripción a Actividad

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint inscripción a actividad

Caso de Prueba PFTDZGR06-460: Inscripción a actividad no existente

Resumen:

Verifica la respuesta del sistema al intentar inscribirse a una actividad inexistente.

Precondiciones:

ID de actividad no existente en la base de datos.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado tiene rol admin

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/inscripciones/{idActividad} Seleccionar una actividad que no existe	
2	Incluir los siguientes encabezados:	

	• Content-Type: application/json • Authorization: Bearer <token>	
4	Enviar solicitud	El código de respuesta: **404 Not Found.** La respuesta contiene un mensaje claro describiendo que la actividad no existe.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-356: Registrar inscripción duplicada

Resumen:

Verificar que el sistema no permita inscribir dos veces al mismo usuario en la misma actividad.

Precondiciones:

Usuario ya inscripto en la actividad.

Token válido.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/inscripciones/{idActividad} Seleccionar una actividad a la que el usuario ya pertenece	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	Enviar solicitud	El código de respuesta: **409 Conflict.** La respuesta contiene un mensaje claro describiendo el error. mensaje: Ya existe una inscripción para el usuario en la actividad" Ningún dato de la actividad o el usuario es modificado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-357: Inscripción a actividad inactiva o dada de baja

Resumen:

Verificar que el sistema no permita inscribirse en una actividad que esté inactiva o dada de baja.

Precondiciones:

Actividad con estado inactiva.

Usuario autenticado (token válido).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/inscripciones/{idActividad}, incluir id de actividad con estado inactivo	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar solicitud	Código de respuesta: **422 Unprocessable Entity**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "La actividad seleccionada está inactiva y no acepta inscripciones"
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-358: Inscripción a actividad pasada

Resumen:

Verificar que el sistema no permita inscribirse en una actividad que ya ocurrió.

Precondiciones:

Actividad con fecha de actividad pasada.

Usuario autenticado.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/inscripciones/{idActividad} Seleccionar una actividad que ya ocurrió	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar solicitud	Código de respuesta: **422 Unprocessable Entity**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No es posible inscribirse a una actividad ya finalizada."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-359: Inscripción a actividad fuera del periodo habilitado (aún no iniciado)

Resumen:

Verificar que el sistema **no permita realizar la inscripción** cuando la actividad aún **no ha iniciado su periodo de inscripción** según las fechas configuradas.

Precondiciones:

Actividad existente con:

- Fecha de inicio de inscripción en el futuro.

Usuario autenticado con rol válido para inscribirse.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/inscripciones/{idActividad} Seleccionar una actividad en la que su periodo de inscripciones aún no ha empezado	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	Enviar solicitud	Código de respuesta: **422 Unprocessable Entity**

		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No es posible inscribirse antes del inicio del período de inscripción."
		}

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

Caso de Prueba PFTDZGR06-360: Inscripción a actividad cancelada

Resumen:

Verificar que el sistema no permita inscribirse en una actividad que está cancelada.

Precondiciones:

Actividad con estado cancelada.

Usuario autenticado (token válido).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/inscripciones/{idActividad} Seleccionar una actividad con estado CANCELADA	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar solicitud	Código de respuesta: **422 Unprocessable Entity**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No es posible inscribirse a una actividad cancelada."
		}

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

Caso de Prueba PFTDZGR06-355: Registrar inscripción exitosamente

Resumen:

Verificar que un usuario válido pueda inscribirse correctamente en una actividad disponible.

Precondiciones:

El usuario está autenticado (token válido).

La actividad existe, está activa y requiere de inscripción.

Endpoint disponible: POST /api/actividades/inscripciones.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/inscripciones/{idActividad}	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar solicitud	El código de respuesta es **201 Created**.
		El cuerpo de la respuesta contiene un mensaje y el objeto de la inscripción recién creada:
		{
		"message": "Su inscripción ha sido registrada correctamente.",
		"data": {
		"idActividad": 1,
		"actividadNom": "Clase de danza",
		"actividadDescripcion": "CLase con maestro especializado",
		...
		}
		}

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

Suite de Pruebas : Cancelación de Inscripción

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint cancelar inscricpción

Caso de Prueba PFTDZGR06-361: Cancelación de inscripción exitosa

Resumen:

Verificar que un usuario pueda cancelar correctamente una inscripción vigente.

Precondiciones:

Usuario autenticado.

Inscripción existente

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/inscripciones/{idActividad}/cancelar	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar solicitud	El código de respuesta es **200 OK**.
		El cuerpo de la respuesta contiene un mensaje y el objeto de la inscripción cancelada:
		{
		"message": "Su inscripción ha sido registrada correctamente.",
		"data": {
		"idActividad": 1,
		"actividadNom": "Clase de danza",
		"actividadDescripcion": "CLase con maestro especializado",
		...
		}
		}

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

Caso de Prueba PFTDZGR06-362: Cancelación de inscripción inexistente

Resumen:

Verificar que el sistema devuelva un error si se intenta cancelar una inscripción que no existe.

Precondiciones:

ID de inscripción inexistente.

Usuario autenticado (Token válido).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/inscripciones/{idActividad}/cancelar Seleccionar id de actividad donde no existe inscripción	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	Enviar solicitud	El código de respuesta: **404 Not Found.** La respuesta contiene un mensaje claro describiendo que la inscripción no existe.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-363: Cancelación de inscripción ya cancelada

Resumen:

Verificar que el sistema no permita cancelar una inscripción que ya fue cancelada previamente y avice el estado actual de la inscripción.

Precondiciones:

Inscripción con estado cancelada.

Usuario autenticado (token válido)

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/inscripciones/{idActividad}/cancelar Seleccionar id de actividad donde la inscripción fue cancelada posteriormente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	Enviar solicitud	El código de respuesta: **409 Conflict.** La respuesta contiene un mensaje claro describiendo el error.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Suite de Pruebas : Reporte de Inscripciones

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint reportes de inscripciones

Caso de Prueba PFTDZGR06-364: Generar reporte de inscripciones por fechas exitosamente

Resumen:

Verificar que el sistema genere correctamente el reporte de inscripciones cuando se proporcionan filtros válidos (por ejemplo, rango de fechas o actividad específica).

