**Лабораторная работа №4**

**«Реализация распределенного приложения (dApp) для голосования.»**

## 1 ХОД РАБОТЫ

1. Был создан каталог dapp, в котором был инициализирован шаблон приложения из фреймворка Truffle.

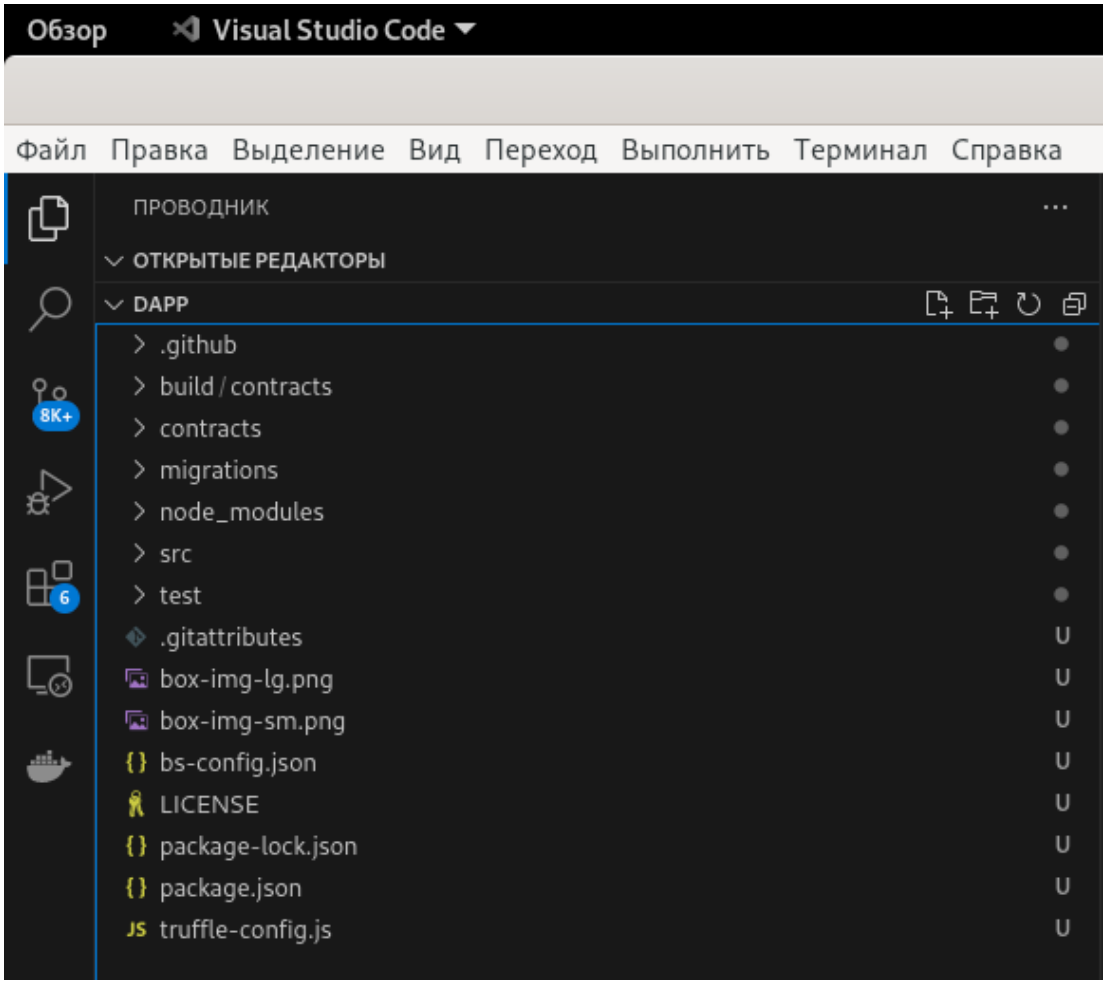


Рисунок 1 – Инициализированное приложение pet_shop

2. Был создан контракт Election.sol, а также скрипт миграции 2_deploy_contracts.js

Листинг 1 – Контракт Election.sol

```
pragma solidity >=0.4.2;
```

```
contract Election {
    // Read/write candidate
    string public candidate;

    // Constructor
    constructor() public {
        candidate = "Candidate 1";
    }
}
```

Листинг 2 – Скрипт миграции 2_deploy_contracts.js

```
var Election = artifacts.require("./Election.sol");

module.exports = function(deployer) {
  deployer.deploy(Election);
};
```

3. Был установлен Ganache, а также изменена конфигурация проекта.

