Borbély Katalin - CDLEBS

Szigeti Soma - BQHM70

S15\_2

# Feladatkiírás

HW15: Virtual server rental

Problem statement

Implement a smart contract for virtual server rental.

The contract should have exactly one owner (the server's owner).

Clients can buy access to virtual servers for time slots of 30 minutes.

The owner sets the price per time slot.

Access to time slots can be transferred between clients.

Anyone can check if a client has access to the machine at any given time.

The owner can withdraw the profits.

Example scenario

Alice owns a server park. She creates an instance of this contract to manage access to one of her servers, setting the price per time slot to 0.01 ether.

Bob reserves the time slot between 11:00am and 11:30am this Sunday using the time slot's number since contract creation. Later, he decides to transfer his reservation to Claire. After 11:00am, Claire sends her address and a packet signed with her private key to the server. The server validates the packet, checks is she has access during the current time slot, and grants access to Claire.

Contract interface

constructor: function ReserveVS(uint256 pricePerSlot)

The creator of the contract (msg.sender) is the owner.

pricePerSlot is the price for renting the server for a single time slot.

reserve: function reserve(uint24 slotId) payable

Reserve the slotId'th time slot since contract creation. For example, if the contract was created today at 00:00, then slot number 16 would be the one from 07:30 to 08:00, while slot number 64 would be the same time slot on the next day.

This transaction succeeds only if the time slot has not yet been reserved, and the sender transfers the correct amount of ether.

transfer: function transfer(uint24 slotId, address to)

Transfer ownership of the slotId'th time slot to address to.

Succeeds only if the given time slot originally belonged to the sender of the transaction (msg.sender).

validate: function validate(address client) view returns (bool)

Returns true if client is entitled to use the machine at the time of the validation.

Can be called by anyone. (In our use case, this will be called by the server itself.)

withdraw: function withdraw()

Transfer profit to the sender.

Only the owner of the contract can call it.

Contract skeleton

pragma solidity ^0.4.21;

contract ReserveVS {

// <contract\_variables>

// </contract\_variables>

function ReserveVS(uint256 pricePerSlot) public {

// TODO

}

function reserve(uint24 slotId) public payable {

// TODO

}

function transfer(uint24 slotId, address to) public {

// TODO

}

function validate(address client) public view returns (bool) {

// TODO

}

function withdraw() public {

// TODO

}

}

# Okos szerződés funkciók

Egy szerverpark megvalósítása a cél. A szerverparkhoz tartozik egy gazda, akitől lehet bérelni félórás időtartamokban a szervereket. Egy időpontban csak egy szerver bérelhető. Egy szervert egyszerre csak egy ember tud bérelni/használni.

A szerver bérlésének feltétele: az adott időpontra ne legyen még foglalás. A bérlő a foglalás időpontjában rendelkezzen a foglaláshoz szükséges etherrel.

Egy már lefoglalt szerver átadható egy másik embernek. Csak olyan időpont adható át, melyet a bérlő foglalt le. A folyamatot a bérlő kezdeményezi. A lefoglalt szerver használati jogának átadásakor is az eredeti bérlőtől kerül levonásra az ether.

A szervert csak foglalás időtartalma alatt használhatja a bérlő és csak a bérlő.

A foglalás pillanatában a bérlés költsége zárolásra kerül a bérlő számláján. Melynek levonását és a saját számláján való jóváírását később a szerverpark gazdája kezdeményezheti.

Konstruktor: function ReserveVS(uint256 pricePerSlot)

A konstruktorban létrehozunk egy szerverparkot. A szerverpark gazdája a hívó(msg.sender). Szerverpark létrehozásához szükséges még megadni egy szerver egy időtartamra való bérlésének az árát.

Foglalás: function reserve(uint24 slotId) payable

Foglalás csak a szerverpark létrehozása után történhet. A foglalónak meg kell adnia a slotId-t azaz, hogy a szerverpark létrehozása óta melyik blokkban szeretne szervert bérelni. Minden blokk egy félórás idő intervallumot takar. Sikeres a foglalás, ha az adott intervallumot még nem foglalta le senki, illetve a bérlő rendelkezik a megfelelő mennyiségű etherrel. Egyszerre csak egy intervallumot lehet lefoglalni. Amennyiben több intervallumot szeretne a bérlő lefoglalni, úgy többször kell foglalást kezdeményeznie.

Sikertelen foglalás esetén hibaüzenettel kerül értesítésre a foglaló.

Sikeres foglalás esetén zárolásra kerül a bérlési díj és lefoglalásra kerül az intervallum a bérlőnek.

Hely átadás: function transfer(uint24 slotId, address to)

A bérlő átadhatja az általa lefoglalt időpontot egy másik embernek. Ehhez szükséges megadni az intervallumot és az új bérlő ID-ját. Egyszerre csak egy időpont adható át más embernek. Sikeres az átadás, ha a megadott idő intervallum az eredeti bérlőé. Ekkor az adott intervallum bérlőjét kicseréljük az új bérlőre.

Sikertelen átadás esetén hibaüzenettel értesítjük a bérlőt.

Validálás belépéshez: function validate(address client) view returns (bool)

Hozzáféréskor ellenőrzésre kerül a bérlő. Sikeres hozzáférés akkor történik ha a lefoglalt intervallumba próbál belépni a bérlője. Ekkor használhatja a bérlő a szervert.

Amennyiben nem a bérlő próbál hozzáférni a szerverhez, vagy a bérlő, de rossz idő intervallumban úgy a szerverhez való hozzáférés megtagadásra kerül.

Ellenérték felvétele: function withdraw()

A foglalási díjat a szerverpark gazdája beszedi.

# Specifikációnak való megfelelés

A kiadott specifikáció pontjainak és az elkészített megoldás elemeinek megfeleltetése felsorolásszerűen. Pl.: a B követelményt az X metódus Y része/sora valósítja meg.

# Teszt szcenárió futtatása

A specifikációban megadott szcenárió „végigjátszásához” szükséges lépések részletes dokumentálása (tehát a hívás során átadott paraméterek értékeivel együtt).