Precondiciones:

Existen inscripciones registradas en el sistema.

Usuario autenticado con rol **administrador**.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/reportes/inscripciones/por-fecha	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Construir body con el formato JSON.
	Ejemplo:
	{
	"fechaDesde": "2025-09-01",
	"fechaHasta": "2025-10-31",
	"tipoMovimiento": "AMBAS",
	"actividadesIds": [2]
	}	

4	Enviar solicitud	Código de respuesta: **200 OK** El cuerpo de la respuesta contiene un objeto con el reporte
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-365: Generar reporte de inscripciones por tipo de actividad exitosamente

Resumen:

Verificar que el sistema genere correctamente el reporte de inscripciones cuando se proporcionan filtros válidos (por ejemplo, rango de fechas o actividad específica).

Precondiciones:

Existen inscripciones registradas en el sistema.

Usuario autenticado con rol **administrador**.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/reportes/inscripciones/por-tipo-actividad	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Construir body con el formato JSON.
	{
	"fechaDesde": "2025-09-01",
	"fechaHasta": "2025-10-31",
	"tipoMovimiento": "CANCELACIONES",
	"tiposIds": [1]
	}	

4	Enviar solicitud	Código de respuesta: **200 OK** El cuerpo de la respuesta contiene un objeto con el reporte
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-367: Generar reporte con filtros inválidos o formato incorrecto

Resumen:

Verificar la respuesta del sistema cuando los filtros enviados son inválidos o tienen formato incorrecto.

Precondiciones:

Usuario autenticado de tipo *admin*

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/reportes/inscripciones/por-tipo-actividad o /api/reportes/inscripciones/por-fecha	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Construir body con el formato JSON, incluyendo datos inválidos.
	Ejemplo:
	{
	"fechaDesde": "2025-19-01",
	"fechaHasta": "2025-10-31",
	"tipoMovimiento": null,
	"actividadesIds": [2]
	}	

4	Enviar solicitud	El código de respuesta: **400 Bad Request** La respuesta incluye un **mensaje de error** describiendo los campos inválidos y su formato correcto.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-368: Generar reporte por usuario no autorizado

Resumen:

Verificar que los usuarios sin rol o permiso de administración no puedan acceder al reporte de inscripciones.

Precondiciones:

Usuario autenticado sin privilegios de reporte (por ejemplo, rol “socio”).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/reportes/inscripciones/por-tipo-actividad o /api/reportes/inscripciones/por-fecha	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Construir body con el formato JSON.
	Ejemplo:
	{
	"fechaDesde": "2025-09-01",
	"fechaHasta": "2025-10-31",
	"tipoMovimiento": "AMBAS",
	"actividadesIds": [2]
	}	

4	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-369: Generar reporte con token inválido o ausente

Resumen:

Verificar que el sistema rechace solicitudes sin token o con token inválido.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/reportes/inscripciones/por-tipo-actividad o /api/reportes/inscripciones/por-fecha	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token inválido>	

3	Construir body con el formato JSON.
	Ejemplo:
	{
	"fechaDesde": "2025-09-01",
	"fechaHasta": "2025-10-31",
	"tipoMovimiento": "AMBAS",
	"actividadesIds": [2]
	}	

4	Enviar solicitud	Código de respuesta: **401 Unauthorized**
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Suite de Pruebas : Gestión de Auditoría

Conjunto de pruebas diseñado para validar el correcto funcionamiento de todos los endpoints del módulo de auditorías, garantizando la integridad de las operaciones CRUD, seguridad y rendimiento de las APIs.

Suite de Pruebas : Reportes de Auditorías

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint reporte de auditoría

Caso de Prueba PFTDZGR06-278: Consulta de auditoría exitosa

Resumen:

Validar que un usuario con permisos de administrador pueda consultar correctamente el reporte de auditoría con los filtros permitidos.

Precondiciones:

El usuario autenticado debe tener el rol Administrador.

Se debe obtener token JWT (Se obtiene al iniciar sesión)

Existen registros de auditoría en el sistema.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/auditorias/{tipoAuditoria} los tipos pueden ser: Usuario, Actividad, Pago	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	Enviar solicitud	Código de respuesta: **200 OK** Devuelve una lista con la información de auditoria del tipo seleccionado

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	4.00
Prioridad:	Alta

Caso de Prueba PFTDZGR06-279: Consulta sin filtros

Resumen:

Verificar que al no enviar filtros, el sistema devuelva los registros más recientes o todos, según la configuración del endpoint.

Precondiciones:

El usuario autenticado debe tener el rol Administrador.

Se debe obtener token JWT (Se obtiene al iniciar sesión)

Existen registros de auditoría en el sistema.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/auditorias/{tipoAuditoria} los tipos pueden ser: Usuario, Actividad, Pago	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	Enviar solicitud	Código de respuesta: **200 OK** Devuelve una lista completa de todas las auditorias

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	2.00
Prioridad:	Alta

Caso de Prueba PFTDZGR06-280: Consulta con filtros inválidos

Resumen:

Probar que el sistema maneje correctamente filtros incorrectos (por ejemplo, fecha con formato inválido o usuario inexistente).

Precondiciones:

El usuario autenticado debe tener el rol Administrador.

Se debe obtener token JWT (Se obtiene al iniciar sesión)

Existen registros de auditoría en el sistema.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/auditorias/{tipoAuditoria} los tipos pueden ser: Usuario, Actividad, Pago	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	Enviar solicitud	El código de respuesta: **400 Bad Request** La respuesta incluye un **mensaje de error** describiendo los campos inválidos.

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	2.00
Prioridad:	Alta

Caso de Prueba PFTDZGR06-281: Acceso sin token o token inválido

Resumen:

Validar que solo usuarios autenticados puedan acceder al reporte de auditoría.

