Opracowanie Tokenu Flotea

Projekt: “Przeprowadzenie prac badawczych i rozwojowych umożliwiających wdrożenie inteligentnego kontraktu opartego o technologię blockchain”

Autorzy:

- Blicharski Bartłomiej,
- Martin Morawiec,
- Tomasz Sienkowiec

Kontekst: Opracowanie tokenu (wirtualnej waluty), która będzie używana w ramach planowanych funkcjonalności.

Spis treści:

Autorzy:

- Blicharski Bartłomiej
- Martin Morawiec

Fundusze Europejskie Program Regionalny | Rzeczpospolita Polska | Śląskie. | Unia Europejska Europejski Fundusz Rozwoju Regionalnego

1. Token Flotea - służy do opłat pomiędzy pasażerem, przewoźnikiem i agencjami.

2. Kontrakt konsensus

- Każdy może dodawać trasy, różnić się będą statusem walidującym.
- Walidacje wykonuje Flotea, przewoznik, lub agencja, ktora kupila Tokeny z ICO.
    - Czy Flotea będzie miala 51% tokenów?
- Do grupy wykonawców (konsensus) wariacje może dodać każdy z grupy, ale dodaje się aż po głosowaniu. Usuwanie sie dzieje tak samo.
    - Trzeba ustawić minimalną ilość głosujących.
- Brakuje wymyślić nagrody dla ludzi w konsensusie
- Agencja wybiera jakie bilety i od kogo chce sprzedawać.

Flotea Token

Token jest ograniczony do 100 milionów tokenów FTN podzielone na dwa miejsca po przecinku (10^10). Kontrakt tokenu pozwala przesyłać tokeny na dowolny portfel Etherea za pomocą funkcji `transfer(address _to, uint256 _value) public returns (bool)`. Funkcja `balanceOf(address _owner) public constant returns (uint256 bal)` odzyskamy informacje o saldzie. Pozwala żeby trzecia osoba mogła dysponować określoną ilością tokena `approve(address _spender, uint256 _value) public returns (bool)`.

Caly kod zrodlowy:

```
pragma solidity ^0.4.21;

import "./SafeMath.sol";

/**
 * @title Token
 * @dev Token with ERC20 standard
 */
contract Token {
  using SafeMath for uint256;

  mapping (address => mapping (address => uint256)) allowed;
  mapping(address => uint256) balances;
  uint256 public totalSupply;

  event Transfer(address indexed _from, address indexed _to, uint256
value);
  event Approval(address indexed _owner, address indexed _spender,
uint256 value);
```

