Relatório Técnico de Verificação e Validação de Software (V&V)

Disciplina: Verificação e Validação de Software

Grupo:

- Gabriel Just
- Renan Fischer
- Henrique Friske
- Felipe Pisoni

1. Introdução e Contexto do Projeto

O presente relatório descreve as estratégias de qualidade de software implementadas no projeto **API Calvão de Cria**. O objetivo principal desta etapa foi elevar a maturidade do projeto através da implementação de uma pirâmide de testes completa e da automação de processos via CI/CD (Integração e Entrega Contínuas).

Diferente de abordagens anteriores onde a validação era manual, este ciclo de desenvolvimento focou na **automatização total**, garantindo que regressões sejam detectadas antes mesmo de o código ser integrado à branch principal (main ou FINAL).

2. Escopo e Estratégia de Testes

Para garantir a cobertura de código e a integridade das regras de negócio, adotamos uma estratégia baseada em três níveis de granularidade, utilizando **Vitest** para testes de baixo nível e **Cypress** para testes de ponta a ponta.

2.1. Testes Unitários (Unit Tests)

Foco: Isolamento da lógica de negócio nos Services.
Tecnologia: Vitest + Mocks (vi.mock).
Nesta camada, o objetivo foi validar exclusivamente a lógica interna dos serviços, sem depender de banco de dados ou conexões externas. Para isso, utilizamos **Mocks** e **Stubs** para simular o comportamento dos Repositórios.

Exemplo Prático (user.service.spec.ts):
Ao testar o método getUserProfile, simulamos (mockResolvedValue) a resposta do userRepository. Isso garante que, se o teste falhar, o erro está na lógica do serviço e não numa falha de conexão com o banco.

> *"A implementação de mocks foi essencial para garantir que os testes unitários sejam rápidos e determinísticos, rodando em milissegundos."*

2.2. Testes de Integração e Top-Down

Foco: Validação de fluxos e comunicação entre componentes (Controller -> Service -> Repository).
Tecnologia: Vitest + Supertest + MongoDB Memory Server.
Aqui validamos se as peças do sistema conversam corretamente. Utilizamos uma abordagem **Top-Down**, enviando requisições HTTP reais para a API que roda em um ambiente controlado com banco de dados em memória. Isso valida rotas, middlewares de autenticação, validação de DTOs e persistência de dados.

2.3. Testes End-to-End (E2E)

Foco: Simulação do usuário final e fluxos críticos.
Tecnologia: Cypress.
Cobrimos os cenários críticos de uso da aplicação, garantindo que o sistema funcione como um todo do ponto de vista do cliente.

- **Fluxos Testados:**
  - Autenticação (Login/Registro).
  - Gestão de Carrinho de Compras.
  - Fluxo de Checkout completo.
  - Administração de Produtos.

3. Pipeline de CI/CD com GitHub Actions

A automação foi configurada utilizando **GitHub Actions**, definida no arquivo .github/workflows/ci-cd.yml. O pipeline é acionado automaticamente em *pushes* e *pull requests* para as branches principais (main, develop, renan, FINAL).

3.1. Estágios do Workflow

O job build-and-test executa os seguintes passos sequenciais em um ambiente Ubuntu:

1. **Checkout & Setup:** Prepara o ambiente e instala o Node.js v20.
2. **Instalação de Dependências:** Executa npm ci para uma instalação limpa e rápida (usando cache).
3. **Linting:** Roda o ESLint para garantir a padronização do código e prevenir erros de sintaxe ou "code smells".
4. **Execução de Testes (Vitest):** Roda a suíte de testes e gera um relatório em formato JSON (test-report.json).
5. **Build Check:** Verifica se o projeto transpila o TypeScript sem erros (npm run build).

3.2. Feedback Visual e Notificações (Atitude)

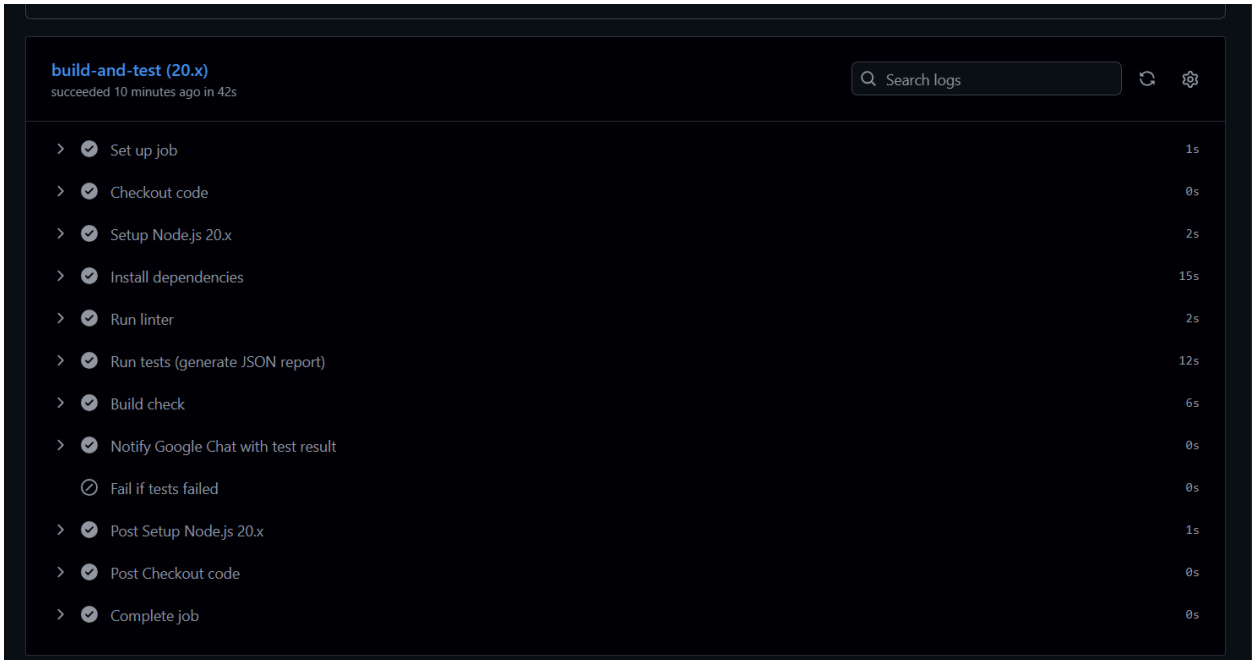
Para garantir que a equipe tenha visibilidade imediata sobre a saúde do projeto, implementamos uma integração com o **Google Chat via Webhook**.