Precondiciones:

Solicitud sin token o con token vencido

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/auditorias/{tipoAuditoria} los tipos pueden ser: Usuario, Actividad, Pago	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token inválido>	
3	Enviar solicitud	Código de respuesta: **401 Unauthorized**

Tipo de ejecución:	Manual

Duración estimada de la ejec. (min):	2.00
Prioridad:	Alta

Caso de Prueba PFTDZGR06-282: Acceso a auditoría de usuario no autorizado

Resumen:

Asegurar que usuarios no administradores (por ejemplo, socios) no puedan acceder a la auditoría.

Precondiciones:

Usuario autenticado con rol distinto a administrador.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/auditorias/{tipoAuditoria} los tipos pueden ser: Usuario, Actividad, Pago	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	2.00
Prioridad:	Alta

Caso de Prueba PFTDZGR06-283: Listado sin registros encontrados

Resumen:

Validar que el sistema responda correctamente cuando no existen registros que cumplan con los filtros aplicados.

Precondiciones:

Usuario administrador autenticado.

Filtros que no correspondan a ningún registro o lista vacía.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/auditorias/{tipoAuditoria} los tipos pueden ser: Usuario, Actividad, Pago	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar solicitud	Código de respuesta: **404 Not Found**
		El cuerpo de la respuesta contiene un objeto con el listado vacío de auditorias y un mensaje de informativo, por ejemplo:
		{
		"mensaje": "No se encontraron auditorias que coincidan con los filtros.",
		"data": [ ]
		}

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	4.00
Prioridad:	Media

Suite de Pruebas : Gestión de Espacios

Conjunto de pruebas diseñado para validar el correcto funcionamiento de todos los endpoints del módulo de espacios, garantizando la integridad de las operaciones CRUD, seguridad y rendimiento de las APIs.

Ejemplos de parámetros válidos para los campos de Gestión de Espacios:

- Nombre de Espacio: Salón Principal
- Capacidad: 100
- Precio de Socio: 15.75
- Precio de No Socio: 35.75
- Vigencia de Precio: 2025-12-31
- Observaciones: Salón climatizado con proyectores
- Estado: ACTIVO

Ejemplos para parámetros inválidos:

- Nombre de Espacio: "" | "A" * 201 | "Salon@2024"
- Capacidad: -1 | null | ""
- Precio de Socio: -5.75 | "" | trecientos
- Precio de No Socio: -5.75 | trecientos | ""
- Vigencia de Precio: 2023-12-31 | ""
- Observaciones: "" | "A" * 1001
- Estado: activo | "" | 0

Ejemplos de parámetros válidos para los campos de Gestión de Reservas:

- Fecha de Reserva: 2025-12-24
- Hora de Actividad: 14:30:00
- Duración: 90
- Cantidad de Personas: 80
- Id de Usuario: 2
- Id de Espacio: 3
- Tipo de Pago: EFECTIVO

Ejemplos para parámetros inválidos:

- Fecha de Reserva: 2023-05-01 | hoy | null
- Hora de Actividad: 25:00:00 | tres | null | ""
- Duración: 0 | 200 | null | sesenta
- Cantidad de Personas: 200 (mayor al espacio seleccionado) | 0
- Id de Usuario: 199 | 0 | "" | null | juan
- Id de Espacio: 199 | 0 | null | "" | cancha

Suite de Pruebas : Registro de Espacio

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint registro de espacio

Caso de Prueba PFTDZGR06-373: Registrar espacio por parte de usuario no autorizado

Resumen:

Verifica que solo un usuario con rol *admin* pueda registrar nuevos espacios.

Precondiciones:

El endpoint POST /api/espacios está disponible y operativo.

Se dispone de un **token JWT** (Se obtiene al iniciar sesión)

Usuario autenticado sin rol admin (por ejemplo, socio o no socio)

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/espacios	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Construir una solicitud con el siguiente **body JSON.**
	Ejemplo:
	{
	"nomEspacio": "",
	"capacidad": 0,
	"precioSocio": 0,
	"precioNoSocio": 0,
	"vigPrecio": "",
	"espObservacion": ""
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	Código de respuesta: 403 Forbidden
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	4.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-374: Registrar espacio con datos duplicado

Resumen:

Verificar que el endpoint /api/espacios **rechace solicitudes de registro** cuando los datos enviados ya existen en la base de datos, asegurando que **no se creen espacios duplicados**.
Esto incluye campos únicos como **nombre**.

Precondiciones:

El endpoint /api/espacios está activo y accesible.

EL espacio no está previamente registrado

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/espacios	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el **body JSON** los datos de un espacio que ya exista en la base de datos. Ejemplo:
	Construir una solicitud con el siguiente **body JSON.**
	Ejemplo:
	{
	"nomEspacio": "",
	"capacidad": 0,
	"precioSocio": 0,
	"precioNoSocio": 0,
	"vigPrecio": "",
	"espObservacion": ""
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar la solicitud.	Código de respuesta: **409 Conflict**. Mensaje de error en la respuesta indicando que el espacio ya existe. No se crea un nuevo espacio en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-375: Registrar espacio con campos obligatorios faltantes

Resumen:

Verificar que el endpoint /api/espacios rechace correctamente solicitudes de registro cuando uno o más **campos obligatorios** no son enviados o están vacíos.

Precondiciones:

El endpoint /api/espacios está activo y accesible.

El espacio no está previamente registrado

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/espacios	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el **body JSON** los campos del usuario, dejando vacío uno o varios campos obligatorios. Ejemplo:
	Construir una solicitud con el siguiente **body JSON.**
	Ejemplo:
	{
	"nomEspacio": "",
	"capacidad": null,
	"precioSocio": 0,
	"precioNoSocio": null,
	"vigPrecio": "",
	"espObservacion": ""
	}
	(null y "" cuentan como vacios)	

4	Enviar la solicitud	El código de respuesta es **400 Bad Request**. La respuesta contiene un mensaje claro describiendo los **campos obligatorios faltantes**. Ningún espacio es creado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-376: Registrar espacio con datos inválidos

Resumen:

Verificar que el endpoint /api/espacios rechace correctamente solicitudes de registro cuando uno o más campos contienen **datos inválidos**, respetando las reglas de validación del sistema.

