3. Modularização — Reutilizando Funções

Por que isso existe?

Porque em projetos profissionais você **não copia lógica de login** pra 10, 30 ou 100 scripts.

Você cria **um módulo** (ex: `auth.js`) que faz o login uma vez e qualquer script pode usar.

Isso torna seu projeto:

- mais limpo
- mais rápido de manter
- mais profissional

---

auth.js — módulo de login reutilizável

```
import http from 'k6/http';
import { check } from 'k6';

export function fazerLogin(baseUrl) {
  const payload = JSON.stringify({
    email: "fulano@qa.com",
    password: "teste",
  });

  const headers = { 'Content-Type': 'application/json' };

  const res = http.post(`${baseUrl}/login`, payload, { headers });

  check(res, {
    'login bem sucedido': (r) => r.status === 200,
  });

  const token = res.json('authorization');
```

```
    console.log(`Token obtido: ${token}`);

    return token;
}
```

O módulo:

- Faz login
- Valida o status
- Retorna o token
- É reutilizável em **qualquer** teste

---

teste.js — usando o módulo

```
import { fazerLogin } from '../utils/auth.js';
import http from 'k6/http';
import { check } from 'k6';

export const options = {
    vus: 1,
    duration: '10s'
};

export function setup() {
    const baseUrl = __ENV.BASE_URL || 'http://localhost:3000';
    const authToken = fazerLogin(baseUrl);

    return { baseUrl, authToken };
}

export default function (data) {

    const headers = {
        'Content-Type': 'application/json',
```

```
    'Authorization': data.authToken
  };

  const res = http.get(`${data.baseUrl}/produtos`, { headers });

  check(res, {
    'consulta produtos': (r) => r.status === 200,
  });
}
```

Por que isso melhora sua vida?

- Você vira **profissional**, não “copiador de código”
- Login fica centralizado em **um único arquivo**
- Se amanhã muda `/login` → `/auth`, você altera **uma única linha**
- Reutiliza o módulo em:
  - testes de carga
  - smoke test
  - stress test
  - ramping
  - CRUD
  - controladoras diferentes