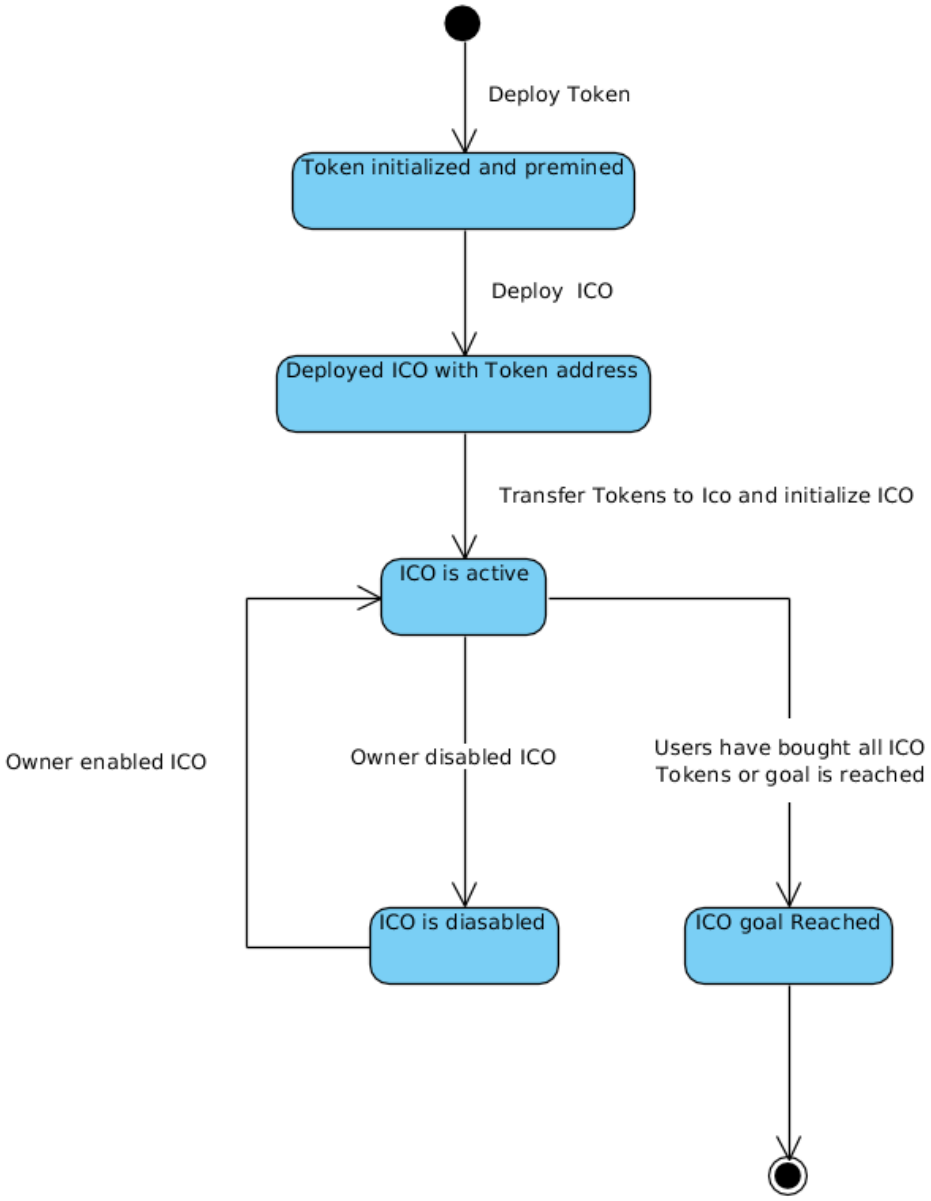
```
  /**
   * Token transfer function
   * @dev transfer token for a specified address
   * @param _to address to transfer to.
   * @param _value amount to be transferred.
   */
  function transfer(address _to, uint256 _value) public returns (bool) {
    //Safemath fnctions will throw if value is invalid
    balances[msg.sender] = balances[msg.sender].sub(_value);
    balances[_to] = balances[_to].add(_value);
    emit Transfer(msg.sender, _to, _value);
    return true;
  }

  /**
   * Token transferFrom function
   * @dev Transfer tokens from one address to another
   * @param _from address to send tokens from
   * @param _to address to transfer to
   * @param _value amout of tokens to be transfered
   */
  function transferFrom(address _from, address _to, uint256 _value)
public returns (bool) {
    uint256 _allowance = allowed[_from][msg.sender];
    // Safe math functions will throw if value invalid
    balances[_to] = balances[_to].add(_value);
    balances[_from] = balances[_from].sub(_value);
    allowed[_from][msg.sender] = _allowance.sub(_value);
    emit Transfer(_from, _to, _value);
    return true;
  }

  /**
   * Token balanceOf function
   * @dev Gets the balance of the specified address.
   * @param _owner address to get balance of.
   * @return uint256 amount owned by the address.
   */
  function balanceOf(address _owner) public constant returns (uint256
bal) {
    return balances[_owner];
  }

  /**
   * Token approve function
```

```
   * @dev Aprove address to spend amount of tokens
   * @param _spender address to spend the funds.
   * @param _value amount of tokens to be spent.
   */
  function approve(address _spender, uint256 _value) public returns
(bool) {
    // To change the approve amount you first have to reduce the
addresses`
    // allowance to zero by calling `approve(_spender, 0)` if it is not
    // already 0 to mitigate the race condition described here:
    // @notice
https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
    assert((_value == 0) || (allowed[msg.sender][_spender] == 0));

    allowed[msg.sender][_spender] = _value;
    emit Approval(msg.sender, _spender, _value);
    return true;
  }

  /**
   * Token allowance method
   * @dev Ckeck that owners tokens is allowed to send to spender
   * @param _owner address The address which owns the funds.
   * @param _spender address The address which will spend the funds.
   * @return A uint256 specifing the amount of tokens still available
for the spender.
   */
  function allowance(address _owner, address _spender) public view
returns (uint256 remaining) {
    return allowed[_owner][_spender];
  }
}

/**
 * @title Flotea Token
 * @dev Simple ERC20 Token with standard token functions.
 */
contract FloteaToken is Token {
  string public constant NAME = "Flotea Token";
  string public constant SYMBOL = "FTN";
  uint256 public constant DECIMALS = 2;

  uint256 public constant INITIAL_SUPPLY = 100 * 10**6 * 10**2; // 100
milions * 100
```

```
  /**
   * Flotea Token Constructor
   * @dev Create and issue tokens to msg.sender.
   */
  constructor() public {
    totalSupply = INITIAL_SUPPLY;
    balances[msg.sender] = INITIAL_SUPPLY;
  }
}
```

