Projeto de Implementação: Implementando Tokens no Ethereum

Murilo Cadedo Fiorin[1]
[1]Departamento Acadêmico de Computação (DACOM)
Universidade Tecnológica Federal do Paraná (UTFPR)

Abstract

This article delves into the implementation of smart contracts and tokenization on the Ethereum platform, emphasizing tools such as Truffle, Ganache, and test networks like Goerli. Tokenization, the central concept of this study, enables the digital representation of fungible and non-fungible assets through standards like ERC-20 and ERC-721. Based on modern development practices, the study outlines the process of creating, testing, and deploying smart contracts, illustrating functionalities such as balance inquiries. Highlights blockchain's potential to revolutionize industries by fostering decentralization, security, and accessibility in digital applications. This work serves as a practical guide for developers, providing a comprehensive introduction to the Ethereum ecosystem.

Resumo

Este artigo explora a implementação de contratos inteligentes e tokenização na plataforma Ethereum, destacando ferramentas como Truffle, Ganache e Remix. A tokenização, conceito central deste estudo, permite a representação digital de ativos fungíveis e não fungíveis por meio de padrões como ERC-20 e ERC-721. Com base em práticas modernas de desenvolvimento, o estudo aborda a criação, teste e implantação de contratos inteligentes, exemplificando funcionalidades como a consulta de saldos. Ressalta o potencial da blockchain para transformar setores inteiros, promovendo descentralização, segurança e acessibilidade em aplicações digitais. O estudo serve como um guia prático para desenvolvedores, oferecendo uma introdução abrangente ao ecossistema Ethereum.

1 Introdução

As tecnologias blockchain revolucionaram diversos setores ao introduzir um sistema distribuído e descentralizado que garante transparência, segurança e imutabilidade. Inicialmente popularizadas por moedas digitais como Bitcoin, essas tecnologias evoluíram para suportar aplicações mais complexas, como contratos inteligentes e tokenização. Este artigo foca no *Ethereum*, uma plataforma que vai além das transações financeiras e oferece uma máquina virtual robusta para execução de contratos inteligentes. A tokenização, por sua vez, é o processo de representação de ativos reais ou digitais por meio de tokens em um *blockchain*, permitindo maior liquidez, divisibilidade e acessibilidade a bens tradicionalmente ilíquidos.

Corresponding author(s):

2 Ethereum

Ethereum, segundo Antonopoulos & Wood (2018), é uma plataforma descentralizada que utiliza a *Ethereum Virtual Machine* (EVM), permitindo a execução de contratos inteligentes e dApps (aplicativos descentralizados). A EVM é responsável por executar códigos baseados em *Solidity*, garantindo que os contratos funcionem de maneira consistente em toda a rede. *Ethereum* possui uma rede principal (*mainnet*) para transações reais e diversas redes de teste (como *Ropsten, Kovan* e *Rinkeby*) que são usadas para desenvolvimento e validação de projetos antes da implementação final. Essas redes são fundamentais para desenvolvedores, reduzindo riscos e custos de erros na implantação.

3 Contratos Inteligentes

Introduzidos por Szabo (1997), contratos inteligentes são programas autoexecutáveis que armazenam e processam regras contratuais diretamente em um *blockchain*. Eles permitem a automação de acordos entre partes, eliminando intermediários e reduzindo custos operacionais. A utilidade dos contratos inteligentes é ampla, abrangendo desde sistemas financeiros até a tokenização de ativos, garantindo segurança, transparência e descentralização.

4 Solidity e implementação de Contratos Inteligentes

De acordo com , *Solidity* é a principal linguagem de programação usada para desenvolver contratos inteligentes na plataforma *Ethereum*. Inspirada por linguagens como *JavaScript* e *C++*, é uma linguagem orientada a contratos que facilita a implementação de funcionalidades como transferência de tokens e controle de acesso. A implementação de contratos inteligentes envolve definir a lógica em *Solidity*, compilar o código para *bytecode* compatível com a EVM e implantá-lo na *blockchain*. Ferramentas como *Truffle* e *OpenZeppelin* simplificam esse processo, fornecendo bibliotecas e padrões testados para desenvolvimento seguro.

5 Tokenização

Antonopoulos & Wood (2018) descreve tokens como representações digitais de ativos que residem em um *blockchain*. Eles podem ser fungíveis (como o padrão ERC-20, que representa moedas digitais ou pontos de recompensa) ou não fungíveis (ERC-721, usados em itens únicos como obras de arte digitais). A tokenização oferece diversas vantagens, como segurança aprimorada, maior liquidez e divisibilidade de ativos físicos ou intangíveis. *Ethereum* se destaca nesse contexto por suportar múltiplos padrões de token, como ERC-20 e ERC-721, que definem regras para interoperabilidade e uso consistente. Aplicações de tokenização incluem financiamento coletivo (ICOs), propriedade fracionada e DeFi (finanças descentralizadas).

