Documentação da Aplicação de Blogging para educação

Back end - Fase 2

Gabriel Alves de Melo | RM: 364289

Igor da Silva Santos | RM: 362093

Kainan Guerra Silva | RM: 361060

Kaue Luiz de Borba | RM: 363421

Leticia Rodrigues Sena | RM: 361203

São Paulo

2025

I. Introdução

O presente projeto tem como objetivo o desenvolvimento do back-end de uma plataforma digital voltada à publicação de atividades pedagógicas por professores, destinadas aos seus respectivos alunos. Esta iniciativa busca facilitar a organização, distribuição e acompanhamento de conteúdos educacionais de forma prática, centralizada e acessível.

Para a construção da aplicação, foi utilizada a linguagem Node.js, conhecida por sua eficiência na criação de servidores escaláveis e sua ampla aceitação no desenvolvimento web. A estrutura de dados do sistema foi implementada com um banco de dados relacional SQL, permitindo maior consistência, segurança e integridade na manipulação das informações, como os registros de usuários, atividades e interações.

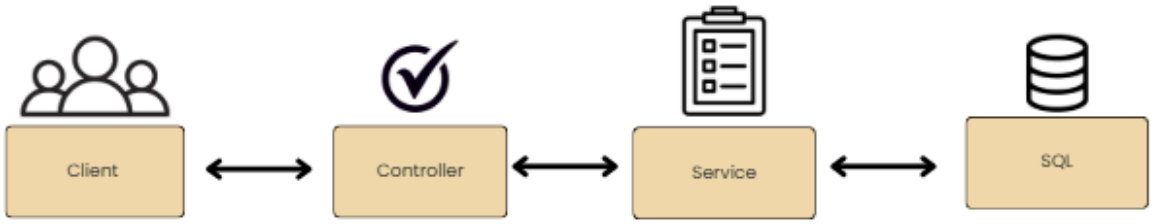
Esse projeto acadêmico visa não apenas o aprofundamento prático em tecnologias amplamente utilizadas no mercado, mas também a proposição de uma solução viável para a rotina educacional, alinhando conceitos técnicos de desenvolvimento com necessidades reais do ambiente escolar.

II. Solução

A solução proposta por este projeto consiste no desenvolvimento de uma API robusta e escalável, responsável por gerenciar toda a lógica de negócios e a comunicação entre o banco de dados e o front-end da plataforma educacional. Por meio dessa API, professores poderão criar, editar, organizar e publicar atividades pedagógicas direcionadas aos seus alunos, enquanto os estudantes terão acesso simplificado a essas atividades, com possibilidade de visualização e acompanhamento.

