# **Decisões Técnicas do Projeto DoeCamp**

O desenvolvimento do projeto **DoeCamp** seguiu uma abordagem organizada e voltada para boas práticas de engenharia de software. Abaixo, estão as decisões técnicas adotadas:

## **Front-end**

### **Prioridade no Design e Usabilidade**

A implementação começou pelo front-end, com foco em proporcionar uma interface visual agradável e responsiva para os usuários. O objetivo inicial foi garantir que todas as funcionalidades visuais estivessem prontas e funcionais.

### **Componentização**

Foi adotada a estratégia de componentização utilizando **React**, o que facilitou a reutilização de código e a manutenção de diferentes partes da interface. Cada componente foi isolado em pequenas partes modulares, o que torna o projeto mais fácil de expandir e escalar.

### **Rotas e Navegação**

As rotas foram gerenciadas de forma simples utilizando o sistema de rotas do **Next.js**, que facilita a navegação entre as diferentes páginas da aplicação.

## **Back-end (API)**

### **Arquitetura REST**

No back-end, foi adotada a arquitetura REST para organizar os endpoints da API, garantindo uma interface consistente e fácil de entender. Os endpoints foram projetados para gerenciar campanhas e doações.

### **Mongoose e MongoDB Atlas**

O banco de dados **MongoDB Atlas** foi utilizado em conjunto com o **Mongoose**, o que proporcionou um gerenciamento eficiente dos dados no banco NoSQL. Mongoose foi fundamental na modelagem dos dados e nas operações de CRUD.

## **Organização do Projeto**

### **Kanban e GitFlow**

Para manter a organização e rastreabilidade do progresso, foi utilizado o **quadro Kanban** do GitHub. Esse método facilitou o planejamento das tarefas e a visualização do avanço do projeto.

Além disso, o método **GitFlow** foi empregado para o controle das versões, garantindo que o desenvolvimento fosse feito de forma organizada na branch de desenvolvimento (develop), com commits descritivos e ramificações adequadas para novas features e correções.

### **Boas Práticas de Programação**

O código foi desenvolvido seguindo boas práticas, como o uso de variáveis autodescritivas e commits padronizados utilizando convenções como feat, style, e refactor.