**Conceitos**

API (Interface de Programação de Aplicações) é um tipo de implementação de software que permite expor um código backend com uma interface sem a necessidade de acesso ao código fonte.

Uma das arquiteturas se chama REST (Transferência Representacional de Estado).

O ciclo de uso de uma API funciona basicamente assim: o usuário, através de uma interface gráfica, consegue buscar, inserir, alterar ou excluir dados em uma aplicação sem necessidade de utilizar codificação, apenas com comandos que passarão os métodos ao back-end e banco de dados.

Usuário > Interface Gráfica > API > Envio do Método > Back-end / Banco de Dados.

A comunicação entre as Interfaces são chamadas de requisições.

Uma requisição é composta por:

* Metodo (GET, POST, PUT, DELETE)
* URI: Endereço onde está a API junto com o recurso.
* Cabeçalho: são informações técnicas (usuário autenticado, permissões, token, etc)
* Body: é informado os dados que serão editados.

**Entendendo o JSON**

JSON (Notação de Objetos JavaScript) é o formato de troca de dados normalmente utilizado nas APIs Rest.

Nas documentações é possível verificar os tipos de valores a serem preenchidos e como enviar os métodos.

A documentação de uma API pode ser criada de diversas formas. A mais conhecida é no formato Swagger, nela podemos descrever todas as informações necessárias para integração com aquela API.

Os agrupamentos de comandos que podem ser realizados na API são chamados de controladores.

**Postman**

Para manipulação e testes usaremos a ferramenta postman que possui diversos recursos para interagir com as APIs.

Na aba collection criamos um grupo de interações para um API.

Sequência de configuração postman:

* Criar um coleção e abrir uma nova requisição;
* Escolher o método e colocar URI + instrução da requisição;
* Configurar o Header com a Key e Value;
* Na seção Body escolher “Raw” e arquivo no formato JSON;
* Enviar a requisição.

Na aba environments é possível criar variáveis globais, como por exemplo uma variável com o link base da api, otimizando o processo.

**Métodos**

POST: usado para enviar informações dentro da requisição. Nesse modo é enviado informações de forma mais segura.

GET: utilizado para buscar uma ou muitas informações relacionadas a um recurso específico.

Nesse método é possível colocar filtros para buscar por algo específico utilizando parâmetros do tipo query que é passado depois do endereço do recurso, utilizando “?” e o parâmetro. **Ex: {{uri}}/v2/produtos?produtoNome=playstatio 5**

hg

PUT: utilizado para alterar um recurso existente.

DELETE: utilizado para alterar um recurso existente.

**Status Code ou código de estado HTTP**

400 Bad Request: erro relacionado a algo que foi feito de forma errada por quem está solicitando a request, por exemplo ter colocado alguma informação incorreta no body.

401 unauthorized: falta de autenticação para realizar a request, por exemplo a não especificar o token no header.

403 forbiden: erro relacionado a permissionamento.

404 Not found: erro de não encontrado, por exemplo ao buscar um produto que não está cadastrado no sistema.

409 conflict: erro de conflito de informação, exemplo ao tentar cadastrar um usuário já existente.

422: pode estar relacionado a regra de negócio, ao tentar fazer alguma inserção não permitida no sistema definido pela regra.

500 internal error: erro interno não tratado pelo desenvolvedor que impede o funcionamento do software.

**Heurística de testes de API Rest VADER**

(V)erbos:

Testar outros métodos não especificados para determinada ação, por exemplo na criação de usuário.

* Utilizei um método permitido para registro de usuário e tive sucesso.
* Outros métodos para o endpoint registro de usuário não são permitidos.

(A)utenticação e autorização:

Testar métodos que precisarão de autenticação e permissões.

* Token correto, sucesso.
* Sem token, erro 401 como esperado.
* Token invalido, retorno com erro 500 ao invés de 401.

(D)ados:

Testar valores diferentes dentro do body.

* Enviado valorProduto dooble em aspas igual string é permitido o envio.
* Enviei o produtoNome como true (booleano) e salvou como “1”, isso era esperado?
* Quando o produtoNome tem um texto de 644 caracteres, ao tentar registrar, recebe um status code igual a 201 e vejo no body que o id do produto não muda, além disso vejo que o componente informado é adicionado à lista de componentes do último produto cadastrado.

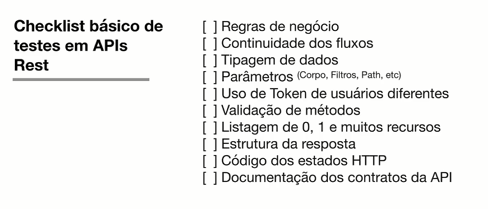
(E)rros:

Teste dos erros ja documentados.

(R)esponsividade:

Verificar o tempo de resposta em cada request. Realize mais de um teste, anote os tempos e calcule a média de todos os tempos, verifique com o time qual tempo estipulado para cada teste.

**Checklist testes em APIs Rest**



-Testar os casos de testes;

-Em continuidade dos fluxos é preciso testar se todos itens dependentes estão funcionando;

-Tipagem de dados verifica que o retorno está como esperado na documentação;

-Parâmetros é testados os parametros documentados da request;

-Testar o uso de outros tokens com diferentes permissões;

-Validação dos métodos fazendo requests com métodos diferentes;

-Testar listagens, exemplo lista sem nenhum produto cadastrado e ir aumentando a quantidade.

- verificar estrutura de resposta conforme documentação;

-Verificar status code conforme documentação;

**Exercício prático**

**Grupo usuário**

**Obter token do usuário**

* Testei a request obter token do usuário sem passar login e senha e apresentou o status code 404 not found.

**Adicionar um novo usuário**

* Ao tentar enviar a request sem preencher os campos é apresentado um status code 404 Not Found.
* Ao tentar criar o usuraio com o preenchimento conforme pede a documentação apresenta o status code 404 Not Found.

**Automação API Rest**