



Insper

Ativos Digitais e Blockchain

Ricardo Rocha
Raul Ikeda

Objetivo

- Programando SmartContracts

Relembrando

Site: <http://ethereum.org/developers/>

- Smart Contract Language:
 - Solidity (javascript)
 - Bamboo
 - Vyper (tipo Python)
 - Flint
- Compiler:
 - Truffle
 - Waffle
 - Brownie
 - etc
- Network:
 - Ganache
 - Ethnode
 - Infura
 - etc.

Teste rápido:

Acessar: <http://remix.ethereum.org/>

Ele é dividido em três áreas:

Área de Edição



Área de Compilação

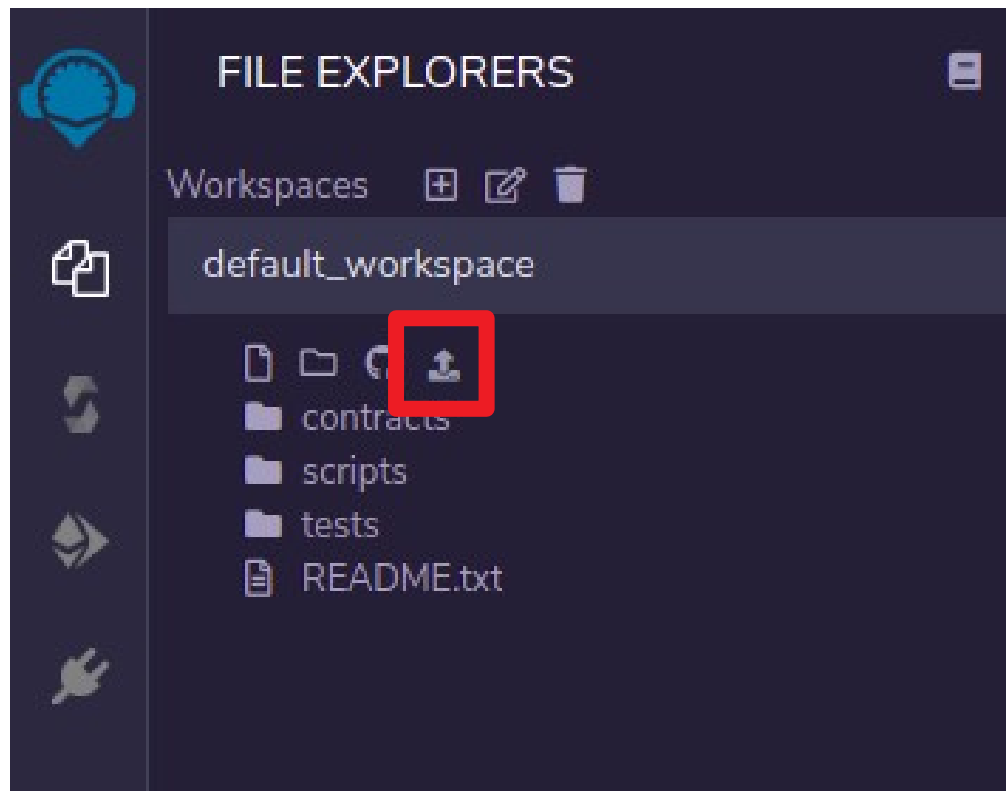


Área de Deploy



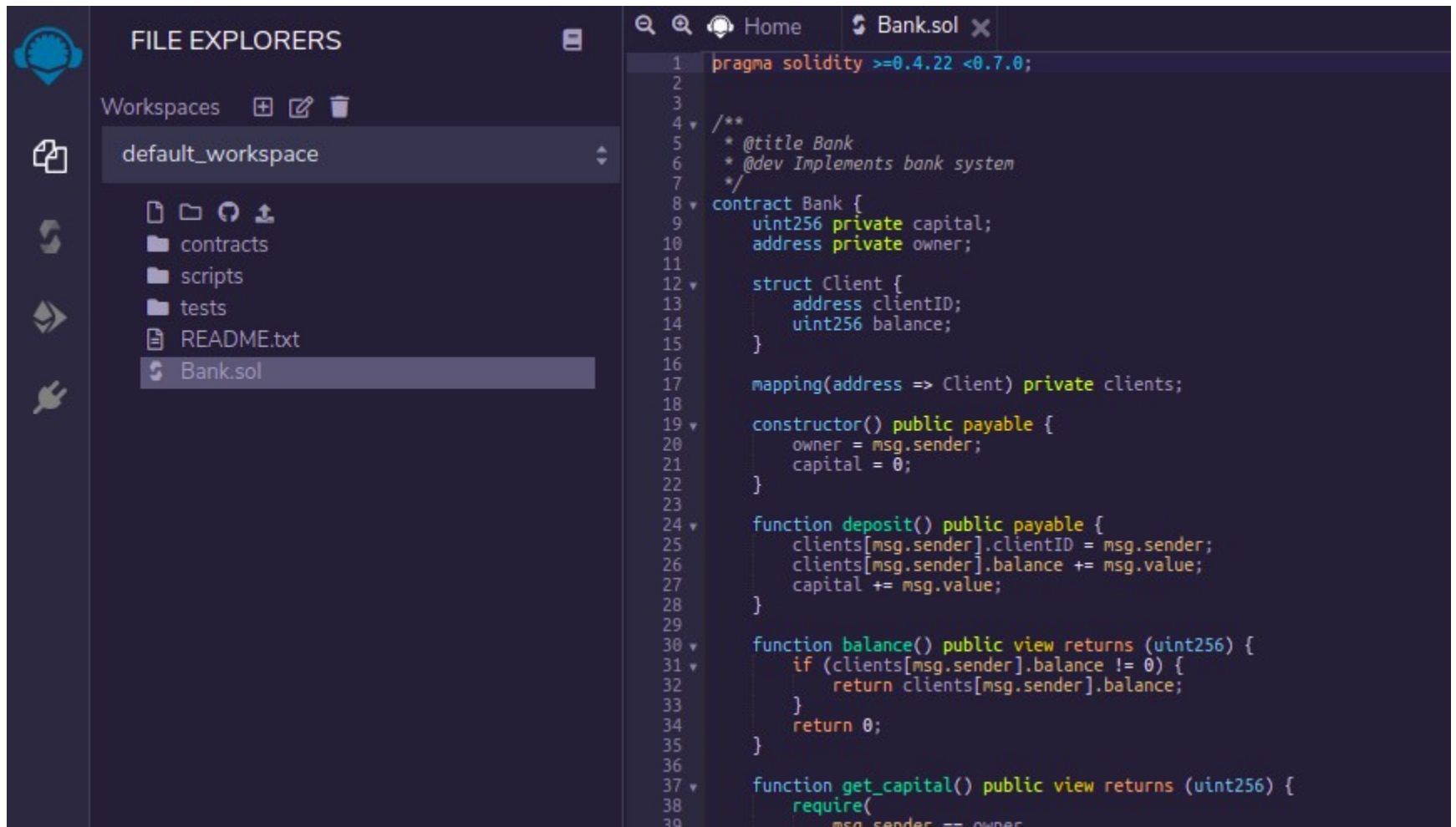
Área de edição

Onde os contratos são editados



Fazer o upload do arquivo Bank.sol

Área de edição

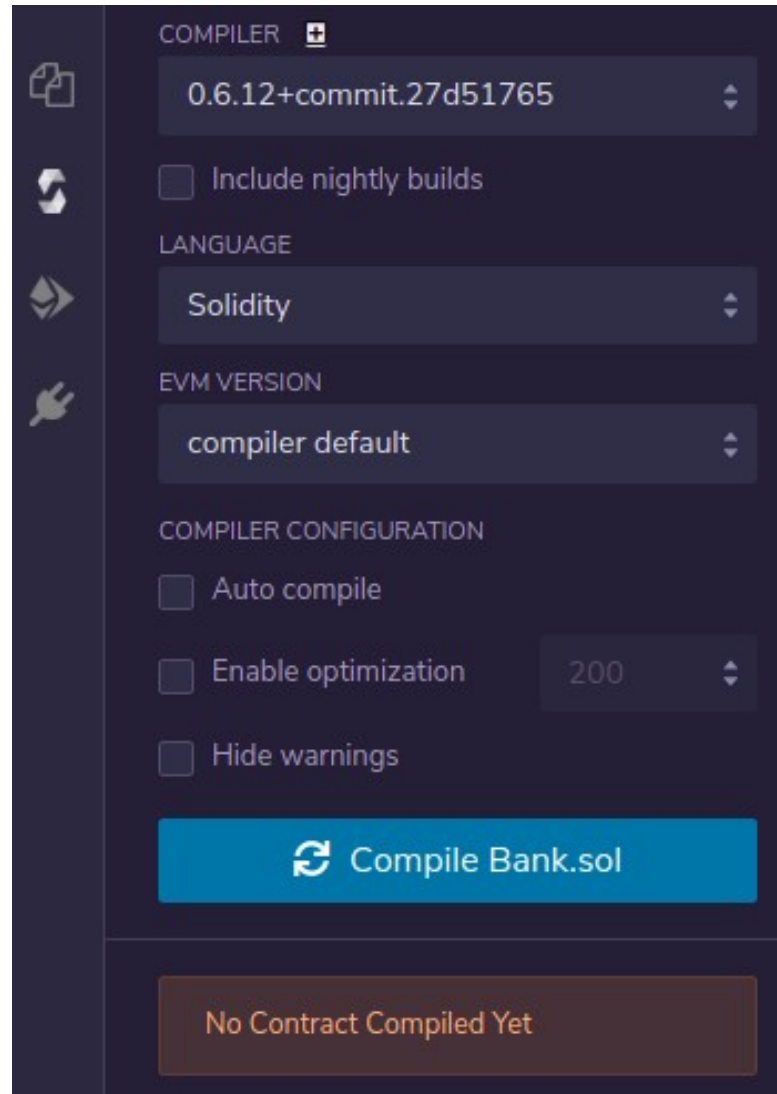