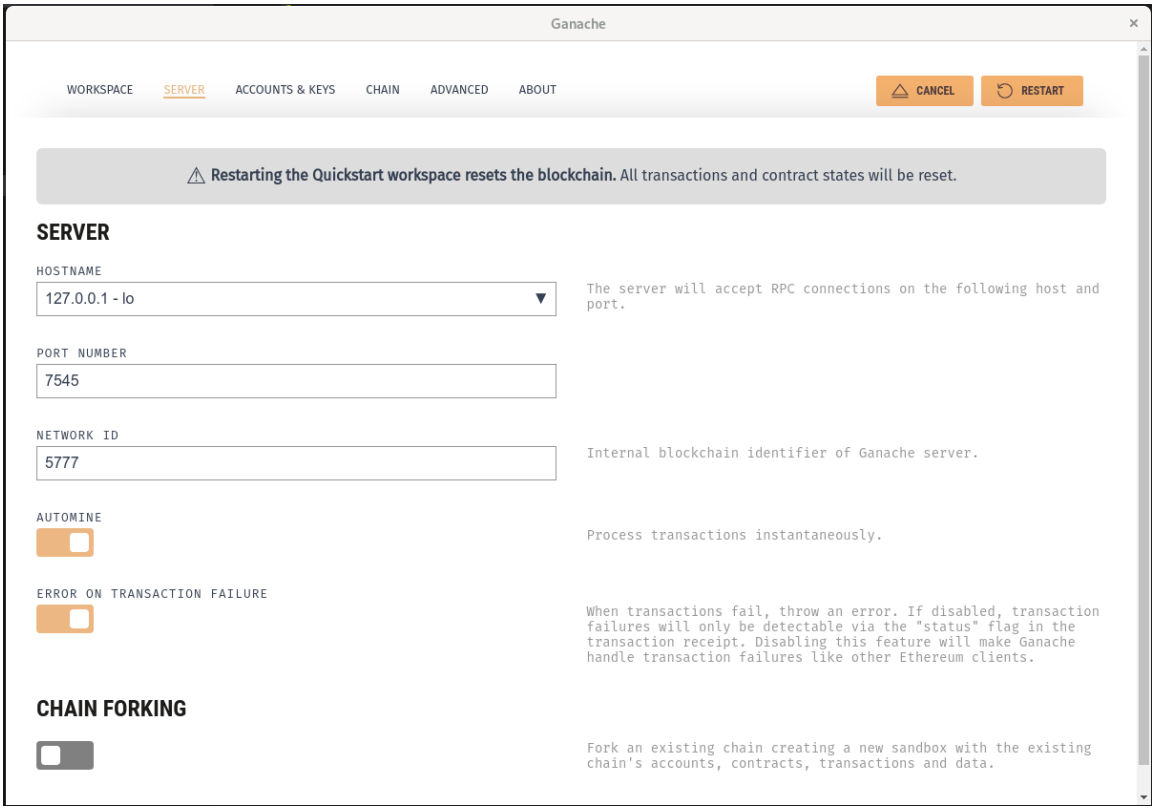


Рисунок 2 – Настройки сервера в Ganache

Листинг 3 – Файл truffle-config.js

```
module.exports = {
  // See <http://truffleframework.com/docs/advanced/configuration>
  // for more about customizing your Truffle configuration!
  networks: {
    development: {
      host: "127.0.0.1",
      port: 7545,
      network_id: "5777" // Match any network id
    },
    develop: {
      port: 8545
    }
  }
};
```

4. После чего, были запущены миграции (truffle migrate), а также инициализировано приложение.

```
   > Blocks: 0            Seconds: 0
   > contract address:    0xcb131797192545f417D3038a807052fC35eeF453
   > block number:        1
   > block timestamp:     1683305749
   > account:             0xc1d578f96D0c032Fade15f497bf6f87bFf41EF78
   > balance:             99.999347804875
   > gas used:            193243 (0x2f2db)
   > gas price:           3.375 gwei
   > value sent:          0 ETH
   > total cost:          0.000652195125 ETH

   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:     0.000652195125 ETH


2_deploy_contracts.js
=====================

   Deploying 'Election'
   --------------------
   > transaction hash:    0xd5cbb8dc3a9641a5249a81ac5c1530724bb46b2ad85391fa897a8cccb5c1231e
   > Blocks: 0            Seconds: 0
   > contract address:    0x9AB4924A86fc7b3743cd47b36A8154bA17EE91f9
   > block number:        3
   > block timestamp:     1683305749
   > account:             0xc1d578f96D0c032Fade15f497bf6f87bFf41EF78
   > balance:             99.998678544669541001
   > gas used:            163567 (0x27eef)
   > gas price:           3.176737487 gwei
   > value sent:          0 ETH
   > total cost:          0.000519609420536129 ETH

   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:     0.000519609420536129 ETH

Summary
=======
> Total deployments:   2
> Final cost:          0.001171804545536129 ETH


➜  dapp git:(main) ✗ truffle console
truffle(development)> Election.deployed().then(function(instance) { app = instance })
undefined
truffle(development)> app.candidate()
'Candidate 1'
truffle(development)>
```

Рисунок 4 – Корректная работа окружения

5. Был изменён контракт Election.sol, а также был написан тест для него.