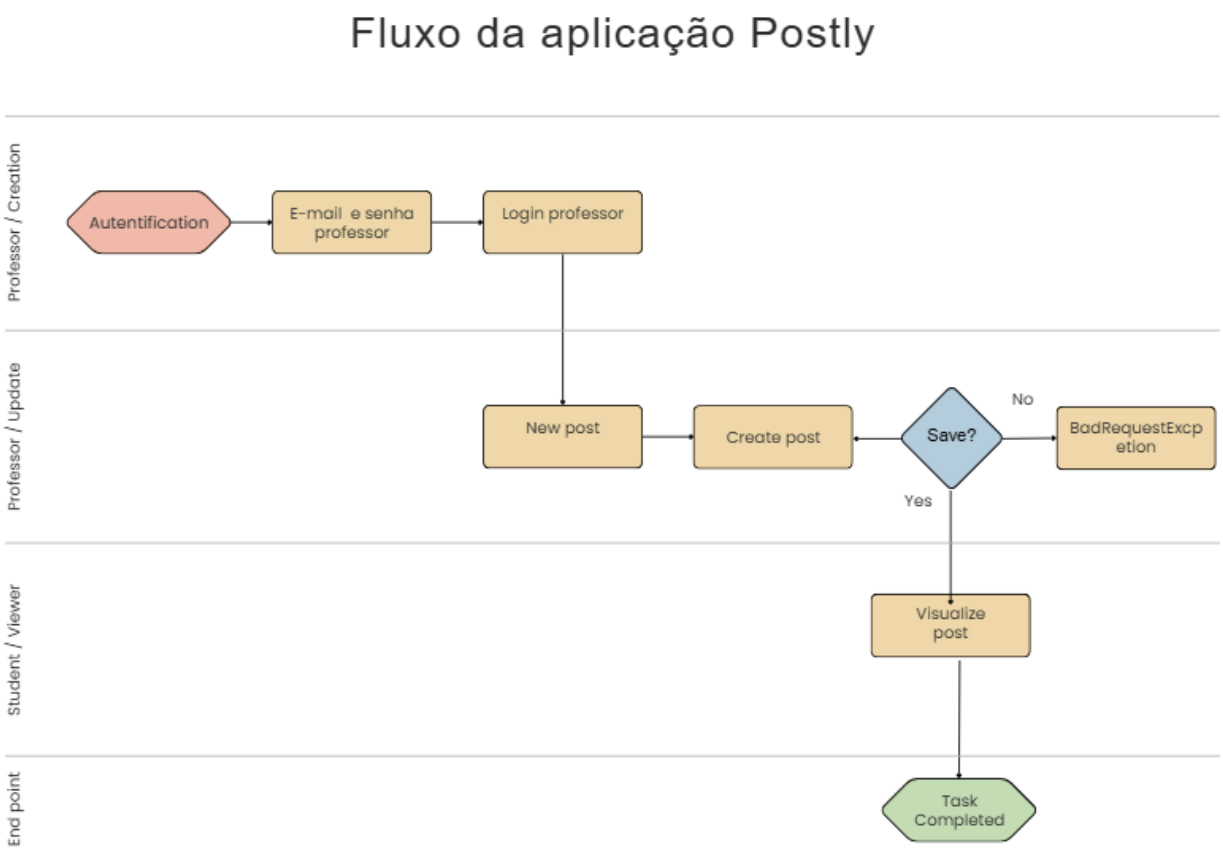
III. Arquitetura de Sistema

A arquitetura do sistema foi feita da seguinte forma usamos o node.js como compilador, Nest.js como framework, Postgres como banco, TypeORM como ORM e Arquitetura hibrida usando Event-Driven e DDD.



IV. Fluxo de funcionalidade

O Fluxo de funcionalidade do projeto está organizada da seguinte maneira:



- **Creation**: Acesso do professor por autenticação para criação de posts.
- **Update**: Alteração e salvamento de posts.
- **Viewer**: Visualização dos posts por alunos.
- **End point**: Finaliza o fluxo da aplicação.

V. Funcionalidades Implementadas

- Cadastro e autenticação de usuários (professores)
- Criação, listagem, edição e exclusão de atividades (CRUD)
- Controle de acesso por tipo de usuário
- Registro de datas e horários de publicações

VI. Autenticação

Para garantir maior segurança e evitar alterações não autorizadas nas postagens, implementamos um sistema robusto de autenticação e controle de acesso. Esse sistema é baseado em **dois tipos de usuário (roles)** — **Professor** e **Estudante** —, com permissões bem definidas:

- **Professores** têm permissão para:
    - Criar novos posts;
    - Atualizar seus próprios posts;
    - Visualizar todos os posts publicados;
    - Deletar seus próprios posts.
- **Estudantes** podem apenas:
    - Visualizar todos os posts publicados.

Essa abordagem protege a integridade dos conteúdos, assegura que apenas usuários autorizados possam realizar alterações e promove um ambiente confiável e organizado para todos os envolvidos na plataforma.

