Guia de Construção e Implantação: Criando o "Qual L2 Você É?"

Guia elaborado por xAI para andre e messias

May 16, 2025

Abstract

Este guia oferece um passo a passo detalhado e didático para construir e implantar um projeto como o "Qual L2 Você É?" um Mini App que ajuda usuários a descobrir qual blockchain Layer 2 (Base, Arbitrum, Optimism ou zkSync) combina com sua personalidade por meio de um quiz interativo. O projeto inclui salvamento de resultados na blockchain Base e compartilhamento no Warpcast. Você aprenderá desde a preparação do ambiente até a implantação no Vercel e integração com o Warpcast, permitindo que qualquer pessoa replique e publique o app.

1 Introdução

O "Qual L2 Você É?" é um Mini App educativo e interativo desenvolvido para o Base Batch LatAm buildathon. Este guia ensina como criar um projeto semelhante do zero, utilizando Next.js, MiniKit e Warpcast, e implantá-lo no Vercel com funcionalidades onchain na Base. Ao final, você terá um app funcional e poderá compartilhá-lo no Warpcast, assim como foi feito originalmente.

2 Preparação do Ambiente

2.1 Requisitos

Antes de começar, você precisa instalar algumas ferramentas no seu computador:

- **Node.js e npm**: Necessários para rodar o projeto. Baixe a versão mais recente em https://nodejs.org.
- **Git**: Para versionamento e clonagem do repositório. Baixe em https://git-scm.com.
- **VS Code**: Um editor de código (opcional, mas recomendado). Baixe em https://code.visualstudio.com.
- **Vercel CLI**: Para implantação. Instale globalmente após o Node.js.
- **MetaMask**: Uma carteira para testes onchain (instale como extensão no navegador).

2.2 Instalando Node.js e npm

1. Acesse https://nodejs.org e baixe a versão LTS (Long Term Support).
2. Execute o instalador e siga as instruções.
3. Verifique a instalação abrindo o terminal (Prompt de Comando no Windows ou Terminal no Mac/Linux) e digitando:

```
node --version
npm --version
```

Você deve ver algo como `v20.12.2` para o Node.js e `10.5.0` para o npm.

2.3 Instalando Git

1. Acesse https://git-scm.com e baixe o instalador para o seu sistema operacional.
2. Siga as instruções de instalação (deixe as opções padrão, a menos que saiba o que está alterando).
3. Verifique a instalação no terminal:

```
git --version
```

Você deve ver algo como `git version 2.44.0`.

2.4 Instalando Vercel CLI

1. No terminal, digite:

```
npm install -g vercel
```

2. Verifique a instalação:

```
vercel --version
```

Você deve ver algo como `Vercel CLI 32.5.3`.

3 Criando o Projeto

3.1 Inicializando um Novo Projeto com Next.js

1. No terminal, crie um diretório para o projeto e entre nele:

```
mkdir l2match
cd l2match
```

2. Inicialize um projeto Next.js com TypeScript:

```
npx create-next-app@latest . --ts
```

Escolha as opções padrão (pressione Enter para todas as perguntas, exceto se preferir Tailwind CSS, que usaremos).

3. Após a instalação, abra o projeto no VS Code:

```
code .
```

3.2 Instalando Dependências

1. Instale Tailwind CSS e outras dependências:

```
npm install tailwindcss postcss autoprefixer @coinbase/
   onchainkit wagmi viem @upstash/redis
```

2. Configure o Tailwind CSS:

```
npx tailwindcss init -p
```

3. Atualize o arquivo `tailwind.config.ts` com:

```
module.exports = {
  content: [
    "./pages/**/*.{js,ts,jsx,tsx}",
    "./components/**/*.{js,ts,jsx,tsx}",
    "./app/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

4. Adicione as diretivas do Tailwind ao arquivo `app/globals.css`:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

3.3 Configurando Variáveis de Ambiente

1. Crie um arquivo `.env.local` na raiz do projeto com:

```
NEXT_PUBLIC_ALCHEMY_API_KEY=sua_chave_alchemy
NEXT_PUBLIC_WARPCAST_API_KEY=sua_chave_warpcast
```

2. Para obter as chaves:

   - **Alchemy**: Crie uma conta em https://www.alchemy.com, crie um app na Base Sepolia testnet e copie a chave API.
   - **Warpcast**: Registre-se em https://warpcast.com, acesse a API de desenvolvedor (se disponível) e obtenha a chave.

3. Adicione `.env.local` ao `.gitignore`:

```
.env.local
```

4 Desenvolvendo o Quiz

4.1 Estrutura do Projeto

1. Crie uma pasta `components` dentro de `app` (`app/components`).
2. Crie uma pasta `pages` dentro de `app` (`app/pages`), caso não exista.
3. Crie uma pasta `utils` na raiz para funções utilitárias.

4.2 Criando o Quiz

1. Crie um arquivo `app/pages/index.tsx` para a página inicial do quiz:

