DApps

Identificar o conceito de DApps, identificando aspectos funcionais

Prof. Jose Emiliano

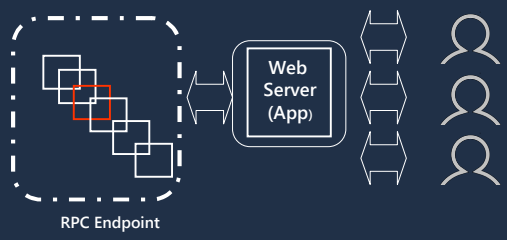
DApp

- É um aplicativo que roda sobre uma rede blockchain em vez de servidores centralizados.
  - Combina uma interface tradicional (como um site ou app) com um backend descentralizado, geralmente implementado por meio de contratos inteligentes.

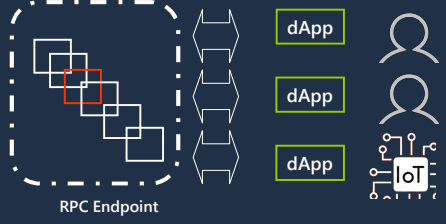
Prof. Jose Emiliano

Web 3

Mixed Web 3 - Centralizado



Full Web 3 - Descentralizado



Modelo híbrido da Web 3.0	Modelo padrão da Web 3.x / 4.0

Prof. Jose Emiliano

DApp

- Características
  - Descentralização
    - O backend roda em uma blockchain, sem controle de uma única entidade
  - Código Aberto
    - O código geralmente é público e auditável
  - Contratos Inteligentes
    - A lógica do app é executada por contratos na blockchain

Prof. Jose Emiliano

DApp

- Características
  - Tokenização
    - Pode usar tokens para acesso, governança ou recompensas
  - Resistência à Censura
    - Como roda em rede distribuída, é difícil de ser derrubada ou censurada

DApp

- Partes componentes:
  - Backend
    - Rede blockchain
    - Contrato inteligente (código escrito solidity)
  - Frontend
    - Endereço de deploy
    - ABI
    - Html (react, angular, vue) e bibliotecas javascript:
      - Web3.js
      - Ether.js

Front End

- Utiliza chamadas assíncronas de métodos constantes do contrato
  - Uso de async...await
  - Variáveis mínimas (js)
    - provider – conector com a blockchain: RPC,Metamask
    - signer – o usuário que assina transações
    - contract – o objeto JS que representa o contrato on-chain
    - abi – As funções que o contrato dispõe (JSON)
    - contractAddress – endereço do contrato já deployed

Etapas

- Principais métodos:
  - ethereum.request()
    - parte do Ethereum Provider API, empacota chamadas JSON-RPC .

Etapas

- Detectar a carteira do usuário

```
await window.ethereum.request({ method: 'eth_requestAccounts' });
```

- Conectar um provider (provedor), uma interface para um nó Ethereum
  - Nó Ethereum – Software que mantem cópia da bc e participa da rede (pode ser full node ou light node)
  - Provider – Interface de acesso a um ou mais nós usada por bibliotecas como ethers.js ou web3.js para interação com a rede.

```
const provider = new
  ethers.providers.Web3Provider(window.ethereum);
```


Etapas

- Obter o signer

```
const signer = provider.getSigner();
```

- Definir ABI e endereço do contrato

```
const abi = [ /* ... ABI do contrato ... */ ];
// Endereço do contrato na rede
const contratoEndereco = "0x123...";
```

Etapas

- Criar o contrato (ethers.js):
  - Para escrita:

```
const contrato =
     new ethers.Contract(contratoEndereco, abi, signer);
```

  - Para Leitura:

```
const contrato = new ethers.Contract(contratoEndereco,
  abi, provider);
```


Etapas

- Criar o contrato (web3.js):
-

```
const contrato = new web3.eth.Contract(abi,
contratoEndereco);
```

- Chamada de funções (view/pure)

```
const nome = await contrato.consultarNome("0xabc...");
```