Usuários de autenticação:

Login	Senha	Role
professor@fiap.com	Fiap@2025	Professor
professor2@fiap.com	Fiap@2025	Professor
estudante@fiap.com	Fiap@2025	Student

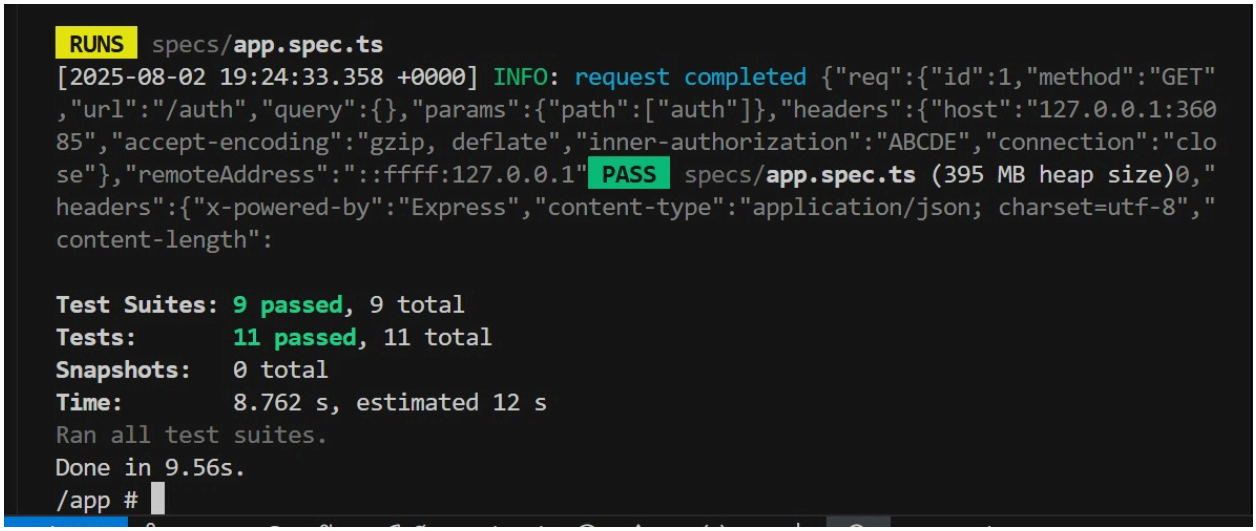
VII. Testes e Uso da Aplicação

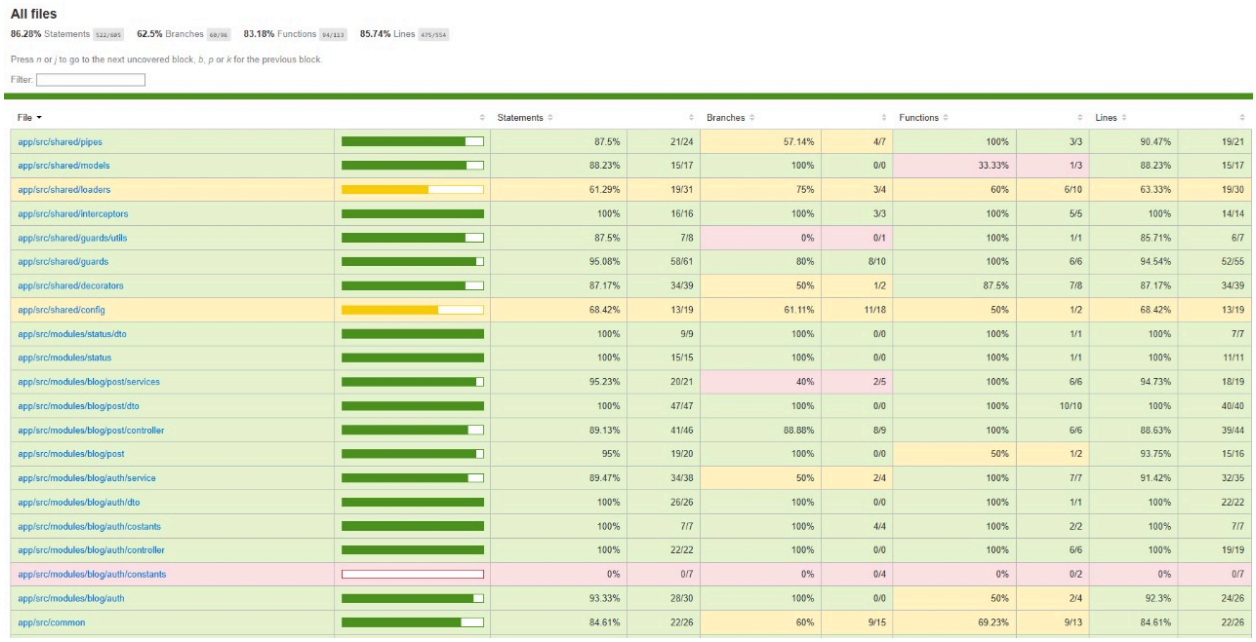
Os testes automatizados deste projeto foram desenvolvidos com Jest, com foco em testes de feature. Eles validam funcionalidades completas da aplicação, garantindo que os principais fluxos de uso funcionem corretamente.

