

# Relatório Técnico: Projeto GraphLab

Eduardo Versiani de Melo Penna

Lucas Fernandes Nascimento

2025

## Resumo

O presente relatório descreve o desenvolvimento e a estrutura da aplicação GraphLab, uma ferramenta web interativa voltada para o ensino e simulação de algoritmos em grafos. O projeto tem como objetivo facilitar a compreensão de conceitos fundamentais da Teoria dos Grafos, permitindo a visualização dinâmica de algoritmos como BFS, DFS e Dijkstra. A aplicação foi construída utilizando uma arquitetura moderna, separando a interface visual (React/Next.js) da lógica de processamento (Node.js/Express). Este documento detalha as decisões de design, a modelagem de dados adotada para garantir a animação dos algoritmos e as tecnologias empregadas.

**Palavras-chaves:** Grafos. Algoritmos. Educação. React. Typescript.

## Introdução

O estudo de algoritmos em grafos é fundamental para a Ciência da Computação, oferecendo soluções para problemas complexos de roteamento, redes e otimização. No entanto, a visualização abstrata dessas estruturas pode apresentar desafios para estudantes iniciantes.

O objetivo principal do projeto \*\*GraphLab\*\* é mitigar essa dificuldade fornecendo uma aplicação web interativa que apoie o aprendizado prático. A ferramenta visa permitir que estudantes e usuários criem, visualizem, editem e executem algoritmos clássicos de grafos de forma intuitiva ([NASCIMENTO; PENNA, 2025](#)). A aplicação foca em demonstrar o funcionamento passo-a-passo de algoritmos como Busca em Largura (BFS), Busca em Profundidade (DFS) e Dijkstra, facilitando a compreensão das estruturas de dados subjacentes e do comportamento de visitação dos nós ([CORMEN et al., 2012](#)).

Este relatório técnico detalha o desenvolvimento da aplicação, cobrindo desde a escolha da stack tecnológica até a implementação dos algoritmos e modelagem de dados.

# 1 Desenvolvimento

O GraphLab é uma aplicação *fullstack* desenvolvida para a simulação de algoritmos. O desenvolvimento foi pautado em requisitos funcionais que priorizam a interatividade, como a criação de grafos via *drag and drop* e a animação fluída das execuções (NASCIMENTO; PENNA, 2025).

## 1.1 Arquitetura e Stack Tecnológica

O projeto adota uma arquitetura monolítica dividida em dois diretórios principais, facilitando o gerenciamento do código fonte:

- **Frontend:** Localizado no diretório `front`, foi desenvolvido com **Next.js**, **React** e **TypeScript**. A biblioteca **React Flow** (Webkid, 2024) é utilizada como motor gráfico para renderização dos nós e arestas, enquanto a estilização utiliza Tailwind CSS e componentes shadcn/ui. O gerenciamento de estado é realizado através de Hooks customizados.
- **Backend:** Localizado no diretório `back`, foi implementado em **Node.js** com **Express**. É responsável por receber a representação JSON do grafo, executar os algoritmos logicamente e retornar os resultados processados para a animação.

## 1.2 Modelagem de Dados

Um dos desafios centrais do desenvolvimento foi a conversão entre a representação visual (necessária para o navegador) e a representação lógica (necessária para os algoritmos). A estrutura de dados foi documentada para garantir a consistência entre as camadas (TEAM, 2025).

- **Modelo Visual:** O frontend armazena coordenadas  $(x, y)$ , rótulos e IDs únicos. As arestas contêm metadados visuais e o peso é armazenado dentro de um objeto `data`.
- **Modelo Lógico:** Antes de enviar dados ao backend, uma função de conversão transforma o estado visual em um objeto contendo listas puras de nós e arestas, abstraindo posições de tela.
- **Estrutura Interna:** O servidor converte o JSON recebido em uma **Lista de Adjacência**, estrutura escolhida pela eficiência na iteração de vizinhos durante as buscas.

## 1.3 Implementação dos Algoritmos

Os algoritmos foram implementados como classes estáticas no backend, garantindo isolamento e testabilidade. Todos retornam um objeto padronizado contendo o status da execução, os passos (**steps**) para animação e o resultado final (TEAM, 2025).

### 1.3.1 Busca em Profundidade (DFS)

A implementação da DFS utiliza recursão e um conjunto (**Set**) para controle de nós visitados. O algoritmo registra a “fronteira” atual e o nó corrente em cada passo recursivo, permitindo que o frontend anime a pilha de execução visualmente.

### 1.3.2 Busca em Largura (BFS)

Implementada com uma fila (*queue*), a BFS explora o grafo nível por nível. O algoritmo armazena o caminho percorrido utilizando um mapa de predecessores, o que permite a reconstrução do caminho mínimo em grafos não ponderados ao final da execução.

### 1.3.3 Algoritmo de Dijkstra

Utilizado para encontrar caminhos mínimos em grafos ponderados. A implementação inclui validações robustas:

- **Validação:** Impede a execução caso existam pesos negativos, o que violaria as premissas do algoritmo.
- **Execução:** Utiliza uma lógica de fila de prioridade para selecionar o nó com menor distância acumulada a cada iteração. O relaxamento das arestas atualiza as distâncias e os predecessores (CORMEN et al., 2012).

## 1.4 Execução e Visualização

A comunicação entre as camadas ocorre via API REST. Ao receber a resposta, o frontend processa o array de passos (`steps`), alterando as cores dos nós e arestas sequencialmente para criar a animação solicitada nos requisitos. O ambiente de desenvolvimento é executado localmente via `npm run dev` em ambos os diretórios.

## Considerações finais

O projeto GraphLab cumpre seu objetivo de servir como uma ferramenta de apoio pedagógico para a disciplina de Algoritmos em Grafos. A separação clara entre as responsabilidades de visualização (Frontend/React Flow) e lógica de processamento (Backend/Node.js) permitiu a criação de uma interface responsiva e moderna, sem sacrificar a precisão dos algoritmos.

A implementação bem-sucedida da animação dos algoritmos BFS, DFS e Dijkstra oferece aos usuários uma visão detalhada do comportamento interno dessas estruturas, validando os requisitos funcionais e não funcionais estabelecidos inicialmente.

O repositório pode ser acessado em: <<https://github.com/lucsfn/graphlab>>. Já o vídeo de demonstração encontra-se disponível em: <<https://youtu.be/2cIIUkdsYIg>>.

## Referências

CORMEN, T. H. et al. *Algoritmos: Teoria e Prática*. 3. ed. Rio de Janeiro: Campus, 2012. Citado 2 vezes nas páginas [1](#) e [3](#).

NASCIMENTO, L. F.; PENNA, E. V. d. M. *GraphLab: Documentação do Repositório*. [S.l.], 2025. Disponível no arquivo README.md e requirements.md do projeto. Citado 2 vezes nas páginas [1](#) e [2](#).

TEAM, G. *Especificação de Estrutura de Dados do GraphLab.* [S.l.], 2025. Arquivo data\_structure.md. Citado na página 2.

Webkid. *React Flow Documentation.* 2024. Acesso em: 2025. Disponível em: <<https://reactflow.dev/>>. Citado na página 2.