Листинг 4 – Контракт Election.sol

```
pragma solidity >=0.4.2;

contract Election {
    // Модель данных кандидата
    struct Candidate {
        uint id;
        string name;
        uint voteCount;
    }

    // Хранилище конадидатов
    // Получаем отсюда же, без геттеров
    mapping(uint => Candidate) public candidates;

    // Счетчик кандидатов
    uint public candidatesCount;

    constructor() public {
        addCandidate("Candidate 1");
        addCandidate("Candidate 2");
    }

    function addCandidate (string memory _name ) private {
        candidatesCount ++;
           candidates[candidatesCount] = Candidate(candidatesCount,
_name, 0);
    }
}
```

Листинг 5 – Тест election.js

```
var Election = artifacts.require("./Election.sol");

contract("Election", function(accounts) {
  var electionInstance;
```

```
        it("initializes with two candidates", function() {
          return Election.deployed().then(function(instance) {
            return instance.candidatesCount();
          }).then(function(count) {
            assert.equal(count, 2);
          });
        });
         it("it initializes the candidates with the correct values",
function() {
          return Election.deployed().then(function(instance) {
            electionInstance = instance;
            return electionInstance.candidates(1);
          }).then(function(candidate) {
            assert.equal(candidate[0], 1, "contains the correct id");
                assert.equal(candidate[1], "Candidate 1", "contains the
correct name");
              assert.equal(candidate[2], 0, "contains the correct votes
count");
            return electionInstance.candidates(2);
          }).then(function(candidate) {
            assert.equal(candidate[0], 2, "contains the correct id");
                assert.equal(candidate[1], "Candidate 2", "contains the
correct name");
              assert.equal(candidate[2], 0, "contains the correct votes
count");
          });
        });
      });
```

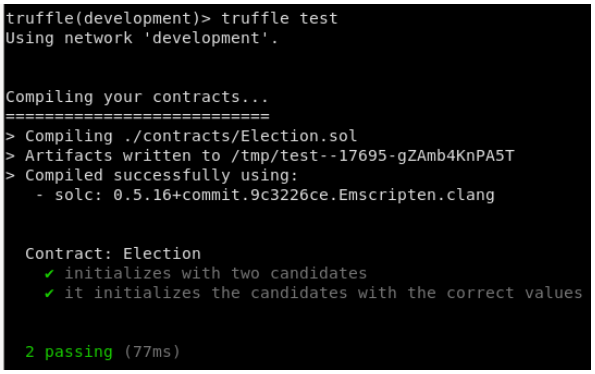


Рисунок 5 – Успешное прохождение теста

6. Была исправлена базовая разметка интерфейса приложения, а также js код для него.

Листинг 6 – Файл index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
     <meta  name="viewport"  content="width=device-width,  initial-
scale=1">
  <title>Election Results</title>

  <!-- Bootstrap -->
  <link href="css/bootstrap.min.css" rel="stylesheet">
</head>

<body>
  <div class="container" style="width: 650px;">
    <div class="row">
      <div class="col-lg-12">
        <h1 class="text-center">Election Results</h1>
        <hr />
        <br />
        <div id="loader">
          <p class="text-center">Loading...</p>
        </div>
        <div id="content" style="display: none;">
          <table class="table">
            <thead>
              <tr>
                <th scope="col">#</th>
                <th scope="col">Name</th>
                <th scope="col">Votes</th>
              </tr>
            </thead>
            <tbody id="candidatesResults">
```

```
                </tbody>
              </table>
              <hr />
              <p id="accountAddress" class="text-center"></p>
            </div>
          </div>
        </div>
      </div>

      <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
                                                                  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"><
/script>
      <!-- Include all compiled plugins (below), or include individual
files as needed -->
      <script src="js/bootstrap.min.js"></script>
      <script src="js/web3.min.js"></script>
      <script src="js/truffle-contract.js"></script>
      <script src="js/app.js"></script>
    </body>

