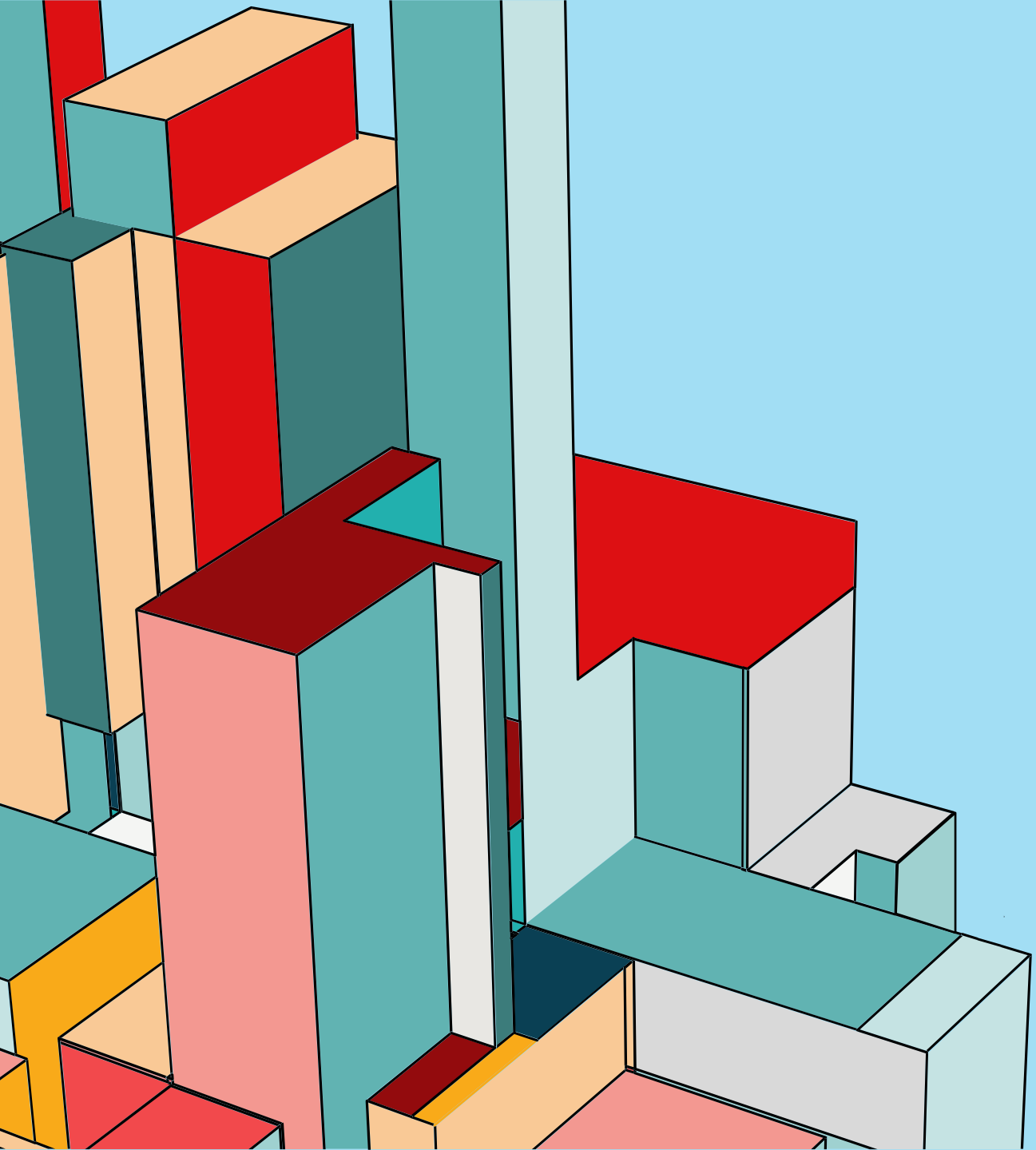
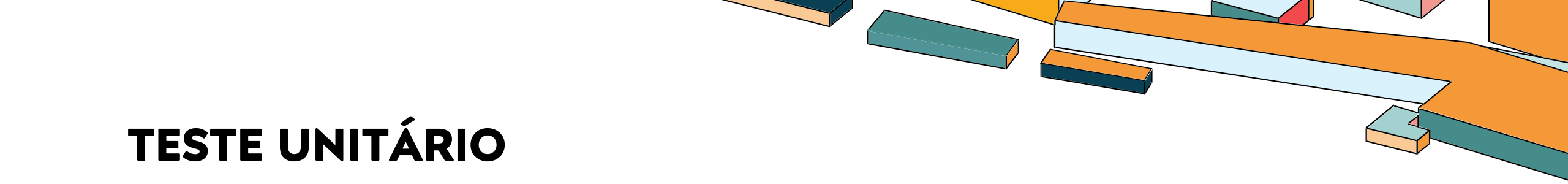