Flotea ICO

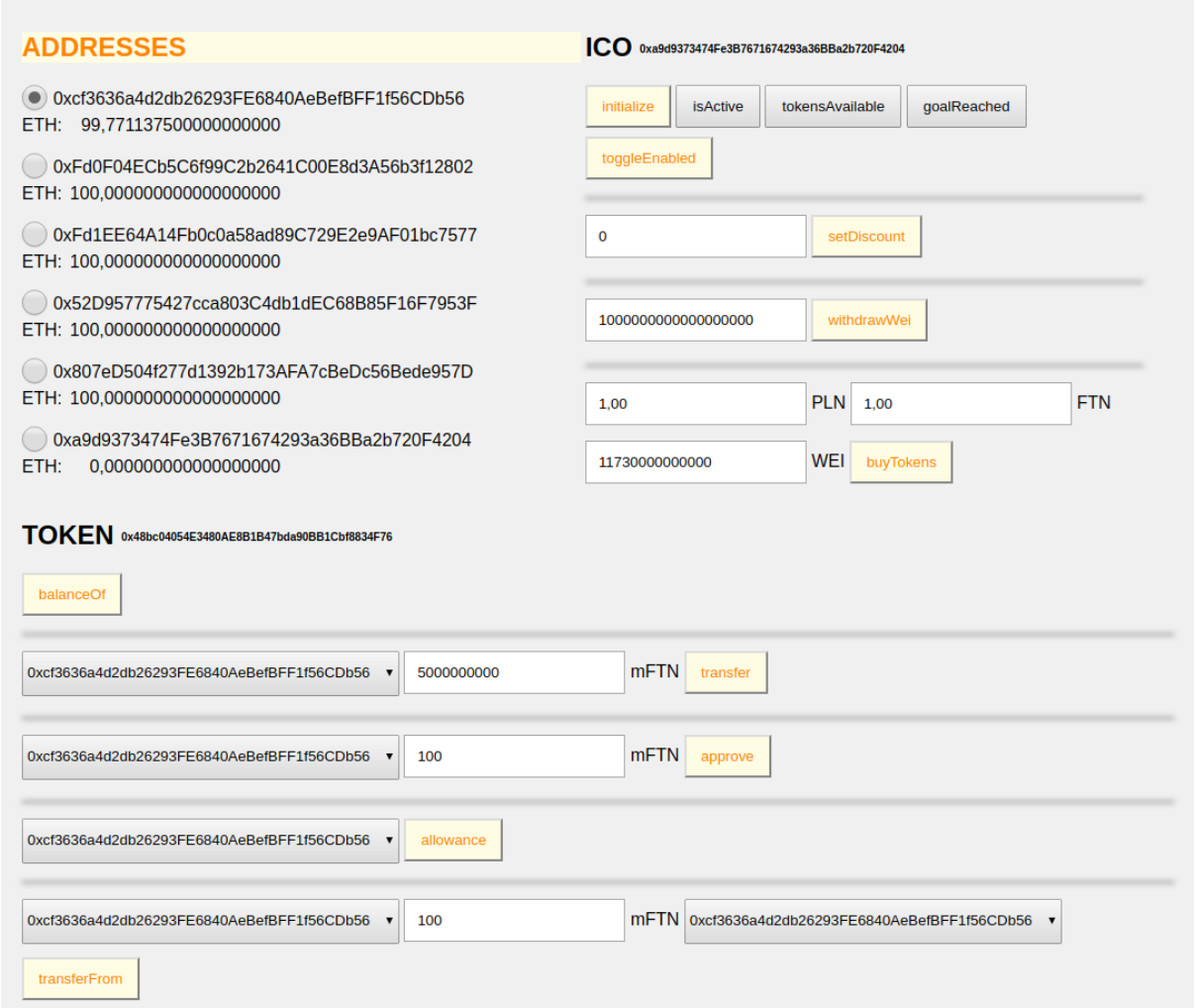
Na obrazku poniżej widać stany w których się może Token i ICO znaleźć.



