

Программирование интерфейса

Коваленко Г. А.

СОДЕРЖАНИЕ

1. Введение	3
2. Практика	5
2.1. «Программирование узла блокчейн»	5
2.2. Консольный интерфейс	5
2.3. Графический интерфейс	5
2.4. «Ethereum»	25
2.5. Загрузка контракта	25
2.6. Консольный интерфейс	37
2.7. Графический интерфейс	63
3. Скриншоты	115
3.1. «Программирование узла блокчейн»	115
3.2. «Ethereum»	118
4. Литература	123
5. Исходный код	123

1. Введение

Интерфейс есть важная составляющая приложений, благодаря которому исполняются все необходимые действия заложенные в ядро программы. Так например, интерфейсами можно считать API сайтов, внешние (extern) функции библиотек, входные аргументы программы. Но когда речь заходит об обычных пользователях, то масштаб термина «интерфейс» сужается всего до двух его составляющих: CLI (консольный интерфейс) и GUI (графический интерфейс).

Плюсом консольного интерфейса, со стороны лёгкости его написания, является отсутствие (либо предельный минимализм) дизайна. Благодаря этому качеству, сначала первым пишется консольный интерфейс (для первоначальной проверки работоспособности всей программы), а лишь потом графический.

Плюсом графического интерфейса является конечно же «дружелюбность» к обычным пользователям, не знакомым ни с какими терминалами и консолями. Это возможно благодаря более высокому уровню дизайна и действиям, рассчитанным не только на клавиатуру, но и на работу с мышью.

Иногда графический интерфейс могут специально реализовывать таким образом, чтобы он имел вид консольного интерфейса, но при обратном действии, консольный интерфейс не способен принять вид графического (по крайней мере полноценно). Из этого следует, что CLI можно рассматривать как подмножество GUI, и потому сам CLI имеет меньшую сложность при реализации.

Одним из интересных способов реализации графического интерфейса служит создание и поднятие локального сайта, вместо создания обычного десктопного (или мобильного) приложения. Такой способ выстраивания GUI имеет множество положительных оттенков. Во-первых, приложение реализованное таким образом имеет свойство кроссплатформенности. То-есть способно корректно запускаться на Windows, Linux, Mac OS, Android, IOS и других платформах за счёт использования обычных браузеров. Во-вторых, так как используются браузеры и GUI приложение представляет из себя обычный

сайт, то и является возможным использовать безграничное количество фреймворков и библиотек связанных с WEB-разработкой, не ограничиваясь при этом стандартными (либо подгружаемыми) библиотеками выбранного языка программирования. Для такого способа реализации GUI достаточно лишь возможности использовать HTTP протокол.

2. Практика

2.1. «Программирование узла блокчейн»

Привязка интерфейса будет осуществляться к блокчейн сети, реализовывая функции проверки баланса, просмотра блоков, создания транзакций, регистрации и авторизации.

2.2. Консольный интерфейс

CLI был реализован в методическом пособии [1, с.70].

2.3. Графический интерфейс

При реализации GUI будет использоваться язык программирования Go (<https://golang.org/>) [2] представляющий HTTP сервер, а также языки HTML/CSS и фреймворк Bootstrap (<https://getbootstrap.com/>).

Графический клиент представляет отдельное приложение и потому именуется как «main».

```
package main
```

(1.1) Функция init.

```
func init() {  
    err := json.Unmarshal([]byte(readFile(ADDR_FILE)), &Addresses)  
    if err != nil {  
        panic("failed: load addresses")  
    }  
    if len(Addresses) == 0 {
```

```
        panic("failed: len(Addresses) == 0")
    }
}
```

Загружает список адресов к которым клиент подключается, как к узлам блокчейн-сети. Используется константа ADDR_FILE (1.2), а также глобальная переменная Addresses и функция readFile, которые были реализованы ещё в консольном интерфейсе [1].

(1.2) Константа ADDR_FILE.

```
const (
    ADDR_FILE = "addr.json"
)
```

(1.3) Функция main.

```
func main() {
    fmt.Println("Server is running ...")

    http.Handle("/static/", http.StripPrefix(
        "/static/",
        handleFileServer(http.Dir(STTC_PATH))),
    )

    http.HandleFunc("/", indexPage)
    http.HandleFunc("/login", loginPage)
    http.HandleFunc("/signup", signupPage)
    http.HandleFunc("/logout", logoutPage)
    http.HandleFunc("/account", accountPage)
    http.HandleFunc("/transaction", transactionPage)
    http.HandleFunc("/blockchain", blockchainPage)
    http.HandleFunc("/blockchain/", blockchainXPage)
```

```
    http.ListenAndServe(":7545", nil)
}
```

Точка входа в программу. Сначала указывает директорию со статичными файлами (css) при помощи константы STTC_PATH (1.4) и функции handleFileServer (1.5). Далее, для маршрутизации используются функции indexPage (1.6), loginPage (1.7), signupPage (1.8), logoutPage (1.9), accountPage (1.10), transactionPage (1.11), blockchainPage (1.12), blockchainXPage (1.13). В конце запускается прослушивание порта 7545.

