

# Aula 22 - Ethereum

## Ferramentas de Desenvolvimento e Frameworks

---

Prof. Rogério Aparecido Gonçalves<sup>1</sup>

rogerioag@utfpr.edu.br

<sup>1</sup>Universidade Tecnológica Federal do Paraná (UTFPR)  
Departamento de Computação (DACOM)  
Campo Mourão - Paraná - Brasil

Programa de Pós Graduação em Ciência da Computação

**Mestrado em Ciência da Computação**

PPGCC17 - Tópicos em Redes de Computadores e Cibersegurança



# Agenda i

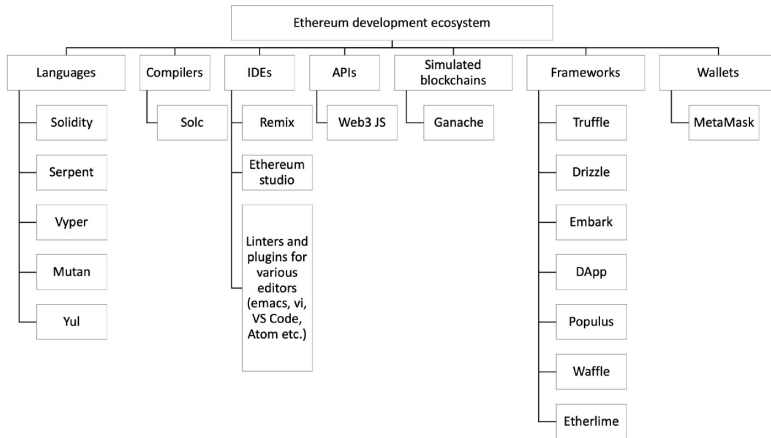
1. Introdução
2. Próximas Aulas
3. Referências

# Introdução

---

- Apresentação de Ferramentas de Desenvolvimento e *Frameworks*.
- Linguagens, Compiladores, Ferramentas e Bibliotecas, *Frameworks*, Desenvolvimento e implantação de contratos e Linguagem Solidity.

# Taxonomia do Ecossistema de Componentes de Desenvolvimento Ethereum i



- **Solidity:** Tem se tornado a linguagem padrão para escrever contratos para *Ethereum*. O código precisa ser compilado e transformado em *bytecode*, é necessário utilizar o compilador **solc**.
- **Vyper:** Essa linguagem é uma linguagem experimental semelhante ao Python que está sendo desenvolvida para trazer segurança, simplicidade e auditabilidade ao desenvolvimento de contratos inteligentes.
- **Yul:** Esta é uma linguagem intermediária que tem a capacidade de compilar para diferentes back-ends, como EVM e eWasm. Os objetivos de projeto do Yul incluem principalmente legibilidade, fluxo de controle fácil, otimização, verificação formal e simplicidade.
- **Mutan:** Esta é uma linguagem de estilo Go, que foi descontinuada no início de 2015 e não é mais usada.

- **LLL:** Linguagem semelhante ao *Low-Level Lisp-Like*, daí o nome **LLL**, também não é mais usada.
- **Serpent:** Esta é uma linguagem simples e limpa parecida com Python. Ela não é mais usado para desenvolvimento de contratos e não é suportado pela comunidade.
- Leia mais sobre Solidity e Recursos de Desenvolvimento de **DApps** em **DAPP DEVELOPMENT FRAMEWORKS**<sup>1</sup>

---

<sup>1</sup><http://ethdocs.org/en/latest/contracts-and-transactions/developer-tools.html#developer-tools>

- O compilador **Solidity** (solc)
- Compilador usado para compilar código de contratos inteligentes e converter eles para *bytecode*.



- **Ganache**
  - Simula um Blockchain Ethereum pessoal com uma interface com usuário (UI), comumente usada no desenvolvimento e testes.
- **Ganache-cli**
  - Versão linha de comando do **Ganache** tem como pre-requisito **NodeJS**.

# Ferramentas e Bibliotecas ii

ACCOUNTS

BLOCKS

TRANSACTIONS

CONTRACTS

EVENTS

LOGS

SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK  
0

GAS PRICE  
20000000000

GAS LIMIT  
6721975

HARDFORK  
PETERSBURG

NETWORK ID  
5777

RPC SERVER  
HTTP://127.0.0.1:7545

MINING STATUS  
AUTOMINING

WORKSPACE  
JAZZY-GIRL

SWITCH

MNEMONIC

kick abstract strong shrug forward enlist puppy reunion elephant hip suffer base

HD PATH  
m/44'/60'/0'/0/account\_index

ADDRESS 0x2366e9848803cB00CB82E6E6De3F6D17C4AA9ADA	BALANCE 100.00 ETH	TX COUNT 0	INDEX 0	
ADDRESS 0x6805940005154aEdfe6a00A39C588ED668E64D9D	BALANCE 100.00 ETH	TX COUNT 0	INDEX 1	
ADDRESS 0x56C21294F4e17dF32486b3d4D12E72D023861edF	BALANCE 100.00 ETH	TX COUNT 0	INDEX 2	
ADDRESS 0xAfACDB553412071bB538E36d57ED92d6745C5f94	BALANCE 100.00 ETH	TX COUNT 0	INDEX 3	
ADDRESS 0x694AE93a42C43B8E3d10c3F6769ADf2566C1863B	BALANCE 100.00 ETH	TX COUNT 0	INDEX 4	

- **Truffle**
  - Framework de desenvolvimento para *Ethereum* com recursos para implantação, teste e depuração.