6 Ferramentas de Desenvolvimento

O *Ganache* é uma ferramenta que simula uma *blockchain Ethereum* local, permitindo que desenvolvedores testem contratos inteligentes em um ambiente controlado. Ele cria uma rede local que não exige custos com gas, utilizando ETH falso para realizar transações. O *Ganache* pode ser usado via interface gráfica (GUI), que exibe informações detalhadas sobre transações, blocos e consumo de gas, ou via linha de comando (CLI) para integração em fluxos de trabalho automatizados. Essa ferramenta é ideal para depurar contratos, testar funcionalidades e simular interações complexas entre contas, fornecendo uma base sólida para o desenvolvimento local

antes da implantação em redes públicas.

O *Remix* é um IDE online altamente acessível, usado para escrever, depurar e implantar contratos inteligentes em *Solidity*. Sua interface baseada em navegador permite que desenvolvedores iniciem projetos rapidamente sem precisar configurar ambientes locais. O *Remix* inclui um depurador poderoso, que permite executar contratos linha por linha, e suporta plugins para análise de código, segurança e gerenciamento de dependências. Além disso, ele pode ser conectado a redes como *Goerli, Ropsten, mainnet* e até mesmo *Ganache*, tornando-o uma ferramenta versátil tanto para iniciantes quanto para desenvolvedores avançados que desejam desenvolver contratos de forma ágil.

O *Truffle* é um framework completo que facilita o gerenciamento de projetos de contratos inteligentes, desde a criação até a implantação. Ele organiza os contratos em um formato estruturado, automatiza o processo de compilação e migração para redes *Ethereum*, e oferece suporte a testes automatizados com *JavaScript* ou *Solidity*. O *Truffle* se integra perfeitamente com o *Ganache*, permitindo que desenvolvedores testem contratos em um ambiente local antes de implantar em redes de teste ou na *mainnet*. Além disso, sua capacidade de gerenciamento de migrações e interações com contratos o torna uma escolha popular para projetos de grande escala.

7 Especificação e Desenvolvimento do Projeto

O projeto foi realizado em uma máquina com Windows 10, A instalação do *Truffle* pode ser feita através do gerenciador de pacotes npm, que pode ser instalado conforme é explicado no site Instalação Node, como já estava instalado, apenas foi conferido as versões como demostrado abaixo.

```
PS C:\Users\MuriDani\Desktop\Mestrado\Tópicos de redes\projeto-token> node -v
v18.20.4
PS C:\Users\MuriDani\Desktop\Mestrado\Tópicos de redes\projeto-token> npm -v
10.7.0
PS C:\Users\MuriDani\Desktop\Mestrado\Tópicos de redes\projeto-token>
```