Cada teste cobre a integração entre diferentes camadas do sistema, simulando cenários reais de uso. O objetivo é detectar regressões e aumentar a confiabilidade da aplicação sem depender de testes manuais.

Funcionalidades críticas, como autenticação e operações de negócio, foram priorizadas nos cenários de teste. Este conjunto de testes contribui diretamente para a qualidade e estabilidade das entregas.

As imagens abaixo representam os testes realizados





VIII. Vídeo

Video Tech Challenge Fase 2 Fiap

IX CI/CD - Integração e Entrega Contínuas

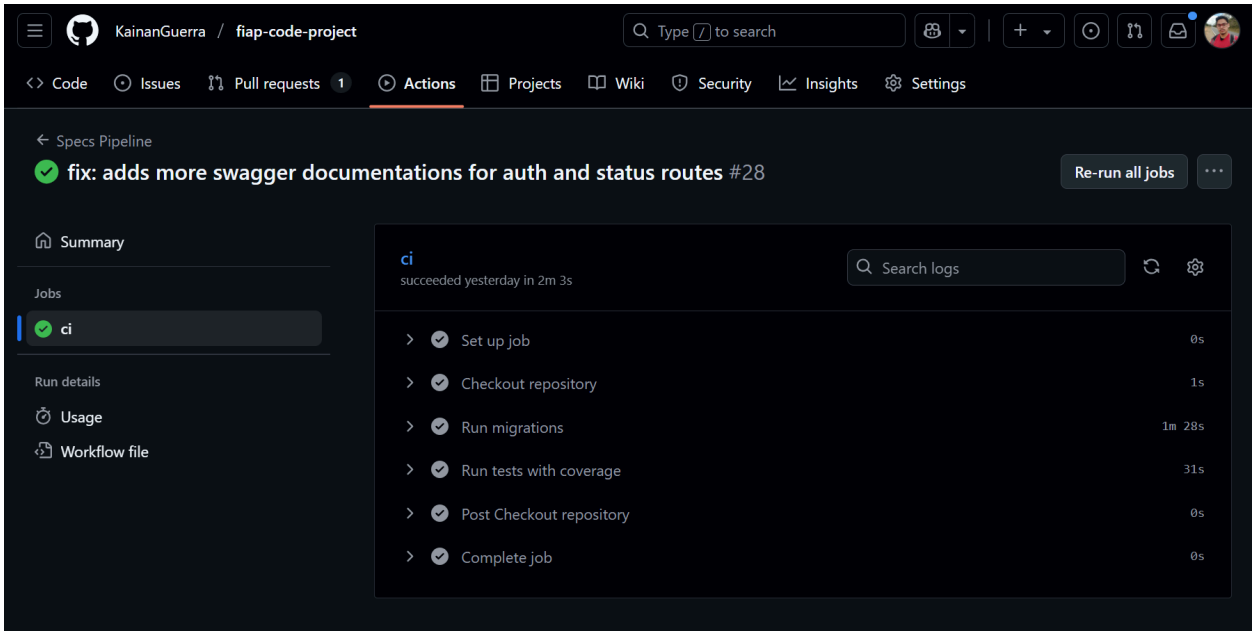
A automação dos processos de integração e entrega contínuas (CI/CD) garante mais segurança, qualidade e agilidade no desenvolvimento do projeto.

