Sui

Do contexto ao desenvolvimento

Primeiro vamos entender como é estruturada a rede, como ela surgiu, e como sua estrutura única se comporta.

Sui

SUI: Uma nova abordagem

Co-fundadores	Lançamento	Consenso	Linguagem
Ex-engenheiros do Facebook/Meta (Diem project)	Mainnet em maio de 2023	Narwhal & Bullshark (sem blocos tradicionais)	A Sui utiliza a linguagem de programação Move, criada originalmente para ela

Sui

2018-2019:

Meta Diem Project

- **Equipe:** Ex-engenheiros Microsoft, Apple, VMware
- **Objetivo:** Moeda digital para 3+ bilhões de usuários
- **Problema:** Linguagens existentes inadequadas para assets digitais

2019-2020:

Design e Implementação

- **Lead:** Sam Blackshear (PHD Carnegie Mellon)
- **Co-lead:** Evan Cheng (LLVM Creator, Apple)
- **Inspiração:** Rust ownership + Linear types

2022:

Open source e Sui

- Meta abandona Diem
- Equipe forma Mysten Labs
- Move adaptado para Sui blockchain

Sui

Problemas das blockchains tradicionais

Sui

Escalabilidade	Custos elevados	Experiencia do usuário	Programação complexa
Ethereum ~15 TPS Bitcoin ~7 TPS	Gas fees podem passar de $50	Transações lentas e complexas	Solidity tem limitações de segurança

Sui

Arquitetura única da SUI

Sui

Object-Centric Model

- Tudo é um objeto com ID único
- Objetos têm propriedades (owned, shared, immutable)
- Transações paralelas quando não há conflito
  - Resultado: Até 297,000 TPS em testes

Abordagem orientada a Objetos

Principais características de um Objeto:

Identificação única
Propriedade (Ownership)
Controle de Acesso

Sui

Tipos de Propriedade (ownership)

O encapsulamento da SUI impõe uma propriedade para evitar cópias não autorizadas

Propriedade exclusiva (Address- Owned)

O objeto só pode ser alterado pelo seu proprietário

Objetos compartilhados (Shared)

São mutáveis por qualquer pessoa, mas exigem consenso para a alteração

Objetos Imutáveis (Immutable)

Não podem ser alterados após criação

Sui

Move Language: Por que é especial?

Sui

Características principais

Resource-oriented	Linear types	Formal Verification	Ausência de reentrancy attacks
Ativos digitais são "resources" que não podem ser copiados	Previne bugs como double-spending automaticamente	Código pode ser matematicamente provado	Por design da linguagem

Sui

Comparação com outras linguagens

Sui

Característica	Ethereum	Solana	Sui
TPS	~15	~3000	297,000+
Finalidade	~13min	~13s	<1s
Linguagem	Solidity	Rust/C	Move
Gas fees	Alto	Baixo	Muito baixo
Paralelização	Não	Limitada	Completa

Sui

Casos de uso ideais

Onde a Sui brilha

Gaming	DeFi	Social	Iot	Enterprise
NFTs e in-game assets com alta performance	Trading de alta frequência com baixas taxas	Apps com muitas interações simultâneas	Micropagamentos e dados de sensores	Aplicações corporativas que precisam de throughput

Sui

Com isso vemos uma nova forma de programar smart contracts com escalabilidade e segurança de ponta a ponta

Sui

Move: Dos conceitos ao Deploy

Primeiro vamos entender como é estruturada a rede, como ela surgiu, e como sua estrutura única se comporta.

Sui

Conceito de Pacote

O que é um Pacote?

Um pacote é a unidade de deploy na SUI. Ele é como um "projeto" completo, que contém todos os arquivos e configurações necessários.

Para que serve?

Ele é o arquivo de configuração principal, contendo o nome do projeto, dependências e endereços de deploy. É similar ao package.json no Node.js.

Onde o definimos?

No arquivo Move.toml

Sui

Conceito de Módulo

O que é um módulo em Move?

É a unidade fundamental de código em Move. É a nossa caixinha que armazena todo nosso contrato inteligente.

Onde o criamos?

Cada módulo é um arquivo separado, geralmente dentro da pasta source do seu pacote.

Para que serve?

Os módulos definem as estruturas (structs) dos seus objetos e as funções que operam sobre eles. Eles ajudam a organizar o projeto de forma lógica e segura.

