# Introdução a **Testes**

Por que são tão importantes e muitos ignoram

# **Por que** escrever testes?

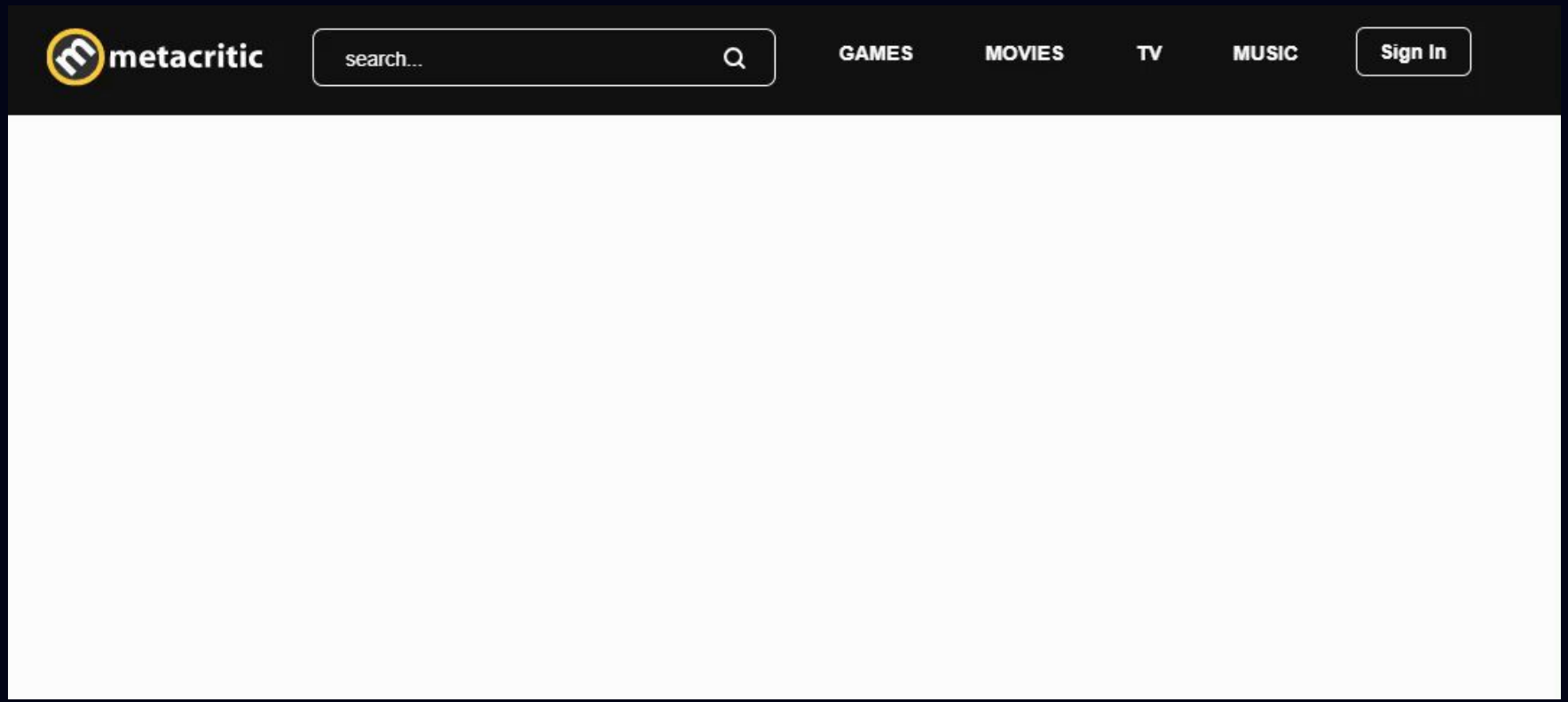
- Código complexo não é simples de se debuggar só com olho
- Testar é uma forma robusta de **validar software**
  - Funciona como eu espero?
  - Funciona como o usuário espera?
  - Se eu atualizar esse trecho, o código quebra?
- Testes funcionam como uma primeira camada de documentação

# Que tipos de teste existem?

- Testes Unitários
  - Testam **isoladamente** pequenas unidades de código
  - O código está se comportando como o desenvolvedor espera?
- Testes Funcionais
  - Checa se as unidades funcionam também entre si
  - Testes podem conter múltiplas classes, métodos, etc
  - O programa funciona de acordo com o que o usuário espera?