- Chamada de funções (que registram no contrato)

```
const tx = await contrato.registrarNome("Jose");
await tx.wait(); // Aguarda confirmação na rede
```

Resumo

Etapa	ethers.js	web3.js
Detectar carteira	ethereum.request({ method: 'eth_requestAccounts' })	Mesma coisa
Criar conexão	new ethers.providers.Web3Provider()	new Web3(window.ethereum)
Obter conta/signer	getSigner()	web3.eth.getAccounts()
Criar instância do contrato	new Contract(endereço, abi, signer/provider)	new web3.eth.Contract(abi, endereço)
Chamar função	contrato.minhaFuncao(...)	contrato.methods.minhaFuncao().call()/send()

Prof. Jose Emiliano

Contrato de exemplo

```
contract SeedGuard {
    struct Usuario {
        address endereco;
        string mensagem;
    }
    mapping(address => Usuario) private usuarios;
    constructor() {}
 function guardarMensagem(string memory mens) public returns (bool sucesso)
  {
        Usuario memory u = Usuario(msg.sender, mens);
        usuarios[msg.sender] = u;
        return true;
    }
  function lerMensagem() public view returns (string memory) {
        return usuarios[msg.sender].mensagem;
    }
}
```

Prof. Jose Emiliano

Estrutura

```
<script src="https://cdn.jsdelivr.net/npm/ethers@6.7.0/dist/ethers.umd.min.js"></script>
<script>
const contratoAddress="0xADDR";
const contratoABI=[{...}];
//variáveis padrão
let provider,signer,contrato;
//conexão com a carteira Metamask
async function conectar(){
   if(!window.ethereum)return alert("Metamask Error!");
   await ethereum.request({method:'eth_requestAccounts'});
    provider = new ethers.BrowserProvider(window.ethereum);
    signer = await provider.getSigner();
    contrato = new ethers.Contract(contratoAddress, contratoABI, signer);
}
...
```


Estrutura

```
//funções do contrato (especificadas na ABI)
   //function guardarMensagem(string memory mens)...
async function guardar(){
   const text=document.getElementById("msg-input").value;//do documento html
   const tx=await contrato.guardarMensagem(text);//vide ABI
   await tx.wait(); alert("Message sent!");
}
   //function lerMensagem()public return (string memory)...
async function ler(){
   const.msg=await contrato.lerMensagem();//vide ABI
   document.getElementById("msg-output").textContent=msg;
   alert("This is your message!");
}

</script>
```

Scaffold-ETH

Identificar ferramentas para a criação de DApps de forma profissional

Prof. Jose Emiliano

Scaffold

- é um kit de ferramentas open-source criado para facilitar o desenvolvimento de aplicações descentralizadas (dApps) na blockchain Ethereum
- Para que serve?
  - Prototipagem rápida: Ideal para testar idéias de contratos inteligentes com uma interface funcional.
  - Aprendizado prático: Ótimo para quem está começando a entender como funcionam contratos inteligentes e dApps.
  - Desenvolvimento completo: Serve tanto para projetos simples quanto para aplicações robustas e prontas para produção.

Prof. Jose Emiliano

Scaffold

- O que ele oferece?
  - Atualização automática da interface conforme você modifica o contrato inteligente.
  - Interface pronta com React (Next.js) conectada ao contrato.
  - Integração com carteiras como MetaMask e WalletConnect quanto para aplicações robustas e prontas para produção.
  - Faucets próprias e integradas à plataforma

Componentes

- Ambiente Local
  - Rede Local Hardhat: Ambiente de desenvolvimento isolado para testes rápidos
  - Contrato Desenvolvimento: Versão de teste dos seus contratos inteligentes
  - Frontend NextJS: Interface do usuário em desenvolvimento
- Testnet
  - Rede de teste pública para validação em ambiente real
  - Contrato Testnet: Versão pré-produção dos contratos
  - Faucet: Sistema para obter ethers de teste gratuitos
- Produção
  - Mainnet: Rede principal da Ethereum
  - Contrato Produção: Versão final dos contratos inteligentes
  - Vercel/IPFS: Hospedagem do frontend em produção

