

PLANTILLA PARA CREAR UN PLAN DE PRUEBAS

Contenido

PLANTILLA PARA CREAR UN PLAN DE PRUEBAS	1
1. Descripción general.....	3
1.1 Introducción	3
1.2 Alcance	3
1.3 Glosario.....	3
1.4 Referencias	4
1.5 Visión General del Documento.....	4
2. Objetivos y Criterios de Prueba	4
2.1 Objetivos de Prueba.....	4
2.2 Criterios de Aceptación	4
2.3 Criterios de Suspensión y Reanudación.....	4
2.4 Criterios de Terminación.....	4
3. Objetivos y Criterios de Prueba	5
3.1 Objetivos de Prueba.....	5
3.2 Niveles de Pruebas	5
3.3 Tipos de Pruebas	5
3.4 Entregables de Prueba	5
3.5 Ambiente de Pruebas	5
3.6 Herramientas de Prueba	6
3.7 Roles y Responsabilidades	6
4. Diseño de las pruebas	6
4.1 Identificación de Casos de Prueba	6
4.2 Priorización de Casos de Prueba.....	7
4.3 Trazabilidad de Requisitos a Casos de Prueba.....	7
4.4 Requisitos de Datos de Prueba.....	7



5.	Ejecución de Pruebas	7
5.1	Preparación de Pruebas	7
5.2	Ejecución de Casos de Prueba	7
5.3	Registro de Resultados de Prueba	7
5.4	Informe de Defectos	7
6.	Riesgos y Contingencias.....	7
6.1	Identificación de Riesgos	7
6.2	Evaluación de Riesgos	8
6.3	Planes de Mitigación y Contingencia	8
7.	Aprobaciones	8
7.1	Aprobaciones del Plan de Pruebas	8
7.2	Firmas de Aprobación	8

1. Descripción general

1.1 Introducción

El presente documento tiene como objetivo definir la estrategia, alcance, recursos y casos de prueba para validar el correcto funcionamiento del sistema colaborativo para intercambio de material académico EduShare, asegurando que cumpla con los requisitos establecidos en la Especificación de Requisitos de Software.

1.2 Alcance

En plan cubrirá los siguientes módulos y middlewares del sistema:

1. Módulo de Acceso
2. Módulo de Catalogo
3. Módulo de Comentarios
4. Módulo de Notificaciones
5. Módulo de Perfil
6. Módulo de Publicación
7. Módulo de Seguimiento
8. Middleware ValidarAdmin
9. Middleware ValidarAdminOPropietario

Se excluyen del alcance pruebas de estrés y pruebas de compatibilidad con dispositivos o sistemas operativos no especificados

1.3 Glosario

SQL Server: Un sistema de gestión de bases de datos relacional desarrollado por Microsoft para almacenar y recuperar datos solicitados por otras aplicaciones.

Node.js: Node.js es un entorno de ejecución de JavaScript diseñado para ejecutar código en el lado del servidor. Basado en eventos y construido sobre el motor V8 de Google.

Npm: Node Packet Manager es el administrador de paquetes predeterminado para el tiempo de ejecución de JavaScript Node.js. Consiste en una interfaz de línea de comandos (CLI) para la publicación y descargar de paquetes y un repositorio en línea que alberga estos paquetes.

Mssql: El módulo mssql es una biblioteca de Node.js diseñada para interactuar con bases de datos MSSQL (Microsoft SQL Server) de manera asíncrona.

Jest: Marco de pruebas de JavaScript mantenido por Facebook con un enfoque en la simplicidad.

Eslint: ESLint es una herramienta para identificar y reportar patrones encontrados en código ECMAScript/JavaScript. Muestra errores de sintaxis, seguimiento de malas prácticas y provee sugerencias.

Babel: Babel es un transpilador para JavaScript. Permite transformar código de versiones más recientes de JS en código que sea entendido por navegadores más antiguos.

Dotenv: Dotenv es un módulo con cero dependencias que carga las variables de entorno de un archivo .env en process.env. Almacenar la configuración en el entorno, separada del código, se basa en la metodología de The Twelve-Factor App.

Nodemailer: Nodemailer es un módulo de Node.js que permite el envío de emails desde un servidor con facilidad.

Log4js: Log4js es un framework escrito en JavaScript para loguear eventos de una aplicación. Muy similar a Log4j

Zod: Biblioteca de validación centrada en TypeScript. Permite validar datos simples como una cadena hasta objetos anidados complejos.