CI – Integração Contínua

No fluxo de **CI**, toda nova *feature branch* enviada ao repositório dispara automaticamente uma pipeline no GitHub Actions que:

- Instala as dependências do projeto;
- Roda todos os testes automatizados;
- Garante que o código esteja funcionando corretamente antes de ser integrado à branch principal (`main`).

Este processo reduz o risco de quebrar funcionalidades existentes e promove um desenvolvimento mais confiável.



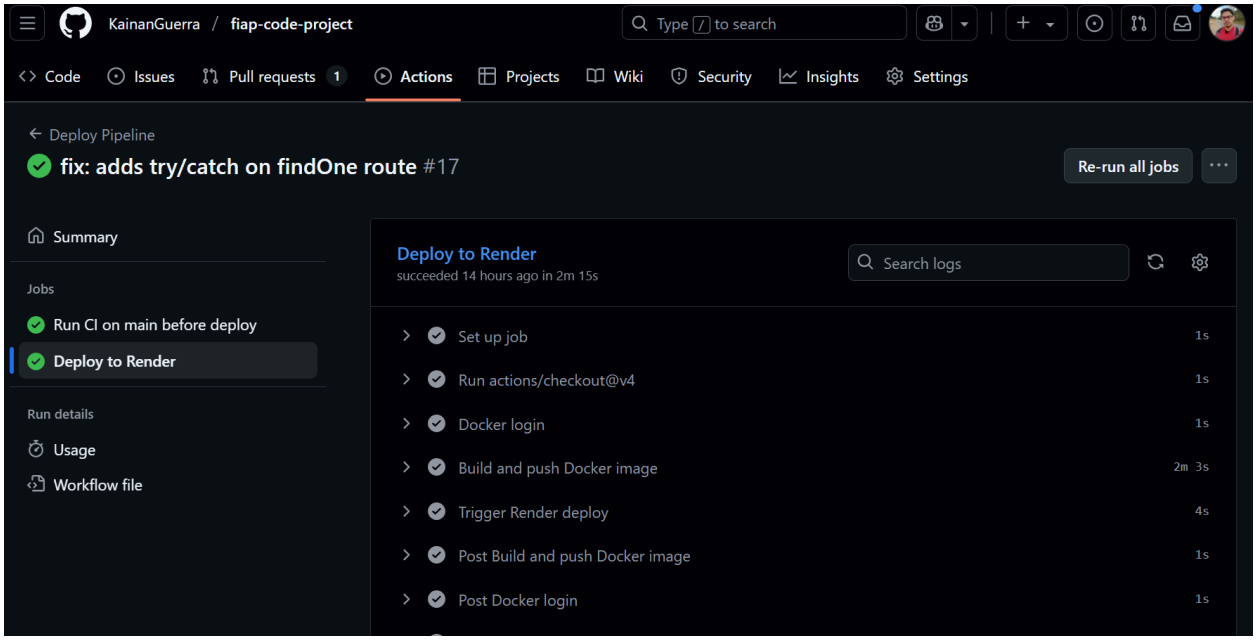
(Imagem: fluxo de CI no GitHub)

CD – Entrega Contínua

O processo de **CD** entra em ação após qualquer alteração na branch `main`. Ele:

- Executa novamente os testes para validar a estabilidade do projeto;
- Troca a imagem Docker no DockerHub;
- Realiza o deploy automático da aplicação para o ambiente de produção na plataforma Render.

