Documentação de Testes

Testes do Backend

No backend para executamos os testes unitários usamos a biblioteca Vitest (https://vitest.dev/) que é uma plataforma de teste automatizada que permite a criação, execução e gerenciamento de testes de software de forma eficiente. É uma ferramenta desenvolvida para facilitar e otimizar os processos de teste, ajudando equipes de desenvolvimento a garantir a qualidade e a confiabilidade de seus produtos.
A definição para o Vitest na documentação de testes pode ser a seguinte:

Vitest é uma plataforma de teste automatizada projetada para simplificar e aprimorar o processo de teste de software. Ele permite que os desenvolvedores criem, executem e gerenciem testes de forma eficiente, ajudando a identificar problemas e garantir a qualidade dos aplicativos. Com o Vitest, as equipes de desenvolvimento podem automatizar testes repetitivos, acelerar o ciclo de testes e obter resultados confiáveis. A plataforma fornece recursos abrangentes para criar cenários de teste, realizar testes de regressão, integrar-se a pipelines de CI/CD e gerar relatórios detalhados. Com o Vitest, as equipes podem melhorar a eficiência do teste, reduzir erros e entregar software de alta qualidade aos usuários finais.

Essa definição destaca as principais funcionalidades e benefícios do Vitest, incluindo automação de testes, gerenciamento de testes, integração com pipelines de CI/CD e relatórios detalhados. Ela ressalta como o Vitest pode ajudar a equipe de desenvolvimento a melhorar a eficiência do teste e a entregar software de qualidade. Certifique-se de personalizar a definição para atender às necessidades e ao contexto específico de sua documentação de testes.

Exemplo de testes unitários no Backend:

Teste de criação de usuario

Aqui testamos se a criação de usuário será feita corretamente

```
import { faker } from '@faker-js/faker'
import { PrismaClient } from '@prisma/client'
import { test, expect, describe, afterAll, beforeAll } from 'vitest'

import { createUser } from '../../models/user'

const prisma = new PrismaClient()

describe('createUser', () => {
 beforeAll(async () => {
  await prisma.user.delete({ where: {} })
 })

 afterAll(async () => {
  await prisma.user.delete({ where: {} })
  await prisma.$disconnect()
 })

 test('Should create an user', async () => {
  const user = {
   username: faker.internet.userName(),
   email: faker.internet.email(),
   password: faker.internet.password(),
   createdAt: new Date(),
   updatedAt: new Date()
  }

  const createdUser = await createUser(user)

  expect(createdUser.username).toBe(user.username)
  expect(createdUser.email).toBe(user.email)
  expect(createdUser.password).toBe(user.password)
 })
})

```

Testes do Frontend

No Angular para testes de componentes usamos o Karma que é é uma ferramenta de teste automatizado desenvolvida para o ecossistema do Angular. Ela fornece um ambiente robusto e flexível para executar testes de unidade em aplicações Angular de forma eficiente. Com o Karma, desenvolvedores podem automatizar a execução de testes em vários navegadores, garantindo a consistência e a qualidade do software em diferentes ambientes.
A definição para o Karma na documentação pode ser a seguinte:

Karma é uma ferramenta de teste automatizado projetada para facilitar e agilizar o processo de teste de unidade em aplicações Angular. Com o Karma, você pode executar testes em diferentes navegadores, verificar a correta funcionalidade de componentes, serviços, diretivas e outros elementos do Angular, e obter relatórios detalhados sobre os resultados dos testes. Além disso, o Karma permite a automação de testes contínuos, recarregando automaticamente os testes quando ocorrem alterações nos arquivos de origem. Ele é amplamente utilizado em ambientes de integração contínua (CI) e implantação contínua (CD) para garantir a qualidade do software em cada etapa do processo de desenvolvimento. Com o Karma, você pode ter confiança na estabilidade e no comportamento esperado de suas aplicações Angular, melhorando a eficiência e a confiabilidade do processo de teste.

Essa definição destaca as principais funcionalidades e benefícios do Karma, como a execução de testes em vários navegadores, a verificação de componentes e a integração com CI/CD. Certifique-se de personalizar a definição para atender às necessidades e ao contexto específico de sua documentação de testes.

Exemplo de teste do Frontend:
Aqui testamos se o componente SignInComponent está sendo renderizado corretamente

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { SignInComponent } from './sign-in.component';

describe('SignInComponent', () => {
  let component: SignInComponent;
  let fixture: ComponentFixture<SignInComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ SignInComponent ]
    })
    .compileComponents();

    fixture = TestBed.createComponent(SignInComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});

```

Grupo 9 – Pass Manager

Guilherme Poleti - 22023873
Heitor Dias - 22023799
Mary Alice Aparecida Guilherme - 22023051
Mauricio Cardoso - 22023081
Mateus Quintino - 22023854
Thais Landim - 22023274