Estrutura de pastas

```
my-dapp/
├── contracts/        → contratos Solidity
├── frontend/         → aplicação React (Next.js)
├── scripts/          → deploys ou tarefas customizadas
├── packages/         → configurações e utilitários
```


Criando primeiro dApp

- Na pasta:
  - C:/projetos_node/
    - Digite: npx create-eth@latest
  - ou
    - git clone https://github.com/scaffold-eth/scaffold-eth-2.git
    - cd scaffold-eth-2
    - yarn install

Criando primeiro dApp

- Na pasta:
  - C:/projetos_node/
    - Digite: npx create-eth@latest
  - Esse comando inicia uma instalação guiada, onde você escolhe:
    - Nome do projeto
    - Framework de Solidity (Hardhat ou Foundry)
    - Extensões opcionais (como ERC-20, Subgraph, etc.)
    - Informe o nome do projeto: scaffold-proj
    - Prossiga a instalação dos módulos (1.5GB)

Criando primeiro dApp

- Após a instalação:
  - `Yarn chain` - inicializa o nó local de uma testnet com vários endereços e valores
  - `Yarn deploy` – inicializa a implantação do contrato inteligente e da aplicação front-end
  - `Yarn start` – executa o servidor na porta padrão (3000)
  - `Yarn vercel` – para o deploy inicial
  - `Yarn vercel --prod` – para o deploy de produção (Vercel ou IPFS)

Pastas importantes

- As seguintes pastas contem arquivos importantes:
  - contracts
    - YourContract.sol – o contrato que está sendo deployed
  - deploy
    - 00_deploy_your_contract.ts – o script de deploy do contrato principal do seu projeto
  - nextjs/app
    - page.tsx – landing page gerada automaticamente para o projeto

Pastas importantes

- Para o deploy em produção:
  - nextjs/utils
    - scaffold.config.ts – landing page gerada automaticamente para o projeto

```
export const config = {
    targetNetwork: chains.mainnet,
    alchemyApiKey: process.env.NEXT_PUBLIC_ALCHEMY_API_KEY!,
    walletConnectProjectId:
    process.env.NEXT_PUBLIC_WALLET_CONNECT_PROJECT_ID!
};
```

Vercel X IPFS

- Vercel
  - CDN global: Rede de distribuição de conteúdo que coloca a aplicação em servidores próximos aos usuários em todo o mundo
  - Automatização de builds: Processo automático que prepara o código para produção
  - Integração com Git: Conecta-se diretamente ao repositório de código, fazendo deploy automaticamente
- IPFS
  - Nodes ativos: Computadores que mantêm uma cópia dos seus arquivos e os compartilham com a rede
  - Identificação por conteúdo: Em vez de usar URLs tradicionais, os arquivos são identificados por um hash único baseado no seu conteúdo (CID)
  - Redundância automática: Os arquivos são automaticamente copiados para múltiplos nodes da rede

Prof. Jose Emiliano

Vercel X IPFS

- Quando usar cada um?
- Use Vercel quando:
  - Quiser uma solução rápida e fácil para hospedar aplicações web
  - Precisar de automação completa no processo de deploy
  - Tiver uma aplicação que precisa de boa performance global
  - Estiver desenvolvendo projetos menores ou médios
- Use IPFS quando:
  - Precisar de armazenamento verdadeiramente descentralizado
  - Quiser que seus arquivos permaneçam disponíveis mesmo se seu site principal cair
  - Necessitar de resistência a censura
  - Tiver conteúdo que precisa ser imutável (não pode ser alterado)

Prof. Jose Emiliano

Vercel X IPFS

- Resumo
  - Vercel é uma plataforma de hospedagem tradicional, focada em facilitar o deploy de aplicações web, enquanto
  - IPFS é um sistema de armazenamento descentralizado, focado em durabilidade e resistência a falhas.
    - Embora ambos possam ser usados para hospedar conteúdo web, eles têm propósitos e abordagens muito diferentes.

Fim