The screenshot displays the Insper IDE interface. On the left, the 'FILE EXPLORERS' sidebar shows a project structure with a 'default_workspace' containing folders for 'contracts', 'scripts', and 'tests', along with 'README.txt' and 'Bank.sol'. The 'Bank.sol' file is selected and open in the main editor. The editor shows Solidity code for a 'Bank' contract, including a pragma statement, a comment, a struct for 'Client', and functions for 'deposit', 'balance', and 'get_capital'. The code is syntax-highlighted and includes line numbers on the left margin.

```
1 pragma solidity >=0.4.22 <0.7.0;
2
3
4 /**
5  * @title Bank
6  * @dev Implements bank system
7  */
8 contract Bank {
9     uint256 private capital;
10    address private owner;
11
12    struct Client {
13        address clientID;
14        uint256 balance;
15    }
16
17    mapping(address => Client) private clients;
18
19    constructor() public payable {
20        owner = msg.sender;
21        capital = 0;
22    }
23
24    function deposit() public payable {
25        clients[msg.sender].clientID = msg.sender;
26        clients[msg.sender].balance += msg.value;
27        capital += msg.value;
28    }
29
30    function balance() public view returns (uint256) {
31        if (clients[msg.sender].balance != 0) {
32            return clients[msg.sender].balance;
33        }
34        return 0;
35    }
36
37    function get_capital() public view returns (uint256) {
38        require(
39            msg.sender == owner
```

Área de Compilação

Pressione para compilar
O arquivo Bank.sol



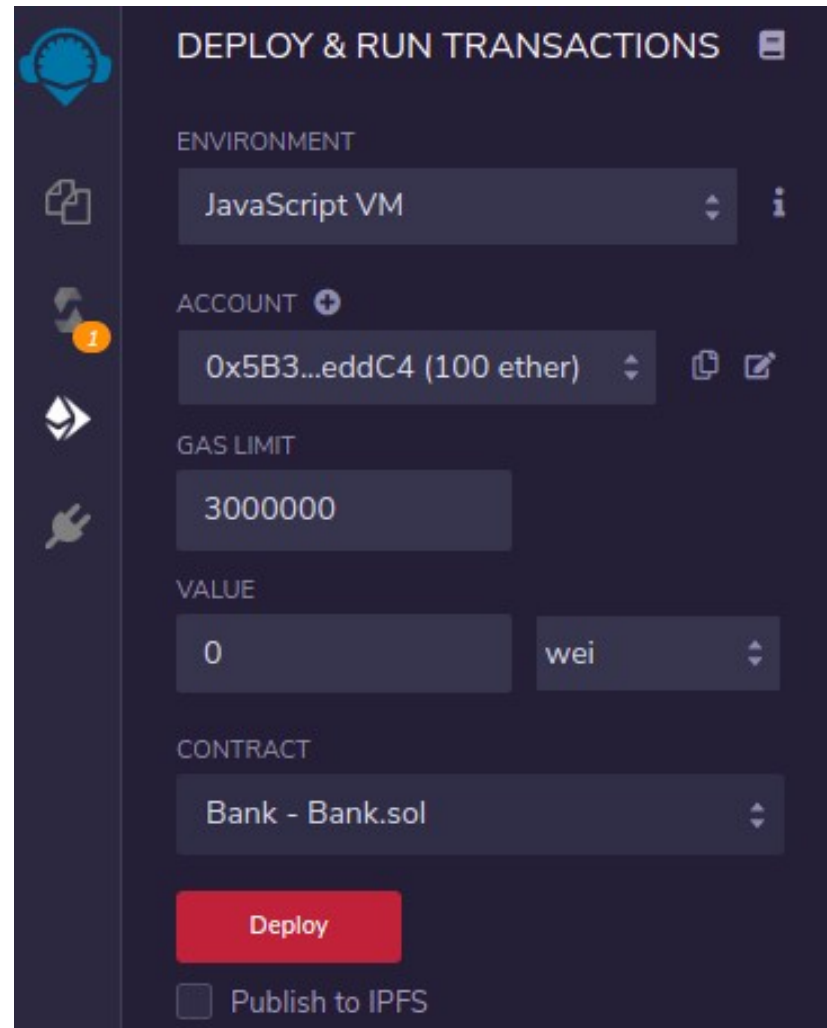
The screenshot shows the 'COMPILER' settings panel in the Insper IDE. On the left is a sidebar with icons for file explorer, console, compiler, and a plug. The main panel has the following sections:

