Uso de CI/CD, IaC e Testes Automatizados como Base para Garantia de Qualidade no Projeto da Calculadora

• O que é CI/CD e por que importa:

CI/CD é a combinação de Continuous Integration (CI) e Continuous Delivery / Continuous Deployment (CD).
CI envolve integrar mudanças de código frequentemente e automaticamente construir/testar o software a cada commit.
CD garante que, após passar por testes e validações automatizadas, o software esteja sempre pronto para ser liberado, com deploy automático ou com simples aprovação manual.
Em prática, CI/CD transforma um fluxo manual e suscetível a erro em um pipeline automatizado, confiável e repetível.
Para o projeto de calculadora, CI/CD significa que qualquer alteração, seja ajuste no HTML, JS ou CSS, será automaticamente validada: o build, os testes unitários, de integração e e2e rodarão sem intervenção manual. Isso dá confiança de que a aplicação continua funcional antes de "liberar" alterações.

• Benefícios principais

Detecção precoce de bugs e regressões: erros são capturados logo após o commit (antes de "vazar" para produção).
Entrega mais rápida e confiável: mudanças pequenas e frequentes, com deploy constante possível, sem risco de quebrar funcionalidades.
Menor esforço manual e menor chance de erro humano: automatização reduz trabalho repetitivo e o risco de esquecer de rodar testes ou deploy.
Maior disciplina de desenvolvimento: obriga a manter um repositório versionado, com histórico, testes e validações automáticas, que disciplina a própria qualidade do código.

• O papel dos Testes Automatizados (unitários, integração, E2E)

Testes automatizados são peça-chave de uma pipeline CI/CD robusta.
Testes unitários: verificam funções isoladas, ideal para checar lógica de cálculos da sua calculadora. São rápidos e baratos de rodar.
Testes de integração: verificam a interação entre módulos (por exemplo: integração entre lógica de cálculo e UI, ou entre diferentes componentes JS). Isso detecta erros que só surgiriam quando partes diferentes forem combinadas.
Testes E2E (end-to-end): simulam o comportamento real do usuário, no caso da calculadora, algo como "usuário abre página, clica em botões, vê resultado correto", garantindo que tudo funciona "da ponta a ponta".
Quando combinados (padrão da "piramide de testes"), eles fornecem uma cobertura ampla e eficiente: muitos testes rápidos (unitários), alguns de integração, poucos E2E pesados. Isso equilibra velocidade com confiabilidade.

Para o projeto, isso significa que mudanças superficiais (ex: estilo CSS, layout) dificilmente quebrarão a lógica, e mudanças na lógica serão rapidamente detectadas.
Além disso, dentro do CI/CD: testes automatizados integrados ao pipeline garantem que nenhuma mudança chegue ao “main branch” ou produção sem passar pelas verificações. Isso evita regressões acidentais.

• IaC, Quando faz sentido (e limites, dado o escopo)

Infrastructure as Code (IaC) é a prática de definir infraestrutura (servidores, ambientes, configurações) de forma declarativa via código, em vez de configurar manualmente.
Wikipedia
Em aplicações simples como sua calculadora (frontend JS/HTML/CSS), a necessidade de IaC costuma ser menor, você pode nem ter infraestrutura complexa para provisionar.
Porém, se o projeto evoluir para algo mais robusto (backend, deployment, múltiplos ambientes, escalabilidade, containers etc.), IaC garante que ambientes de desenvolvimento, teste e produção permaneçam idênticos e configurados de forma automatizada, reduzindo erros de “ambiente diferente”.
Ou seja: no escopo atual, IaC não traz tanto valor prático imediato, mas é um bom hábito de maturidade técnica para manter em mente, se o projeto escalar.

• Resumo, Por que usar CI/CD + Testes no projeto

Garante que todo commit seja validado automaticamente (build + testes), evitando regressões.
Permite deploys rápidos, frequentes e confiáveis, mesmo em projetos pequenos.
Reduz dependência de testes manuais e erros humanos.
Dá confiança para modificar código, saber que erros serão detectados cedo.
Cria disciplina e estrutura desde o início, facilitando a escalabilidade se o projeto crescer.

Alunos: Cézar, Caio, Bruno Aguiar, Gustavo e Wallace