# MockApi Test-Driven Java Solutions

## MockApi - Detalhes e Consumo em Java (8, 11 e 17)

O **MockApi** é uma ferramenta poderosa que simula APIs reais para testes, desenvolvimento e prototipagem.

Com o MockApi, você pode definir rotas e respostas, permitindo simular operações de uma API de maneira realista.

Este guia apresenta exemplos práticos de consumo do MockApi com os métodos HTTP e HTTPS (GET, POST, PUT, PATCH, DELETE) nas versões Java 8, 11 e 17, oferecendo um conteúdo didático e de fácil aplicação.

## Acesso ao MockApi

1. **Site**: MockApi.io
2. **Configuração**:
    - Crie uma conta gratuita.
    - Configure endpoints como /users e defina respostas em formato JSON para simular diferentes interações.

## Funcionamento do MockApi

1. **Configuração de Endpoints**: Defina rotas como /users ou /products.
2. **Definição de Respostas**: O MockApi permite definir dados para cada rota simulada, facilitando o desenvolvimento e testes.
3. **Simulação de Métodos HTTP e HTTPS**: Suporta métodos como GET, POST, PUT, PATCH e DELETE, cobrindo operações CRUD.
4. **Testes e Integração**: Permite testar o consumo de APIs e validar o fluxo de dados sem a necessidade de um backend real.

# Consumo de MockApi em Java

## **Exemplo 1:** Java 8 com **HttpURLConnection** para métodos GET, POST, PUT, PATCH e DELETE

```
import javax.net.ssl.HttpsURLConnection;
import java.io.*;
import java.net.URL;

public class MockApiExampleJava8 {
    public static void main(String[] args) {
        try {
            // Método GET
            URL getUrl = new URL("https://mockapi.io/api/v1/users");
            HttpsURLConnection getConn = (HttpsURLConnection) getUrl.openConnection();
            getConn.setRequestMethod("GET");
            BufferedReader in = new BufferedReader(new
InputStreamReader(getConn.getInputStream()));
            String inputLine;
            StringBuilder content = new StringBuilder();
            while ((inputLine = in.readLine()) != null) {
                content.append(inputLine);
            }
            in.close();
            System.out.println("Resposta GET: " + content.toString());

            // Método POST
            URL postUrl = new URL("https://mockapi.io/api/v1/users");
            HttpsURLConnection postConn = (HttpsURLConnection) postUrl.openConnection();
            postConn.setRequestMethod("POST");
            postConn.setDoOutput(true);
            postConn.setRequestProperty("Content-Type", "application/json");
            String jsonInputString = "{\"name\":\"John Doe\"}";
            try(OutputStream os = postConn.getOutputStream()) {
                byte[] input = jsonInputString.getBytes("utf-8");
                os.write(input, 0, input.length);
            }
            System.out.println("Resposta POST: " + postConn.getResponseCode());

            // Método PUT
            URL putUrl = new URL("https://mockapi.io/api/v1/users/1");
            HttpsURLConnection putConn = (HttpsURLConnection) putUrl.openConnection();
            putConn.setRequestMethod("PUT");
            putConn.setDoOutput(true);
            putConn.setRequestProperty("Content-Type", "application/json");
            String jsonPutString = "{\"name\":\"Jane Doe\"}";
            try(OutputStream os = putConn.getOutputStream()) {
                byte[] input = jsonPutString.getBytes("utf-8");
                os.write(input, 0, input.length);
            }
            System.out.println("Resposta PUT: " + putConn.getResponseCode());

            // Método PATCH
            URL patchUrl = new URL("https://mockapi.io/api/v1/users/1");
            HttpsURLConnection patchConn = (HttpsURLConnection) patchUrl.openConnection();
            patchConn.setRequestMethod("PATCH");
            patchConn.setDoOutput(true);
            patchConn.setRequestProperty("Content-Type", "application/json");
            String jsonPatchString = "{\"name\":\"John Smith\"}";
```

```
        try(OutputStream os = patchConn.getOutputStream()) {
            byte[] input = jsonPatchString.getBytes("utf-8");
            os.write(input, 0, input.length);
        }
        System.out.println("Resposta PATCH: " + patchConn.getResponseCode());

        // Método DELETE
        URL deleteUrl = new URL("https://mockapi.io/api/v1/users/1");
        HttpsURLConnection deleteConn = (HttpsURLConnection) deleteUrl.openConnection();
        deleteConn.setRequestMethod("DELETE");
        System.out.println("Resposta DELETE: " + deleteConn.getResponseCode());

    } catch (Exception e) {
        e.printStackTrace();
    }
  }
}
```

