Coingrid

Principais preços e gráficos de criptomoedas, listados por capitalização de mercado.
Acesso gratuito a dados atuais e históricos do Bitcoin e de milhares de altcoins.

Objetivos

Solução

Criar uma solução completa para busca, histórico e visualização de dados de criptomoedas.

Camadas

Backend em .NET 9
Web App em React (Vite)
Mobile App em React Native (Expo)

Arquitetura

Utilizar uma arquitetura simples, porém moderna, focada em extensibilidade.

Grupo 11

Camilla Manzini

Giovanni Shintaku

André Fernandes

Princípios Técnicos que Guiaram o Desenvolvimento

01 **Simplicidade com Escalabilidade**

MongoDB permite expandir coleções facilmente, enquanto a API foi construída de forma modular e simples de estender. Além disso, o frontend foi planejado para receber novos componentes e páginas sem causar quebras, garantindo evolução contínua do sistema.

02 **Reutilização de Lógica entre Web e Mobile**

Hooks, serviços e chamadas HTTP compartilham a mesma estrutura mental, o que facilita a manutenção e reduz duplicação. Essa reutilização consistente torna o desenvolvimento mais fluido entre diferentes partes do projeto.

03 **Integração Limpa com MongoDB**

As coleções foram modeladas para suportar histórico sem necessidade de retrabalho, permitindo consultas rápidas e diretas. Dessa forma, evita-se a complexidade de joins ou estruturas pesadas, mantendo o desempenho mesmo com grande volume de dados.

04 **Responsividade e UX como prioridades**

Web e mobile compartilham a mesma identidade visual, oferecendo uma experiência alinhada entre plataformas. A interface prioriza busca rápida, leitura clara e telas minimalistas, favorecendo uma navegação simples e eficiente.

Avaliação dos Frameworks e Tecnologias Utilizadas

.NET 9	MongoDB	SmarterASP .NET	React + Vite	React Native + Expo
Minimal APIs, alta performance e arquitetura simples de manter.	Flexível para dados dinâmicos e histórico, consultas rápidas sem joins.	Publicação rápida, testes online e baixa complexidade.	Desenvolvimento veloz, build otimizado e interface responsiva.	Hot reload eficiente, testes fáceis e experiência consistente.

Propostas de Melhoria

01 — **PRs e Issues** — Não trabalhamos com issues e tivemos problemas de colaboração que poderiam ser evitados com o barramento de commits diretos.

02 — **RFs** — Maior detalhamento de requisitos antes da implementação.

03 — **RFs** — Requisitos funcionais essenciais assumidos por membros engajados desde o início.

04 — **Deploy API** — Deploy da API logo na entrega do backend para melhor desenvolvimento do front. Fazer também deploy dos workers em vez de apenas da API.