```
import { useState } from 'react';
import { useRouter } from 'next/router';

export default function Home() {
  const [answers, setAnswers] = useState<number[]>([]);
  const router = useRouter();

  const questions = [
    "Você prefere velocidade ou segurança?",
    "Você é mais otimista ou realista?",
    "Gosta de explorar novas tecnologias?",
    "Prefere soluções simples ou complexas?",
    "Você é mais comunitário ou independente?",
  ];

  const handleAnswer = (questionIndex: number, answer: number
     ) => {
    const newAnswers = [...answers];
    newAnswers[questionIndex] = answer;
    setAnswers(newAnswers);

    if (questionIndex + 1 < questions.length) {
      // Próxima pergunta
    } else {
      // Calcula o resultado
      const scores = { base: 0, arbitrum: 0, optimism: 0,
         zksync: 0 };
      newAnswers.forEach((answer, i) => {
        if (i === 0) scores.base += answer;
        if (i === 1) scores.optimism += answer;
        if (i === 2) scores.arbitrum += answer;
        if (i === 3) scores.zksync += answer;
        if (i === 4) scores.base += answer;
```

```
      });

      const result = Object.keys(scores).reduce((a, b) =>
        scores[a] > scores[b] ? a : b
      );
      router.push(`/result?score=${result}`);
    }
  };

  return (
    <div className="min-h-screen flex items-center justify-
       center bg-gray-100">
      <div className="bg-white p-6 rounded-lg shadow-lg">
        <h1 className="text-2xl font-bold mb-4">Qual L2 Você
           É?</h1>
        {questions.map((question, index) => (
          <div key={index} className={answers[index] !==
             undefined ? 'hidden' : ''}>
            <p className="text-lg mb-2">{question}</p>
            <button
              onClick={() => handleAnswer(index, 1)}
              className="bg-blue-500 text-white px-4 py-2
                 rounded mr-2"
            >
              Sim
            </button>
            <button
              onClick={() => handleAnswer(index, 0)}
              className="bg-gray-500 text-white px-4 py-2
                 rounded"
            >
              Não
            </button>
          </div>
        ))}
      </div>
    </div>
  );
}
```

2. Crie um arquivo `app/pages/result.tsx` para a página de resultados:

```
import { useRouter } from 'next/router';

export default function Result() {
  const router = useRouter();
  const { score } = router.query;

  return (
    <div className="min-h-screen flex items-center justify-
       center bg-gray-100">
      <div className="bg-white p-6 rounded-lg shadow-lg">
```

```
        <h1 className="text-2xl font-bold mb-4">Seu Resultado
           </h1>
        <p className="text-lg mb-4">Você é: {score}</p>
        <button
          className="bg-purple-500 text-white px-4 py-2
             rounded"
          onClick={() => alert('Funcionalidade de
             compartilhamento no Warpcast será adicionada!')}
        >
          Compartilhar Resultado
        </button>
      </div>
    </div>
  );
}
```

5 Adicionando Salvamento Onchain com MiniKit

5.1 Configurando MiniKit

1. Crie um arquivo `utils/wallet.ts` para configurar o MiniKit e Wagmi:

```
import { createWalletClient, custom } from 'viem';
import { base } from 'viem/chains';
import { MiniKit } from '@coinbase/onchainkit';

const walletClient = createWalletClient({
  chain: base,
  transport: custom(window.ethereum),
});

export const connectWallet = async () => {
  await MiniKit.connect();
  return walletClient;
};
```

2. Atualize `app/pages/result.tsx` para incluir salvamento onchain:

```
import { useState } from 'react';
import { useRouter } from 'next/router';
import { connectWallet } from '../utils/wallet';
import { writeContract } from 'wagmi/actions';

export default function Result() {
  const router = useRouter();
  const { score } = router.query;
  const [isSaving, setIsSaving] = useState(false);

  const saveResultOnchain = async () => {
    setIsSaving(true);
```

```
    try {
      const walletClient = await connectWallet();
      // Exemplo de contrato (você precisa criar um contrato na Base)
      const tx = await writeContract({
        address: '0xSeuContratoAqui',
        abi: [
          {
            name: 'saveResult',
            type: 'function',
            inputs: [{ name: 'result', type: 'string' }],
            outputs: [],
          },
        ],
        functionName: 'saveResult',
        args: [score],
      });
      alert('Resultado salvo na blockchain!');
    } catch (error) {
      console.error(error);
      alert('Erro ao salvar na blockchain.');
    } finally {
      setIsSaving(false);
    }
  };

  return (
    <div className="min-h-screen flex items-center justify-center bg-gray-100">
      <div className="bg-white p-6 rounded-lg shadow-lg">
        <h1 className="text-2xl font-bold mb-4">Seu Resultado</h1>
        <p className="text-lg mb-4">Você é: {score}</p>
        <button
          onClick={saveResultOnchain}
          disabled={isSaving}
          className="bg-green-500 text-white px-4 py-2 rounded mb-2"
        >
          {isSaving ? 'Salvando...' : 'Salvar na Blockchain'}
        </button>
        <button
          className="bg-purple-500 text-white px-4 py-2 rounded"
          onClick={() => alert('Funcionalidade de compartilhamento no Warpcast será adicionada!')}
        >
          Compartilhar Resultado
        </button>
      </div>
    </div>
```