Precondiciones:

El endpoint /api/espacios está activo y accesible.

El espacio no está previamente registrado

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/espacios	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el **body JSON** los campos requeridos, reemplazando uno o varios con valores inválidos. Ejemplo:
	Construir una solicitud con el siguiente **body JSON.**
	Ejemplo:
	{
	"nomEspacio": "",
	"capacidad": 0,
	"precioSocio": 0,
	"precioNoSocio": 0,
	"vigPrecio": "",
	"espObservacion": ""
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar la solicitud.	El código de respuesta es **400 Bad Request** (o el código definido para validaciones). La respuesta incluye un **mensaje de error** describiendo los campos inválidos. Ningún espacio es creado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	5.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-377: Registrar espacio exitosamente

Resumen:

Registro de espacio con todos los campos del formulario completos con datos válidos

Precondiciones:

El espacio no debe estar registrado en el sistema.

Endpoint POST /api/espacios disponible.

Nº:	Pasos:	Resultados Esperados:

1	Construir una solicitud POST al endpoint /api/espacios	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Construir una solicitud con el siguiente **body JSON.**
	Ejemplo:
	{
	"nomEspacio": "",
	"capacidad": 0,
	"precioSocio": 0,
	"precioNoSocio": 0,
	"vigPrecio": "",
	"espObservacion": ""
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar la solicitud	El código de respuesta es **201 Created**.
		El cuerpo de la respuesta contiene un mensaje y el objeto del espacio recién creado:
		{
		"message": "Espacio registrado correctamente",
		"data": {
		"idEspacio ": <autogenerado>,
		"nomEspacio ": "Cancha",
		"estado": "INACTIVO",
		...
		}
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Suite de Pruebas : Listado de Espacios

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint listado de espacios

Caso de Prueba PFTDZGR06-387: Acceso a listado de espacios exitosamente

Resumen:

Verificar que el endpoint GET /api/espacios devuelva correctamente la lista de espacios registrados en el sistema, asegurando que el formato de respuesta y el código HTTP sean los esperados.

Precondiciones:

El endpoint GET /api/espacios está disponible.

Existe al menos un espacio registrado en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado debe ser de tipo *Admin*

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/espacios	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar la solicitud.	Código de respuesta: **200 OK**
		El cuerpo de la respuesta contiene un objeto con el listado de espacios y un mensaje de exito, por ejemplo:
		{
		"mensaje": "Listado de espacios obtenido correctamente.",
		"data": [
		{
		"idEspacio": 1,
		"nomEspacio": "Cancha",
		"estado": "ACTIVO",
		...
		},
		{
		"idEspacio": 2,
		"nomEspacio": "Salón",
		"estado": "ACTIVO",
		}
		]
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-388: Acceso a listado de espacios por Usuario Admin

Resumen:

Verificar que el endpoint GET /api/espacios devuelva correctamente la lista de espacios registrados si el usuarios es de tipo ADMIN

Precondiciones:

El endpoint GET /api/espacios está disponible.

Existe al menos una espacio registrado en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado tiene perfil ADMIN

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/espacios	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar solicitud	Código de respuesta: **200 OK**
		El cuerpo de la respuesta contiene un objeto con el listado de espacios y un mensaje de exito, por ejemplo:
		{
		"mensaje": "Listado de espacios obtenido correctamente.",
		"data": [
		{
		"idEspacio": 1,
		"nomEspacio": "Cancha",
		"estado": "ACTIVO",
		...
		},
		{
		"idEspacio": 2,
		"nomEspacio": "Salón",
		"estado": "ACTIVO",
		}
		]
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-389: Acceso a listado de espacios por Usuario Socio/No Socio

Resumen:

Verificar que el endpoint GET /api/espacios no devuelva la lista de espacios registrados en el sistema si el usuario que intenta acceder es de tipo Socio o No Socio, asegurando que el formato de respuesta y el código HTTP sean los esperados.

Precondiciones:

El endpoint GET /api/espacios está disponible.

Existe al menos un espacio registrado en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado debe ser de tipo Socio o No Socio

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/espacios	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-390: Filtrado del listado de espacios

Resumen:

Verificar que el endpoint GET /api/espacios devuelva correctamente la lista de espacios registrados en el sistema de acuerdo al filtro seleccionado, asegurando que el formato de respuesta y el código HTTP sean los esperados.

Precondiciones:

El endpoint GET /api/espacios está disponible.

Existe al menos un espacio registrado en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado debe ser ADMIN

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/espacios	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	(Opcional) Agregar parámetros de filtro válidos, por ejemplo: • /api/actividades?nombre=Taller • /api/actividades?estado=ACTIVO Los filtros se pueden combinar usando &	

4	Enviar solicitud	Código de respuesta: **200 OK**
		El cuerpo de la respuesta contiene un objeto con el listado de espacios de acuerdo a los filtros incluidos y un mensaje de exito, por ejemplo:
		{
		"mensaje": "Listado de espacios obtenido correctamente.",
		"data": [
		{
		"idEspacio": 1,
		"nomEspacio": "Cancha",
		"estado": "ACTIVO",
		...
		},
		{
		"idEspacio": 2,
		"nomEspacio": "Salón",
		"estado": "ACTIVO",
		}
		]
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-391: Acceder a listado sin espacios registrados

Resumen:

Verificar que el endpoint GET /api/espacios devuelva una lista vacía de espacios y el código HTTP sea lo esperado.

Precondiciones:

El endpoint GET /api/espacios está disponible.

No existen espacios registrados en la base de datos.

Se dispone de un **token de autenticación válido** (Se obtiene al iniciar sesión).