Código 1: Versão do Node.js e npm

Com o *Node.js* operante executei o comando `npm install -g truffle` para instalar o *truffle*. Abaixo tem o log da instalação.

```
PS C:\Users\MuriDani\Desktop\Mestrado\Tópicos de redes\projeto-token> npm
    install -g truffle
npm warn deprecated testrpc@0.0.1: testrpc has been renamed to ganache-cli,
    please use this package from now on.
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks
    memory. Do not use it. Check out lru-cache if you want a good and tested way
    to coalesce async requests by a key value, which is much more comprehensive
    and powerful.
npm warn deprecated mkdirp-promise@5.0.1: This package is broken and no longer
    maintained. 'mkdirp' itself supports promises now, please switch to that.
npm warn deprecated @truffle/source-map-utils@1.3.119: Package no longer
    supported. Contact Support at https://www.npmjs.com/support for more info.
npm warn deprecated rimraf@2.7.1: Rimraf versions prior to v4 are no longer
    supported
npm warn deprecated har-validator@5.1.5: this library is no longer supported
npm warn deprecated @truffle/promise-tracker@0.1.7: Package no longer supported.
     Contact Support at https://www.npmjs.com/support for more info.
npm warn deprecated @truffle/db-loader@0.2.36: Package no longer supported.
    Contact Support at https://www.npmjs.com/support for more info.
npm warn deprecated @truffle/error@0.2.2: Package no longer supported. Contact
    Support at https://www.npmjs.com/support for more info.
```

```
11 npm warn deprecated glob@7.2.0: Glob versions prior to v9 are no longer
       supported
12 npm warn deprecated apollo-datasource@3.3.2: The `apollo-datasource` package is
       part of Apollo Server v2 and v3, which are now end-of-life (as of October 22
       nd 2023 and October 22nd 2024, respectively). See https://www.apollographql.
       com/docs/apollo-server/previous-versions/ for more details.
13 npm warn deprecated apollo-server-errors@3.3.1: The `apollo-server-errors`
       package is part of Apollo Server v2 and v3, which are now end-of-life (as of
       October 22nd 2023 and October 22nd 2024, respectively). This package's
       functionality is now found in the `@apollo/server` package. See https://www.
       apollographql.com/docs/apollo-server/previous-versions/ for more details.
14 npm warn deprecated @truffle/provider@0.3.13: Package no longer supported.
       Contact Support at https://www.npmjs.com/support for more info.
15 npm warn deprecated apollo-server-plugin-base@3.7.2: The `apollo-server-plugin-
       base` package is part of Apollo Server v2 and v3, which are now end-of-life
16 (as of October 22nd 2023 and October 22nd 2024, respectively). This package's
       functionality is now found in the `@apollo/server` package. See https://www.
       apollographql.com/docs/apollo-server/previous-versions/ for more details.
17 npm warn deprecated apollo-server-types@3.8.0: The `apollo-server-types` package
        is part of Apollo Server v2 and v3, which are now end-of-life (as of October
        22nd 2023 and October 22nd 2024, respectively). This package's functionality
        is now found in the `@apollo/server` package. See https://www.apollographql.
       com/docs/apollo-server/previous-versions/ for more details.
18 npm warn deprecated @truffle/spinners@0.2.5: Package no longer supported.
       Contact Support at https://www.npmjs.com/support for more info.
19 npm warn deprecated @truffle/dashboard-message-bus-common@0.1.7: Package no
       longer supported. Contact Support at https://www.npmjs.com/support for more
       info.
20 npm warn deprecated apollo-server-express@3.13.0: The `apollo-server-express`
       package is part of Apollo Server v2 and v3, which are now end-of-life (as of
21 October 22nd 2023 and October 22nd 2024, respectively). This package's
       functionality is now found in the `@apollo/server` package. See https://www.
       apollographql.com/docs/apollo-server/previous-versions/ for more details.
22 npm warn deprecated apollo-server@3.13.0: The `apollo-server` package is part of
        Apollo Server v2 and v3, which are now end-of-life (as of October 22nd 2023
       and October 22nd 2024, respectively). This package's functionality is now
       found in the `@apollo/server` package. See https://www.apollographql.com/docs
       /apollo-server/previous-versions/ for more details.
23 npm warn deprecated multicodec@1.0.4: This module has been superseded by the
       multiformats module
24 npm warn deprecated uuid@2.0.1: Please upgrade  to version 7 or higher.  Older
       versions may use Math.random() in certain circumstances, which is known to be
        problematic.  See https://v8.dev/blog/math-random for details.
25 npm warn deprecated @truffle/events@0.1.25: Package no longer supported. Contact
        Support at https://www.npmjs.com/support for more info.
26 npm warn deprecated apollo-reporting-protobuf@3.4.0: The `apollo-reporting-
       protobuf` package is part of Apollo Server v2 and v3, which are now end-of-
       life
27 (as of October 22nd 2023 and October 22nd 2024, respectively). This package's
       functionality is now found in the `@apollo/usage-reporting-protobuf` package.
        See https://www.apollographql.com/docs/apollo-server/previous-versions/ for
       more details.
28 npm warn deprecated uuid@3.4.0: Please upgrade  to version 7 or higher.  Older
       versions may use Math.random() in certain circumstances, which is known to be
        problematic.  See https://v8.dev/blog/math-random for details.
29 npm warn deprecated request@2.88.2: request has been deprecated, see https://
       github.com/request/request/issues/3142
30 npm warn deprecated @truffle/code-utils@3.0.4: Package no longer supported.
       Contact Support at https://www.npmjs.com/support for more info.
31 npm warn deprecated multibase@0.6.1: This module has been superseded by the
```

```
    multiformats module
32  npm warn deprecated multibase@0.7.0: This module has been superseded by the
    multiformats module
33  npm warn deprecated apollo-server-env@4.2.1: The `apollo-server-env` package is
    part of Apollo Server v2 and v3, which are now end-of-life (as of October 22
    nd 2023 and October 22nd 2024, respectively). This package's functionality is
     now found in the `@apollo/utils.fetcher` package. See https://www.
    apollographql.com/docs/apollo-server/previous-versions/ for more details.
34  npm warn deprecated @truffle/config@1.3.61: Package no longer supported. Contact
     Support at https://www.npmjs.com/support for more info.
35  npm warn deprecated multicodec@0.5.7: This module has been superseded by the
    multiformats module
36  npm warn deprecated @truffle/dashboard-message-bus-client@0.1.12: Package no
    longer supported. Contact Support at https://www.npmjs.com/support for more
    info.
37  npm warn deprecated @truffle/compile-common@0.9.8: Package no longer supported.
    Contact Support at https://www.npmjs.com/support for more info.
38  npm warn deprecated @truffle/interface-adapter@0.5.37: Package no longer
    supported. Contact Support at https://www.npmjs.com/support for more info.
39  npm warn deprecated @truffle/abi-utils@1.0.3: Package no longer supported.
    Contact Support at https://www.npmjs.com/support for more info.
40  npm warn deprecated cids@0.7.5: This module has been superseded by the
    multiformats module
41  npm warn deprecated @ensdomains/ens@0.4.5: Please use @ensdomains/ens-contracts
42  npm warn deprecated @ensdomains/resolver@0.2.4: Please use @ensdomains/ens-
    contracts
43  npm warn deprecated @truffle/debugger@12.1.5: Package no longer supported.
    Contact Support at https://www.npmjs.com/support for more info.
44  npm warn deprecated apollo-server-core@3.13.0: The `apollo-server-core` package
    is part of Apollo Server v2 and v3, which are now end-of-life (as of October
    22nd 2023 and October 22nd 2024, respectively). This package's functionality
    is now found in the `@apollo/server` package. See https://www.apollographql.
    com/docs/apollo-server/previous-versions/ for more details.
45  npm warn deprecated @truffle/db@2.0.36: Package no longer supported. Contact
    Support at https://www.npmjs.com/support for more info.
46  npm warn deprecated @truffle/codec@0.17.3: Package no longer supported. Contact
    Support at https://www.npmjs.com/support for more info.
47
48  added 1159 packages in 4m
49  npm notice
50  npm notice New minor version of npm available! 10.7.0 -> 10.9.1
51  npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.9.1
52  npm notice To update run: npm install -g npm@10.9.1
53  npm notice
54  PS C:\Users\MuriDani\Desktop\Mestrado\Tópicos de redes\projeto-token> truffle --
    version
55  Truffle v5.11.5 (core: 5.11.5)
56  Ganache v7.9.1
57  Solidity v0.5.16 (solc-js)
58  Node v18.20.4
59  Web3.js v1.10.0
```

