# Desafio #5
# Planejamento e Automação de Testes de Microsserviços

## Descrição

Planejar e automatizar os testes para os *endpoints* desenvolvidos nos microsserviços. Os testes devem seguir a perspectiva funcional dos Casos de Uso, ou seja, devem ser criados testes que cubram os fluxos definidos nos Casos de Uso. Assim, as chamadas aos *endpoints* deverão refletir a sequência de passos do fluxo que está sendo testado.

## Instruções

Esse desafio é composto de duas entregas. A primeira é o planejamento dos Casos e Roteiros de Teste conforme definido na tabela a seguir. Nesse planejamento devem constar os Casos de Teste (com número, dados de entrada e resultado esperado) e os Roteiros de Teste (com a sequência de passos a serem seguidos nos testes juntamente com os CTs a serem aplicados). A segunda entrega são os scripts de teste implementados em xUnit. Os scripts devem, obrigatoriamente, refletir o que está definido no planejamento e os nomes dos métodos de teste devem, de alguma forma, referenciar os Cas/Roteiros de Teste que implementam.

| Equipe | Planejamento | Automação |
|---|---|---|
| Aluguel | UC01, UC02, UC03, UC04, UC10 | UC01, UC02, UC03, UC04 |
| Equipamento | UC08, UC10, UC11, UC13, UC14, UC03 | UC08, UC10, UC11, UC13, UC14 |
| Externo | UC01, UC03, UC04, UC07, UC10 | UC01, UC02, UC03, UC07, UC10 |

## Entregas

- Primeira entrega: 22/03/2023
- Segunda entrega: 05/04/2023

## Dicas

1. Para implementar os scripts de teste crie um projeto de teste na mesma *solution* do projeto dos microsserviços:
    a. Tipo do projeto: Projeto de Teste do xUnit
    b. Nomes: TesteAluguel, TesteEquipamento e TesteExterno
2. O projeto de teste deve referenciar o projeto com os microsserviços.

3. Para os projetos Aluguel e Externo, colocar na última linha do Program.cs, o código

   ```
   public partial class Program { }
   ```

   para que a classe `Program` possa ser visualizada no projeto de teste.

4. Na classe de teste criar os atributos:

   ```
   using Microsoft.AspNetCore.Mvc.Testing;

   private readonly WebApplicationFactory<Program>
   application;
   private readonly HttpClient client;
   ```

5. Inicializar esses atributos no construtor da classe de teste:

   ```
   application = new WebApplicationFactory<Program>()
         .WithWebHostBuilder(builder =>
         {
               // Configure test services
         });

   client = application.CreateClient();
   ```

6. Usar o `client` nos métodos de teste para fazer as chamadas aos *endpoints*.
7. Consulte o repositório TesteMicroservico no Azure para um exemplo dessas dicas.