El usuario logueado es de tipo ADMIN

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/espacios	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Enviar solicitud	Código de respuesta: **404 Not Found**
		El cuerpo de la respuesta contiene un objeto con el listado vacío de espacios y un mensaje de informativo, por ejemplo:
		{
		"mensaje": "No se encontraron espacios que coincidan con los filtros.",
		"data": [ ]
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-392: Acceder a listado de espacios sin token de autorización

Resumen:

Validar que el sistema no permita acceder al listado de espacios cuando el token de autenticación está ausente o es inválido, devolviendo el código de estado y mensaje de error correspondiente.

Precondiciones:

El endpoint GET /api/espacios requiere autenticación mediante token JWT

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/espacios	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token inválido>	
3	Enviar solicitud	Código de respuesta: **401 Unauthorized**
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Suite de Pruebas : Modificación de Espacio

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint modificación de espacio

Caso de Prueba PFTDZGR06-380: Modificar espacio no existente

Resumen:

Verificar que el sistema retorne un error cuando se intenta modificar un espacio que no existe.

Precondiciones:

El endpoint PUT /api/espacios/{id} está disponible y operativo.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado es de tipo *Admin*

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/espacios/{id}, incluir id de espacio inexistente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Construir una solicitud con el siguiente **body JSON**:
	{
	"nomEspacio": "",
	"capacidad": 0,
	"precioSocio": 0,
	"precioNoSocio": 0,
	"vigPrecio": "",
	"espObservacion": "",
	"estado": ""
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar la solicitud	El código de respuesta: **404 Not Found.** La respuesta contiene un mensaje claro describiendo que el espacio no existe.

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	2.00
Prioridad:	Alta

Caso de Prueba PFTDZGR06-381: Modificar espacio exitosamente

Resumen:

Verificar que el sistema permita modificar correctamente los datos de un espacio existente cuando la solicitud es válida y proviene de un usuario autorizado.

Precondiciones:

El endpoint PUT /api/espacios/{id} está disponible y operativo.

Existe al menos un espacio en la base de datos

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

El usuario logueado posee rol de *admin*.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/espacios/{id}	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Construir una solicitud con el siguiente **body JSON**:
	{
	"nomEspacio": "",
	"capacidad": 0,
	"precioSocio": 0,
	"precioNoSocio": 0,
	"vigPrecio": "",
	"espObservacion": "",
	"estado": ""
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar la solicitud	El código de respuesta es **200 OK**.
		El cuerpo de la respuesta contiene un mensaje y el objeto del espacio modificado:
		{
		"message": "El espacio ha sido actualizado correctamente.",
		"data": {
		"idEspacio ": 2,
		"nomEspacio ": "Cancha",
		"estado": "INACTIVO",
		...
		}
		}

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	2.00
Prioridad:	Alta

Caso de Prueba PFTDZGR06-382: Modificar espacio con datos obligatorios faltantes

Resumen:

Verifica que el sistema no permita actualizar un espacio si faltan campos obligatorios.

Precondiciones:

El endpoint PUT /api/espacios/{id} está disponible y operativo.

Existe al menos un espacio registrado en la base de datos

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado es de tipo *Admin*

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/espacios/{id}	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	Construir una solicitud con el siguiente **body JSON**, dejar campos obligatorios vacíos. Ejemplo: { "nomEspacio": "", "capacidad": null, "precioSocio": null, "precioNoSocio": null, "vigPrecio": "",	

	"espObservacion": "", "estado": "" } (Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.) (null y "" cuentan como vacíos)	
4	Enviar la solicitud	El código de respuesta: **400 Bad Request** La respuesta incluye un **mensaje de error** describiendo los campos faltantes. Ningún espacio es modificado en la base de datos
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-383: Modificar espacio con datos inválidos

Resumen:

Verifica que el sistema rechace modificaciones con datos en formato incorrecto o inválido.

Precondiciones:

El endpoint PUT /api/espacios/{id} está disponible y operativo.

Existe al menos un espacio registrado en la base de datos

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado es de tipo *Admin*

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/espacios/{id}	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Construir una solicitud con el siguiente **body JSON**, ingresar datos inválidos.
	Ejemplo:
	{
	"nomEspacio": "",
	"capacidad": 0,
	"precioSocio": 0,
	"precioNoSocio": 0,
	"vigPrecio": "",
	"espObservacion": "",
	"estado": ""
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar la solicitud	El código de respuesta: **400 Bad Request** La respuesta incluye un **mensaje de error** describiendo los campos inválidos y su formato correcto. Ningún espacio es modificado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-384: Modificar espacio con datos duplicados

Resumen:

Verificar que el sistema no permita modificar un espacio asignándole un nombre que ya esté registrado en otro espacio.

Precondiciones:

El endpoint PUT /api/espacios/{id} está disponible y operativo.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

El usuario logueado en de tipo *Admin*

Existen al menos un espacio en el sistema.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/espacios/{id}	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Construir una solicitud con el siguiente **body JSON**, ingresar datos existentes en otro espacio.
	Ejemplo:
	{
	"nomEspacio": "",
	"capacidad": 0,
	"precioSocio": 0,
	"precioNoSocio": 0,
	"vigPrecio": "",
	"espObservacion": "",
	"estado": ""
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar la solicitud	Código de respuesta: **409 Conflict**. Mensaje de error en la respuesta indicando que ya existe un espacio con esa información.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-385: Modificar espacio por parte de un usuario no autorizado

Resumen:

Verificar que los usuarios que no poseen el rol administrador no puedan modificar espacios.

Precondiciones:

El endpoint PUT /api/espacios/{id} está disponible y operativo.

Existe un espacio en la base de datos

Se dispone de un **token JWT** (Se obtiene al iniciar sesión)

Usuario autenticado sin rol admin (por ejemplo, socio o no socio)

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/espacios/{id}	
2	Incluir los siguientes encabezados:	

	• Content-Type: application/json • Authorization: Bearer <token>	

3	Construir una solicitud con el siguiente **body JSON**.
	Ejemplo:
	{
	"nomEspacio": "",
	"capacidad": 0,
	"precioSocio": 0,
	"precioNoSocio": 0,
	"vigPrecio": "",
	"espObservacion": "",
	"estado": ""
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar la solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Suite de Pruebas : Baja/Reactivación de Espacio

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint baja de espacio y reactivación de espacio

Caso de Prueba PFTDZGR06-393: Validar restricción de uso de espacios dados de baja

Resumen:

Verificar que el sistema **no permita asociar ni relacionar** un espacio que se encuentra dado de baja (inactivo) con otros registros, como usuarios u otras entidades dependientes.