Código 2: Instalando o Truffle

Agora só falta instalar o *Ganache* para começar nosso projeto, ele está disponível no Instalação Ganache na versão CLI ou GUI, aqui foi usado o GUI como é demostrado na Figura 1.

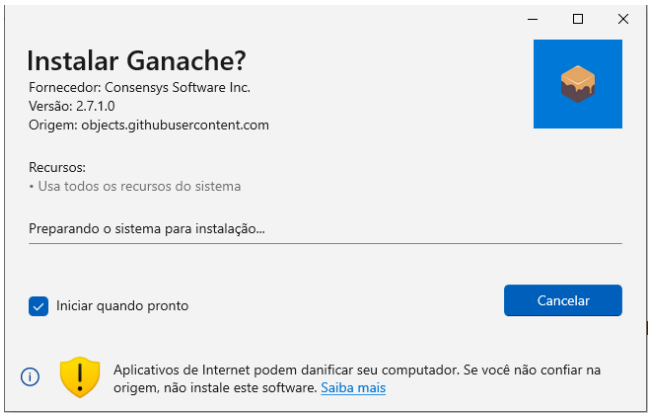

Figura 1: Instalando Ganache

Para começar um rede local com o *Ganache* é simples, basta clicar em *New Workspace* e será criado uma rede, 10 contas com saldo e gás para as transações, o resultado é mostrado na Figura 2.