Migracja tokena FTN i ICO na testową sieć Ethereum.

```
martin@martin-XPS-15-9560:~/eth-contract/token-test$ truffle migrate --reset
Using network 'development'.

Running migration: 1_initial_migration.js
  Replacing Migrations...
  ... 0x3842a5543ea022d6ffb83e67caed8a1fe654464fabdd34a72a32493b36d845a7
  Migrations: 0x6a4c32c0d4da1b81a3fb272dc4606813afa2a59d
Saving successful migration to network...
  ... 0x95f073d215fd1830664ec52de3d6a37d24c04a4faa7399573912741a4f01d44a
Saving artifacts...
Running migration: 2_deploy_contracts.js
  Replacing FloteaToken...
  ... 0x45bdff15a5c129b98f8074fb728485c1786bb30e1f0790fca2996dbbb04c5a4d
  FloteaToken: 0x48bc04054e3480ae8b1b47bda90bb1cbf8834f76
Saving successful migration to network...
  Deploying FloteaICO...
  ... 0xb56aec93bffdd77ac7e32f1a13d7f1db98787dbf9d506863249a29373565e3ea
Saving artifacts...
```

Webowa strona oparta o web3.js - Ethereum JavaScript API, dla testowania tokena i ICO:



Fundusze Europejskie Program Regionalny
Rzeczpospolita Polska
Śląskie.
Unia Europejska Europejski Fundusz Rozwoju Regionalnego

Kontrakt do tworzenie ofert przewoźników i token do zakupu biletu - eksperymenty

Token jest oparty o standard ERC20, pozwala właścicielowi przy stworzeniu ustawić początkową ilość, nazwę, ilość miejsc po przecinku i symbol `constructor(uint256 initialSupply, string tokenName, uint8 decimalUnits, string tokenSymbol) public` . Właściciel może tworzyć dodatkowe tokeny `mintToken(address target, uint256 mintedAmount) onlyOwner public`, albo też ustawiać cenę zakupu i sprzedaży tokena za Ethereum `setPrices(uint256 newSellPrice, uint256 newBuyPrice) onlyOwner public` .

Kod źródłowy tokena:

```
pragma solidity ^0.4.24;

import "./Owned.sol";
import "./TokenRecipient.sol";

contract Token is Owned {
    /* Public variables of the token */
    string public standard = 'Token 0.1';
    string public name;
    string public symbol;

    uint8 public decimals;

    uint256 public totalSupply;
    uint256 public sellPrice;
    uint256 public buyPrice;

    /* This creates an array with all balances */
    mapping (address => uint256) public balanceOf;
    mapping (address => mapping (address => uint256)) public allowance;

    /* This generates a public event on the blockchain that will notify
clients */
    event Transfer(address indexed from, address indexed to, uint256
value);

    /* Initializes contract with initial supply tokens to the creator of
the contract */
    constructor(uint256 initialSupply, string tokenName, uint8
decimalUnits, string tokenSymbol) public{
        balanceOf[msg.sender] = initialSupply;
```

```
        // Give the creator all initial tokens
        totalSupply = initialSupply;
        // Update total supply
        name = tokenName;
        // Set the name for display purposes
        symbol = tokenSymbol;                                   // Set the
symbol for display purposes
        decimals = decimalUnits;                                // Amount of
decimals for display purposes
    }

    /* Send coins */
    function transfer(address _to, uint256 _value) public{
        require (balanceOf[msg.sender] >= _value);           // Check if
the sender has enough
        require (balanceOf[_to] + _value >= balanceOf[_to]); // Check
for overflows
        balanceOf[_to] += _value;                            // Add the
same to the recipient
        balanceOf[msg.sender] -= _value;                     // Subtract
from the sender
        emit Transfer(msg.sender, _to, _value);                   //
Notify anyone listening that this transfer took place
    }

    /* Allow another contract to spend some tokens in your behalf */
    function approve(address _spender, uint256 _value) public
        returns (bool success){
        allowance[msg.sender][_spender] = _value;
        return true;
    }

    /* Approve and then communicate the approved contract in a single tx
*/
    function approveAndCall(address _spender, uint256 _value, bytes
_extraData) public
        returns (bool success) {
        TokenRecipient spender = TokenRecipient(_spender);
        if (approve(_spender, _value)) {
            spender.receiveApproval(msg.sender, _value, this,
_extraData);
            return true;
        }
    }
```

```
    /* A contract attempts to get the coins */
    function transferFrom(address _from, address _to, uint256 _value)
public returns (bool success){
        require (balanceOf[msg.sender] >= _value);           // Check if
the sender has enough
        require (balanceOf[_to] + _value >= balanceOf[_to]); // Check
for overflows
        require (_value <= allowance[_from][msg.sender]);    // Check
allowance
        balanceOf[_to] += _value;                            // Add the
same to the recipient
        balanceOf[_from] -= _value;                          // Subtract
from the sender
        allowance[_from][msg.sender] -= _value;
        emit Transfer(_from, _to, _value);
        return true;
    }

    function mintToken(address target, uint256 mintedAmount) onlyOwner
public{
        balanceOf[target] += mintedAmount;
        totalSupply += mintedAmount;
        emit Transfer(0, this, mintedAmount);
        emit Transfer(this, target, mintedAmount);
    }

    function setPrices(uint256 newSellPrice, uint256 newBuyPrice)
onlyOwner public{
        sellPrice = newSellPrice;
        buyPrice = newBuyPrice;
    }

    function buy() payable public{
        uint amount = msg.value / buyPrice;                // calculates
the amount
        require (balanceOf[this] >= amount);               // checks if
it has enough to sell
        balanceOf[msg.sender] += amount;                   // adds the
amount to buyer's balance
        balanceOf[this] -= amount;                         // subtracts
amount from seller's balance
        emit Transfer(this, msg.sender, amount);               //
execute an event reflecting the change
    }
```

```
    function sell(uint256 amount) public{
        require (balanceOf[msg.sender] >= amount );        // checks if
the sender has enough to sell
        balanceOf[this] += amount;                         // adds the
amount to owner's balance
        balanceOf[msg.sender] -= amount;                   // subtracts
the amount from seller's balance
        if (!msg.sender.send(amount * sellPrice)) {        // sends
ether to the seller. It's important
            require(false);                                          //
to do this last to avoid recursion attacks
        } else {
            emit Transfer(msg.sender, this, amount);             //
executes an event reflecting on the change
        }
    }

    /* This unnamed function is called whenever someone tries to send
ether to it */
    function () public{
        assert(false);     // Prevents accidental sending of ether
    }
}
```