Precondiciones:

Existe un espacio previamente dado de baja (estado inactivo).

Usuario autenticado con rol administrador.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud POST al endpoint /api/reservas/confirmar	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON con la información de la reserva. Ejemplo:
	{
	"fecReservaActividad": "2025-12-28",
	"horaReservaActividad": "10:00:00",
	"duracion": 60,
	"cantidadPersonas": 300,
	"idUsuario": 2,
	"idEspacio": 1,
	"tipoPago": "EFECTIVO"
	}	

4	Enviar solicitud	Código de respuesta: **422 Unprocessable Entity**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "El espacio seleccionado está inactivo y no acepta reservas"
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-394: Baja exitosa de espacio

Resumen:

Verifica que un administrador pueda dar de baja correctamente a un espacio activo.

Precondiciones:

El espacio existe y está en estado **activo**.

El usuario logueado tiene rol admin.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/espacios/baja/{id}	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado del espacio. Ejemplo:
	{
	"nuevoEstado": "INACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **200 OK**.
		La respuesta contiene un mensaje de exito y el objeto del espacio con su nuevo estado
		{
		"mensaje": "El espacio ha sido dado de baja exitosamente.",
		"data": [
		{
		"idEspacio": 1,
		"nomEspacio": "Cancha",
		"estado": "INACTIVO",
		}
		]
		}
		El espacio queda inactivo y no puede ser vinculado a otra entidad o realizar más acciones sobre el.

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-395: Baja de espacio ya inactivo

Resumen:

Verifica que el sistema no permita dar de baja a un espacio que ya se encuentra inactivo.

Precondiciones:

El espacio está en estado **inactivo**.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado tiene rol admin

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/espacios/baja/{id}, incluyendo un id de espacio con estado inactivo	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el estado del espacio. Ejemplo:
	{
	"nuevoEstado": "INACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **409 Conflict.** La respuesta contiene un mensaje claro describiendo el error. Ningún dato del espacio es modificado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Media	

Caso de Prueba PFTDZGR06-396: Baja con espacio no existente

Resumen:

Verifica la respuesta del sistema al intentar dar de baja a un espacio inexistente.

Precondiciones:

ID de espacio no existente en la base de datos.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado tiene rol admin

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/espacios/baja/{id}, incluyendo el id de una actividad inexistente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado del espacio. Ejemplo:
	{
	"nuevoEstado": "INACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **404 Not Found.** La respuesta contiene un mensaje claro describiendo que el espacio no existe.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	1.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-397: Baja de espacio por usuario no autorizado

Resumen:

Verifica que solo un usuario con rol *admin* pueda dar de baja espacios.

Precondiciones:

Usuario autenticado con rol **no administrador (Socio o No Socio)**.

Espacio objetivo existe y está activo.

Se dispone de un **token JWT** (Se obtiene al iniciar sesión).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/espacios/baja/{id}	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado del espacio. Ejemplo:
	{
	"nuevoEstado": "INACTIVO"
	}	

4	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-398: Reactivar espacio ya activo

Resumen:

Verifica que el sistema no permita reactivar un espacio que ya se encuentra activo.

Precondiciones:

Espacio con estado **activo**.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado tiene rol admin

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/espacios/reactivar/{id}, incluyendo el id de un espacio con estado activo	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	Incluir en el body JSON el estado del espacio. Ejemplo: { "nuevoEstado": "ACTIVO"	

	}	
4	Enviar solicitud	El código de respuesta: **409 Conflict.** La respuesta contiene un mensaje claro describiendo el error. Ningún dato del espacio es modificado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Media	

Caso de Prueba PFTDZGR06-399: Reactivar espacio exitosamente

Resumen:

Verifica que un administrador pueda reactivar correctamente un espacio dado de baja.

Precondiciones:

Espacio existente en estado **inactivo**.

El usuario logueado tiene rol admin.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/espacios/reactivar/{id}	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado del espacio. Ejemplo:
	{
	"nuevoEstado": "ACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **200 OK**.
		La respuesta contiene un mensaje de exito y el objeto del espacio con su nuevo estado
		{
		"mensaje": "El espacio ha sido activado exitosamente.",
		"data": [
		{
		"idEspacio": 1,
		"nomEspacio": "Cancha",
		"estado": "ACTIVO",
		}
		]
		}
		El espacio queda activo y puede ser vinculado a otra entidad o realizar más acciones sobre el.

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-400: Reactivar espacio no existente

Resumen:

Verifica la respuesta al intentar reactivar un espacio que no existe.

Precondiciones:

ID de espacio no existente en la base de datos.

Se dispone de un **token JWT válido** (Se obtiene al iniciar sesión)

El usuario logueado tiene rol admin

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/espacios/reactivar/{id}, incluyendo el id de un espacio inexistente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado del espacio. Ejemplo:
	{
	"nuevoEstado": "ACTIVO"
	}	

4	Enviar solicitud	El código de respuesta: **404 Not Found.** La respuesta contiene un mensaje claro describiendo que el espacio no existe.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-401: Reactivar espacio por usuario no autorizado

Resumen:

Verifica que solo administradores puedan reactivar espacios.

Precondiciones:

Usuario autenticado con rol **no administrador (Socio o No Socio)**.

Espacio objetivo existe y está activo.

Se dispone de un **token JWT** (Se obtiene al iniciar sesión).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud PUT al endpoint /api/espacios/reactivar/{id}	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado del espacio. Ejemplo:
	{
	"nuevoEstado": "ACTIVO"
	}	

4	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Suite de Pruebas : Registro de Reserva

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint registro de reserva de espacio

Caso de Prueba PFTDZGR06-404: Registrar reserva con campos obligatorios faltantes

Resumen:

Verificar que el endpoint /api/reservas rechace correctamente solicitudes de registro cuando uno o más **campos obligatorios** no son enviados o están vacíos.