Sui

Conceito de Structs

O que é um módulo em Move?

É a unidade fundamental de código em Move. É o nosso cabeça dos smart-contracts.

Onde o criamos?

Cada módulo é um arquivo separado, geralmente dentro da pasta source do seu pacote.

Para que serve?

Os módulos definem as estruturas (structs) dos seus objetos e as funções que operam sobre eles. Eles ajudam a organizar o projeto de forma lógica e segura.

Sui

Abilities

Copy

O valor pode ser copiado.

Drop

O valor pode ser descartado (destruído) no final de um escopo.

Store

O valor pode ser armazenado dentro de outras Structs.

Key

O valor pode ser usado como uma chave de recurso global; no Sui, torna a Struct um Objeto.

Sui

Entry Functions

O que são?

- São funções que servem como pontos de entrada para um contrato
- Elas são chamáveis diretamente a partir de transações

Para que servem as Entry Functions?

- Implementam a lógica de negócios on-chain
- São usadas em PTBs (Programmable Transaction Blocks) para operações atômicas

Sui

Linguagem Move

Qualidade

- Extremamente Segura
- Flexível
- Focada em Smart Contracts

Orientada a Recursos

- Move gerencia ativos com segurança
- Bloqueio de reentrância eficiente
- Sua estrutura previne vulnerabilidades comuns

Uso na SUI

- Controla objetos on-chain, garantindo gerenciamento de dados eficiente e seguro
- Permite definir estruturas e funções para a organização do seu contrato inteligente

Sui

Momento mão na massa

Vamos codificar agora nosso primeiro smart contract em move abordando os conceitos de base que vimos anteriormente

Sui

sui client

https://github.com/SilvaCleverson/bootcampSui/tree/main/sui-one-click-installer

Sui

Escaneie o QR-Code a seguir para ter acesso ao repositorio do nosso material oficial e vamos codar juntos nossos primeiros contratos em move

Construindo um Dapp na Sui

Entender o processo de construção de dApps no Sui, incluindo desenvolvimento de contratos inteligentes, integração front-end e conectividade de carteira.

Agenda

1. Conectando Front-End aos Contratos Inteligentes do Sui
2. Integração de Carteira
3. Construindo um App de Cunhagem de NFT
4. Construindo um App DeFi Básico
5. Exercícios Práticos

Conectando Front-End aos Contratos Inteligentes do Sui

Sui TypeScript SDK

- Interação blockchain de baixo nível
- Chamar funções de contrato inteligente

dApp Kit for React

- SuiClientProvider para rede
- WalletProvider para gerenciamento de carteira
- ConnectButton para login do usuário

Exemplos de Uso

Envolver app com provedores; usar ConnectButton; obter endereço da carteira

Pré requisitos

Node.js

Download and install Node.js

Instalação das Ferramentas (os SDKs)

@mysten/dapp-kit

@mysten/sui

@tanstack/react-query

Integração de Wallet

Wallet Role

Sign transactions, manage user assets

Uso do dApp Kit

Assinar transações, gerenciar ativos do usuário

Developer Tip

Acessar informações da carteira com o hook useCurrentAccount

Construindo um App de Mintagem de NFT

Smart Contract

Definir struct NFT com atributos key e store

Mint Function

Transferir NFT recém-criado para o remetente

Front-End

Usar SDK/dApp Kit para chamar mint com metadados

*Código de exemplo disponível para implementações Move e JavaScript.

Creando um app DeFi básico

Contrato Inteligente de Pool de Empréstimos

- Definir struct do pool de empréstimos
- Funções de depósito, empréstimo e pagamento
- Tratamento de juros incluído

Front-End Interface

- Página de depósito SUI
- Pedir emprestado usando garantia
- Página de pagamento de empréstimo

SDK Integration

- Invocar funções de contrato a partir da UI

Otimizando Integração Front-End

React Hooks

Usar hooks para gerenciamento de estado e conta

Envolvimento de Provedor

Envolver app com provedores SuiClient e Wallet

User Experience

Mostrar status de conexão da carteira e tratamento de erros

Exercícios Práticos

Criando um app de Mintagem de NFT

- ✓ Código Move para struct NFT, Typescript para chamada de mint do front-end

Construindo um app Defi básico

- ✓ Usando SDK: Chamar funções de contrato inteligente do front-end para depósitos, empréstimos, pagamentos

