Automatización API Backend - Reqres IN

Plan de pruebas

Versión 1.0

Historial de revisiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Versión** | **Descripción** | **Autor** |
| 07/may/24 | 1.0 | Creación del documento | Sebastian Muñoz |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Tabla de Contenidos

**Contenido**

[1. Introducción 4](#_Toc166069336)

[1.1 Objetivo del documento 4](#_Toc166069337)

[1.2 Resumen de la funcionalidad 4](#_Toc166069338)

[1.3 Alcance 4](#_Toc166069339)

[2. Requerimientos para pruebas 5](#_Toc166069340)

[3. Estrategia de prueba 5](#_Toc166069341)

[3.1 Herramientas 6](#_Toc166069342)

[4. Hitos del proyecto 6](#_Toc166069343)

[5. Entregables 6](#_Toc166069344)

# Introducción

## Objetivo del documento

Identificar información existente del proyecto y componentes del software a probar.

garantizar que el software desarrollado cumpla con los requisitos y expectativas del cliente y estándares establecidos.

## Resumen de la funcionalidad

Se requiere la implementación de pruebas automatizadas para los componentes de API del Backend de la empresa Mieles S.A.S, Se automatizarán cuatro funcionalidades principales clave del sistema correspondientes a la búsqueda, registro, actualización, modificación y eliminación de usuarios, así como los recursos desconocidos en la API ReqRes.

Se requiere garantizar el correcto funcionamiento de las solicitudes y respuestas de la API, al igual que la correcta parametrización de las variables basadas en la documentación entregada.

## Alcance

**Dentro del Alcance:**

Se automatizará las pruebas Backend del servicio API reqres.in, en base a lo descrito dentro de la solicitud de pruebas realizada, se automatizarán los casos de prueba de forma que permitan realizar las validaciones esperadas basado en la documentación disponible para la API en cuestión.

Se realizará validaciones al servicio API reqres.in, basado en la documentación entregada contenida dentro del paquete, se realizarán validaciones correspondientes a cuatro funcionalidades de la API, esta cual maneja los recursos y usuarios, su registro, actualización, modificación y eliminación al igual que la búsqueda general e individual de cada recurso y usuario.

Se realizará la debida y correcta validación y verificación de entradas y respuestas del servicio, obligatoriedad de campos, ruta critica y flujos alternativos del servicio, se realizará la validación en base a la documentación entregada y las especificaciones y esquemas encontrados dentro del swagger de la API usando los protocolos debidos y flujos alternos posibles dentro de lo documentado para cada funcionalidad.

Riesgos:  
\* Inestabilidad en el ambiente de pruebas

\* Cambios en la documentación

\* Cambios en la API sin previo aviso

\* Falta de documentación o especificaciones claras

\* Disponibilidad del Servicio

**Fuera del Alcance:**

\* Cualquier otra funcionalidad no descrita dentro del alcance

\* Pruebas de Integración y Alcances mayores a los descritos en el documento

\* Productos diferentes mencionados dentro del alcance

\* No se realizará validación de módulos, funcionalidades o sistemas diferentes a los descritos; o implícitos en la documentación entregada.

\* Se mantendrán políticas, lineamientos y estándares con los que el aplicativo cuenta actualmente.

# Requerimientos para pruebas

La siguiente lista identifica los items, casos de uso y requerimientos que fueron identificados para las pruebas:

\* Documentación de la API: Obtención del acceso a la documentación de la API a través del enlace <https://reqres.in> cual detalla los endpoints disponibles, parámetros aceptados, métodos HTTP admitidos, y lo códigos de respuesta y estructura esparados.

\* Herramientas de automatización: Tener lista las herramientas de automatización utilizadas para las pruebas, Postman y un repositorio GIT para gestión y documentación de las actividades relacionadas a pruebas.

