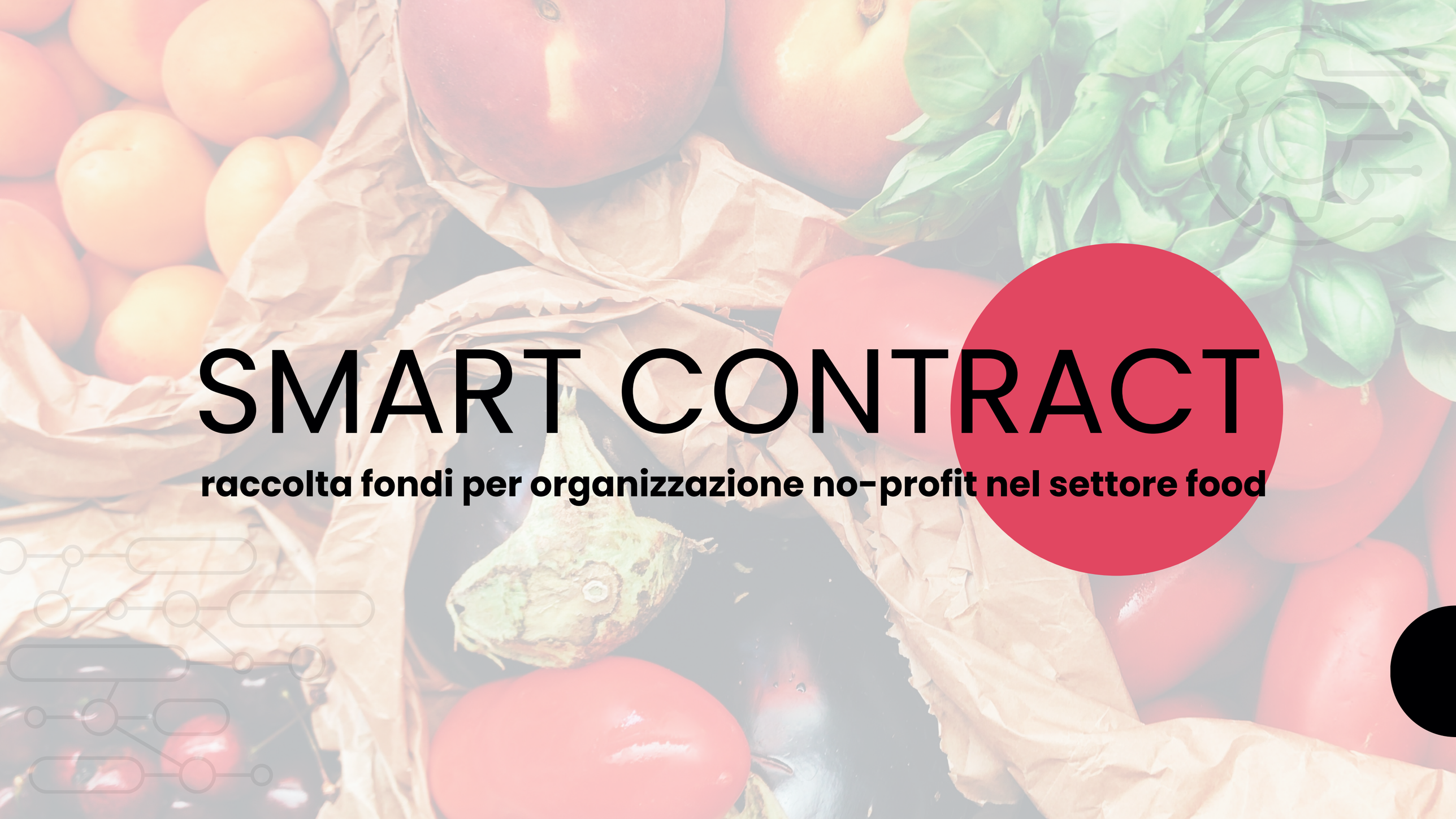


SMART CONTRACT

raccolta fondi per organizzazione no-profit nel settore food

INIZIALIZZAZIONE VARIABILI

In questa parte di codice sono state definite e inizializzate le variabili richieste:

- **saldo --> Rappresenta il saldo raccolto**
- **owner -> L'indirizzo del Manager della raccolta fondi**
- **goal -> l'obiettivo della raccolta**
- **donators -> i donatori che vengono inizializzati a 0**
- **closed -> se la raccolta fondi è chiusa questo risulta a true**

Inizializziamo nel costruttore l'owner (rappresnta il manager, ovvero il creatore del contratto) e il goal inserito moltiplicato per 1 ether in quanto andremo a calcolare i valori in ether. Ho creato una funzione apposta etherValue in quanto questo procedimento servirà anche in altre parti di codice dello smart contract.

