**O que é Cypress?**

**Cypress** é um framework de teste **end-to-end (E2E)** para aplicações web modernas.

**Características principais:**

* Focado em testes que simulam um usuário real interagindo com a aplicação
* Executa testes em um navegador real
* Tem um runner interativo que permite ver a execução dos testes
* Inclui ferramentas de debugging poderosas
* Suporta testes de frontend e backend (API)

**O que é Jest?**

**Jest** é um framework de teste **unitário** desenvolvido pelo Facebook.

**Características principais:**

* Focado em testes de unidades individuais de código
* Extremamente rápido (executa em Node.js, sem navegador)
* Inclui recursos como mocking, spies e cobertura de código
* Muito popular para testar JavaScript/TypeScript, especialmente com React

**Diferença entre Teste Unitário e Teste de Integração**

**Teste Unitário**

* **Foco**: Testa unidades individuais de código isoladamente
* **Escopo**: Uma função, método ou classe específica
* **Velocidade**: Muito rápido (milissegundos por teste)
* **Dependências**: Usa mocks para isolar a unidade testada
* **Objetivo**: Garantir que cada parte funciona corretamente sozinha

**Exemplo**: Testar uma função que calcula imposto, mockando qualquer chamada de API ou banco de dados.

**Teste de Integração**

* **Foco**: Testa como múltiplas unidades trabalham juntas
* **Escopo**: Módulos, serviços ou componentes interagindo
* **Velocidade**: Mais lento que unitário (depende das integrações)
* **Dependências**: Pode usar instâncias reais ou mocks mais simples
* **Objetivo**: Garantir que as partes se comunicam corretamente

**Exemplo**: Testar um serviço que chama o banco de dados e depois processa os dados.

**1. Mensageria e Filas**

* **O que é?**: Sistemas que permitem a comunicação **assíncrona** entre aplicações por meio de mensagens.
* **Para que serve?**: Garantir que mensagens sejam processadas em ordem, sem perdas, mesmo sob alta carga.
* **Exemplo de uso**: Processamento de transações financeiras, pedidos em e-commerce, etc.

**2. RabbitMQ**

* **O que é?**: Um *broker* de mensagens que segue o protocolo **AMQP**.
* **Características**:
  + **Desacoplamento**: Serviços não precisam estar ativos ao mesmo tempo.
  + **Confiança**: Mensagens podem ser confirmadas e reenviadas em caso de falha.
  + **Escalabilidade**: Suporta múltiplos consumidores e filas.
* **Cenário típico**: Filas com prioridades (ex.: transações VIP vs. comuns).

**3. Apache Kafka**

* **O que é?**: Plataforma de **streaming distribuída** orientada a logs.
* **Características**:
  + **Alto throughput**: Milhões de mensagens por segundo.
  + **Persistência**: Mensagens ficam salvas em disco e podem ser reprocessadas.
  + **Tempo real**: Ideal para processamento contínuo de grandes volumes.
* **Cenário típico**: Análise de fraudes em transações financeiras em tempo real.

**4. Diferenças Principais**

| **RabbitMQ** | **Apache Kafka** |
| --- | --- |
| Broker de mensagens | Plataforma de streaming |
| Foco em filas e troca de mensagens | Foco em fluxo contínuo de dados |
| Ideal para tarefas assíncronas | Ideal para processamento em tempo real |
| Mensagens são removidas após consumo | Mensagens persistem por tempo configurável |

**5. Aplicação Prática (MoneyFlow)**

* **Problema**: Atrasos em horários de pico.
* **Solução com RabbitMQ**: Filas prioritárias para transações VIP.
* **Solução com Kafka**: Tópicos com múltiplos consumidores para processamento paralelo.

**6. Vantagens da Mensageria**

* Escalabilidade
* Tolerância a falhas
* Desacoplamento entre serviços
* Organização e priorização de tarefas