\* Identificación de Endpoints a probar: Se identifican los endpoints a probar en las pruebas, siendo los principales definidos los cuatro esenciales, con método get /api/users para listar los usuarios, con método post /api/users para la creación de un usuario, con método put /api/users para actualizar un usuario y con método delete /api/users para la eliminación de un usuario

\* Datos de prueba: Definir un conjunto de datos de prueba representativos que cubran los diferentes casos de uso, incluyendo datos de entrada validos e inválidos para cada endpoint, así como datos limite para verificar la robustez de la API.

# Estrategia de prueba

Para las opciones definidas en el alcance de pruebas, se identifican y generan escenarios de prueba que permitan en la ejecución, confrontar la funcionalidad del sistema con las especificaciones en la documentación que soportan la trayectoria del proyecto, y permitan validar el comportamiento de la API frente a un comportamiento esperado.

Se realizará la ejecución de pruebas con el fin de evidenciar que lo desarrollado este de acuerdo con lo solicitado por el cliente, esto de la siguiente forma:

1. Análisis de Requerimientos y Documentación de la API: se revisará la documentación entregada de la API para comprender los endpoints disponibles, los métodos HTTP admitidos, los parámetros requeridos y las respuestas esperadas, se identificarán los casos de uso claves que pueden ser cubiertos por las pruebas de automatización.
2. Identificación de Escenarios de prueba: se define una lista de escenarios de prueba basados en los casos de uso identificados y los requerimientos funcionales del api, se realiza las validaciones iniciales en la herramienta Postman contra la API con la finalidad de validar y verificar los escenarios identificados y tener una base con la cual preparar una base sobre la cual diseñar los scripts de pruebas.
3. Desarrollo de Scripts de prueba: Se realiza los scripts de prueba con Maven como Gestor de dependencias, utilizando Karate como Framework para la automatización de las pruebas, se incluyen la preparación de datos y ejecución de la solicitud de la API para la validación de la respuesta recibida.
4. Gestión y documentación del proyecto: Se utilizará un repositorio GIT para subir y mantener un registro actualizado de todo el trabajo realizado, facilitando la colaboración y seguimiento de las actividades realizadas.
5. Manejo y reporte de BUGS: en caso de encontrar bugs se generarán informes detallados que incluyan información sobre las pruebas ejecutadas, los resultados obtenidos y problemas identificados sobre las pruebas que permitan al equipo de desarrollo abordar y corregir de manera eficiente cualquier problema identificado.
6. Retest: una vez resuelto los bugs, se Re ejecutaran los scripts y generaran los reportes actualizados para dar seguimiento y cierre a los incidentes corregidos o reapertura con los comentarios actualizados.
7. Cierre de pruebas: Una vez resuelto los incidentes se alojará la documentación y scripts correspondientes del proceso de pruebas en un repositorio GIT y se hará entrega del trabajo realizado.

## Herramientas

Las siguientes herramientas fueron utilizadas en el proyecto:

|  |  |  |
| --- | --- | --- |
|  | Herramienta | Version |
| Gestor de dependencias | Apache-Maven | 3.9.6 |
| Versionamiento y Control | Git | 2.45.0 |
| Repositorio de Código fuente | Github | ~ |
| Documentación y seguimiento de pruebas | Microsoft Excel | 18.0 |
| Framework de pruebas | Karate-junit5 | 1.5.0.RC4 |
| Herramienta de validación preeliminar | Postman | 10.24.16 |

## 

# Hitos del proyecto

|  |  |  |  |
| --- | --- | --- | --- |
| **Milestone Task** | **Ezfuero (en horas)** | **Fecha inicio** | **Fecha fin** |
| Test Plan | 2 | 5/07/2024 | 5/07/2024 |
| Diseño de Casos | 4 | 5/07/2024 | 5/07/2024 |
| Desarrollo de Scripts | 8 | 5/08/2024 | 5/17/2024 |
| Ejecución y Reporte | 4 | 5/17/2024 | 5/17/2024 |
| Documentación y Cierre |  |  |  |