Membros da Equipe:

Danilo Rafael de Lima Cabral

Henrique Ferraz Arcoverde

Título do Projeto: Paradigma Orientado a Contratos

Descrição:

A ideia principal do projeto é estender a linguagem orientada a objetos 1 para o paradigma orientado a contratos. Esse paradigma foi proposto por linguagens de programação projetadas para a criação de contratos inteligentes em blockchain.

Blockchain pode ser considerado, de forma simplificada, como um banco de dados descentralizado, sendo inicialmente proposta em 2009 como a tecnologia por trás do funcionamento do Bitcoin.

Quando o Bitcoin foi lançado (juntamente com a sua blockchain), ele trouxe consigo o conceito de Bitcoin script, que é uma linguagem de programação baseada em pilha e projetada basicamente para prover condições de pagamento entre endereços da rede Bitcoin, derivando-se daí os chamados contratos inteligentes.

Apesar da revolução trazida pelo Bitcoin, foi o Ethereum que popularizou o conceito de contratos inteligentes, ao integrar a sua blockchain com uma máquina virtual capaz de executar uma linguagem turing completa, chamada de Solidity.

O Solidity utiliza uma especialização do paradigma orientado a objetos, uma vez que ele não permite a declaração de classes genericamente. Ao invés disso, nessa linguagem, o conceito de classes foi especializado para contratos.

Um contrato pode ser declarado e instanciado como uma classe, porém conta com algumas propriedades e métodos nativos. Algumas de suas principais particularidades são:

1 – Assim como os usuários do Ethereum, todo contrato é associado a um endereço na blockchain.

2 – As propriedades (ou atributos) de um contrato inteligente são chamadas de variáveis de estado e sempre têm os seus valores armazenados na blockchain.

3 – Todo contrato possui uma propriedade nativa responsável armazenar o saldo (em ether) pertencente a ele.

4 – Os métodos de um contrato podem ter um modificador chamado “payable”, tornando obrigatório o envio de algum valor de ether sempre que for chamado.

5 – Todo contrato possui um método nativo responsável por transferir saldo do contrato para algum endereço de usuário na blockchain, passado como parâmetro.

Em nossa proposta, pretendemos estender a linguagem orientada a objetos 1 da seguinte forma:

1 – O maior nível de abstração da linguagem será um contrato (que funcionará como um tipo de classe estática e especializada).

2 – Um contrato será declarado com o comando **contract**.

3 – Um usuário será um tipo nativo da linguagem e poderá ser declarado, através do comando **user**, seguinte sintaxe: **user** nome do usuário **=** valor.

4 – Todo contrato terá um atributo chamado **balance**, que armazenará o seu saldo.

5 – Um contrato poderá ter dois tipos de métodos: nativos e definidos pelo usuario.

6 – Todo contrato terá um método nativo chamado **transfer**(user, valor), o qual será responsável por enviar um valor para o usuário passado como parâmetro (caso o contrato possua tal valor).

7 – Qualquer método definido pelo usuário poderá ser declarado com o modificador **payable**. Se um método for payable, ele deverá ser chamado com a seguinte sintaxe: **proc** nome do método **with** valor **by** usuário.

O fluxo para a declaração, instanciação e utilização de contratos deverá ser o seguinte:

1 – Primeiramente, um usuário deverá ser declarado com um determinado valor que será o seu saldo inicial.

2 – Após isso, um contrato deverá declarado com um valor passado para ele por um usuário válido.

3 – Uma vez que um contrato for declarado, ele funcionará como uma classe estática e não poderá ser instanciada.

4 – Um contrato receberá algum valor exclusivamente através de chamadas a algum método **payable**.

5 – Uma vez que um contrato possua algum saldo, será possível realizar o saque desse valor para algum usuário através do método **transfer**(user, valor).

Um exemplo de fluxo para a declaração, instanciação e utilização de contratos poderia ser o seguinte:

```
user Danilo = 100;

{ contract Banco {

        proc depositar() payable {

                // Aqui poderia ser declarada alguma funcionalidade ao contrato.

        }

        }

};

{ proc Banco.depositar() with 50 by Danilo; write(Danilo); write(Banco.balance); }
```