- **Script Customizado:** Criamos o script scripts/google-chat-notify.js que lê o resultado dos testes.
- **Execução Condicional:** O passo de notificação roda com a condição if: always(), garantindo que o time seja avisado tanto em caso de **SUCESSO** quanto de **FALHA**.

4. Evidências de Execução

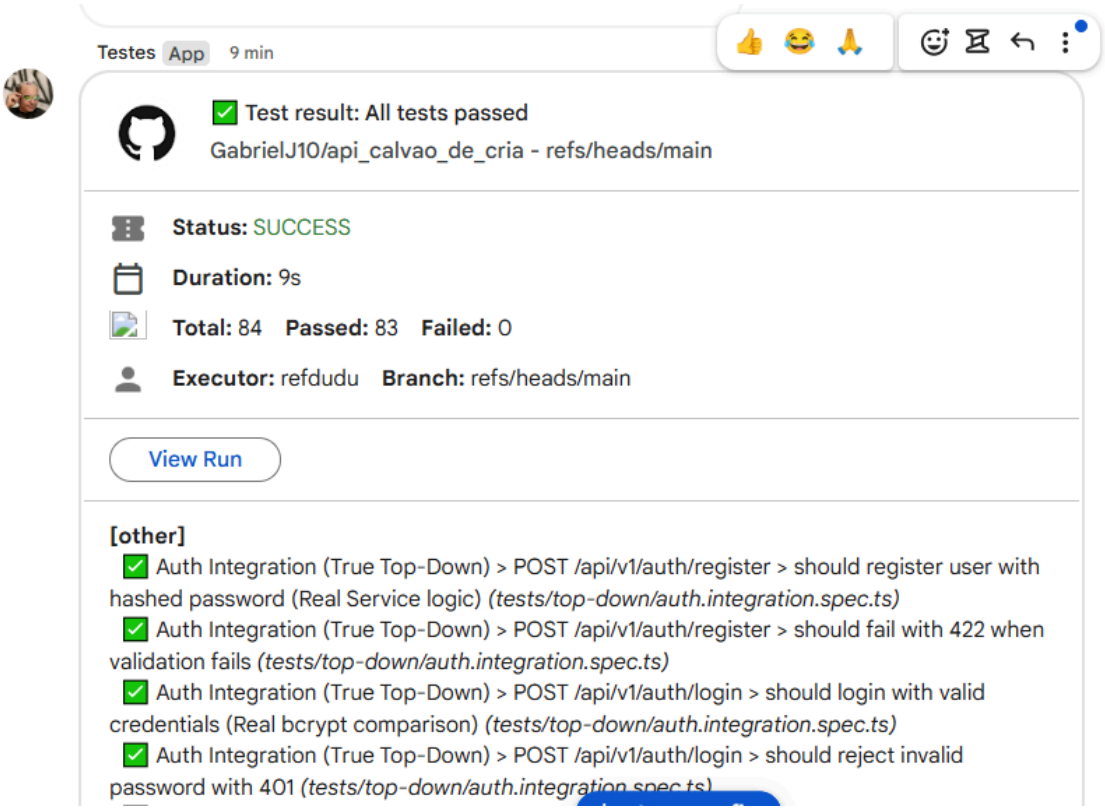
4.1. Execução do Pipeline no GitHub

Abaixo, a evidência de que o workflow executa todas as etapas, desde o lint até a notificação.



4.2. Notificação Automatizada no Google Chat

O bot configurado envia um feedback instantâneo com o status do build, permitindo reação rápida da equipe.



5. Conclusão

A implementação das práticas de V&V transformou o fluxo de desenvolvimento do projeto **API Calvão de Cria**. A combinação de testes unitários isolados com testes E2E robustos criou uma malha de segurança que permite refatorações seguras.

Além disso, a configuração do pipeline de CI/CD eliminou a necessidade de testes manuais repetitivos e centralizou o feedback de qualidade. A atitude de integrar notificações no chat corporativo (Google Chat) demonstrou a preocupação do grupo não apenas com o código, mas com a comunicação e a eficiência do processo de desenvolvimento.