Aula 22 - Ethereum

Ferramentas de Desenvolvimento e Frameworks

Prof. Rogério Aparecido Gonçalves[1]
rogerioag@utfpr.edu.br

[1]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

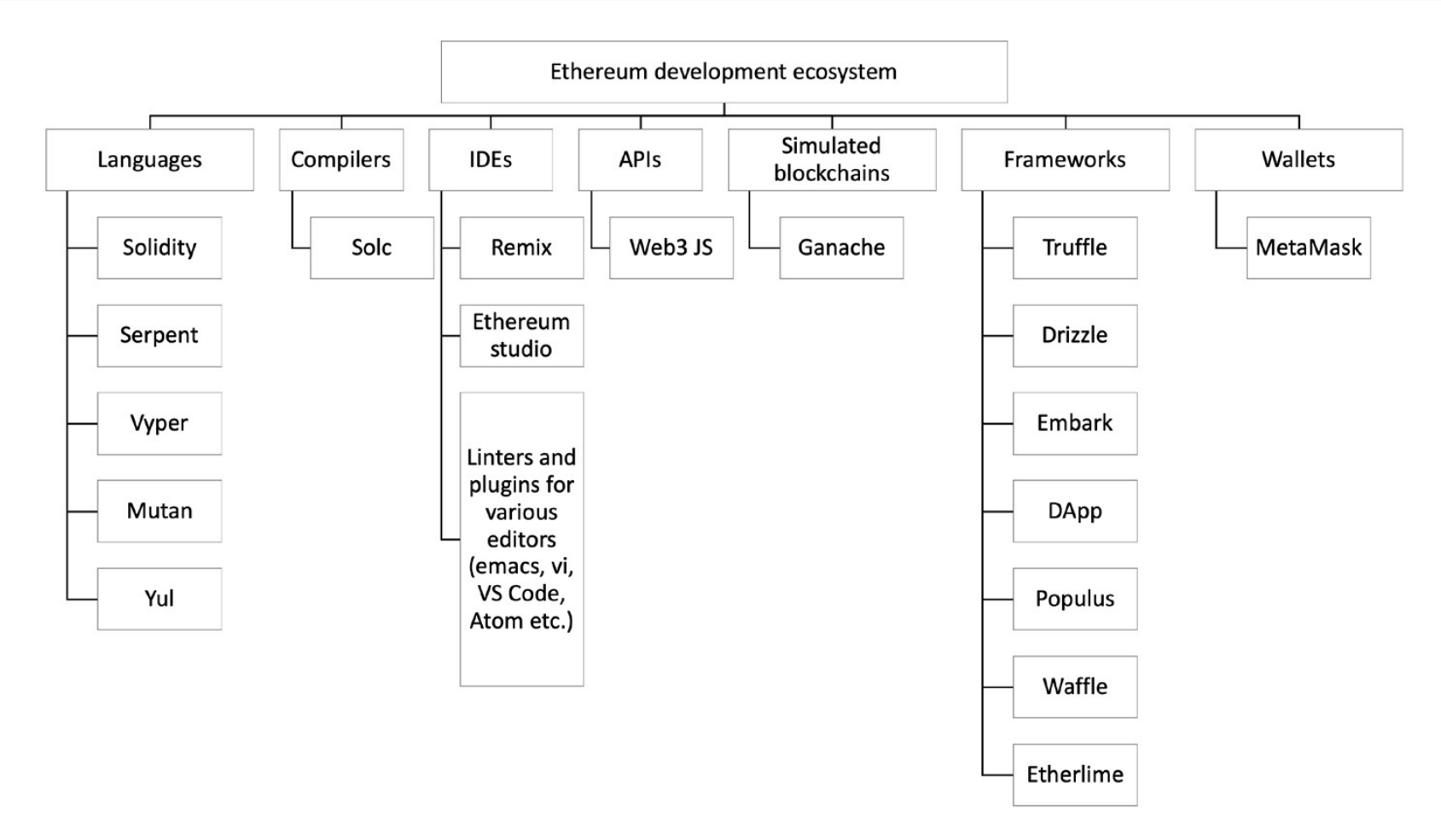
Agenda i

Introdução

Objetivos

- 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



Linguagens 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.