(https://mirror.xyz/greymate.eth/_P2NXvVoh9wISj_mqgavDymIERCnW2DgC1gigJNrmUI)

(https://docs.sui.io/guides/developer/app-examples)

```move
module meu_nft_exemplo::meu_nft {

    use sui::display;
    use std::string::{Self, String};
    use sui::package::{Self, Publisher};

    // --- Definição do Objeto NFT ---

    /// One-Time Witness para o Publisher
    public struct MEU_NFT has drop {}

    /// A estrutura principal do nosso NFT.
    public struct MeuNFT has key, store {
        id: UID,
        name: String,
        description: String,
        /// URL para a imagem do NFT (idealmente, um link de gateway IPFS https:///).
        url: String
    }

    // --- Funções de Inicialização ---

    fun init(otw: MEU_NFT, ctx: &mut TxContext) {
        // Cria um novo objeto Publisher para o nosso tipo `MeuNFT`.
        let publisher = package::claim(otw, ctx);
        // Transfere o Publisher para a pessoa que está publicando o contrato.
        // Isso garante que apenas o criador do contrato possa mudar como os NFTs são exibidos.
        transfer::public_transfer(publisher, tx_context::sender(ctx));
    }

    // --- Funções Públicas (Entry Functions) ---

    /// Cria ("minta") uma nova instância do nosso NFT e a envia para o chamador.
    entry fun mint(
        name: vector<u8>,
        description: vector<u8>,
        url: vector<u8>,
        ctx: &mut TxContext
    ) {
        let nft = MeuNFT {
            id: object::new(ctx),
            name: string::utf8(name),
            description: string::utf8(description),
            url: string::utf8(url),
        };
        // Transfere o NFT recém-criado para a carteira que chamou esta função.
        transfer::public_transfer(nft, tx_context::sender(ctx));
    }
```

Features Avançadas

Aprender como construir IA no Sui e interagir com contratos inteligentes, experiência simples de integração com login ZK e Transação Patrocinada

Agenda

1. Como o zkLogin Funciona
2. Transações Patrocinadas - Gerenciamento de Taxas de Gas
3. Oráculos do Sui - Trazendo Dados Off-Chain para On-Chain
4. Pontes Cross-Chain no Sui
5. Construindo Agentes de IA Autônomos no Sui
6. Exercícios Práticos

Como zkLogin Funciona

Fazer login em dApps do Sui com credenciais web familiares e privacidade via provas de conhecimento zero.

Login do usuario

Autenticação com login do Google ou facebook

Preservação de privacidade

Verificação de identidade sem mostrar informações do usuário

Geração de provas

Criar zkProofs usando JWT e salt para segurança de conta

https://docs.sui.io/concepts/cryptography/zklogin

segment	cabbage	shuffle	mosquito
nephew	furnace	cinnamon	garment
wine	wing	wink	stumble
saddle	nominee	grief	vacuum

