Relatório - CodeReview

Quem fez o CodeReview: Ana Carolina Corrêa Rosa

Autor do código: Marcos Paulo Alves de Freitas

API: Restful-booker

Link do repositório: GitHub - Marcosdev03/Estagio-compass-uol

Branch avaliada: Main

Resultado do CodeReview:

Pontos positivos:

- Todos os 7 testes foram executados com sucesso.
- Separação das pastas em "resources" e "test", seguindo boas práticas de organização.
- Criação de uma keyword para a montagem do corpo da reserva, promovendo reaproveitamento de código.
- Validação do Status Code.
- Foram testadas todas as requisições solicitadas (GET, POST, PUT, DELETE).
- Geração de dados aleatórios que garante que a API funciona corretamente com diferentes valores, evitando depender de dados fixos.

Pontos a melhorar:

- O arquivo "variables.robot" poderia ser adicionado na pasta "resources".
- Os comandos presentes na seção "Test Cases" poderiam ser colocados em keywords específicas, como o comando "GET On Session".
- Há dois erros na execução do teste exibido no Log que recomenda o uso de `${resp.json()}` ao invés da keyword `'RequestsLibrary.To Json'`, que está obsoleta.

- As keywords `Verificar Status code 200`, `Verificar Status Code PUT 200` e `Verificar Status code 200` executam a mesma validação.

Sugestões de mudanças de acordo com os pontos a melhorar:

- Adicionar arquivo "variables.robot" na pasta "resources" para facilitar possíveis manutenções.
- Refatorar os cenários na seção "Test Cases" para conter apenas chamadas às keywords criadas. A lógica detalhada pode ser implementada no arquivo "keywords.robot", promovendo melhor organização, padronização e facilitando futuras manutenções.
- Realizar mudança de `'RequestsLibrary.To Json'` para `${resp.json()}` na linha 57 no arquivo "keywords.robot" e na linha 19 no arquivo "test_api.robot".
- Pode ser usado somente uma das keywords que valida o Status Code para todos os cenários para evitar repetição.

Conclusão:

O projeto apresenta uma boa execução dos testes, cobrindo os principais fluxos da API e garantindo seu funcionamento com dados variados. No entanto, pode ser aprimorado com uma melhor organização dos testes, reduzindo repetições e centralizando comandos comuns para facilitar a manutenção e o entendimento do código.