Kontrakt Connections używa tokena powyszej do sprzedaży biletów. Pozwala na tworzenie tras funkcją `createConnection(address carrierWallet, bytes32 from, bytes32 to, uint8 allSlots, uint8 occupiedSlots, uint16 price) public` . Kdzie można ustawić adres portfela przewoźnika, identyfikator stacji skąd i dokąd, ilość wszystkich i zablokowanych miejsc i cenę biletu. Stworzoną trasę można odczytać pod funkcją **`showConnection`**`(uint connectionNumber) public view returns (address, uint8, uint16)`, gdzie jest potrzeba podać identyfikator trasy. Zakup biletu sie dzieje przy pomocy funkcji **`buyTicket`**`(uint connectionNumber, uint8 slots) public`, która wymaga identyfikator trasy i ilości zakupionych biletów. Przewoźnik może zablokować miejsce, które na przykład sprzedał poza blockchain **`lockSlots`**`(uint connectionNumber, uint8 slotsToLock) public` , funkcja wymaga identyfikator trasy i ilość zablokowanych miejsc.

Cały kod źródłowy kontraktu:

```
pragma solidity ^0.4.24;

import "./Owned.sol";
import "./Token.sol";

contract Connections is Owned, Token(1000000, "Flotea Token", 2, "FLT")
{
```

```
    uint public connectionsCount;
    uint public soldTicketsCount;

    struct Connection {
        address creatorWallet;
        uint8 freeSlots;
        uint16 price;
    }

    mapping (uint => Connection) public connectionArray;
    mapping (address => mapping (uint => uint8)) public
soldTicketsArray;

    constructor() public{
        connectionsCount = 0;
        soldTicketsCount = 0;
    }

    function showConnection(uint connectionNumber) public view returns
(address, uint8, uint16) {
        return (connectionArray[connectionNumber].creatorWallet,
connectionArray[connectionNumber].freeSlots,
connectionArray[connectionNumber].price);
    }

    function lockSlots(uint connectionNumber, uint8 slotsToLock) public{
        require (connectionNumber < connectionsCount);
        require (connectionArray[connectionNumber].creatorWallet ==
msg.sender);
        require (connectionArray[connectionNumber].freeSlots >=
slotsToLock);
        connectionArray[connectionNumber].freeSlots =
connectionArray[connectionNumber].freeSlots - slotsToLock;
    }

    function buyTicket(uint connectionNumber, uint8 slots) public{
        require(freeSlots >= slots);
        require (slots > 0);

        uint tokenAmount =
connectionArray[connectionNumber].price*slots;
        address creatorWallet =
connectionArray[connectionNumber].creatorWallet;
        uint8 freeSlots = connectionArray[connectionNumber].freeSlots;
        transfer(creatorWallet, tokenAmount);
```

```
        soldTicketsArray[msg.sender][connectionNumber] += slots;
        connectionArray[connectionNumber].freeSlots -= slots;
        soldTicketsCount++;
    }

    function createConnection(address carrierWallet, bytes32 from,
bytes32 to,
        uint8 allSlots, uint8 occupiedSlots, uint16 price) public {

        require(allSlots >= occupiedSlots);
        bytes32 placeholder;
        placeholder = from;
        placeholder = to;
        placeholder = placeholder;
        connectionArray[connectionsCount].freeSlots = allSlots -
occupiedSlots;
        connectionArray[connectionsCount].creatorWallet = carrierWallet;
        connectionArray[connectionsCount].price = price;
        connectionsCount = connectionsCount + 1;
    }
}
```