# Frameworks ii

Truffle v5.1.11 – a development framework for Ethereum

Usage: truffle <command> [options]

## Commands:

build	Execute build pipeline (if configuration present)
compile	Compile contract source files
config	Set user-level configuration options
console	Run a console with contract abstractions and commands available
create	Helper to create new contracts, migrations and tests
debug	Interactively debug any transaction on the blockchain (experimental)
deploy	(alias for migrate)
develop	Open a console with a local development blockchain
exec	Execute a JS module within this Truffle environment
help	List all commands or provide information about a specific command
init	Initialize new and empty Ethereum project
install	Install a package from the Ethereum Package Registry
migrate	Run migrations to deploy contracts
networks	Show addresses for deployed contracts on each network
obtain	Fetch and cache a specified compiler
opcode	Print the compiled opcodes for a given contract
publish	Publish a package to the Ethereum Package Registry
run	Run a third-party command
test	Run JavaScript and Solidity tests
unbox	Download a Truffle Box, a pre-built Truffle project
version	Show version number and exit
watch	Watch filesystem for changes and rebuild the project automatically

See more at <http://truffleframework.com/docs>

- **Drizzle**
  - Um conjunto de bibliotecas de *frontend* para o desenvolvimento de interfaces *web*.
  - Torna o desenvolvimento *frontend* para **DApps** fácil.
  - Tem o NodeJS como pré-requisito.
  - Baseado no *Redux store*.
  - Mantém uma biblioteca de componentes **React**.

- **Embark:** powerful developer platform for building and deploying DApps
- **Brownie:** framework for Ethereum smart contract development and testing
- **Waffle:** another framework for smart contract development and testing
- **Etherlime:** framework that allows DApp development, debugging, testing, and testing in Solidity and Vyper
- **OpenZeppelin:** a toolkit for smart contract development

- A escrita de contratos inteligentes é basicamente a escrita de código fonte do contrato em **Solidity** em um editor de texto.
- Existem vários *plugins* e extensões disponíveis para os editores mais comuns, tais como Vim, Atom, VSCode, que fornecem *syntax highlighting* e formatadores para código fonte **Solidity**.

# Desenvolvimento e Implantação ii

```
1  pragma solidity ^0.4.0;
2  contract PatentIdea {
3      mapping (bytes32 => bool) private hashes;
4      bool alreadyStored;
5      event IdeaHashed(bool);
6
7      function saveHash(bytes32 hash) private {
8          hashes[hash] = true;
9      }
10     function SaveIdeahash(string idea) public returns (bool){
11         var hashedIdea = HashtheIdea(idea);
12         if (alreadyHashed(HashtheIdea(idea))) {
13             alreadyStored=true;
14             IdeaHash(false);
15             return IdeaHashed (event in PatentIdea) IdeaHashed(bool)
16         } else {
17             SaveIdeahash
18             saveHash(hashedIdea);
19             ideahashed(true);
20         }
21     }
22
23     function alreadyHashed(bytes32 hash) constant private returns(bool) {
24         return hashes[hash];
25     }
26
27     function isAlreadyHashed(string idea) constant public returns (bool) {
28         var hashedIdea = HashtheIdea(idea);
29         return alreadyHashed(hashedIdea);
30     }
31
32     function HashtheIdea(string idea) pure private returns (bytes32) {
33         return keccak256(idea);
34     }
35
36 }
```

Ln 14, Col 13 Spaces: 2 UTF-8 LF Solidity



# The layout of a Solidity source code file i

```
1  pragma solidity ^0.5.0; //specify the solidity compiler version
2  /*
3   this is a simple value checker contract that checks the value provided
4   and returns boolean value (true or false) based on the condition expression
5   evaluation
6   */
7  import "./mapping.sol"; //import a file
8  contract valuechecker {
9      uint price = 10;
10     //price variable declared and initialized with a value of 10
11     event valueEvent(bool returnValue);
12     function Matcher (uint8 x) public returns (bool) {
13         if (x >= price )
14         {
15             emit valueEvent(true);
16             return true;
17         }
18     }
19 }
```

- Uma Linguagem de Domínio Específico (DSL)
- *Contract-oriented language*
- JavaScript / C-like
- Amplamente utilizada
- Estaticamente Tipada



## Leitura Recomendada

### Capítulo 14: Development Tools and Frameworks

**Livro:** IMRAN BASHIR. Mastering Blockchain : Distributed Ledger Technology, Decentralization, and Smart Contracts Explained, 2nd Edition.

## Próximas Aulas

---

- Ambientes de Desenvolvimento e Ferramentas.

## Referências

---

Imran, Bashir. 2018. *Mastering Blockchain : Distributed Ledger Technology, Decentralization, and Smart Contracts Explained, 2nd Edition*. Packt Publishing. <https://search.ebscohost.com/login.aspx?direct=true&db=e000xww&AN=1789486&lang=pt-br&site=eds-live&scope=site>.