"

Vamos imaginar um simples **componente de busca**


metacritic
search...
GAMES
MOVIES
TV
MUSIC
Sign In

# Algumas features do componente

- Expande quando em foco
- Envia um request para um banco com a palavra digitada
- Recebe dados e renderiza na tela a lista
- Destaca dentro dos itens a string digitada
- Quando não possui nenhum item, mostra uma mensagem

# Para os testes unitários, como faríamos?

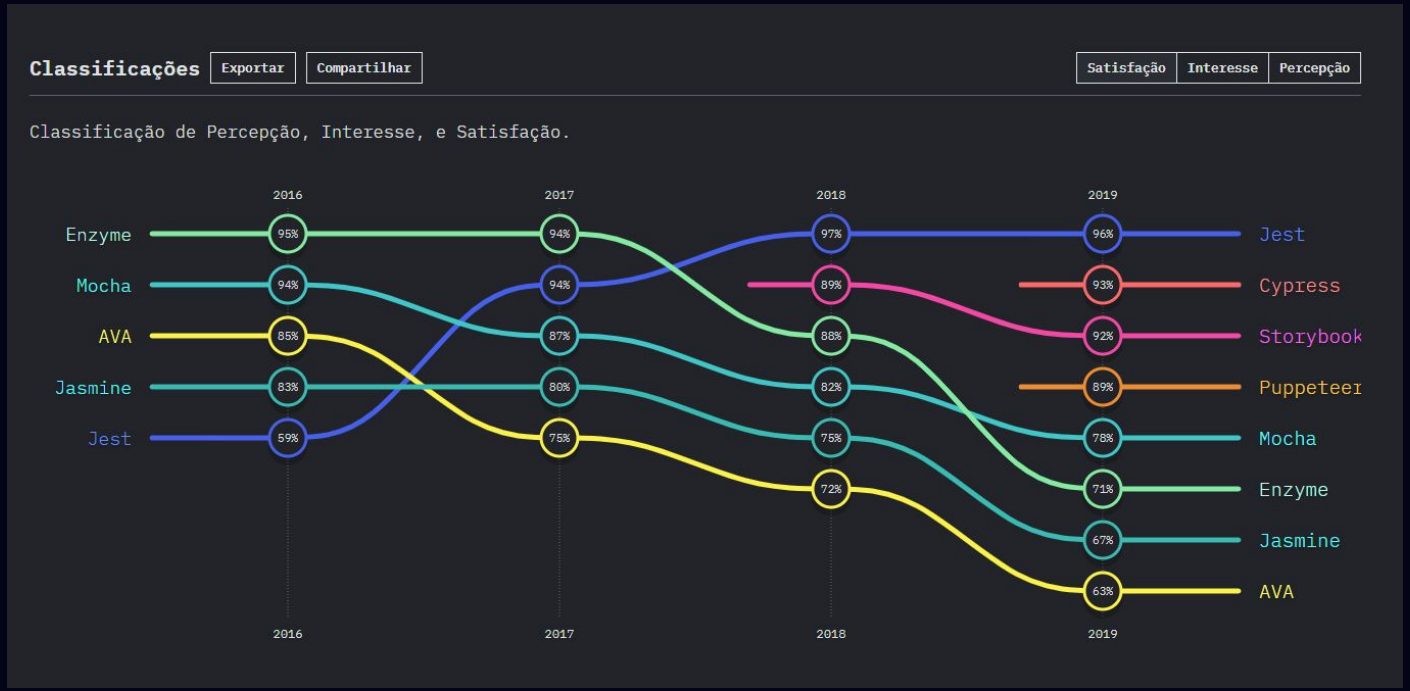
- Isolamos só o componente de busca (criando mock para o serviço)
- E então testamos cada um dos comportamentos de **forma separada**
  - Se o campo recebe focus, ele expande? (Adição de classe, por exemplo)
  - Ao ter uma string no campo, o método de fetch da API é chamado?
  - A string do campo é passada corretamente para a API e os dados retornados são condizentes?
  - Dado um conjunto de dados recebido? A lista é renderizada?
  - Se não tiver dados, uma mensagem é preenchida?

# Para os testes funcionais, como faríamos?

- Aqui não há mais isolamento entre os métodos e nós realizamos o fluxo completo
  - Simula usuário clicando
  - Simula usuário digitando e apagando
  - Simula usuário clicando num item da pesquisa
- E analisa todo o fluxo para ver se o comportamento funciona como esperado

# Test-driven development (TDD)

# E as ferramentas?




https://2019.stateofjs.com/testing/

"

Baseados nos níveis de **satisfação** e **adoção**, nosso framework escolhido foi o **Jest.**

Além dele possuir várias ferramentas já incluídas, como facilidade para mocks, relatório de cobertura, métodos para tratar intervals, snapshots e outros detalhes mais.

"

Para testar nossos componentes em React a escolha foi a biblioteca **React Testing Library,** por ela possuir melhor suporte as novas tecnologias do React como os **hooks** e também uma **renderização real** do componente.

"

E nossa biblioteca para end-to-end foi o **Cypress**, também devido aos níveis de satisfação e adoção na comunidade, além de ser a mais performática dentre as bibliotecas do tipo.