```
  );
}
```

6 Integrando o Warpcast

6.1 Adicionando Meta Tags para o Warpcast

1. Atualize `app/pages/result.tsx` para incluir meta tags `fc:frame` no `<head>`:

```
import { useState } from 'react';
import { useRouter } from 'next/router';
import { connectWallet } from '../utils/wallet';
import { writeContract } from 'wagmi/actions';
import Head from 'next/head';

export default function Result() {
  const router = useRouter();
  const { score } = router.query;
  const [isSaving, setIsSaving] = useState(false);

  const saveResultOnchain = async () => {
    setIsSaving(true);
    try {
      const walletClient = await connectWallet();
      const tx = await writeContract({
        address: '0xSeuContratoAqui',
        abi: [
          {
            name: 'saveResult',
            type: 'function',
            inputs: [{ name: 'result', type: 'string' }],
            outputs: [],
          },
        ],
        functionName: 'saveResult',
        args: [score],
      });
      alert('Resultado salvo na blockchain!');
    } catch (error) {
      console.error(error);
      alert('Erro ao salvar na blockchain.');
    } finally {
      setIsSaving(false);
    }
  };

  const shareUrl = `https://l2match.vercel.app/result?score=${score}`;

  return (
```

```
    <>
      <Head>
        <meta property="fc:frame" content="vNext" />
        <meta property="fc:frame:image" content="https://
           seusite.com/imagem-resultado.jpg" />
        <meta property="fc:frame:button:1" content="
           Compartilhar Resultado" />
        <meta property="fc:frame:button:1:action" content="
           post" />
        <meta property="fc:frame:post_url" content={shareUrl}
            />
      </Head>
      <div className="min-h-screen flex items-center justify-
         center bg-gray-100">
        <div className="bg-white p-6 rounded-lg shadow-lg">
          <h1 className="text-2xl font-bold mb-4">Seu
             Resultado</h1>
          <p className="text-lg mb-4">Você é: {score}</p>
          <button
            onClick={saveResultOnchain}
            disabled={isSaving}
            className="bg-green-500 text-white px-4 py-2
               rounded mb-2"
          >
            {isSaving ? 'Salvando...' : 'Salvar na Blockchain
               '}
          </button>
          <button
            className="bg-purple-500 text-white px-4 py-2
               rounded"
            onClick={() => window.open(`https://warpcast.com
               /~/compose?text=Eu sou ${score}! Descubra qual
                L2 você é: ${shareUrl}`, '_blank')}
          >
            Compartilhar no Warpcast
          </button>
        </div>
      </div>
    </>
  );
}
```

2. Para a meta tag `fc:frame:image`, você precisará de uma imagem pública (não incluída aqui, mas você pode hospedar uma imagem no Vercel ou outro serviço).

7 Implantando no Vercel

7.1 Configurando o Repositório no GitHub

1. Inicialize o Git no seu projeto:

```
git init
git add .
git commit -m "Primeiro commit do projeto Qual L2 Você É?"
```

2. Crie um repositório no GitHub (https://github.com/new) chamado `l2match`.
3. Vincule o repositório local e envie o código:

```
git remote add origin https://github.com/seu-usuario/l2match.
    git
git branch -M main
git push -u origin main
```

7.2 Implantando no Vercel

1. No terminal, faça login no Vercel:

```
vercel login
```

   Siga as instruções para autenticar.

2. Implante o projeto:

```
vercel deploy
```

3. Configure as variáveis de ambiente no Vercel:
   - Acesse https://vercel.com, vá para o seu projeto.
   - Em "Settings" > "Environment Variables", adicione:
   – $\texttt{NEXT}_{P}UBLIC_{A}LCHEMY_{A}PI_{K}EY = sua_{c}have_{a}lchemy$
4. Após a implantação, você receberá uma URL (e.g., https://l2match.vercel.app).

8 Publicando no Warpcast

8.1 Testando o App

1. Acesse a URL do Vercel (e.g., https://l2match.vercel.app).
2. Responda ao quiz e vá até a página de resultados.
3. Conecte a MetaMask e salve o resultado na blockchain Base (se configurado).

8.2 Compartilhando no Warpcast

1. Na página de resultados, clique em "Compartilhar no Warpcast".
2. Você será redirecionado para o Warpcast com um texto pré-preenchido (e.g., "Eu sou base! Descubra qual L2 você é: [sua URL]").
3. Publique o post.
4. Verifique se o botão "Compartilhar Resultado" aparece no Warpcast, graças às meta tags `fc:frame`.

9 Conclusão

Parabéns! Você criou e implantou um Mini App como o "Qual L2 Você É?". Agora você tem um quiz funcional, com salvamento onchain na Base e integração com o Warpcast. Continue explorando as possibilidades da Web3 com ferramentas como MiniKit e Warpcast para criar experiências ainda mais incríveis!