UNIVAS
UNIVERSIDADE DO VALE DO SAPUCAÍ

TESTE DE SOFTWARE

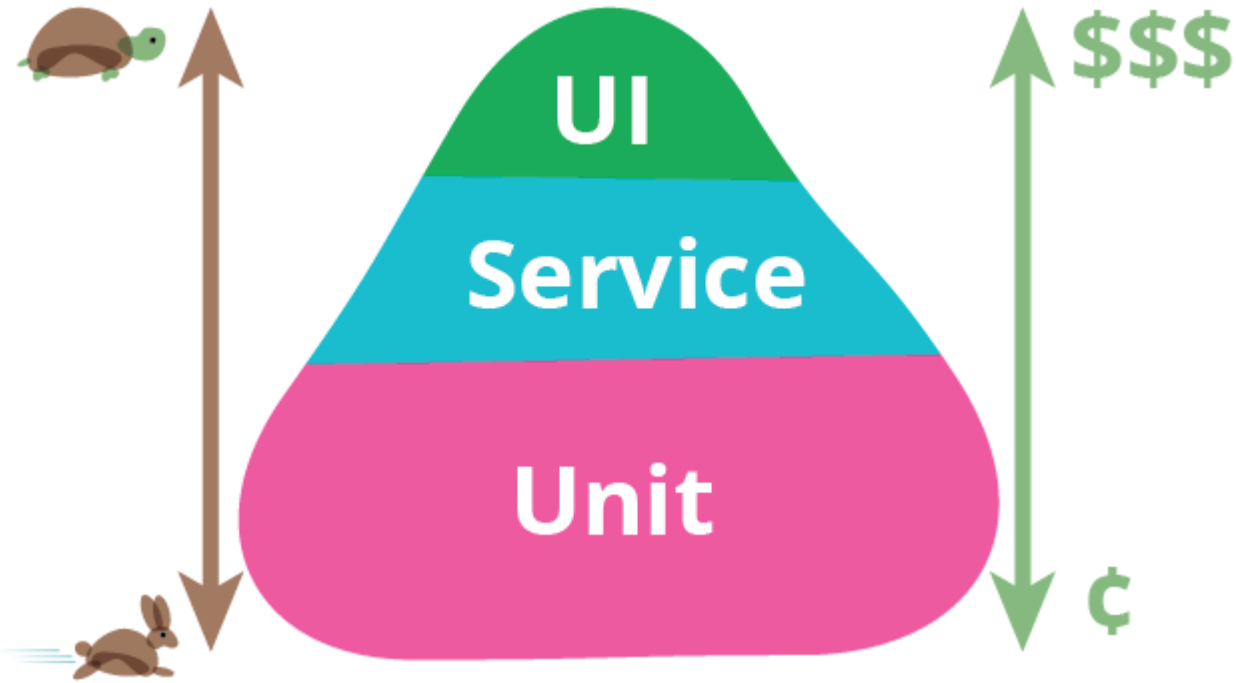
TESTES UNITÁRIOS

Prof. Flávio Belizário da Silva Mota
Universidade do Vale do Sapucaí – UNIVAS
Sistemas de Informação