Linguagens ii

- **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[1]

[1] http://ethdocs.org/en/latest/contracts-and-transactions/developer-tools.html#developer-tools

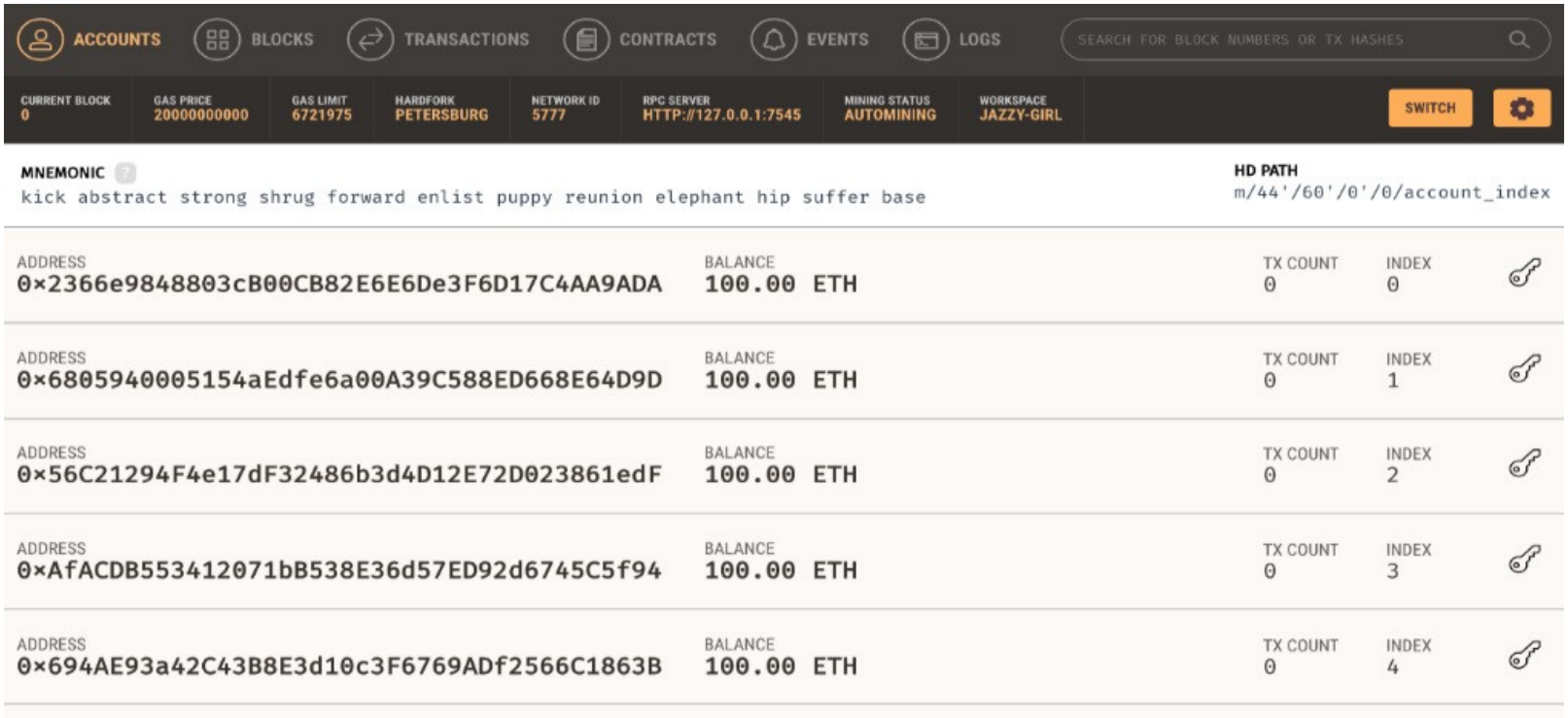
Compiladores i

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

Ferramentas e Bibliotecas i

- `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



Frameworks i

- 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
```

Frameworks iii

- 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`.

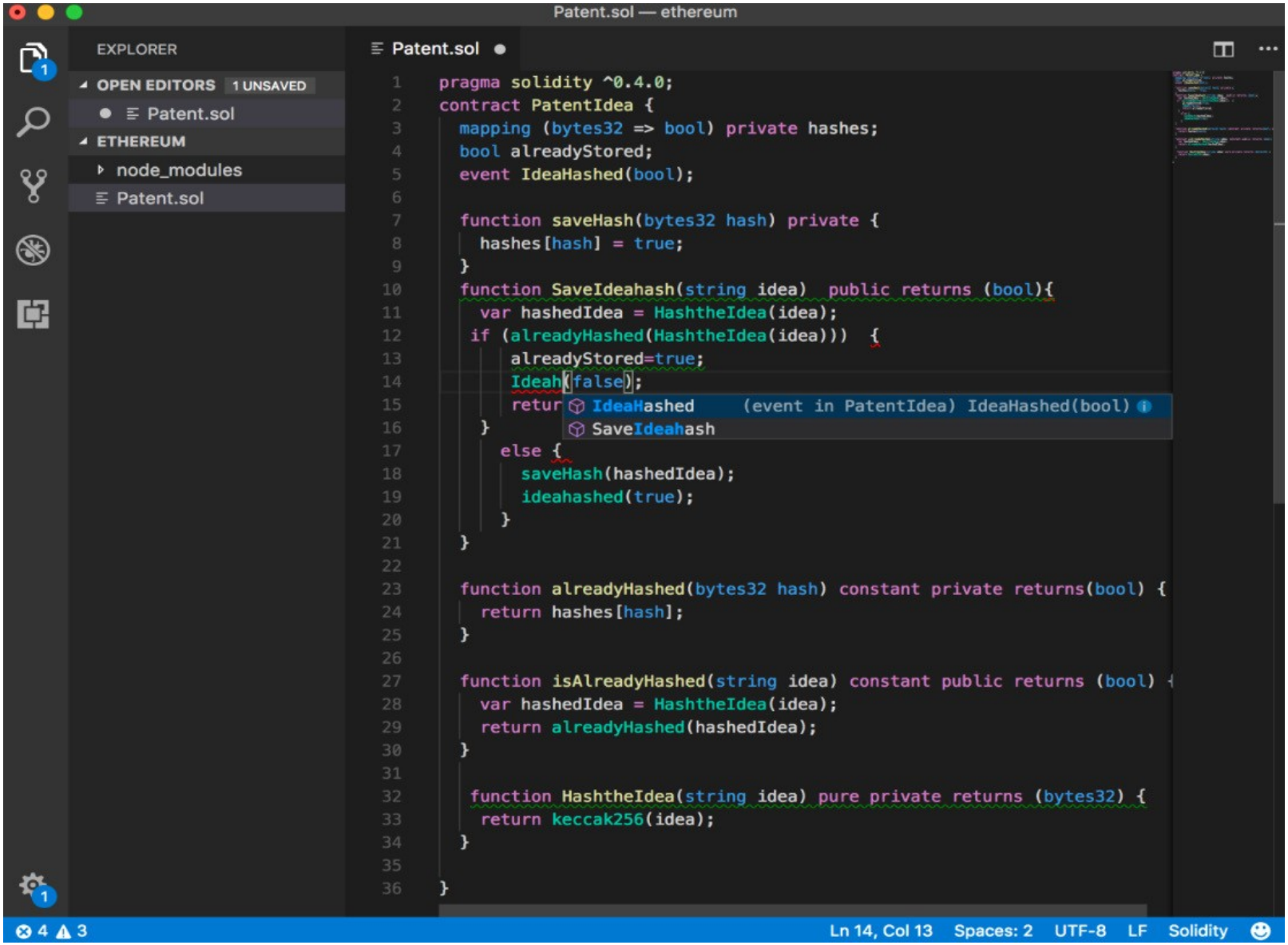
Outras Ferramentas i

- **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

Desenvolvimeto e Implantação i

- 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`.

Desenvolvimeto e Implantação ii



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 }
```

Linguagem Solidity i

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

Linguagem Solidity i

Leitura Recomendada

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

Próximas Aulas

- Ambientes de Desenvolvimento e Ferramentas.

Referências

Referências i

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.