Possibilidade de reutilizar metodos existentes

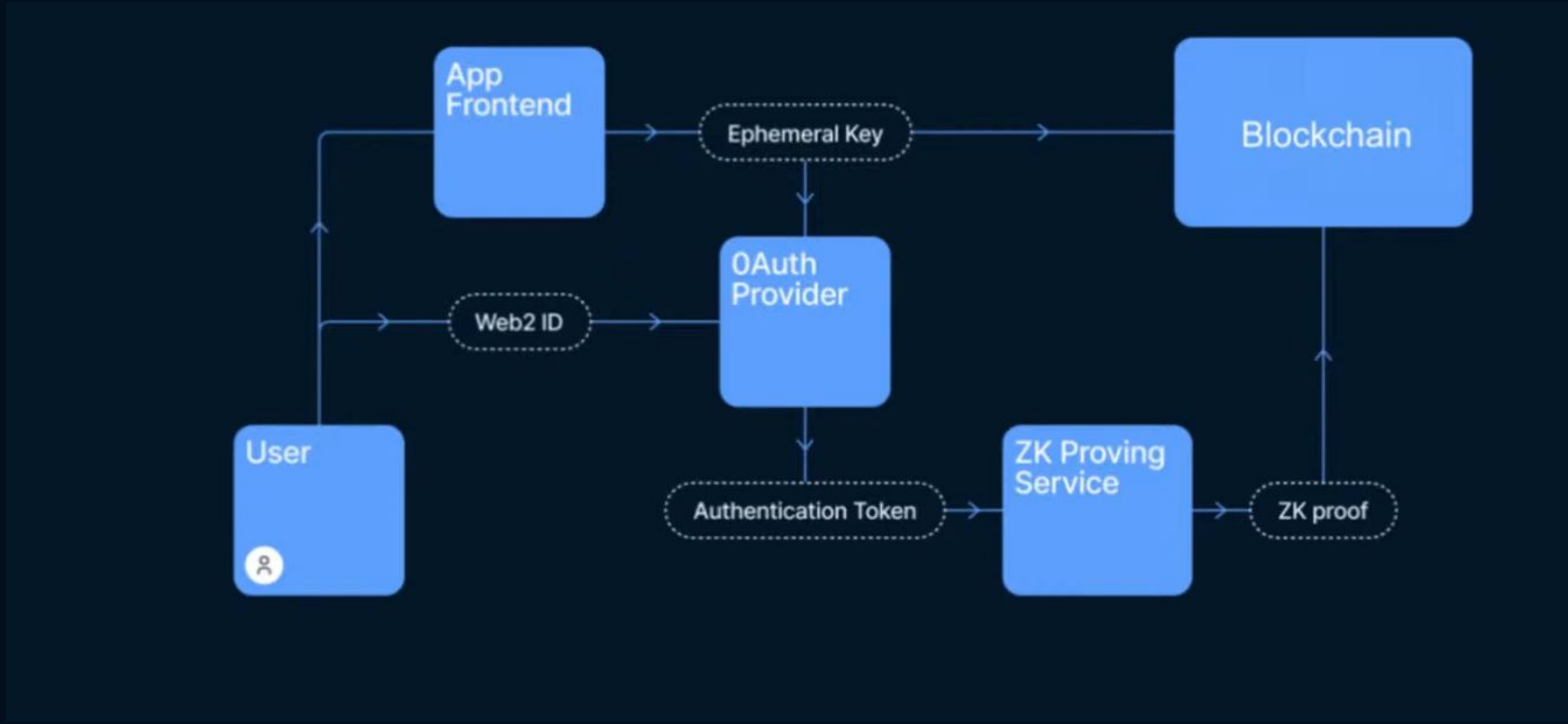
Fluxo de login familiar com Google OAuth/OpenID Connect

Toda transação requer sua aprovação, mas de forma simplificada.

```
{
  "iss": "https://accounts.google.com",
  "azp": "575519204237-msop9ep45u2uo98hapqmngv8d84qdc8k.apps.googleusercontent.com",
  "aud": "575519204237-msop9ep45u2uo98hapqmngv8d84qdc8k.apps.googleusercontent.com",
  "sub": "110463452167303598383",
  "nonce": "16637918813908060261870528903994038721669799613803601616678155512181273289477",
  "iat": 1682002642,
  "exp": 1682006242,
  "jti": "a8a0728a3ffd5dc81ecfd0ea81d0d33d803eb830"
}
```


- Como Identificar o usuário de forma persistente?
  - **address = H(sub, iss, aud)**
- Como assinar uma transação?
  - Gerar um par de chaves efêmeras: **sk, pk**
  - Inserir **pk** dentro do nonce
  - Usar **sk** para assinar as transações

Fluxo ZKLogin



Exercício — Integração zkLogin

- Left half: code snippet from `ex1/src/pages/api/login.ts` showing stub:

ts Copy Edit

```ts
export default async function login(req, res) {
  // TODO: call Google OAuth, generate proof
  res.status(200).json({ proof: '0x…' });
}
```

- Right half: terminal screenshot of `npm run dev` and browser hitting `/api/login` returning JSON

Sua tarefa: complete o handler login.ts. Use @googleapis/oauth2 para autenticar e o helper generateZkProof do @sui/sui.js.

