In-Source Test

O Vitest também fornece uma maneira de executar testes em seu código-fonte junto com a implementação, semelhante aos testes de módulo do Rust.

Isso faz com que os testes compartilhem o mesmo fechamento que as implementações e sejam capazes de testar contra estados privados sem exportar. Enquanto isso, também traz um ciclo de feedback mais próximo para o desenvolvimento.

apps > my-app > src > App.tsx > ...

```tsx
import { useState } from 'react';
import './App.css';
import { Button } from 'ziegler-ui';
import { userService } from './services';

export function add(...args: number[]) {
  return args.reduce((a, b) => a + b, 0);
}

function App() {
  const [user, setUser] = useState<any>(null);

  const handleOnClick = async () => {
    const { data } = await userService.getUserProfile(1);

    setUser(data);
  };

  return (
    <div className="App">
      <Button label="Fetch User" onClick={handleOnClick} />
      {user && <p>{user.name}</p>}
    </div>
  );
}

export default App;

if (import.meta.vitest) {
  const { it, expect, test, describe } = import.meta.vitest;
  describe('App', () => {
    test('Button be defined', async () => {
      it('add', () => {
        expect(add()).toBe(0);
        expect(add(1)).toBe(1);
        expect(add(1, 2, 3)).toBe(6);
      });
    });
  });
}

```

Test Context

Inspirado em Playwright Fixtures, o contexto de teste do Vitest permite definir utils, estados e fixtures que podem ser usados em seus testes.

```ts
interface LocalTestContext {
  buttonLabel: string;
}

describe('App', () => {
  setupTests();
  beforeEach<LocalTestContext>(async (context) => {
    // typeof context is 'TestContext & LocalTestContext'
    context.buttonLabel = 'Fetch User';
  });
  test<LocalTestContext>('Button be defined', async (context) => {
    render(<App />);

    const button = await screen.findByText(context.buttonLabel);
    expect(button).toBeDefined();
  });
  test<LocalTestContext>('Fetch user if clicked on button', async (context) => {
    const wrapper = render(<App />);

    const button = wrapper.getByText(context.buttonLabel);
    userEvent.click(button);
    await wrapper.findByText(/john/i);
  });
});
```