    </html>
```

Листинг 7 – Файл app.js

```
App = {
  web3Provider: null,
  contracts: {},
  account: '0x0',

  init: function() {
    return App.initWeb3();
  },

  initWeb3: function() {
    if (typeof web3 !== 'undefined') {
      // If a web3 instance is already provided by Meta Mask.
      App.web3Provider = web3.currentProvider;
      web3 = new Web3(web3.currentProvider);
```

```
      } else {
        // Specify default instance if no web3 instance provided
                                        App.web3Provider      =      new
Web3.providers.HttpProvider('http://localhost:7545');
        web3 = new Web3(App.web3Provider);
      }
      return App.initContract();
    },

    initContract: function() {
      $.getJSON("Election.json", function(election) {
        // Instantiate a new truffle contract from the artifact
        App.contracts.Election = TruffleContract(election);
        // Connect provider to interact with contract
        App.contracts.Election.setProvider(App.web3Provider);

        return App.render();
      });
    },

    render: function() {
      var electionInstance;
      var loader = $("#loader");
      var content = $("#content");

      loader.show();
      content.hide();

      // Load account data
      web3.eth.getCoinbase(function(err, account) {
        if (err === null) {
          App.account = account;
          $("#accountAddress").html("Your Account: " + account);
        }
      });

      // Load contract data
      App.contracts.Election.deployed().then(function(instance) {
        electionInstance = instance;
```

```
          return electionInstance.candidatesCount();
        }).then(function(candidatesCount) {
          var candidatesResults = $("#candidatesResults");
          candidatesResults.empty();

          for (var i = 1; i <= candidatesCount; i++) {
            electionInstance.candidates(i).then(function(candidate) {
              var id = candidate[0];
              var name = candidate[1];
              var voteCount = candidate[2];

              // Render candidate Result
               var candidateTemplate = "<tr><th>" + id + "</th><td>" +
name + "</td><td>" + voteCount + "</td></tr>"
              candidatesResults.append(candidateTemplate);
            });
          }

          loader.hide();
          content.show();
        }).catch(function(error) {
          console.warn(error);
        });
      }
    };

    $(function() {
      $(window).load(function() {
        App.init();
      });
    });
```

7. Был создан аккаунт MetaMask и подключен к сайту.

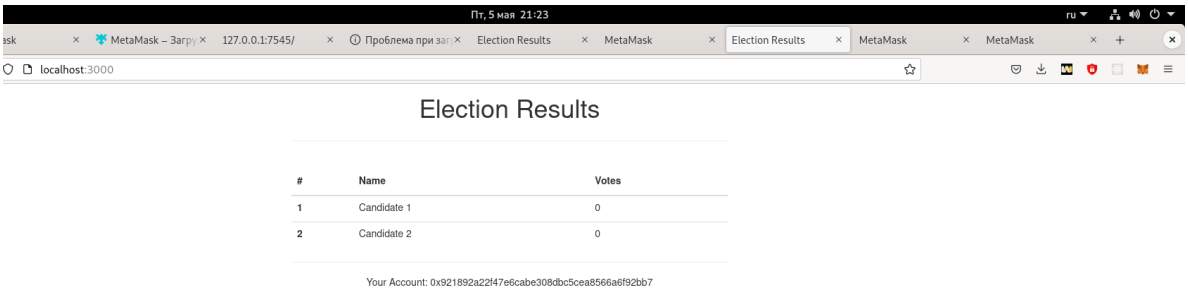


Рисунок 6 – Вход на сайт

8. Была добавлена возможность голосования, также она была покрыта тестами.

Листинг 8 – Файл Election.sol

```
pragma solidity >=0.4.2;

contract Election {
    // Модель данных кандидата
    struct Candidate {
        uint id;
        string name;
        uint voteCount;
    }

    // Хранилище конадидатов
    // Получаем отсюда же, без геттеров
    mapping(uint => Candidate) public candidates;

    // Store accounts that have voted
    mapping(address => bool) public voters;

    // Счетчик кандидатов
    uint public candidatesCount;

    constructor() public {
```

```
            addCandidate("Candidate 1");
            addCandidate("Candidate 2");
        }

        function addCandidate (string memory _name ) private {
            candidatesCount ++;
               candidates[candidatesCount] = Candidate(candidatesCount,
_name, 0);
        }



        function vote (uint _candidateId) public {
            // require that they haven't voted before
            require(!voters[msg.sender]);

            // require a valid candidate
                      require(_candidateId > 0 && _candidateId <=
candidatesCount);

            // record that voter has voted
            voters[msg.sender] = true;