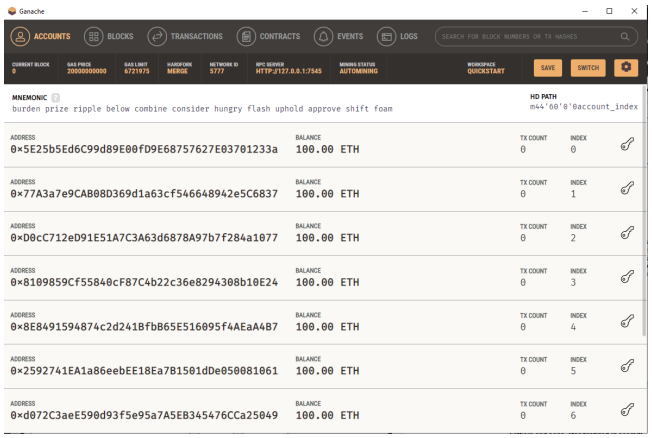

Figura 2: QuickStart Ganache

7.1 Iniciando Token

Para criar o nosso *Token* usaremos o *framewoek Truffle*, assim vamos criar um diretório e inicializar o projeto usando os seguintes comandos e responder como padrão para qualquer pergunta.

```
1  PS C:\Users\MuriDani\Desktop\Mestrado\Tópicos de redes\projeto-token> mkdir
       METoken
2
3
4      Diretório: C:\Users\MuriDani\Desktop\Mestrado\Tópicos de redes\projeto-token
5
6
7  Mode                 LastWriteTime         Length Name
8  ----                 -------------         ------ ----
9  d-----         27/11/2024     21:40                METoken
10
11
12 PS C:\Users\MuriDani\Desktop\Mestrado\Tópicos de redes\projeto-token\METoken>
       truffle init
13
14 Starting init...
15 ================
16
17 > Copying project files to C:\Users\MuriDani\Desktop\Mestrado\Tópicos de redes\
       projeto-token\METoken
```

```
18
19 Init successful, sweet!
20
21 Try our scaffold commands to get started:
22   $ truffle create contract YourContractName # scaffold a contract
23   $ truffle create test YourTestName         # scaffold a test
24
25 http://trufflesuite.com/docs
26
27 PS C:\Users\MuriDani\Desktop\Mestrado\Tópicos de redes\projeto-token\METoken>
       npm init
28 This utility will walk you through creating a package.json file.
29 It only covers the most common items, and tries to guess sensible defaults.
30
31 See `npm help init` for definitive documentation on these fields
32 and exactly what they do.
33
34 Use `npm install <pkg>` afterwards to install a package and
35 save it as a dependency in the package.json file.
36
37 Press ^C at any time to quit.
38 package name: (metoken) metoken
39 version: (1.0.0)
40 description:
41 entry point: (truffle-config.js)
42 test command:
43 git repository:
44 keywords:
45 author:
46 license: (ISC)
47 About to write to C:\Users\MuriDani\Desktop\Mestrado\Tópicos de redes\projeto-
       token\METoken\package.json:
48
49 {
50   "name": "metoken",
51   "version": "1.0.0",
52   "main": "truffle-config.js",
53   "directories": {
54     "test": "test"
55   },
56   "scripts": {
57     "test": "echo \"Error: no test specified\" && exit 1"
58   },
59   "author": "",
60   "license": "ISC",
61   "description": ""
62 }
63
64
65 Is this OK? (yes)
66
67 PS C:\Users\MuriDani\Desktop\Mestrado\Tópicos de redes\projeto-token\METoken>
```

Código 3: Iniciando projeto

Na Figura 3 pode ser visto a disposição das pastas e arquivos iniciais do projeto criado. O terminal e o Explorer sendo usados são do *Visual Studio Code*.

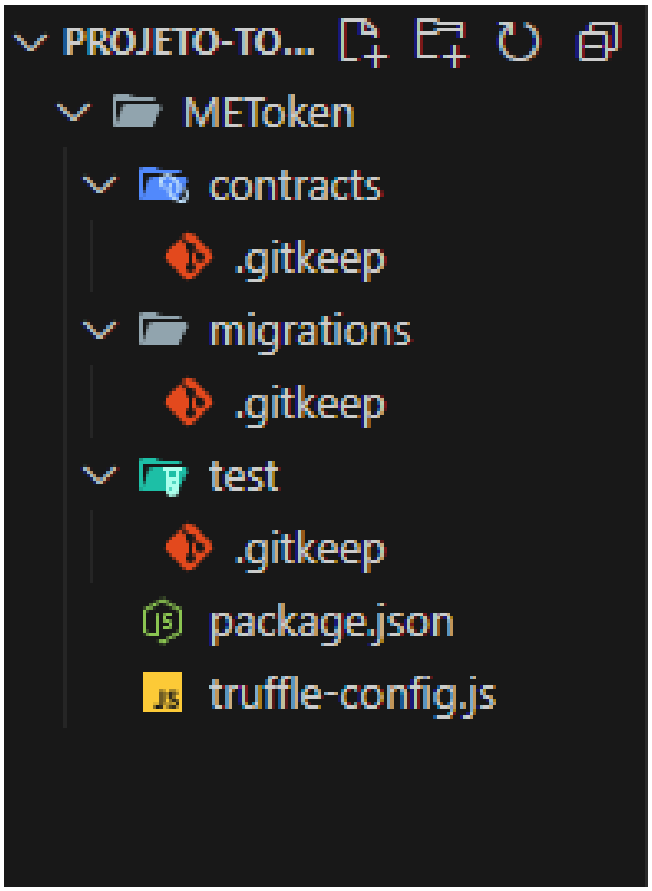

Figura 3: Diretório do projeto

Para facilitar a criação de tokens seguindo o padrão ERC-20, usaremos a biblioteca *OpenZeppelin*, que fornece contratos pré-implementados e seguros, basta seguir o código abaixo.

```
1 PS C:\Users\MuriDani\Desktop\Mestrado\Tópicos de redes\projeto-token\METoken>
      npm install @openzeppelin/contracts
2
3 added 1 package, and audited 2 packages in 3s
4
5 found 0 vulnerabilities
```

Código 4: Instalando OpenZeppelin

Agora criamos o arquivo METoken.sol no diretório `/contracts`. Com o seguinte código.

```
1 PS C:\Users\MuriDani\Desktop\Mestrado\Tópicos de redes\projeto-token\METoken>
      npm install @openzeppelin/contracts
2
3 added 1 package, and audited 2 packages in 3s
4
5 found 0 vulnerabilities
```

Código 5: METoken.sol

Para compilar o contrato criado executamos o comando `truffle compile`. Se bem sucedido como no log abaixo.

```
1 PS C:\Users\MuriDani\Desktop\Mestrado\Tópicos de redes\projeto-token\METoken>
      truffle compile
2
3 Compiling your contracts...
4 ===========================
5  Fetching solc version list from solc-bin. Attempt #1
```

```
 Downloading compiler. Attempt #1.
> Compiling @openzeppelin\contracts\interfaces\draft-IERC6093.sol
> Compiling @openzeppelin\contracts\token\ERC20\ERC20.sol
> Compiling @openzeppelin\contracts\token\ERC20\IERC20.sol
> Compiling @openzeppelin\contracts\token\ERC20\extensions\IERC20Metadata.sol
> Compiling @openzeppelin\contracts\utils\Context.sol
> Compiling .\contracts\METoken.sol
> Artifacts written to C:\Users\MuriDani\Desktop\Mestrado\Tópicos de redes\
    projeto-token\METoken\build\contracts
> Compiled successfully using:
   - solc: 0.8.21+commit.d9974bed.Emscripten.clang
```