Food/contracts/Founds.sol

```
/**
 * Contract **Founds**
 */
contract Founds {
    uint256 public saldo = 0;
    address payable owner;

    uint256 public goal;

    uint256 public donators = 0;

    // the closed variable determines if the collection is completed
    // - True --> closed
    // - False --> open
    bool closed = false;

    function etherValue(uint value) internal pure returns (uint){
        return (value*1 ether);
    }

    constructor(uint256 target) {    infinite gas 445800 gas
        owner = payable(msg.sender);
        goal = etherValue(target);
    }
```

```
modifier ownerCheck() {
    require(owner == msg.sender, "You're not the owner");
    _;
}
modifier saldoCheck(uint value) {
    require(saldo >= etherValue(value), "Saldo is not enough");
    _;
}

/// Function that allow the manager (only) to take an amount of money
function take(uint256 value) public payable ownerCheck saldoCheck(value) {
    uint ethers = etherValue(value);
    (bool sent, bytes memory _data) = owner.call{value: ethers}("");
    require(sent, "Failed sent");

    // decrease of saldo
    saldo -= ethers;

    // reopen the fund if the goal is less than the saldo
    if (!goalCheck()) {
        closed = false;
    }
}
```

PRELIEVI

La funzione take prende in input un valore intero che rappresenta quanti Ether vuole prelevare il manager. Controlla attraverso i modifiers ownerCheck e saldoCheck che il saldo sia maggiore o uguale del valore inserito e che la richiesta sia da parte del manager.

Inoltre, richiamando la funzione goalCheck verifica se il goal ora è minore del saldo allora riapre la raccolta fondi.

```
/// function that check and close the fund in case of reaching the goal
function closeCheck() internal {    28540 gas
    if (goalCheck()) {
        closed = true;
    }
}

/// function that check if the saldo is greater or equal to the goal
function goalCheck() internal view returns (bool) {    4232 gas
    return (saldo >= goal);
}
```

CHECK

La funzione goalCheck verifica se il saldo è maggiore o uguale al goal e restituisce un bool.

La funzione closeCheck utilizza la funzione goalCheck e, se restitutisce true, imposta la varibile closed a true e quindi chiude la raccolta.

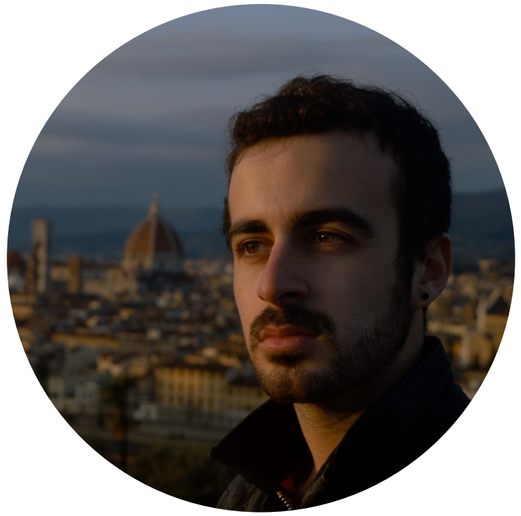
```
modifier goalReached() {
    require(closed == false, "Goal reached, the found is closed");
    _;
}

modifier positiveValue() {
    require(msg.value > 0, "Not money");
    _;
}


/// function used to donate an amount of money
function donate() public payable positiveValue goalReached {
    // Increment the saldo
    saldo += msg.value;

    // Increment donators
    donators++;

    // call the closeCheck function
    closeCheck();

}
```

DONAZIONI

La funzione donate viene utilizzata per effettuare le donazioni. Incrementa il saldo e i donatori. Infine, chiama la funzione closeCheck che chiude la raccolta se l'obiettivo è stato raggiunto.

Controlla attraverso i modifiers positiveValuee goalReached che il valore sia maggiore di 0 e che la raccolta sia aperta.

THANK YOU



Andrea Ballarini