PDF (Portable Document Format): Un formato de archivo utilizado para presentar documentos de manera independiente del software, hardware o sistema operativo en el que se visualizan

1.4 Referencias

ISO 829, ISO/IEC/IEEE 29119-3:2021 y ISO/IEC/IEEE 24748-1:202, Especificación de Requisitos de Software: ERS_EduShare.pdf

1.5 Visión General del Documento

El documento cubre los objetivos y criterios de prueba, tipos y niveles, cronograma, diseño de casos, ejecución, gestión de defectos y riesgos.

2. Objetivos y Criterios de Prueba

2.1 Objetivos de Prueba

Se busca validar el cumplimiento de los requisitos funcionales especificados en el ERS, detectar errores en la lógica de negocio y validaciones y garantizar la seguridad e integridad de los datos.

2.2 Criterios de Aceptación

No deben existir defectos de severidad alta pendientes de corrección. El 100% de los casos de prueba son exitosos.

2.3 Criterios de Suspensión y Reanudación

Al encontrarse un defecto bloqueante en funcionalidades críticas se deberá solucionar el defecto y posteriormente se reanudará el proceso

2.4 Criterios de Terminación

Cuando se haya cubierto al menos un 70% del código total que constituye la API debe estar probado, sin defectos de severidad alta pendientes de corrección (inserciones sin token, modificación de datos de otro usuario, escalada de privilegios) y habiendo cubierto la funcionalidad esencial en un 100%.

3. Objetivos y Criterios de Prueba

3.1 Objetivos de Prueba

Se utilizarán tanto técnicas de caja negra como de caja blanca para comprobar el correcto comportamiento de los distintos flujos de la API.

Caja blanca

- Cobertura de sentencias. Asegurar que cada línea se ejecute al menos una vez.
- Cobertura de decisiones: Asegurar que las condiciones lógicas (if, switch, bucles) se prueben con valores true y false
- Cobertura de caso básico

Caja negra

- Valores límite: Verificar que no se superen los límites establecidos para distintos campos (Ej, nombre de usuario, ruta de foto, contraseña)
- Campos nulos o faltantes: Verificar que no se acepten cuerpos con campos faltantes o headers sin token

3.2 Niveles de Pruebas

Este plan contempla únicamente pruebas unitarias, las cuales se enfocan en verificar el correcto funcionamiento de funciones y encontrar comportamientos inesperados métodos individuales de los módulos de la API.

3.3 Tipos de Pruebas

Se implementarán pruebas unitarias funcionales, enfocadas en validar que cada componente de la API cumpla correctamente con su responsabilidad individual. Estas pruebas se desarrollarán utilizando Jest, permitiendo automatizar la ejecución y verificar el comportamiento esperado de cada función o controlador.

Además, se incluirán pruebas orientadas a aspectos de seguridad, especialmente en torno a:

- Validación del JSON Web Token (JWT),
- Prevención de inyección de datos (e.g., SQL/NoSQL injection),
- Verificación de permisos de administrador,
- Verificación de permisos del propietario del recurso mediante middlewares personalizados.

Todas las pruebas estarán diseñadas bajo un enfoque de regresión, de modo que aseguren que futuras modificaciones en el sistema no introduzcan fallos en las funcionalidades ya existentes.

3.4 Entregables de Prueba

Se deberá acompañar este plan de pruebas con los casos de prueba escritos bajo el framework de pruebas Jest de Facebook y un reporte de ejecución de pruebas donde se muestren casos, entradas, salidas obtenidas, salidas esperadas y si ambas coinciden

3.5 Ambiente de Pruebas

Para garantizar la correcta ejecución de las pruebas, el entorno debe cumplir con las siguientes especificaciones:

El sistema debe ejecutarse en el entorno de desarrollo (dev), con la variable de entorno TEST configurada como TRUE en el archivo .env. La base de datos debe estar correctamente configurada con las credenciales de acceso DB_USUARIOREGISTRADO=UsuarioRegistrado y DB_CONTRASENIA USUARIOREGISTRADO=£A3_*8bRqz1m, asegurando una conexión activa y estable. Adicionalmente, la base de datos debe contener al menos un documento con el id=3 y una categoría con el id=1 para validar los flujos críticos.

Las pruebas deben realizarse en un sistema operativo Windows 10 u 11, con un mínimo de 100 MB de espacio libre para el correcto funcionamiento de la API. Una vez descargado el proyecto desde el repositorio, es necesario ejecutar npm install para instalar todas las dependencias requeridas. Finalmente, las pruebas se lanzan con el comando npm test, con un tiempo estimado de ejecución de 2 a 2.5 minutos. Este entorno controlado garantiza resultados consistentes y confiables en cada ejecución.