Código 6: Compilando contrato

Outra forma de criar o token é com o *Remix*, que está disponível em http://remix.ethereum.org. Não irá mudar da criação do contrato, mas essa IDE online é fácil de usar e de alterar versões do *solidity*. Desta forma fica mais rápido, e não usaremos *Visual Studio Code* e *Truffle*. Para este modo usaremos Ganache-GUI para criar nossa rede local de teste e fazer *deploy* do token.

Abra o Remix criamos um arquivo chamado *MyERC20.sol* e colamos o código descrito Código 7, este código é disponibilizado por Gilad Haimov em ERC20 Token Tutorial.

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.2 <0.9.0;


contract ERC20Basic {

    string public constant name = "ERC20Basic";
    string public constant symbol = "BSC";
    uint8 public constant decimals = 18;


    event Approval(address indexed tokenOwner, address indexed spender, uint
       tokens);
    event Transfer(address indexed from, address indexed to, uint tokens);


    mapping(address => uint256) balances;

    mapping(address => mapping (address => uint256)) allowed;

    uint256 totalSupply_;

    using SafeMath for uint256;


   constructor(uint256 total) {
  totalSupply_ = total;
  balances[msg.sender] = totalSupply_;
    }

    function totalSupply() public view returns (uint256) {
  return totalSupply_;
    }

    function balanceOf(address tokenOwner) public view returns (uint) {
        return balances[tokenOwner];
    }

