

Nome: nilton Dionisio Guerra

Versão nilton

Tendo em vista a complexidade do sistema e tempo disponível foi devido que será utilizado as seguintes estratégias, abordagens e tipos de testes que iremos discorrer abaixo.

Para o desenvolvimento do projeto, a implementação de testes de integração e de testes de caixa preta será de fundamental importância. Dado que o sistema em questão é de médio porte, é indispensável a realização de testes que verifiquem a integração entre os módulos que serão implementados ao longo do desenvolvimento. Além disso, a adoção de testes de caixa preta se mostra essencial para garantir a verificação da funcionalidade do sistema do ponto de vista do usuário final.

Embora os testes de caixa branca sejam geralmente considerados cruciais em qualquer projeto, a decisão de não os utilizar neste caso está relacionada à natureza individual do desenvolvimento, o que poderia tornar a implementação desses testes mais desafiadora e demorada. Da mesma forma, optou-se por não incluir testes de regressão, que, apesar de serem altamente recomendados para projetos de médio e grande porte, exigiriam um investimento significativo de tempo e recursos que não se alinham com as restrições deste projeto.

Os testes de caixa preta que serão realizados têm como objetivo principal validar a usabilidade, simplicidade, clareza do sistema, além de identificar possíveis erros. Para isso, diferentes indivíduos executarão esses testes, garantindo que a avaliação seja abrangente e imparcial. No contexto deste projeto, serão aplicadas várias técnicas específicas de teste, cada uma com um foco distinto, para assegurar a qualidade do sistema.

Primeiramente, o teste de caminho será utilizado para verificar se os fluxos de execução dentro do sistema funcionam conforme esperado, garantindo que todas as rotas possíveis no código sejam testadas e que não haja falhas. Em seguida, o teste de tabela de decisão será implementado para analisar cenários de múltiplas condições, assegurando que todas as combinações de entradas resultem nas saídas corretas.

Outro método importante é o teste de transição de estados, que avaliará como o sistema se comporta ao mudar de um estado para outro, verificando a consistência e a estabilidade durante essas transições. Além disso, o teste de valor limite será empregado para identificar erros em limites extremos de entradas, enquanto o teste de partição de equivalência ajudará a reduzir o número de casos de teste ao agrupar entradas que devem ser tratadas de forma semelhante pelo sistema.

O uso dessas técnicas visa não apenas detectar falhas, mas também aprimorar a segurança e a resiliência do código. A implementação desses testes contribuirá significativamente para uma jornada do usuário mais fluida e satisfatória, assegurando que o sistema atenda aos padrões de qualidade esperados e que funcione corretamente em diversos cenários de uso.

Os testes de integração serão conduzidos principalmente no sistema de backend, utilizando uma abordagem de integração *bottom-up*. Para a realização desses testes, optarei por usar o framework Nest.js, que é baseado no Express.js, construído em Node.js, o qual, por sua vez, utiliza JavaScript. Devido à escolha do JavaScript como a linguagem principal no backend, empregarei as bibliotecas mais populares e eficazes para testes nesse ambiente, a fim de garantir a robustez e a confiabilidade do sistema.

A biblioteca Jest será a principal ferramenta para a execução dos testes de integração, devido à sua ampla aceitação na comunidade JavaScript e sua capacidade de lidar com diversos tipos de testes de forma eficiente. Além do Jest, utilizarei o Supertest, que facilita a simulação de requisições HTTP e a verificação das respostas das APIs, o que é essencial para garantir que os serviços do backend estejam funcionando corretamente quando integrados. Para complementar, a biblioteca Mocha será empregada, especialmente para simular respostas de API de maneira simplificada e flexível, tornando o processo de teste mais ágil e eficaz.

O principal objetivo desses testes de integração é melhorar a segurança do sistema, minimizando a quantidade de problemas que possam surgir durante a integração dos módulos e garantindo que o código seja de alta qualidade. Além disso, a implementação desses testes permitirá uma melhor inserção de outras pessoas no projeto, caso seja necessário no futuro, facilitando a manutenção e a escalabilidade do sistema.

Vale salientar que o projeto não constará com testes de aceitação e testes de sistema, uma vez que para a realização de testes de aceitação seria preciso disponibilizar o sistema para o público usar e como o sistema trabalha com dinheiro seria um pouco difícil disponibilizar para um público grande pois isso poderia condicionar em perdas financeiras para o projeto, e não será usado testes de sistema pois eles devido a quantidade de tempo disponível, não seria viável realizar esse tipo de teste

É importante destacar que o projeto não contará com testes de aceitação e testes de sistema. A ausência de testes de aceitação se deve ao fato de que, para sua realização, seria necessário disponibilizar o sistema ao público para uso. Dado que o sistema lida com transações financeiras, isso poderia representar um risco significativo, incluindo possíveis perdas financeiras, caso algum problema fosse identificado apenas após a disponibilização ao público em si mesmo em pequena escala.

Quanto aos testes de sistema, eles foram excluídos devido à limitação de tempo disponível para o desenvolvimento do projeto. Embora esses testes sejam essenciais para a validação completa do sistema como um todo, a sua implementação demandaria um tempo que, neste momento, não é viável, considerando o cronograma e os recursos disponíveis.