3.6 Herramientas de Prueba

Se utilizará Jest como framework principal de pruebas unitarias y de integración, permitiendo validar el comportamiento esperado de cada componente con ejecución automatizada mediante npm test.

Se puede utilizar Postman como complemento para verificación manual de endpoints o flujos no cubiertos inicialmente en las pruebas automatizadas. Esto servirá para:

- Analizar casos bordes complejos antes de convertirlos en pruebas formales.
- Depurar fallos específicos en APIs durante el desarrollo.
- Documentar nuevos flujos que posteriormente se integrarán al suite de Jest (a menos que ya se hayan cumplido los criterios de terminación definidos).

Cualquier flujo verificado con Postman deberá traducirse a una prueba automatizada en Jest, salvo que su inclusión no aporte valor adicional según los criterios de aceptación establecidos

3.7 Roles y Responsabilidades

Rol	Personal
Coordinador de pruebas	Juan Eduardo Cumplido Negrete
Desarrollador de pruebas	Christian Alberto Vázquez Cruz
Tester funcional	Erick Abdiel Atzin Olarte

4. Diseño de las pruebas

4.1 Identificación de Casos de Prueba

Los casos de prueba han sido identificados con base en los requisitos funcionales (RF) de la ERS y los casos de uso (CU) del sistema EduShare. Cada caso cubre al menos un escenario positivo (funcionamiento esperado) y uno o más negativos (manejo de errores).

Ejemplos por módulo:

- Acceso: ingreso con credenciales válidas/erróneas, ausencia de token
- Publicación: ruta de archivo inválida, categorías faltantes, subida correcta
- Middlewares: ejecución con roles no autorizados, acceso de propietario vs admin

4.2 Priorización de Casos de Prueba

Se han clasificado 4 prioridades para los casos de prueba:

1. Críticas: Funcionalidades críticas del sistema (login, validación de JWT, permisos)
2. Altas: Funcionalidades principales del negocio (ej: publicación de material académico)
3. Media: Operaciones comunes del usuario (comentarios, perfil, búsquedas)
4. Baja: Mejoras o funcionalidades opcionales

4.3 Trazabilidad de Requisitos a Casos de Prueba

Cómo se asegura que todos los requisitos están cubiertos por los casos de prueba.

4.4 Requisitos de Datos de Prueba

- Usuarios: al menos uno con rol de administrador, uno registrado y uno sin permisos
- Base de datos: un documento cargado (id = 3) y una categoría (id = 1)
- JWT: tokens válidos, expirados y malformados
- Datos falsos: campos vacíos, strings largos, tipos inválidos

5. Ejecución de Pruebas

5.1 Preparación de Pruebas

- Clonar el repositorio
- Instalar dependencias: `npm install`
- Configurar `.env` con `TEST=TRUE`
- Asegurar conexión a la base de datos de pruebas
- Ejecutar: `npm test`

5.2 Ejecución de Casos de Prueba

Las pruebas se ejecutarán mediante el comando `npm test`, con Jest como framework principal. Las pruebas están organizadas por módulo y middleware.

5.3 Registro de Resultados de Prueba

Cada ejecución genera resultados de cobertura con `—coverage`, y un reporte estructurado en consola. Adicionalmente se creará el reporte de pruebas, donde se especifique si las salidas esperadas coinciden con las obtenidas.

5.4 Informe de Defectos

De encontrarse un defecto, debe detenerse el proceso de pruebas, depurarse la aplicación y corregirlo

6. Riesgos y Contingencias

6.1 Identificación de Riesgos

- Cambios de último momento en la lógica de negocio
- Datos no consistentes en el entorno de prueba
- Fallas en herramientas de automatización

6.2 Evaluación de Riesgos

S

Riesgo	Probabilidad	Impacto	Nivel
Cambio en requisitos	Media	Alta	Alto
Base de datos inconsistente	Alta	Alta	Alto
Fallo en herramientas	Baja	Media	Medio

6.3 Planes de Mitigación y Contingencia



- Congelamiento de requisitos antes de pruebas
- Scripts de reinicialización de datos, Scripts para poblar tablas.
- No actualizar automáticamente la versión de Jest. No incluir el caret (^) en el package.json

7. Aprobaciones

7.1 Aprobaciones del Plan de Pruebas

El plan de pruebas deberá ser aprobado por los responsables, el coordinador de pruebas, el desarrollador de pruebas y el tester funcional

7.2 Firmas de Aprobación

Rol	Responsable	Firma
Ccoordinador de pruebas	Juan Eduardo Cumplido Negrete	
Desarrollador de pruebas	Christian Alberto Vázquez Cruz	
Tester funcional	Erick Atzin Abdiel Olarte	