# **ADR-001 - Stack de Tecnologias para o Front-end**

* **Status:** Aceito
* **Data:** 2025-05-19
* **Autor:** Carlos H. P. Jacinto

## **Contexto**

Este projeto requer a construção da interface de uma **aplicação bancária** voltada ao **cadastro e gerenciamento de contas e transações financeiras**. A interface precisa ser **responsiva, performática e de fácil manutenção**, com uma experiência de usuário fluida e comunicação eficiente com o back-end via API REST.

A stack de front-end deve ser escolhida com base em critérios como:

* Produtividade no desenvolvimento
* Suporte à escalabilidade e modularização
* Comunidade ativa e documentação sólida
* Integração com TypeScript e ferramentas modernas

## **Decisão**

A stack de tecnologias adotada para o front-end é a seguinte:

* **Next.js (App Router + TypeScript)**  
   Framework React-based moderno que oferece **renderização híbrida** (SSR, SSG e ISR), roteamento baseado em arquivos e arquitetura modular com suporte nativo a TypeScript. Ideal para aplicações escaláveis e com boa performance.
* **Tailwind CSS**  
   Framework utilitário de CSS que facilita a criação de layouts responsivos e customizáveis, com menor dependência de arquivos CSS externos.
* **Axios**  
   Cliente HTTP que simplifica requisições com interceptadores, tratamento de erros e suporte melhorado a promessas, em comparação com fetch.
* **React Hooks (useState, useEffect, etc.)**  
   Utilizados para controle de estado e efeitos colaterais dentro dos componentes funcionais.

## **Consequências**

**Vantagens:**

* Acelera o desenvolvimento com **componentes reutilizáveis** e estilização utilitária eficiente.
* Interfaces **responsivas e com excelente performance** desde o início.
* Estrutura modular com **ótimo suporte a escalabilidade e organização de código**.
* Ferramentas com **comunidade ativa**, excelente documentação e amplo suporte em ecossistemas modernos.
* Integração nativa com **TypeScript**, garantindo segurança de tipos e menor propensão a erros.

**Desvantagens:**

* O uso indevido do Tailwind CSS pode gerar arquivos grandes
* A **curva de aprendizado** pode ser maior para desenvolvedores não familiarizados.
* A **estrutura de roteamento** baseada em pastas do App Router requer organização cuidadosa para evitar complexidade excessiva em aplicações maiores.

## **Alternativas Consideradas**

* **Vue.js + Vuetify**  
   Rejeitado por menor familiaridade e ecossistema menos alinhado com a stack já adotada no back-end (NestJS com TypeScript).
* **React com React Router e CRA (Create React App)**  
   Abandonado em favor do **Next.js**, que oferece mais recursos nativos como SSR e roteamento automático.
* **Fetch API nativa**  
   Rejeitada devido à menor ergonomia e ausência de funcionalidades como interceptadores e tratamento automático de erros/respostas.

## **Referências**

* [Next.Js](https://nextjs.org)
* [Tailwind CSS](https://tailwindcss.com/docs)
* [Axios](https://axios-http.com/docs/intro)
* [React Hooks](https://pt-br.legacy.reactjs.org/docs/hooks-intro.html)