- COMPILER**: A dropdown menu showing '0.6.12+commit.27d51765'.
- ☐ Include nightly builds
- LANGUAGE**: A dropdown menu showing 'Solidity'.
- EVM VERSION**: A dropdown menu showing 'compiler default'.
- COMPILER CONFIGURATION**:
 - ☐ Auto compile
 - ☐ Enable optimization (with a dropdown showing '200')
 - ☐ Hide warnings

At the bottom of the settings panel is a large blue button with a refresh icon and the text 'Compile Bank.sol'. Below this button, in a separate section, is a message: 'No Contract Compiled Yet'.

Área de Deploy

Selecione alguma conta
E pressione Deploy
para implantar o contrato



DEPLOY & RUN TRANSACTIONS

ENVIRONMENT

JavaScript VM

ACCOUNT

0x5B3...eddC4 (100 ether)

GAS LIMIT

3000000

VALUE

0 wei

CONTRACT

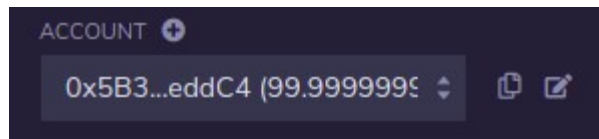
Bank - Bank.sol

Deploy

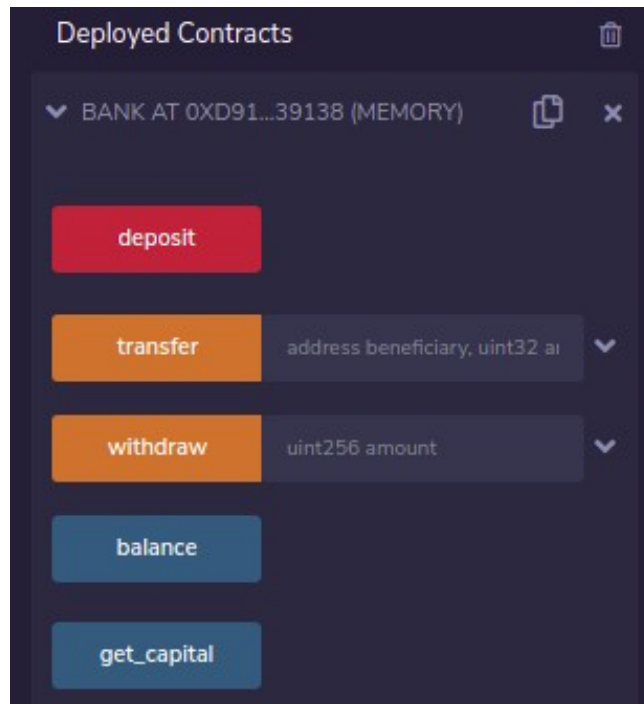
☐ Publish to IPFS

Área de Deploy

Você verá que o saldo da conta diminuirá (custo de implantação)



E aparecerá embaixo em Deployed Contracts:



Analizando o contrato Bank.sol

Atributos:

- capital: total de dinheiro no banco
- owner: endereço do dono do banco
- clients: lista de clientes do banco

Métodos:

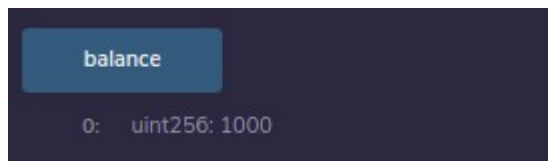
- constructor: roda automaticamente quando implanta o contrato
- deposit(): realizar um depósito na conta
- withdraw(): saca o dinheiro da conta
- transfer(): transfere o dinheiro para outra conta
- balance(): exibe o saldo atual da conta
- get_capital(): exibe o capital total do banco