    function transfer(address receiver, uint numTokens) public returns (bool) {
```

```
39          require(numTokens <= balances[msg.sender]);
40          balances[msg.sender] = balances[msg.sender].sub(numTokens);
41          balances[receiver] = balances[receiver].add(numTokens);
42          emit Transfer(msg.sender, receiver, numTokens);
43          return true;
44      }
45
46      function approve(address delegate, uint numTokens) public returns (bool) {
47          allowed[msg.sender][delegate] = numTokens;
48          emit Approval(msg.sender, delegate, numTokens);
49          return true;
50      }
51
52      function allowance(address owner, address delegate) public view returns (
            uint) {
53          return allowed[owner][delegate];
54      }
55
56      function transferFrom(address owner, address buyer, uint numTokens) public
            returns (bool) {
57          require(numTokens <= balances[owner]);
58          require(numTokens <= allowed[owner][msg.sender]);
59
60          balances[owner] = balances[owner].sub(numTokens);
61          allowed[owner][msg.sender] = allowed[owner][msg.sender].sub(numTokens);
62          balances[buyer] = balances[buyer].add(numTokens);
63          emit Transfer(owner, buyer, numTokens);
64          return true;
65      }
66  }
67
68  library SafeMath {
69      function sub(uint256 a, uint256 b) internal pure returns (uint256) {
70        assert(b <= a);
71        return a - b;
72      }
73
74      function add(uint256 a, uint256 b) internal pure returns (uint256) {
75        uint256 c = a + b;
76        assert(c >= a);
77        return c;
78      }
79  }
```

Código 7: Contrato Remix

Selecione o compilador na versão 0.8.26 e nas configurações avançadas *EVM version* para Istanbul, então compile para verificar se está tudo certo, caso esteja irá aparecer um *check* no ícone do compilador, conforme a Figura 4.

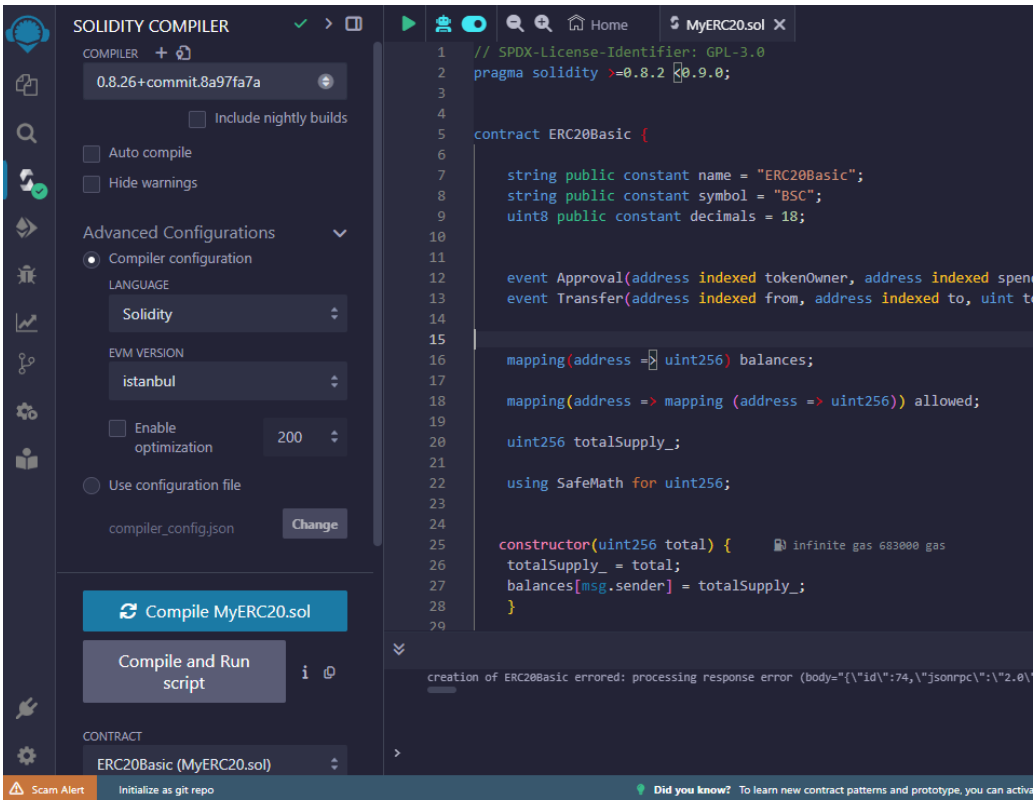

Figura 4: Compilador Remix

Vamos detalhar um pouco melhor o código, primeiramente definimos o compilador de a versões que podem ser usadas com **pragma solidity >=0.8.2 <0.9.0**.

Então criamos o contrato com **contract MyERC20Token** , com o comando **mapping** definimos os mapas usados no contrato inteligente e criamos outras variáveis para nome, simbolo, decimais e quantidade de total na carteira.

Temos evento de transferência e aprovação. Em seguida o **contructor** que é executado quando o contrato é criado.