## **Exemplo 2:** Java 11 com **HttpClient** para métodos GET, POST, PUT, PATCH e DELETE

```
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class MockApiExampleJava11 {
   public static void main(String[] args) throws Exception {
      HttpClient client = HttpClient.newHttpClient();

      // Método GET
      HttpRequest getRequest = HttpRequest.newBuilder()
            .uri(URI.create("https://mockapi.io/api/v1/users"))
            .GET()
            .build();
      HttpResponse<String> getResponse = client.send(getRequest,
HttpResponse.BodyHandlers.ofString());
      System.out.println("Resposta GET: " + getResponse.body());

      // Método POST
      HttpRequest postRequest = HttpRequest.newBuilder()
            .uri(URI.create("https://mockapi.io/api/v1/users"))
            .header("Content-Type", "application/json")
            .POST(HttpRequest.BodyPublishers.ofString("{\"name\":\"John Doe\"}"))
            .build();
      HttpResponse<String> postResponse = client.send(postRequest,
HttpResponse.BodyHandlers.ofString());
      System.out.println("Resposta POST: " + postResponse.body());

      // Método PUT
      HttpRequest putRequest = HttpRequest.newBuilder()
            .uri(URI.create("https://mockapi.io/api/v1/users/1"))
            .header("Content-Type", "application/json")
            .PUT(HttpRequest.BodyPublishers.ofString("{\"name\":\"Jane Doe\"}"))
            .build();
      HttpResponse<String> putResponse = client.send(putRequest,
HttpResponse.BodyHandlers.ofString());
      System.out.println("Resposta PUT: " + putResponse.body());
```

```
        // Método PATCH
        HttpRequest patchRequest = HttpRequest.newBuilder()
                .uri(URI.create("https://mockapi.io/api/v1/users/1"))
                .header("Content-Type", "application/json")
                .method("PATCH", HttpRequest.BodyPublishers.ofString("{\"name\":\"John Smith\"}"))
                .build();
        HttpResponse<String> patchResponse = client.send(patchRequest,
HttpResponse.BodyHandlers.ofString());
        System.out.println("Resposta PATCH: " + patchResponse.body());

        // Método DELETE
        HttpRequest deleteRequest = HttpRequest.newBuilder()
                .uri(URI.create("https://mockapi.io/api/v1/users/1"))
                .DELETE()
                .build();
        HttpResponse<String> deleteResponse = client.send(deleteRequest,
HttpResponse.BodyHandlers.ofString());
        System.out.println("Resposta DELETE: " + deleteResponse.body());
    }
}
```

**Exemplo 3:** Java 17 com **HttpClient** para métodos GET, POST, PUT, PATCH e DELETE

```
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class MockApiExampleJava17 {
    public static void main(String[] args) throws Exception {
        HttpClient client = HttpClient.newHttpClient();

        // Método GET
        HttpRequest getRequest = HttpRequest.newBuilder()
                .uri(URI.create("https://mockapi.io/api/v1/users"))
                .GET()
                .build();
        HttpResponse<String> getResponse = client.send(getRequest,
HttpResponse.BodyHandlers.ofString());
        System.out.println("Resposta GET: " + getResponse.body());

        // Método POST
        HttpRequest postRequest = HttpRequest.newBuilder()
                .uri(URI.create("https://mockapi.io/api/v1/users"))
                .header("Content-Type", "application/json")
                .POST(HttpRequest.BodyPublishers.ofString("{\"name\":\"John Doe\"}"))
                .build();
        HttpResponse<String> postResponse = client.send(postRequest,
HttpResponse.BodyHandlers.ofString());
        System.out.println("Resposta POST: " + postResponse.body());

        // Método PUT
        HttpRequest putRequest = HttpRequest.newBuilder()
                .uri(URI.create("https://mockapi.io/api/v1/users/1"))
                .header("Content-Type", "application/json")
                .PUT(HttpRequest.BodyPublishers.ofString("{\"name\":\"Jane Doe\"}"))
                .build();
```

```
        HttpResponse<String> putResponse = client.send(putRequest,
HttpResponse.BodyHandlers.ofString());
        System.out.println("Resposta PUT: " + putResponse.body());

        // Método PATCH
        HttpRequest patchRequest = HttpRequest.newBuilder()
                .uri(URI.create("https://mockapi.io/api/v1/users/1"))
                .header("Content-Type", "application/json")
                .method("PATCH", HttpRequest.BodyPublishers.ofString("{\"name\":\"John Smith\"}"))
                .build();
        HttpResponse<String> patchResponse = client.send(patchRequest,
HttpResponse.BodyHandlers.ofString());
        System.out.println("Resposta PATCH: " + patchResponse.body());

        // Método DELETE
        HttpRequest deleteRequest = HttpRequest.newBuilder()
                .uri(URI.create("https://mockapi.io/api/v1/users/1"))
                .DELETE()
                .build();
        HttpResponse<String> deleteResponse = client.send(deleteRequest,
HttpResponse.BodyHandlers.ofString());
        System.out.println("Resposta DELETE: " + deleteResponse.body());
    }
}
```

## Conclusão

Utilizar o MockApi com diferentes métodos HTTP é essencial para simular interações de APIs e criar aplicações robustas e preparadas para ambiente real. O conteúdo demonstrou exemplos com Java 8, 11 e 17, que fornecem diversas opções para o consumo de APIs, destacando a importância de usar a versão adequada para o projeto, mantendo desempenho e compatibilidade.

**EducaCiência FastCode para comunidade**