            // update candidate vote Count
            candidates[_candidateId].voteCount ++;
        }

    }
```

Листинг 9 – Файл election.js

```
var Election = artifacts.require("./Election.sol");

contract("Election", function(accounts) {
  var electionInstance;

  it("initializes with two candidates", function() {
    return Election.deployed().then(function(instance) {
      return instance.candidatesCount();
```

```
          }).then(function(count) {
            assert.equal(count, 2);
          });
        });

         it("it initializes the candidates with the correct values",
function() {
          return Election.deployed().then(function(instance) {
            electionInstance = instance;
            return electionInstance.candidates(1);
          }).then(function(candidate) {
            assert.equal(candidate[0], 1, "contains the correct id");
                assert.equal(candidate[1], "Candidate 1", "contains the
correct name");
              assert.equal(candidate[2], 0, "contains the correct votes
count");
            return electionInstance.candidates(2);
          }).then(function(candidate) {
            assert.equal(candidate[0], 2, "contains the correct id");
                assert.equal(candidate[1], "Candidate 2", "contains the
correct name");
              assert.equal(candidate[2], 0, "contains the correct votes
count");
          });
        });

        it("allows a voter to cast a vote", function() {
          return Election.deployed().then(function(instance) {
            electionInstance = instance;
            candidateId = 1;
             return electionInstance.vote(candidateId, { from: accounts[0]
});
          }).then(function(receipt) {
            return electionInstance.voters(accounts[0]);
          }).then(function(voted) {
            assert(voted, "the voter was marked as voted");
            return electionInstance.candidates(candidateId);
          }).then(function(candidate) {
            var voteCount = candidate[2];
```

```
            assert.equal(voteCount, 1, "increments the candidate's vote
count");
          })
        });

        it("throws an exception for invalid candidates", function() {
          return Election.deployed().then(function(instance) {
            electionInstance = instance;
            return electionInstance.vote(99, { from: accounts[1] })
          }).then(assert.fail).catch(function(error) {
              assert(error.message.indexOf('revert') >= 0, "error message
must contain revert");
            return electionInstance.candidates(1);
          }).then(function(candidate1) {
            var voteCount = candidate1[2];
             assert.equal(voteCount, 1, "candidate 1 did not receive any
votes");
            return electionInstance.candidates(2);
          }).then(function(candidate2) {
            var voteCount = candidate2[2];
             assert.equal(voteCount, 0, "candidate 2 did not receive any
votes");
          });
        });

      });
```

```
➜  dapp git:(main) ✗ truffle test
Using network 'development'.


Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.


  Contract: Election
    ✔ initializes with two candidates
    ✔ it initializes the candidates with the correct values
    ✔ allows a voter to cast a vote
    ✔ throws an exception for invalid candidates (79ms)


  4 passing (168ms)
```

Рисунок 7 – Пройденные тесты

9. Была добавлена форма для выбора кандидатов во фронтовскую часть приложения, также был изменён файл app.js.

Листинг 10 – Файл index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta  name="viewport"  content="width=device-width,  initial-
scale=1">
    <title>Election Results</title>

    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">
  </head>
  <body>
    <div class="container" style="width: 650px;">
      <div class="row">
        <div class="col-lg-12">
          <h1 class="text-center">Election Results</h1>
          <hr/>
          <br/>
          <div id="loader">
            <p class="text-center">Loading...</p>
          </div>
          <div id="content" style="display: none;">
            <table class="table">
              <thead>
                <tr>
                  <th scope="col">#</th>
                  <th scope="col">Name</th>
                  <th scope="col">Votes</th>
                </tr>
              </thead>
              <tbody id="candidatesResults">
              </tbody>
            </table>
```

```
                <hr/>
                <form onSubmit="App.castVote(); return false;">
                  <div class="form-group">
                                  <label  for="candidatesSelect">Select
Candidate</label>
                    <select class="form-control" id="candidatesSelect">
                    </select>
                  </div>
                                    <button  type="submit"  class="btn
btn-primary">Vote</button>
                  <hr />
                </form>
                <p id="accountAddress" class="text-center"></p>
              </div>
            </div>
          </div>
        </div>

        <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
                                                                <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js">
/script>
            <!-- Include all compiled plugins (below), or include
individual files as needed -->
        <script src="js/bootstrap.min.js"></script>
        <script src="js/web3.min.js"></script>
        <script src="js/truffle-contract.js"></script>
        <script src="js/app.js"></script>
      </body>
    </html>
```