Então teremos funções que são executadas nas transações sendo, **totalSupply** retorna o valor total de tokens, **balanceOf** retorna o saldo de tokens do endereço, **allowance** retorna o valor restante de tokens, **transfer** envia tokens para um endereço, **approve** aprova uma transferência e **transferfrom** transfere um valor de um endereço para outro.

Agora sabendo tudo o que tem no contrato, vamos fazer deploy dele. Abra o Ganache e inicie uma rede com **New Workspace**, na aba **CHAIN** selecione em **HARDFORK** a opção **Istanbul** e realize o *start* da rede, será criado uma rede e 10 contas com saldo, assim como fizemos no teste anterior. Agora no Remix selecione a aba **Deploy and run** e selecione *Ganache Provider* que será sua rede e insira o IP mostrado na tela do *Ganache*.

Pode conferir que a conta seleciona no Remix é a sua primeira conta no Ganache também, agora basta inserir o número de token que será adicionado a conta realizar o deploy do seu contrato, como é demonstrado Figura 5 dentro do Remix e na Figura 6 nas transações pelo Ganache.

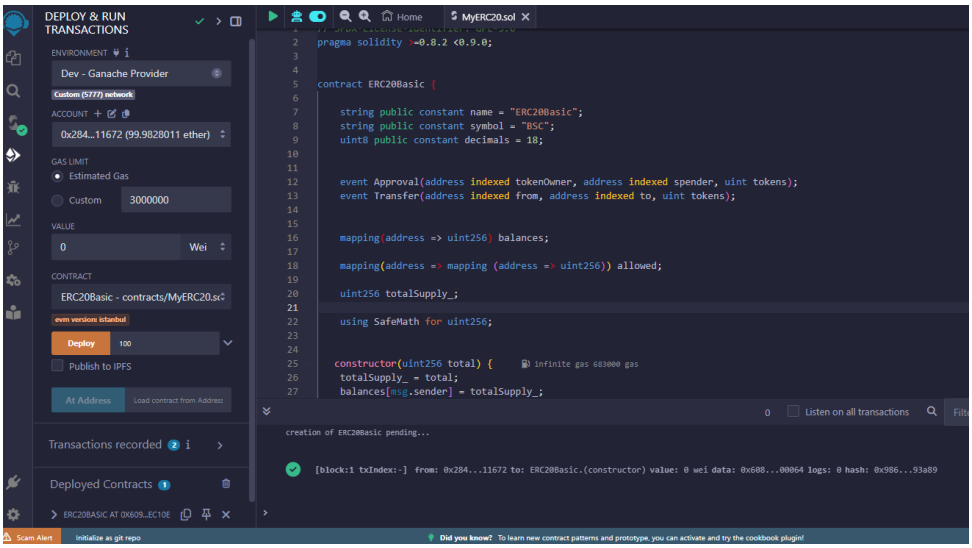

Figura 5: Deploy Remix

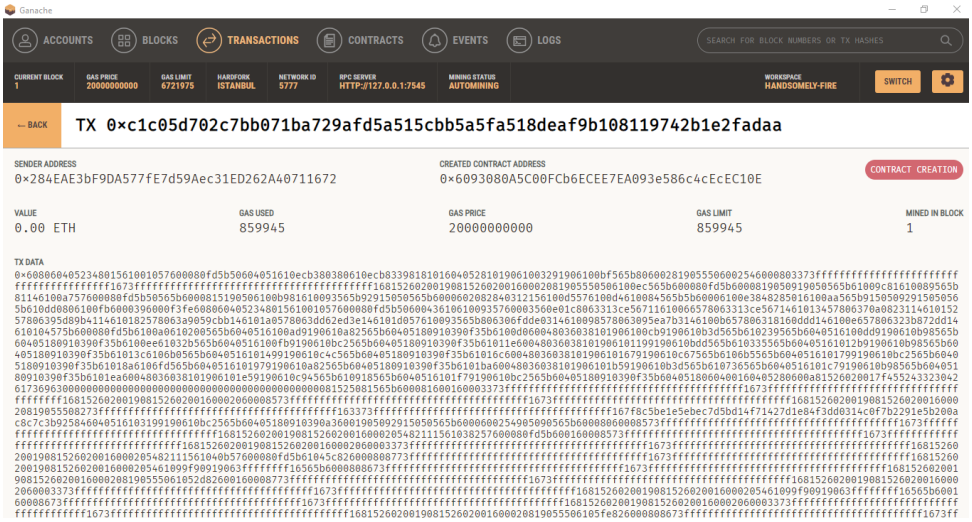

Figura 6: Deploy do contrato visto no Ganache

Na Figura 7, podemos observar que agora nos *Deployed contracts* temos as funções comentadas anteriormente e ao clicar executa-las. Assim, com possibilidade de requisitar nome e valores guardados no contrato e até realizar transações de token.

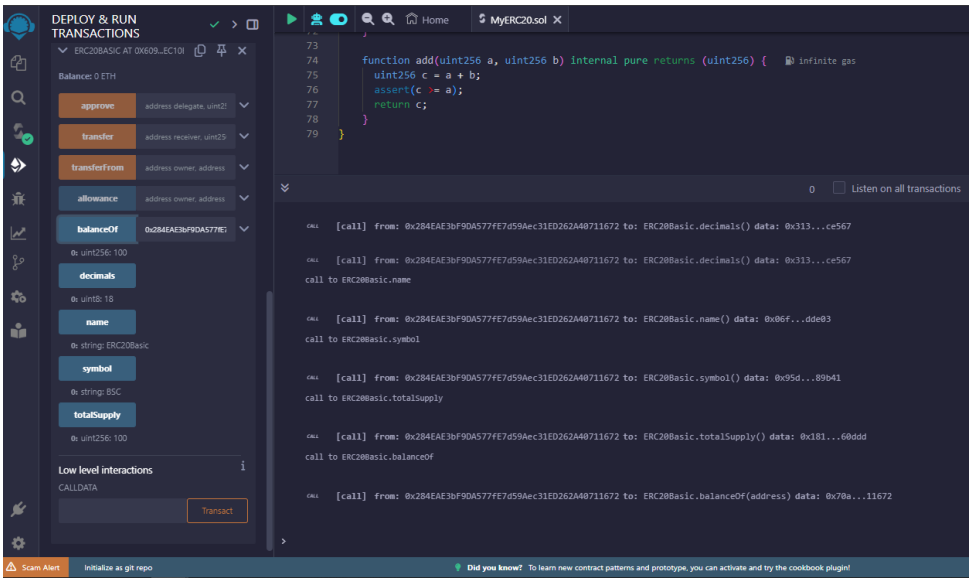

Figura 7: Funções do contrato

8 Considerações Finais

O desenvolvimento de contratos inteligentes e a tokenização representam um marco no avanço das tecnologias *blockchain*, permitindo a criação de sistemas descentralizados, transparentes e altamente seguros. Durante esta discussão, exploramos desde os conceitos fundamentais de *tokenização* até a implementação prática de contratos inteligentes utilizando ferramentas como *Truffle, Ganache e Remix*.

A tokenização, como abordada, redefine a maneira como representamos e transacionamos ativos, sejam eles fungíveis, como os tokens ERC-20, ou não fungíveis, como os ERC-721. Esses padrões garantem interoperabilidade e facilidade de uso em um ecossistema cada vez mais global e diversificado. A flexibilidade dos tokens permite sua aplicação em inúmeros contextos.

Referências

Antonopoulos, A.M. & G. Wood. 2018. *Mastering ethereum: Building smart contracts and dapps*. O'Reilly Media, Incorporated. https://books.google.com.br/books?id=SedSMQAACAAJ.

Szabo, Nick. 1997. Formalizing and securing relationships on public networks. *First Monday* 2(9). http://dblp.uni-trier.de/db/journals/firstmonday/firstmonday2.html#Szabo97.