Exemplo: https://docs.sui.io/guides/developer/cryptography/zklogin-integration/zklogin-example

Transações Patrocinadas - Gerenciamento de Taxas de Gas

Gas Sponsorship Roles

Usuário, Estação de Gas, Patrocinador gerenciam taxas de forma transparente

Service Example

A Estação de Gas da Shinami simplifica o patrocínio de pagamentos de gas

Use Case

dApps de jogos patrocinam transações iniciais para impulsionar usuários

(https://blog.sui.io/shinami-gas-station-tutorial/)

Exercício — Seja sponsor de sua primeira TX

- Screenshot of `docker-compose up` spinning up `gas-station:latest`
- Code snippet from `ex2/src/index.ts` with stubbed call to `GasStationClient.sponsorTransaction(tx)`

Oráculos do Sui - Trazendo Dados Off-Chain para On-Chain

Oráculos Disponíveis

- Chainlink
- Band protocol
- Mysten Labs' simple e meta oracles

Ferramentas do desenvolvedor

Guias para construir oráculos com alto rendimento do Sui

https://docs.sui.io/guides/developer/app-examples/weather-oracle#initialize-the-project
https://github.com/pentagonxyz/move-oracles

Exercício — Feed de Preços On-Chain

- Sample JSON from Oracle REST:

```json
{ "symbol": "ETH/USD", "price": 3456.78 }
```

- Move snippet from `ex3/contracts/oracle_demo.move` with TODO in `entry fun submit_price`

Requisitos: fazer post API http://localhost:8080/price?symbol=ETH/USD, analisar o JSON em seu script Node, então chamar a função submit_price do seu contrato oracle Move

Pontes Cross-Chain no Sui - interoperabilidade

Sui Bridge (Ponte Nativa)

- Ativos : ETH, WETH, USDT, WBTC, LBTC
- Mecanismo: Modelo Lock e mint.

https://bridge.sui.io/

Outras Pontes

https://wormhole.com/docs/tutorials/messaging/sui-connect/

Circle's CCTP (for USDC)

Construindo com Agentes de IA na Sui

Logica do Agent

Definidos em contratos inteligentes Move para ações on-chain

Oráculos e Automações

Conectar feeds de dados; patrocinar transações para automação

Modelos de IA Externos

Integrar IA off-chain para tomada de decisões

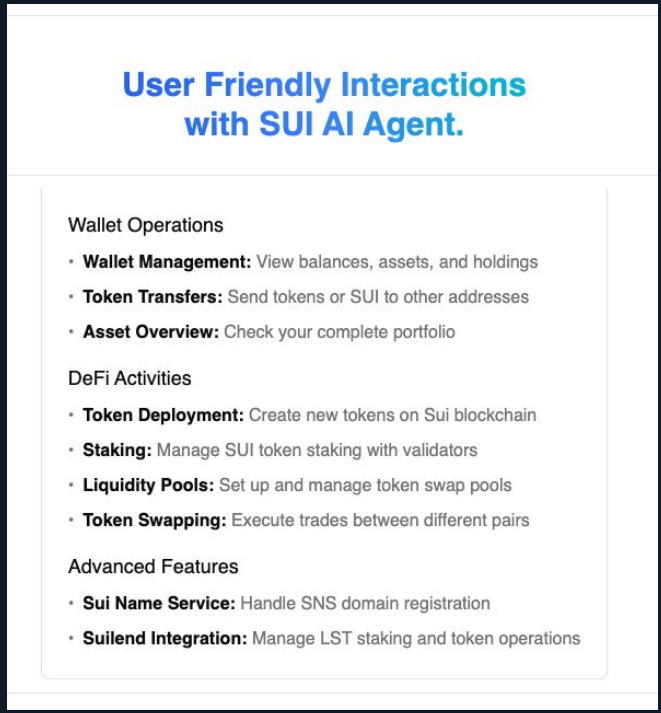
Segurança

Usar capacidades do Move e controles de acesso

Caso de Uso do Agente de IA: Kit de Agente de IA Nimbus

1. Este kit suporta a construção de agente de IA no Sui
2. Suporta obter saldo da carteira
3. Suporta interagir com múltiplos protocolos DeFi do Sui

https://agent.getnimbus.io/



Exercícios Práticos

Construindo Agente de IA Simples para rastrear saldo da carteira

Exemplo de AI agent KIT:
https://docs.getnimbus.io/sui-ai-agent/introduction

SDK: https://www.npmjs.com/package/@flowx-finance/sdk

Obrigado