Uwagi do ulepszenia kontraktu

Kontrakt nie zawiera czasu wyjazdów, można dodać do kontraktu informacje takie jak na przykład tygodniowy harmonogram wyjazdów z czasami i listę zablokowanych miejsc ku konkretnej dacie. Wyszukiwanie tras jest trudne, kontrakt stwarza przewoźnik i ten musi gdzieś udostępnić adres kontraktu i identyfikatory tras. Te dane mogą być w jakimś dalszym ogólnym kontrakcie aby ułatwić wyszukiwanie. W dalszej etapie można dodać pozycje geograficzne przystanków, lub nazwę miasta. Żeby pomóc przewoźnikowi sprzedawać bilety, będzie dodany aktor Agencja, która może sprzedawać bilety wszystkich, lub wybranych przewoźników, na przykład na swoim portale webowym.

Zaprojektować Consensus Przewoźników - DAO bez tokenów.

W ramach bezpieczeństwa zrobić dodatkowy kontrakt Konsensusu DAO, gdzie na początku Flotea spisuje kilku przewoźników, których jest stała liczba np. tych co przystąpili do presale.

```
Bartek, [05.10.18 11:34]
teraz - każdy może dodać do kontraktu połączenia. Kontrakty będą
oznaczone zwalidowany bądź nie i będą są sprawdzeni - inwestorzy /
przewoźnicy, więc wystarczy jeden z przewoźników już autoryzowanych który
na własną odpowiedzialność doda inny portfel, który będzie zwalidowany w
ten sposób

Martin Morawiec, [05.10.18 11:35]
można zrobić vote system przewoźników, to znaczy jak będzie problem, to
się kontrakt przewoźnika zamyka

Bartek, [05.10.18 11:35]
vote system bym zrobił.
Ci w konsensusie będą faktycznie właścicielem kontraktu Flotea...
no ale jak maja takie argument żeby kogoś wywalić i głosuje za tym aż 51%
to chyba mają do tego prawo... trzeba wymyślić jedynie atrakcje... dla
portfeli w konsensusie, żeby chcieli przystępować. Tutaj kolejne
pytanie... czy każdy może wrzucić połączenia ????

Martin Morawiec, [05.10.18 11:45]
no nie może jeśli chcesz walidować przewoźników

Bartek, [05.10.18 11:45]
dodatkowa walidacje można uzyskać poprzez sprzedawanie biletów poza
konsensusem, czyli ktoś wrzuca przejazd i na własnym portalu sprzedaje
bilety jakiś mechanizm sprawdza co się dzieje, że są sprzedaże itp.

Tomasz Sienkowiec, [05.10.18 11:46]
bo jak każdy jak np. w blablacar - to narazimy się na oszustwa

Bartek, [05.10.18 11:46]
Każdy będzie mógł i moim zdaniem - tutaj się nie narażamy. Po prostu
portale nie będą wyświetlały ofert nie zwalidowanych. Będą miały wybór w
API. Firma taka  jak np. Flixbus nie będzie zwalidowana... dajmy na to,
że ich nikt nie dodał, ale może skorzystać z technologii - dla testów,
albo jest mało znaczącym przewoźnikiem... wyświetlać swoje własne oferty
na swoim portalu wtedy decydują agenci czy ufają temu
przewoźnikowi/agentowi i mogą włączyć wyświetlanie i sprzedaż jego ofert
- mimo tego, że jest poza konsensusem
```