Isso garante que a versão mais recente e estável do sistema esteja sempre disponível em produção, sem necessidade de deploys manuais.



(Imagem: fluxo de CD no GitHub)

Todo o processo de CI/CD foi implementado utilizando GitHub Actions, uma ferramenta nativa do GitHub que permite automatizar fluxos de trabalho diretamente no repositório.
Os workflows definidos em arquivos .yml dentro da pasta .github/workflows controlam quando e como os testes são executados e o deploy é realizado. Essa abordagem reduz falhas humanas, agiliza o ciclo de desenvolvimento e garante maior confiabilidade nas entregas.

X Rota de status

Esta rota foi criada para monitoramento da saúde da aplicação. Quando acessada, retorna uma resposta simples indicando se o servidor está ativo e operando corretamente. É útil para verificações de disponibilidade (health checks) em ambientes de produção.

https://fiap-code-project.onrender.com/fiap/status

XI Swagger

A documentação da API está disponível via Swagger, permitindo explorar todas as rotas disponíveis no backend. Nela, é possível visualizar os endpoints, métodos HTTP, parâmetros esperados, respostas possíveis e até testar chamadas diretamente pela interface. Ideal para desenvolvedores que queiram integrar com a API de forma rápida e segura.

https://fiap-code-project.onrender.com/api

XII Repositório

O código-fonte do projeto está disponível publicamente no GitHub. O repositório contém a implementação completa do backend em NestJS, incluindo autenticação, controle de posts, permissões por tipo de usuário, testes automatizados e configuração via Docker. Também inclui instruções de uso e contribuição.

https://github.com/KainanGuerra/fiap-code-project

XIII. Desafios da Equipe

Durante o desenvolvimento da aplicação, a equipe enfrentou diversos desafios técnicos e conceituais:

Modelagem correta do banco de dados: definir as relações entre tabelas como usuários, atividades e turmas exigiu revisões e ajustes constantes.

Controle de acesso: implementar diferentes permissões para professores e alunos foi uma tarefa que demandou estudo e testes com middlewares.

Tratamento de erros e validações: garantir a estabilidade da API mesmo em casos de uso inesperados foi um aprendizado importante.

Organização do código: manter a separação de responsabilidades entre controllers, services e models aumentou a complexidade inicial, mas resultou em um código mais limpo e sustentável.

XIV. Lições Aprendidas

Ao longo do projeto, a equipe teve importantes aprendizados tanto técnicos quanto relacionados ao trabalho em equipe e à resolução de problemas:

Importância da arquitetura: uma boa estrutura inicial de código facilita a escalabilidade e manutenção do projeto.

Revisão de requisitos: entender bem as necessidades do sistema antes de começar o desenvolvimento evita retrabalho.

Versionamento de código: o uso do Git foi essencial para organização e controle do progresso.

Trabalho colaborativo: mesmo em um projeto de back-end, a colaboração entre integrantes foi essencial para identificar problemas e validar soluções.

XV. Considerações Finais

Este projeto representou uma oportunidade significativa de aplicar conhecimentos de desenvolvimento web, banco de dados relacional e boas práticas de programação. A construção do back-end da plataforma educacional mostrou-se desafiadora, mas trouxe resultados satisfatórios em termos de aprendizado técnico e pessoal. Futuramente, o sistema poderá ser expandido com a implementação do front-end, integração com notificações, relatórios e funcionalidades mais avançadas.

XVI. Referências

- Node.js Documentation: https://nodejs.org/en/docs
- Express.js Guide: https://expressjs.com/
- PostgreSQL Docs: https://www.postgresql.org/docs/
- Sequelize ORM: https://sequelize.org/
- Clean Architecture para APIs RESTful