Testando Rapidamente

1. Selecione uma conta
2. Preencha um valor em wei
3. Pressione Deposit

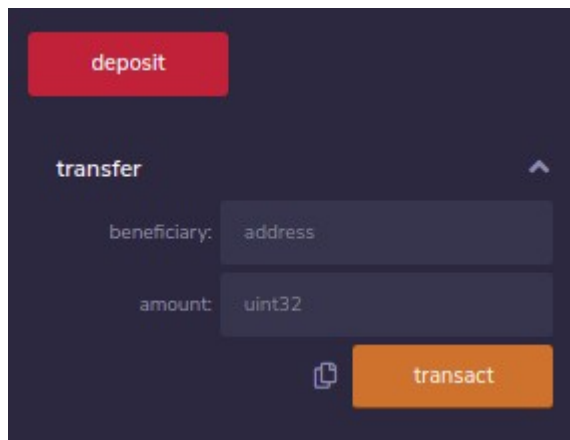
Agora você tem uma conta bancária com um saldo inicial no valor que você preencheu em weis.

Você pode ver seu saldo pressionando Balance:



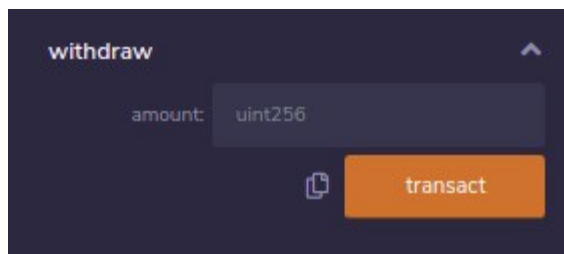
Testando Rapidamente

Você pode transferir para outra conta (de endereço válido):



A screenshot of a web interface for a 'transfer' transaction. At the top, there is a red button labeled 'deposit'. Below it, the word 'transfer' is displayed with an upward arrow. There are two input fields: 'beneficiary:' with the placeholder text 'address' and 'amount:' with the placeholder text 'uint32'. At the bottom right, there is an orange button labeled 'transact' with a copy icon to its left.

Você pode sacar um valor:



A screenshot of a web interface for a 'withdraw' transaction. At the top, the word 'withdraw' is displayed with an upward arrow. There is one input field labeled 'amount:' with the placeholder text 'uint256'. At the bottom right, there is an orange button labeled 'transact' with a copy icon to its left.

Testando Rapidamente

Gaste um tempo revendo o código do contrato e interagindo através da simulação.

Faça dois deploys seguidos para entender o mecanismo de implantação de contratos

Não se preocupe em quebrá-lo. Basta reiniciar o processo para restabelecer a simulação.

Não se preocupe com o código ainda, nas próximas aulas iremos abordar passo a passo os elementos dos contratos.

Vamos agora ver outro contrato: Auction.sol

Contrato Auction.sol

Atributos:

- beneficiary: endereço do vendedor
- auctionEndTime: tempo de término do leilão
- highestBidder: endereço do dono da maior oferta
- highestBid: valor da maior oferta
- pendingReturns: dicionário de ofertas

Métodos:

- constructor: roda automaticamente quando implanta o contrato
- bid(): registra uma oferta
- withdraw(): saca o dinheiro das ofertas, exceto o comprador
- auctionEnd(): encerra o leilão

Eventos:

- HighestBidIncreased: quando há um novo best bid
- AuctionEnded: quando o leilão acaba

Testando o Leilão

Refaça os passos anteriores para realizar a implantação do contrato no ambiente Remix

E se quiséssemos montar em um ambiente local, sem usar o Remix?

Montagem do Ambiente Local

- Instalar o Nodejs 14 LTS
 - <https://nodejs.org/en/>
 - Quem está no Linux: atualizar usando n
- Instalar o Truffle
 - <https://www.trufflesuite.com/>
- Instalar o Ganache
 - <https://www.trufflesuite.com/>
- Instalar Metamask:
 - <https://metamask.io/>

Testando usando Ganache

- Crie uma pasta e acesse via **prompt**
- > truffle init
- Coloque o arquivo auction.sol em contracts
- > truffle compile
- Abrir o arquivo truffle-config.js:
 - Remover o comentário do bloco development
- > truffle migrate
- Um Frontend faz falta agora!

Tutorial

Realizar o tutorial:

- <https://www.trufflesuite.com/tutorial>

Cuidado para configurar o MetaMask corretamente

ATENÇÃO: Se você já possui o MetaMask e se você possui Ethers na rede principal, muito cuidado na hora de “gastar” a simulação. Veja bem qual rede e qual conta você está usando.

Próxima Aula

Continuação do desenvolvimento de SmartContracts