CryptoKitties na Ethereum mreži
Tehnička dokumentacija
Verzija 1.0

Studentski tim: Ivan Mihaljević
Blaž Bagić
Karlo Vrbić
Mateo Gobin

Nastavnik: Federico Matteo Benčić

SADRŽAJ

1. Uvod

CryptoKitties je sustav za implementirati sustav uzgajanje i preprodaju virtualnih mačaka. U CryptoKitties sustavu, korisnici prikupljaju i uzgajaju virtualne mačke! Svaka mačka ima jedinstveni genom koji ju definira. Korisnici mogu uzgajati svoje mačke kako bi stvorili nove mačke koje se raylikuju od ostalih po svom genomu.

CryptoKitties je implementiran kao raspodijeljeni sustav zasnovan na *blockhain* tehnologiji. *Blockhain* je tehnologija pomoću koje implementirati raspodijeljene i decentralizirane sustave, a funkcionira kao lista zapisa, koji nazivamo *blokovima*, a koji su povezani jedni s drugima tako tvoreći jedan lanac blokova. Iako CryptoKitties nije digitalna valuta, nudi istu sigurnost: svaka je mačka jedinstvena i zna se točno njen vlasnik. Ne može se ponoviti, oduzeti ili uništiti.

2. Tehnologija

Sustav je implementiran korištenjem raznih tehnologija i knjižnica kao npr. *Ethereum*, *Truffle*, *JavaScript* i sl.

2.1. Ethereum

Ethereum je decentralizirana platforma koja ima mogućnost pokretanja tzv. *smart contract-a*, tj. aplikacija koje rade točno onako kako su programirane bez ikakve mogućnosti zastoja, cenzure, prijevare ili uplitanja trećih strana. Na taj način možemo grantirati sigurnost pri preprodaji i uzgajanju mačaka.

2.2. Truffle

Za razvoj i testiranje pametnih ugovora korišten je alat *Truffle* koji omogućava simuliranje *Ethereum* mreže lokalno na osobnom računalu. Ovaj alat nam omogućava kompilaciju pametnih ugovora, automatsko testiranje pomoću *JavaScript* knjižnica *Mocha* i *Chai*, skriptiran deployment i migracije i još mnogo funkcionalnosti koje olakšavaju razvoj pametnih ugovora na *Ethereum* mreži.

2.3. JavaScript

JavaScript je skriptni programski jezik koji CryptoKitties sustav koristi za migracije i testiranje pametnih ugovora. Za testiranje funkcionalnosti poput kreiranja, uzgajanja, prodaje ali i otkayivanje prodaje mačaka korištena je posebna *JavaScript* knjižnica imena *Mocha* i *Chai*. Konkretno *Mocha* je knjižnica koja nam omogućava asinkrono testiranje dok je knjižnica *Chai* tzv. *assertion library*, tj. pruža nam metode za provjeru ispravnosti određenih vrijednosti.

2.4. OpenZeppelin

OpenZeppelin je knjižnica za siguran razvoj pametnih ugovora. Ona osigurava implementaciju standarda kao što su ERC20 i ERC721, kao i komponente *Solidity* za izradu prilagođenih ugovora i složenijih decentraliziranih sustava.

3. Tehničke značajke

3.1. Kreiranje mačaka

Korisnicima se za početak mora pounditi određeni broj mačaka. U isječku koda 3.1 vidimo kako se to odvija. Metoda `_createKitty` je odgovorna za stvaranje mačke i emitiranje događaja `Birth` definiranog na način prikazan u odsječku.

3.2. Prodaja mačaka

Jedna od glavnih značajki ovog sustava je prodaja mačaka. Korisnici između sebe mogu prodavati mačke koje posjeduju za određenu cijenu. To podrazumijeva kupovanje mačaka sa tržišta, stavljanje mačke na tržište i povlačenje mačke sa tržišta ukoliko se korisnik predomislio. Pametni ugovor za tržište mačaka je prikazan na 3.3

3.3. Uzgajanje mačaka

Uzgajanje mačaka je značajka koja korisnicima pruža način da parenjem dviju mačaka dobiju dijete mačku. Kako mačke u sustavu nemaju spol nije bitno koje dvije mačke se pare. Pametni ugovor za parenje mačaka prikazan je na **??** isječku koda.

```
event Birth(address owner, uint256 geneticCode, uint32
    _parent1_id, uint32 _parent2_id, uint32 _id);

function _createKitty(
    address _owner,
    uint256 _geneticCode,
    uint32 _parent1_id,
    uint32 _parent2_id,
    uint16 _generation)
    internal
    returns (uint)
{
    Kitty memory _kitty = Kitty({
        geneticCode: _geneticCode,
        parent1_id: _parent1_id,
        parent2_id: _parent2_id,
        generation: _generation,
        birthTime: uint64(now)
    });
    uint32 _kittyId = uint32(kitties.push(_kitty) - 1);
    emit Birth(_owner, _geneticCode, _parent1_id, _parent2_id,
        _kittyId);
    return _kittyId;
}
```

Slika 3.1: Kod za kreiranje mačke

```
contract KittyMarket is KittyOwnership {
   mapping (uint32 => uint256) public kittyIndexToPrice;

   function putKittyUpForSale(uint32 kittyId, uint256 price)
       external {
       require(ownerOf(kittyId) == msg.sender);
       kittyIndexToPrice[kittyId] = price;
   }

   function cancelSale(uint32 kittyId) external {
     require(ownerOf(kittyId) == msg.sender);
     kittyIndexToPrice[kittyId] = 0;
   }

   function buyKitty(uint32 kittyId) external payable {
     uint256 price = kittyIndexToPrice[kittyId];
     address payable owner =
        address(uint160(ownerOf(kittyId)));

     require(owner != msg.sender);
     require(price > 0);
     require(msg.value >= price);

     kittyIndexToPrice[kittyId] = 0;
     _transferFrom(ownerOf(kittyId), msg.sender, kittyId);
     owner.transfer(msg.value);
   }
}
```

Slika 3.2: Pametni ugovor za tržište mačaka

```
contract KittyBreeding is KittyOwnership {

  function breedWith(uint32 parent1Id, uint32 parent2Id)
    public
    returns(uint)
  {
    require(parent1Id >= 0 && parent2Id >= 0);
    require(parent1Id != parent2Id);
    require(ownerOf(parent1Id) == msg.sender);
    require(ownerOf(parent2Id) == msg.sender);

    Kitty storage parent1 = kitties[parent1Id];
    Kitty storage parent2 = kitties[parent2Id];

    require(parent1.birthTime != 0 && parent2.birthTime != 0);

    uint16 generation = uint16(max(parent1.generation,
       parent2.generation) + 1);
    uint256 geneticCode =
       cantorPairingFunction(parent1.geneticCode,
       parent2.geneticCode);
    uint kittyId = _createKitty(msg.sender, geneticCode,
       parent1Id, parent2Id, generation);

    mint(msg.sender, uint256(kittyId));
    return kittyId;
  }

  function cantorPairingFunction(uint a, uint b) private pure
     returns (uint) {
    return (a+b)*(a+b+1)*b/2;
  }

  function max(uint a, uint b) private pure returns (uint) {
     return a > b ? a : b;
  }
}
```

Slika 3.3: Pametni ugovor za uzgajanje mačaka

4. Korištenje sustava

TODO: dodat manual za korištenje sustava

5. Zaključak

Zaključak.