### 1. ****Propuesta de Plan de Pruebas Servicios de**** Booker

#### ****Objetivo****

Automatizar la prueba de las funcionalidades principales del servicio web RESTful de un sistema de reservas (Booker). Esto incluye las operaciones CRUD (Create, Read, Update, Delete) sobre los recursos del servicio.

#### ****Alcance****

* Validar la creación de nuevas reservas (POST).
* Consultar reservas existentes (GET).
* Actualizar una reserva existente (PUT).
* Eliminar una reserva (DELETE).
* Validación de las respuestas HTTP, códigos de estado, y mensajes de error para peticiones mal formadas o no permitidas.
* Verificación de los tiempos de respuesta y posibles regresiones de rendimiento.

#### ****Estrategia de Pruebas****

* **Pruebas Funcionales**: Verificar que las funcionalidades principales (crear, consultar, modificar y eliminar reservas) funcionan correctamente.
* **Pruebas de Integración**: Asegurarse de que los diferentes endpoints del servicio web interactúan de manera correcta.
* **Pruebas de Error y Excepciones**: Verificar que el servicio maneja de manera adecuada los errores, como envíos de datos mal formados o acceso a recursos inexistentes.
* **Pruebas de Rendimiento (Opcional)**: Realizar pruebas de carga y estrés sobre los endpoints para verificar el comportamiento bajo condiciones extremas.

#### ****Casos de Prueba****

* **Crear una reserva válida**: Enviar una solicitud POST con datos válidos y verificar que la respuesta tenga un código 200 o 201.
* **Consultar una reserva existente**: Realizar una solicitud GET a una reserva existente y verificar que los datos sean correctos.
* **Modificar una reserva existente**: Enviar una solicitud PUT para modificar una reserva existente y verificar que los cambios se hayan aplicado.
* **Eliminar una reserva existente**: Realizar una solicitud DELETE a una reserva y verificar que el recurso haya sido eliminado.
* **Errores comunes**: Intentar crear una reserva con datos inválidos y verificar que la respuesta sea un error 400 (Bad Request).
* **Consultar una reserva inexistente**: Realizar un GET a una reserva que no existe y verificar que se devuelva un error 404 (Not Found).

### 2. ****Implementación del Código para el Script de Pruebas Automatizadas****

Utilizando **Selenium** para la interfaz web o **RestAssured** (en Java) o **Requests** (en Python) para interactuar con los endpoints RESTful del servicio. Aquí te muestro un ejemplo en **Python** utilizando el paquete **Requests**:

#### Cobertura de Pruebas

El script cubre las principales operaciones CRUD en el servicio de reservas RESTful. Se están validando:

* Creación de reservas (POST)
* Consulta de reservas (GET)
* Modificación de reservas (PUT)
* Eliminación de reservas (DELETE)

Las validaciones de los códigos de respuesta aseguran que las pruebas cubren tanto los casos de éxito como los posibles errores.

### 3. ****Herramientas y Técnicas Utilizadas****

#### ****Herramientas****

* **Python Requests**: Para realizar peticiones HTTP a los servicios RESTful.
* **Unittest o Pytest**: Frameworks para estructurar y ejecutar las pruebas.
* **Postman**: Para realizar pruebas manuales preliminares y verificar las respuestas de los endpoints antes de automatizar.

#### ****Técnicas****

* **Data-driven Testing**: Para probar varias combinaciones de datos de entrada en las pruebas.
* **Validación de Códigos de Estado HTTP**: Cada prueba valida que el código de estado HTTP devuelto es el esperado.
* **Automatización Basada en Funcionalidades**: Cada funcionalidad crítica se convierte en un script automatizado, asegurando cobertura de las características esenciales del sistema.

Este plan te permitirá automatizar de manera eficiente las pruebas sobre un servicio RESTful como Booker.