TESTE UNITÁRIO

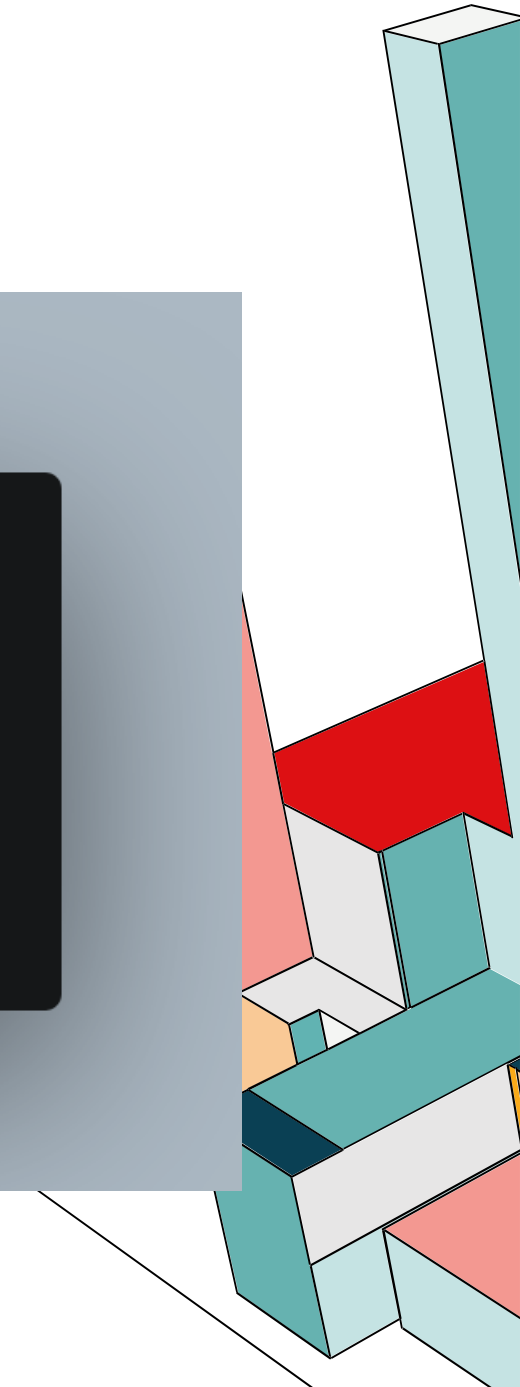
- Um **teste unitário** verifica o comportamento de **uma parte isolada** do código: normalmente uma função, método ou componente.
- São **testes baratos (de baixo custo e esforço) e rápidos de serem executados.**
- São escritos por pessoas desenvolvedoras.



TESTE UNITÁRIO

```
function soma(a, b) {
  return a + b
}

console.assert(soma(2, 3) === 5, 'Erro na função soma!')
```



BENEFÍCIOS DO TESTE UNITÁRIO

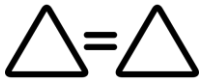
- Evitam regressões quando o código muda
- Facilitam refatorações com segurança
- Servem como uma documentação do sistema
- Melhoram a qualidade e previsibilidade das entregas

Um bom teste unitário vai funcionar como um alarme: ele dispara quando algo sai do lugar.

BOAS PRÁTICAS

Um teste unitário deve ser:

Isolado: não depende de rede, banco ou outros módulos.



Determinístico: sempre gera o mesmo resultado.

Pequeno e rápido: deve rodar em milissegundos.

Claro: fácil de entender o que ele testa.

A ESTRUTURA DE UM TESTE UNITÁRIO — PADRÃO AAA

- Todo teste unitário bem escrito segue uma estrutura lógica conhecida como AAA:

1. **Arrange (Preparar):** montar o cenário do teste (criar dados, instâncias, variáveis).
2. **Act (Agir):** executar a ação a ser testada (chamar a função ou método).
3. **Assert (Verificar):** conferir se o resultado obtido é o esperado.

AUTOMATIZAÇÃO DO TESTE UNITÁRIO

- Nessa disciplina, iremos automatizar os testes com o **Vitest**, um framework para projetos JavaScript e TypeScript.
- O **Vitest** é um framework inspirado no Jest, mas feito para o ecossistema do **Vite**.
- Benefícios:
  - Execução quase instantânea
  - Suporte nativo a ESM e TS
  - Relatórios
  - Mocks e snapshots embutidos

ESTRUTURA BÁSICA DE UM TESTE

soma.js

```
export function soma(a, b) {
  return a + b
}
```

soma.test.js

```
import { expect, test } from 'vitest'
import { soma } from './soma.js'

test('soma dois números', () => {
  expect(soma(1, 2)).toBe(3)
})
```

- O Vitest organiza os testes com `test()` e verifica comportamentos com `expect()`.
- test(nome, função) define o teste.
  expect(resultado).toBe(valorEsperado) faz a verificação