Листинг 11 – Файл app.js

```
App = {
  web3Provider: null,
  contracts: {},
  account: '0x0',

  init: function () {
```

```
    return App.initWeb3();
  },

  initWeb3: function () {
    if (typeof web3 !== 'undefined') {
      // If a web3 instance is already provided by Meta Mask.
      App.web3Provider = web3.currentProvider;
      web3 = new Web3(web3.currentProvider);
    } else {
      // Specify default instance if no web3 instance provided
                                    App.web3Provider      =      new
Web3.providers.HttpProvider('http://localhost:7545');
      web3 = new Web3(App.web3Provider);
    }
    return App.initContract();
  },

  initContract: function () {
    $.getJSON("Election.json", function (election) {
      // Instantiate a new truffle contract from the artifact
      App.contracts.Election = TruffleContract(election);
      // Connect provider to interact with contract
      App.contracts.Election.setProvider(App.web3Provider);

      return App.render();
    });
  },
  render: function () {
    var electionInstance;
    var loader = $("#loader");
    var content = $("#content");

    loader.show();
    content.hide();

    // Load account data
    web3.eth.getCoinbase(function (err, account) {
      if (err === null) {
        App.account = account;
```

```
            $("#accountAddress").html("Your Account: " + account);
          }
        });

        // Load contract data
        App.contracts.Election.deployed().then(function (instance) {
          electionInstance = instance;
          return electionInstance.candidatesCount();
        }).then(function (candidatesCount) {
          var candidatesResults = $("#candidatesResults");
          candidatesResults.empty();

          var candidatesSelect = $('#candidatesSelect');
          candidatesSelect.empty();

          for (var i = 1; i <= candidatesCount; i++) {
            electionInstance.candidates(i).then(function (candidate) {
              var id = candidate[0];
              var name = candidate[1];
              var voteCount = candidate[2];

              // Render candidate Result
               var candidateTemplate = "<tr><th>" + id + "</th><td>" +
name + "</td><td>" + voteCount + "</td></tr>"
              candidatesResults.append(candidateTemplate);

              // Render candidate ballot option
               var candidateOption = "<option value='" + id + "' >" +
name + "</ option>"
              candidatesSelect.append(candidateOption);
            });
          }
          return electionInstance.voters(App.account);
        }).then(function (hasVoted) {
          // Do not allow a user to vote
          if (hasVoted) {
            $('form').hide();
          }
          loader.hide();
```

```
      content.show();
    }).catch(function (error) {
      console.warn(error);
    });
  },

  castVote: function () {
    var candidateId = $('#candidatesSelect').val();
    App.contracts.Election.deployed().then(function (instance) {
      return instance.vote(candidateId, { from: App.account });
    }).then(function (result) {
      // Wait for votes to update
      $("#content").hide();
      $("#loader").show();
    }).catch(function (err) {
      console.error(err);
    });
  }
};

$(function () {
  $(window).load(function () {
    App.init();
  });
});
```

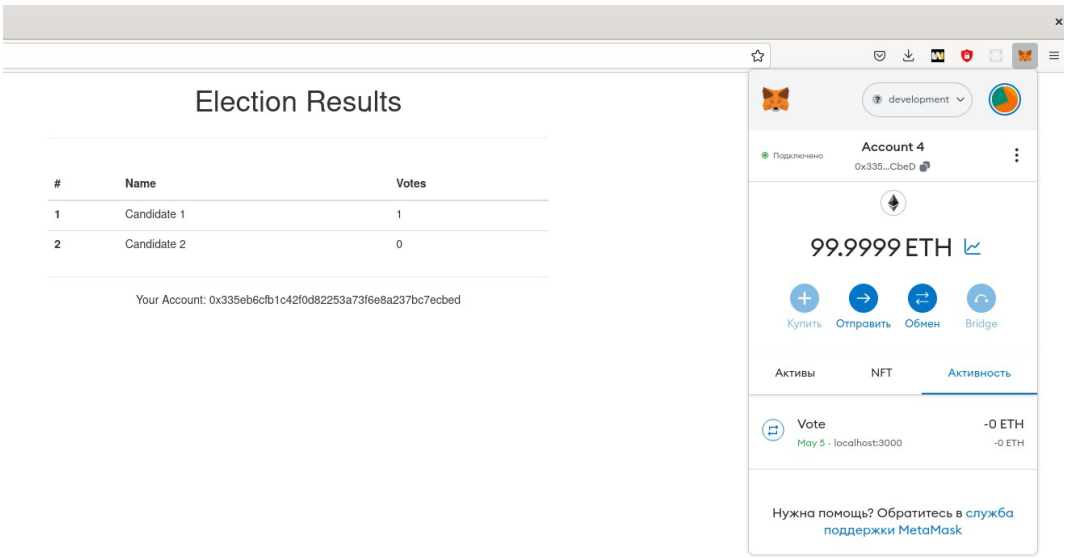


Рисунок 8 – Проведённая транзакция

10. Было добавлено событие о голосовании, параметром которого является кандидат, за которого проголосовали.