Precondiciones:

El endpoint /api/reservas está activo y accesible.

La reserva no está previamente registrado

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud **POST** al endpoint /api/reservas/confirmar.	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el **body JSON** los campos de la reserva, dejando vacío uno o varios campos obligatorios. Ejemplo:
	{
	"fecReservaActividad": "",
	"horaReservaActividad": "00:00:00",
	"duracion": 0,
	"cantidadPersonas": null,
	"idUsuario": 1,
	"idEspacio": 1,
	"tipoPago": null
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)
	(null y "" cuentan como vacios)	

4	Enviar la solicitud	El código de respuesta es **400 Bad Request**. La respuesta contiene un mensaje claro describiendo los **campos obligatorios faltantes**. Ninguna reserva es creado en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-405: Registrar reserva con datos inválidos

Resumen:

Verificar que el endpoint /api/reservas rechace correctamente solicitudes de registro cuando uno o más campos contienen **datos inválidos**, respetando las reglas de validación del sistema.

Precondiciones:

El endpoint /api/reservas está activo y accesible.

La reserva no está previamente registrada

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud **POST** al endpoint /api/reservas/confirmar.	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el **body JSON** los campos de la reserva, ingresando datos inválidos. Ejemplo:
	{
	"fecReservaActividad": "",
	"horaReservaActividad": "00:00:00",
	"duracion": 0,
	"cantidadPersonas": null,
	"idUsuario": 1,
	"idEspacio": 1,
	"tipoPago": null
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar la solicitud.	El código de respuesta es **400 Bad Request** (o el código definido para validaciones). La respuesta incluye un **mensaje de error** describiendo los campos inválidos. Ninguna reserva es creada en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	5.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-406: Registrar reserva exitosamente

Resumen:

Registro de reserva con todos los campos del formulario completos con datos válidos

Precondiciones:

La reserva no debe estar registrada en el sistema.

Endpoint POST /api/reserva disponible.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud **POST** al endpoint /api/reservas/confirmar.	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Construir una solicitud con el siguiente **body JSON**:
	{
	"fecReservaActividad": "",
	"horaReservaActividad": "00:00:00",
	"duracion": 0,
	"cantidadPersonas": 0,
	"idUsuario": 1,
	"idEspacio": 1,
	"tipoPago": ""
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar la solicitud	El código de respuesta es **201 Created**. El cuerpo de la respuesta contiene un mensaje y el objeto de la reserva recién creada: { "message": "La reserva ha sido registrada exitosamente.",

		"data": {
		"idReserva": <autogenerado>,
		"fecReservaActividad": "2025-10-23",
		"idEspacio": 2,
		...
		}
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-407: Registrar reserva con solapamiento de horarios

Resumen:

Verificar que el sistema impida el registro de una reserva cuyo horario coincide parcial o totalmente con otra reserva o actividad ya registrada.

Precondiciones:

Existe otra reserva o actividad con horario que se solapa total o parcialmente.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud **POST** al endpoint /api/reservas/confirmar.	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el **body JSON** los campos de la reserva. Ejemplo:
	{
	"fecReservaActividad": "2025-10-10",
	"horaReservaActividad": "13:00:00",
	"duracion": 0,
	"cantidadPersonas": null,
	"idUsuario": 1,
	"idEspacio": 1,
	"tipoPago": null
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	Código de respuesta: **409 Conflict**. Mensaje de error en la respuesta indicando que el espacio ya está reservado para ese momento. mensaje: El espacio no está disponible en el horario seleccionado No se crea una nueva reserva en la base de datos.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	5.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-408: Registrar reserva consecutivas sin solapamiento

Resumen:

Verificar que el sistema permita el registro de una reserva cuando las reservas o actividades terminan e inician en horarios contiguos (sin superposición).

Precondiciones:

Existe otra reserva o actividad que inicia cuando termina la nueva reserva (con margen de 30 minutos).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud **POST** al endpoint /api/reservas/confirmar.	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el **body JSON** los campos de la reserva, dejando vacío uno o varios campos obligatorios. Ejemplo:
	Construir una solicitud con el siguiente **body JSON**:
	{
	"fecReservaActividad": "2025-08-17",
	"horaReservaActividad": "11:00:00",
	"duracion": 0,
	"cantidadPersonas": null,
	"idUsuario": 1,
	"idEspacio": 1,
	"tipoPago": null
	}
	(Los valores de los campos deben completarse según los datos definidos en la sección “Detalles” de la suite de pruebas.)	

4	Enviar solicitud	El código de respuesta es **201 Created**.
		El cuerpo de la respuesta contiene un mensaje y el objeto de la actividad recién creada:
		{
		"message": "La actividad ha sido registrada correctamente y se encuentra disponible en la lista de actividades.",
		"data": {
		"idActividad": <autogenerado>,
		"fecReservaActividad": "2025-08-17",
		"horaReservaActividad": "11:00:00",
		...
		}
		}

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

Suite de Pruebas : Cancelación de Reserva

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint cancelación de reserva de espacio

Caso de Prueba PFTDZGR06-409: Cancelación de reserva exitosa

Resumen:

Verificar que un usuario pueda cancelar correctamente una reserva vigente.

Precondiciones:

Usuario autenticado.

Reserva existente

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud **PUT** al endpoint /api/reservas/{id}/cancelar.	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	
3	Incluir en el body JSON el nuevo estado de la reserva. Ejemplo:	

	{
	"estadoReserva": "CANCELADA"
	}	

4	Enviar solicitud	El código de respuesta es **200 OK**.
		El cuerpo de la respuesta contiene un mensaje y el objeto de la reserva cancelada:
		{
		"message": "La reserva ha sido cancelada exitosamente.",
		"data": {
		"idReserva": 1,
		"estado": "CANCELADA",
		...
		}
		}

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

Caso de Prueba PFTDZGR06-410: Cancelación de reserva inexistente

Resumen:

Verificar que el sistema devuelva un error si se intenta cancelar una reserva que no existe.

