RESUMO DA AULA

TDD e o conceito de Vermelho, Verde e Refatoração

Nesta aula, vamos falar sobre um conceito fundamental no desenvolvimento de testes automatizados chamado TDD (Test-Driven Development) e entender o processo de Vermelho, Verde e Refatora o. O que o TDD? O TDD, ou desenvolvimento orientado a testes, uma abordagem em que escrevemos os testes antes de desenvolvermos a funcionalidade. Assim, o c digo desenvolvido pensando primeiro no teste e, posteriormente, na implementa o. Essa metodologia ideal para garantir que o c digo atende aos requisitos definidos desde o in cio. O Ciclo Vermelho, Verde e Refatora o O processo de TDD guiado por tr s etapas principais: Vermelho: Escrevemos um teste que inicialmente falha (pois a funcionalidade ainda n o foi implementada). Verde: Escrevemos o c digo m nimo necess rio para passar no teste. Refatora o: Melhoramos o c digo mantendo todos os testes passando. Aplicando o ciclo de Vermelho, Verde e Refatora o Vamos aplicar esse conceito no nosso exemplo. Imagine que agora recebemos um novo requisito: precisamos adicionar um valor extra de 20% no frete para entregas destinadas a estados espec ficos, como RS e SC. Vamos seguir o ciclo: 1. Escrevendo o teste (Vermelho) Come amos criando um teste para garantir que, ao calcular o valor do pedido com um estado espec fico (RS), o valor do frete seja acrescido de 20%. it('deve adicionar 20% no valor da entrega caso o estado do pedido seja RS e valor de produtos menor que 500', () => { const pedidoComEstadoRS = { estado: 'RS', itens: [{ nome: 'Sandu che', valor: 10 }, { nome: 'Bota nova', valor: 400 }, { nome: 'Entrega', valor: 100, entrega: true }] }; const resultado = calcularValorPedido(pedidoComEstadoRS); expect(resultado).toBe(530); // 20% de 100 20, logo 100 + 20 = 120, e 10 + 400 + 120 = 530 }); Rodamos o teste e verificamos que ele falha. Estamos no ponto Vermelho: o c digo atual ainda n o tem a funcionalidade implementada. 2. Escrevendo o c digo para passar no teste (Verde) Agora, adicionamos a l gica m nima para que o teste passe: if (pedido.estado === 'RS') { const acrescimoEntrega = entrega.valor * 0.2; // 20% de acr scimo

entrega.valor += acrescimoEntrega; } Salvamos e verificamos que o teste passa. Isso nos coloca no ponto Verde: o c digo est funcionando, mas ainda pode n o estar otimizado. 3. Refatora o Com o teste passando, agora podemos pensar em melhorar o c digo. Por exemplo, podemos criar uma vari vel constante para a taxa de acr scimo: const taxaAcrescimo = 0.2; if (pedido.estado === 'RS') { entrega.valor += entrega.valor * taxaAcrescimo; } Ap s a refatora o, verificamos novamente se os testes continuam passando. Se tudo estiver correto, seguimos para adicionar mais testes. Adicionando mais cen rios de teste Agora, queremos cobrir mais cen rios, como o caso em que o estado SC e o valor do pedido menor que 500 reais. it('deve adicionar 20% no valor da entrega caso o estado do pedido seja SC e valor de produtos menor que 500', () => { const pedidoComEstadoSC = { estado: 'SC', itens: [{ nome: 'Sandu che', valor: 10 }, { nome: 'Bota nova', valor: 400 }, { nome: 'Entrega', valor: 100, entrega: true }] }; const resultado = calcularValorPedido(pedidoComEstadoSC); expect(resultado).toBe(530); }); Rodamos e, novamente, o teste falha (ponto Vermelho). Adicionamos a corre o para passar no teste: if (pedido.estado === 'RS' || pedido.estado === 'SC') { entrega.valor += entrega.valor * taxaAcrescimo; } Refatora o Final Podemos fazer algumas melhorias no nosso c digo para mant -lo leg vel e eficiente. Por exemplo: Criar uma lista de estados que devem ter o acr scimo de 20%. Verificar se o estado do pedido est presente nessa lista: const taxaAcrescimo = 0.2; const estadosComAcrescimo = ['RS', 'SC']; if (estadosComAcrescimo.includes(pedido.estado)) { entrega.valor += entrega.valor * taxaAcrescimo; } Com essa mudan a, mantemos o c digo limpo e flex vel para futuros ajustes. Verificando a aus ncia de acr scimo para outros estados Por fim, vamos adicionar um teste para garantir que, caso o estado n o seja RS ou SC, o valor do frete permane a inalterado: it('n o deve adicionar 20% no valor da entrega caso estado do pedido seja diferente de SC ou RS e valor de produtos menor que 500', () => { const pedidoComEstadoSP = { estado: 'SP', itens: [{ nome: 'Sandu che', valor: 10 }, { nome: 'Bota nova', valor: 400 }, { nome: 'Entrega', valor: 100, entrega: true }] }; const resultado = calcularValorPedido(pedidoComEstadoSP); expect(resultado).toBe(510); // N o deve adicionar o valor extra }); Resumo Nesta aula, aplicamos o ciclo de Vermelho, Verde e Refatora o usando o

TDD para implementar uma nova funcionalidade com seguran a e confian a. Cada etapa foi validada por testes automatizados, garantindo que nosso c digo n o quebrasse em nenhum ponto. Esse o poder do TDD: nos permite desenvolver de forma iterativa, corrigindo falhas rapidamente e deixando nosso c digo mais robusto e f cil de manter. Espero que este m dulo tenha te mostrado como testes automatizados podem transformar a maneira como voc desenvolve suas funcionalidades, trazendo mais qualidade e tranquilidade para o seu c digo.