Листинг 12 – Файл Election.sol

```
pragma solidity >=0.4.2;

contract Election {
    // Модель данных кандидата
    struct Candidate {
        uint id;
        string name;
        uint voteCount;
    }

    // Хранилище конадидатов
    // Получаем отсюда же, без геттеров
    mapping(uint => Candidate) public candidates;

    // Store accounts that have voted
    mapping(address => bool) public voters;

    // Счетчик кандидатов
    uint public candidatesCount;

    constructor() public {
        addCandidate("Candidate 1");
        addCandidate("Candidate 2");
    }

    function addCandidate (string memory _name ) private {
        candidatesCount ++;
           candidates[candidatesCount] = Candidate(candidatesCount,
_name, 0);
    }

    function vote (uint _candidateId) public {
        require(!voters[msg.sender]);
```

```
                require(_candidateId > 0 && _candidateId <=
candidatesCount);
        voters[msg.sender] = true;
        candidates[_candidateId].voteCount ++;
            // После завершения обработки транзакции - запустим
событие.
        emit votedEvent(_candidateId);
    }

    event votedEvent (
        uint indexed _candidateId
    );

}
```

Листинг 13 – Файл election.js

```
var Election = artifacts.require("./Election.sol");

contract("Election", function(accounts) {
  var electionInstance;

  it("initializes with two candidates", function() {
    return Election.deployed().then(function(instance) {
      return instance.candidatesCount();
    }).then(function(count) {
      assert.equal(count, 2);
    });
  });

   it("it initializes the candidates with the correct values",
function() {
    return Election.deployed().then(function(instance) {
      electionInstance = instance;
      return electionInstance.candidates(1);
    }).then(function(candidate) {
      assert.equal(candidate[0], 1, "contains the correct id");
          assert.equal(candidate[1], "Candidate 1", "contains the
correct name");
```

```
            assert.equal(candidate[2], 0, "contains the correct votes
count");
          return electionInstance.candidates(2);
        }).then(function(candidate) {
          assert.equal(candidate[0], 2, "contains the correct id");
              assert.equal(candidate[1], "Candidate 2", "contains the
correct name");
            assert.equal(candidate[2], 0, "contains the correct votes
count");
        });
      });

      it("allows a voter to cast a vote", function() {
        return Election.deployed().then(function(instance) {
          electionInstance = instance;
          candidateId = 1;
           return electionInstance.vote(candidateId, { from: accounts[0]
});
        }).then(function(receipt) {
          return electionInstance.voters(accounts[0]);
        }).then(function(voted) {
          assert(voted, "the voter was marked as voted");
          return electionInstance.candidates(candidateId);
        }).then(function(candidate) {
          var voteCount = candidate[2];
            assert.equal(voteCount, 1, "increments the candidate's vote
count");
        })
      });

      it("throws an exception for invalid candidates", function() {
        return Election.deployed().then(function(instance) {
          electionInstance = instance;
          return electionInstance.vote(99, { from: accounts[1] })
        }).then(assert.fail).catch(function(error) {
            assert(error.message.indexOf('revert') >= 0, "error message
must contain revert");
          return electionInstance.candidates(1);
        }).then(function(candidate1) {
```

```
          var voteCount = candidate1[2];
           assert.equal(voteCount, 1, "candidate 1 did not receive any
votes");
          return electionInstance.candidates(2);
        }).then(function(candidate2) {
          var voteCount = candidate2[2];
           assert.equal(voteCount, 0, "candidate 2 did not receive any
votes");
        });
      });

      it("allows a voter to cast a vote", function() {
        return Election.deployed().then(function(instance) {
          electionInstance = instance;
          candidateId = 1;
           return electionInstance.vote(candidateId, { from: accounts[2]
});
        }).then(function(receipt) {
                 assert.equal(receipt.logs.length,  1,  "an  event  was
triggered");
           assert.equal(receipt.logs[0].event, "votedEvent", "the event
type is correct");
             assert.equal(receipt.logs[0].args._candidateId.toNumber(),
candidateId, "the candidate id is correct");
        })
      });

    });
```