Precondiciones:

ID de reserva inexistente.

Usuario autenticado (Token válido).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud **PUT** al endpoint /api/reservas/{id}/cancelar. Incluir un id de reserva inexistente	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado de la reserva. Ejemplo:
	{
	"estadoReserva": "CANCELADA"
	}	

4	Enviar solicitud	El código de respuesta: **404 Not Found.** La respuesta contiene un mensaje claro describiendo que la reserva no existe.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-411: Cancelación de reserva ya cancelada

Resumen:

Verificar que el sistema no permita cancelar una reserva que ya fue cancelada previamente y avice el estado actual de la reserva.

Precondiciones:

Reserva con estado cancelada.

Usuario autenticado (token válido)

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud **PUT** al endpoint /api/reservas/{id}/cancelar. Incluir un id de reserva con estado CANCELADO	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	Incluir en el body JSON el nuevo estado de la reserva. Ejemplo:{
	"estadoReserva": "CANCELADA"
	}	

4	Envia solicitud	El código de respuesta: **409 Conflict.** La respuesta contiene un mensaje claro describiendo el error.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Suite de Pruebas : Reportes de Reservas

Conjunto de pruebas diseñado para validar el correcto funcionamiento del endpoint reportes de reservas

Caso de Prueba PFTDZGR06-412: Generar reporte de reservas por fechas exitosamente

Resumen:

Verificar que el sistema genere correctamente el reporte de reservas cuando se proporcionan filtros válidos (por ejemplo, rango de fechas o espacio específico).

Precondiciones:

Existen reservas registradas en el sistema.

Usuario autenticado con rol **administrador**.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud **GET** al endpoint /api/reportes/reservas/por-fecha	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	construir cuerpo de solicitud con formato **JSON**.
	Ejemplo:
	{
	"fechaDesde": "2025-08-21",
	"fechaHasta": "2025-12-31",
	"tipoMovimiento": "AMBAS",
	"espaciosIds": [1]
	}	

4	Enviar solicitud	Código de respuesta: **200 OK** El cuerpo de la respuesta contiene un objeto con el reporte
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	3.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-413: Generar reporte de reservas por espacio exitosamente

Resumen:

Verificar que el sistema genere correctamente el reporte de reservas cuando se proporcionan filtros válidos (por ejemplo, rango de fechas o actividad específica).

Precondiciones:

Existen reservas registradas en el sistema.

Usuario autenticado con rol **administrador**.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud **GET** al endpoint /api/reportes/reservas/por-espacio	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	construir cuerpo de solicitud con formato **JSON**.
	Ejemplo:
	{
	"fechaDesde": "2025-08-21",
	"fechaHasta": "2025-12-31",
	"tipoMovimiento": "AMBAS",
	"espaciosIds": [1]
	}	

4	Enviar solicitud	Código de respuesta: **200 OK** El cuerpo de la respuesta contiene un objeto con el reporte
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-415: Generar reporte con filtros inválidos o formato incorrecto

Resumen:

Verificar la respuesta del sistema cuando los filtros enviados son inválidos o tienen formato incorrecto.

Precondiciones:

Usuario autenticado de tipo *admin*

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/reportes/reservas/por-espacio o /api/reportes/reservas/por-fecha	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	construir cuerpo de solicitud con formato **JSON**, incluyendo datos con formato inválidos.
	Ejemplo:
	{
	"fechaDesde": "2025-08-21",
	"fechaHasta": "2025-12-31",
	"tipoMovimiento": "AMBAS",
	"espaciosIds": [1]
	}	

4	Enviar solicitud	El código de respuesta: **400 Bad Request** La respuesta incluye un **mensaje de error** describiendo los campos inválidos y su formato correcto.
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-416: Generar reporte por usuario no autorizado

Resumen:

Verificar que los usuarios sin rol o permiso de administración no puedan acceder al reporte de reservas.

Precondiciones:

Usuario autenticado sin privilegios de reporte (por ejemplo, rol “socio”).

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/reportes/reservas/por-espacio o /api/reportes/reservas/por-fecha	
2	Incluir los siguientes encabezados: • Content-Type: application/json • Authorization: Bearer <token>	

3	construir cuerpo de solicitud con formato **JSON**.
	Ejemplo:
	{
	"fechaDesde": "2025-08-21",
	"fechaHasta": "2025-12-31",
	"tipoMovimiento": "AMBAS",
	"espaciosIds": [1]
	}	

4	Enviar solicitud	Código de respuesta: **403 Forbidden**
		Objeto de respuesta con mensaje de error:
		{
		"error": "Acceso denegado",
		"mensaje": "No tiene permiso para acceder a este recurso."
		}

Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Caso de Prueba PFTDZGR06-417: Generar reporte con token inválido o ausente

Resumen:

Verificar que el sistema rechace solicitudes sin token o con token inválido.

Nº:	Pasos:	Resultados Esperados:
1	Construir una solicitud GET al endpoint /api/reportes/reservas/por-espacio o /api/reportes/reservas/por-fecha	
2	Incluir los siguientes encabezados:	

	• Content-Type: application/json • Authorization: Bearer <token inválido>	

3	construir cuerpo de solicitud con formato **JSON**.
	Ejemplo:
	{
	"fechaDesde": "2025-08-21",
	"fechaHasta": "2025-12-31",
	"tipoMovimiento": "AMBAS",
	"espaciosIds": [1]
	}	

4	Enviar solicitud	Código de respuesta: **401 Unauthorized**
Tipo de ejecución:	Manual	
Duración estimada de la ejec. (min):	2.00	
Prioridad:	Alta	

Métricas de tiempos de ejecución

Tiempo estimado para ejecutar 193 casos de prueba (horas):7.28

Tiempo usado para ejecutar 193 casos de prueba (horas):4.83