**Indice:**

**1. Introducción**, 1

**2. Herramientas Utilizadas**, 2

**2.1 Postman**, 2

**2.2 Karate DSL**, 2

**2.3 Docker y Docker Compose**, 2

**3. Técnicas de Pruebas Utilizadas**, 3

**3.1 Pruebas de Unidad**, 3

**3.2 Pruebas de Integración**, 3

**3.3 Happy Path (Camino Feliz)**, 3

**3.4 Unhappy Path (Camino Infeliz)**, 4

**3.5 Pruebas de Regresión**, 4

**4. Justificación de Selección**, 4

**5. Arquitectura del Script**, 4

**6. Modo de Ejecución**, 5

**7. Conclusiones**, 5

**Especificación de Herramientas y Técnicas para Pruebas de APIs**

**1. Introducción**

Este documento detalla las herramientas y técnicas utilizadas para implementar pruebas de APIs en el proyecto relacionado con la API proporcionada en [https://restful-booker.herokuapp.com/apidoc/](https://restful-booker.herokuapp.com/apidoc/" \t "_new). El objetivo es asegurar la calidad y fiabilidad de las APIs mediante la automatización de pruebas, utilizando herramientas modernas y prácticas, junto con técnicas de prueba efectivas.

**2. Herramientas Utilizadas**

**2.1 Postman**

**Descripción:**

* Postman es una herramienta de desarrollo y pruebas de APIs ampliamente utilizada que permite crear, enviar y automatizar solicitudes HTTP.
* Proporciona una interfaz gráfica intuitiva para configurar y ejecutar pruebas de API.
* Compatible con diversas operaciones de prueba, incluyendo pruebas de unidad, pruebas de integración y pruebas de regresión.

**Funcionalidades Principales:**

* Creación y envío de solicitudes HTTP.
* Automatización de pruebas mediante la creación de colecciones y scripts.
* Generación de informes detallados de resultados de pruebas.

**2.2 Karate DSL**

**Descripción:**

* Karate DSL es un framework de automatización de pruebas de API en lenguaje de dominio específico (DSL) basado en Java.
* Permite escribir pruebas de API de manera clara y legible utilizando un formato similar a Gherkin.
* Ofrece integración con Java y otras herramientas de automatización como Maven y Cucumber.
* La automatización en Karate tiene un enfoque basado en palabras clave y datos.

**Funcionalidades Principales:**

* Escritura de pruebas en un estilo de lenguaje natural fácil de entender y mantener.
* Validación de respuestas de API utilizando aserciones integradas.
* Soporte para ejecución de pruebas en paralelo para mejorar la eficiencia.

**2.3 Docker y Docker Compose**

**Descripción:**

* Docker es una plataforma de código abierto que permite a los desarrolladores empaquetar, distribuir y ejecutar aplicaciones en contenedores.
* Docker Compose es una herramienta que permite definir y ejecutar aplicaciones Docker multi-contenedor.
* Ambas herramientas se utilizan para crear un entorno de desarrollo y ejecución consistente y reproducible para las pruebas de APIs.

**Funcionalidades Principales:**

* Creación de contenedores Docker para encapsular y ejecutar la aplicación bajo prueba y las herramientas de prueba.
* Definición de un entorno de ejecución consistente utilizando archivos Dockerfile y Docker Compose.
* Integración con pipelines de CI/CD para facilitar la integración y despliegue continuos.

**3. Técnicas de Pruebas Utilizadas**

**3.1 Pruebas de Unidad**

* Se implementan pruebas de unidad para cada endpoint de la API, verificando su funcionalidad de manera aislada.
* Estas pruebas se realizan utilizando las herramientas de automatización de pruebas mencionadas anteriormente.

**3.2 Pruebas de Integración**

* Se realizan pruebas de integración para asegurar que los diferentes componentes de la API funcionen correctamente juntos.
* Se utilizan las herramientas de automatización para simular escenarios complejos que involucren múltiples endpoints y flujos de datos.

**3.3 Happy Path (Camino Feliz)**

* Se enfoca en probar un escenario en el que todo funciona según lo esperado.
* Se prueban casos utilizando datos válidos y siguiendo el flujo de ejecución principal de la aplicación.

**3.4 Unhappy Path (Camino Infeliz)**

* Se centra en probar escenarios donde las cosas no van como se espera.
* Se prueban situaciones de error, excepciones o fallos utilizando datos inválidos, inesperados o incorrectos.

**3.5 Pruebas de Regresión**

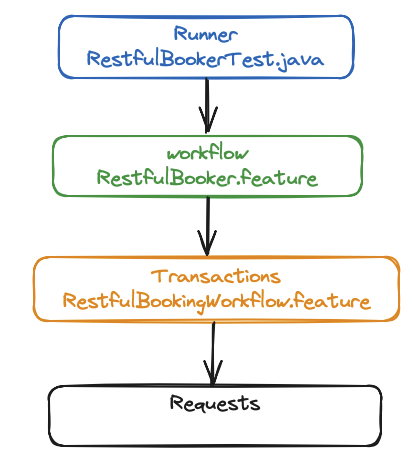
* Se automatizan pruebas de regresión para garantizar que las actualizaciones o cambios en la API no rompan la funcionalidad existente.
* Estas pruebas se ejecutan regularmente como parte de un proceso de integración continua (CI).

**4. Justificación de Selección**

Las herramientas y técnicas seleccionadas ofrecen una combinación sólida y completa para implementar pruebas de APIs en el proyecto. Postman y Karate DSL proporcionan herramientas poderosas y fáciles de usar para la automatización de pruebas, mientras que Docker y Docker Compose garantizan un entorno de ejecución consistente y reproducible. Las técnicas de pruebas utilizadas cubren una amplia gama de escenarios y garantizan la calidad y fiabilidad de las APIs.

**5. Arquitectura del Script**

Esquema de arquitectura del proyecto



**6. Modo de Ejecución:**

Ubicarse en la carpeta del proyecto

Maven:

mvn clean test -Dtest=RestfulBookerTest -Dkarate.env=uat

Docker:

docker-compose build karate-test-Restful-Booker

docker-compose up karate-test-Restful-Booker

**7. Conclusiones**

La combinación de herramientas y técnicas seleccionadas ofrece una solución completa y efectiva para implementar pruebas de APIs en el proyecto relacionado con <https://restful-booker.herokuapp.com/apidoc/>. Estas herramientas y técnicas permiten automatizar las pruebas de manera eficiente, garantizando la calidad y fiabilidad de las APIs y facilitando su integración en un pipeline de CI/CD.