Листинг 14 – Файл app.js

```
App = {
  web3Provider: null,
  contracts: {},
  account: '0x0',

  init: function () {
    return App.initWeb3();
  },
```

```
    initWeb3: function () {
      if (typeof web3 !== 'undefined') {
        // If a web3 instance is already provided by Meta Mask.
        App.web3Provider = web3.currentProvider;
        web3 = new Web3(web3.currentProvider);
      } else {
        // Specify default instance if no web3 instance provided
                                        App.web3Provider      =      new
Web3.providers.HttpProvider('http://localhost:7545');
        web3 = new Web3(App.web3Provider);
      }
      return App.initContract();
    },

    initContract: function () {
      $.getJSON("Election.json", function (election) {
        // Instantiate a new truffle contract from the artifact
        App.contracts.Election = TruffleContract(election);
        // Connect provider to interact with contract
        App.contracts.Election.setProvider(App.web3Provider);
        App.listenForEvents();
        return App.render();
      });
    },
    render: function () {
      var electionInstance;
      var loader = $("#loader");
      var content = $("#content");

      loader.show();
      content.hide();

      // Load account data
      web3.eth.getCoinbase(function (err, account) {
        if (err === null) {
          App.account = account;
          $("#accountAddress").html("Your Account: " + account);
        }
```

```
        });

        // Load contract data
        App.contracts.Election.deployed().then(function (instance) {
          electionInstance = instance;
          return electionInstance.candidatesCount();
        }).then(function (candidatesCount) {
          var candidatesResults = $("#candidatesResults");
          candidatesResults.empty();

          var candidatesSelect = $('#candidatesSelect');
          candidatesSelect.empty();

          for (var i = 1; i <= candidatesCount; i++) {
            electionInstance.candidates(i).then(function (candidate) {
              var id = candidate[0];
              var name = candidate[1];
              var voteCount = candidate[2];

              // Render candidate Result
               var candidateTemplate = "<tr><th>" + id + "</th><td>" +
name + "</td><td>" + voteCount + "</td></tr>"
              candidatesResults.append(candidateTemplate);

              // Render candidate ballot option
               var candidateOption = "<option value='" + id + "' >" +
name + "</ option>"
              candidatesSelect.append(candidateOption);
            });
          }
          return electionInstance.voters(App.account);
        }).then(function (hasVoted) {
          // Do not allow a user to vote
          if (hasVoted) {
            $('form').hide();
          }
          loader.hide();
          content.show();
        }).catch(function (error) {
```

```
      console.warn(error);
    });
  },

  castVote: function () {
    var candidateId = $('#candidatesSelect').val();
    App.contracts.Election.deployed().then(function (instance) {
      return instance.vote(candidateId, { from: App.account });
    }).then(function (result) {
      // Wait for votes to update
      $("#content").hide();
      $("#loader").show();
    }).catch(function (err) {
      console.error(err);
    });
  },

  listenForEvents: function() {
    App.contracts.Election.deployed().then(function(instance) {
      instance.votedEvent({}, {
        fromBlock: 0,
        toBlock: 'latest'
      }).watch(function(error, event) {
        console.log("event triggered", event)
        // Перерисовываем UI по приходу сообщения
        App.render();
      });
    });
  }

};

$(function () {
  $(window).load(function () {
    App.init();
  });
});
```

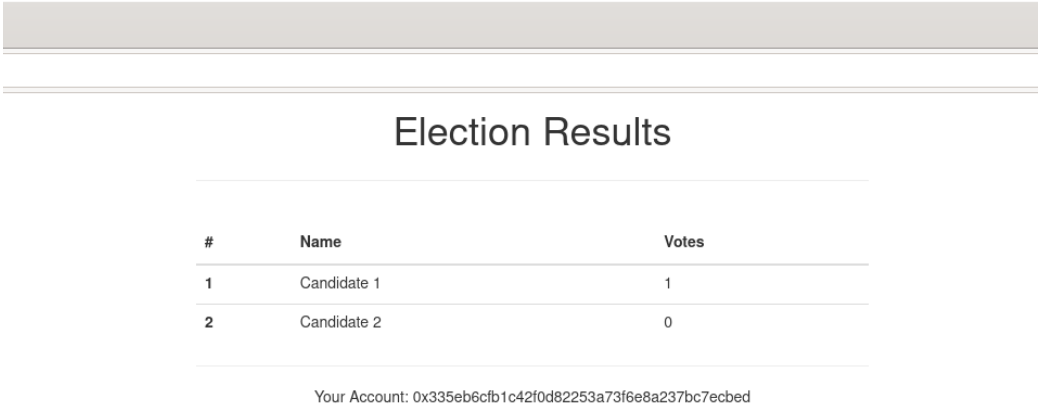


Рисунок 9 – Итоговое приложение