(1.4) Константа STTC_PATH.

```
const (
    STTC_PATH = "static/"
)
```

(1.5) Функция handleFileServer.

```
func handleFileServer(fs http.FileSystem) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        if _, err := fs.Open(r.URL.Path); os.IsNotExist(err) {
            indexPage(w, r)
            return
        }
        http.FileServer(fs).ServeHTTP(w, r)
    })
}
```

(1.6) Функция indexPage.

```

func indexPage(w http.ResponseWriter, r *http.Request) {
    t, err := template.ParseFiles(
        TMPL_PATH+"base.html",
        TMPL_PATH+"index.html",
    )
    if err != nil {
        panic("can't load hmtl files")
    }
    var data struct{
        User *bc.User
    }
    data.User = User
    t.Execute(w, data)
}

```

Использует константу TMPL_PATH (1.14) и подгружает файлы «base.html» (1.15) и «index.html» (1.16). Также используется глобальная переменная User взятая из консольного интерфейса [1].

(1.7) Функция loginPage.

```

func loginPage(w http.ResponseWriter, r *http.Request) {
    t, err := template.ParseFiles(
        TMPL_PATH+"base.html",
        TMPL_PATH+"login.html",
    )
    if err != nil {
        panic("can't load hmtl files")
    }
    var data struct{
        User *bc.User
        Error string
    }
    if r.Method == "POST" {
        r.ParseForm()
    }
}

```



```

    User = bc.LoadUser(r.FormValue("private"))
    if User == nil {
        data.Error = "Load Private Key Error"
    } else {
        http.Redirect(w, r, "/", 302)
        return
    }
}
data.User = User
t.Execute(w, data)
}

```

Подгружает файлы «base.html» и «login.html» (1.17).

(1.8) Функция signupPage.

```

func signupPage(w http.ResponseWriter, r *http.Request) {
    t, err := template.ParseFiles(
        TMPL_PATH+"base.html",
        TMPL_PATH+"signup.html",
    )
    if err != nil {
        panic("can't load hmtl files")
    }
    var data struct{
        User *bc.User
        PrivateKey string
    }
    data.User = User
    if r.Method == "POST" {
        data.PrivateKey = bc.NewUser().Purse()
    }
    t.Execute(w, data)
}

```

Подгружает файлы «base.html» и «signup.html» (1.18).

(1.9) Функция logoutPage.

```
func logoutPage(w http.ResponseWriter, r *http.Request) {  
    User = nil  
    http.Redirect(w, r, "/", 302)  
}
```

(1.10) Функция accountPage.

```
func accountPage(w http.ResponseWriter, r *http.Request) {  
    t, err := template.ParseFiles(  
        TMPL_PATH+"base.html",  
        TMPL_PATH+"account.html",  
    )  
    if err != nil {  
        panic("can't load hmtl files")  
    }  
    var data struct{  
        User *bc.User  
        Address string  
        Balance string  
    }  
    data.User = User  
    if data.User != nil {  
        data.Address = User.Address()  
        res := nt.Send(Addresses[0], &nt.Package{  
            Option: GET_BLNCE,  
            Data: data.Address,  
        })  
        if res != nil {  
            data.Balance = res.Data  
        }  
    }  
}
```

```

    } else {
        http.Redirect(w, r, "/", 302)
        return
    }
    t.Execute(w, data)
}

```

Подгружает файлы «base.html» и «account.html» (1.19).

(1.11) Функция transactionPage.

```

func transactionPage(w http.ResponseWriter, r *http.Request) {
    t, err := template.ParseFiles(
        TMPL_PATH+"base.html",
        TMPL_PATH+"transaction.html",
    )
    if err != nil {
        panic("can't load html files")
    }
    var data struct{
        User *bc.User
        Error string
    }
    data.User = User
    if r.Method == "POST" {
        r.ParseForm()
        if data.User == nil {
            data.Error = "User not authorized"
            t.Execute(w, data)
            return
        }
        receiver := r.FormValue("receiver")
        num, err := strconv.Atoi(r.FormValue("value"))
        if err != nil {
            data.Error = "strconv.Atoi error"

```

```

        t.Execute(w, data)
        return
    }
    flag := false
    for _, addr := range Addresses {
        res := nt.Send(addr, &nt.Package{
            Option: GET_LHASH,
        })
        if res == nil {
            continue
        }
        tx := bc.NewTransaction(User, bc.Base64Decode(res.Data), receiver,
uint64(num))

        res = nt.Send(addr, &nt.Package{
            Option: ADD_TRNSX,
            Data:  bc.SerializeTX(tx),
        })
        if res == nil || res.Data != "ok" {
            continue
        }
        flag = true
    }
    if !flag {
        data.Error = "TX failed"
    } else {
        data.Error = "TX success"
    }
}
t.Execute(w, data)
}

```

Подгружает файлы «base.html» и «transaction.html» (1.20).

(1.12) Функция blockchainPage.

```

func blockchainPage(w http.ResponseWriter, r *http.Request) {
    t, err := template.ParseFiles(
        TMPL_PATH+"base.html",
        TMPL_PATH+"blockchain.html",
    )
    if err != nil {
        panic("can't load hmtl files")
    }
    var data struct{
        Error string
        Size []bool
        Address string
        Balance string
        User *bc.User
    }
    data.User = User
    if r.Method == "POST" {
        data.Address = r.FormValue("address")
        res := nt.Send(Addresses[0], &nt.Package{
            Option: GET_BLNCE,
            Data: data.Address,
        })
        if res != nil {
            data.Balance = res.Data
        }
    }
    res := nt.Send(Addresses[0], &nt.Package{
        Option: GET_CSIZE,
    })
    if res == nil || res.Data == "" {
        data.Error = "Receive error"
        t.Execute(w, data)
        return
    }
    num, err := strconv.Atoi(res.Data)
    if err != nil {

```

```

        data.Error = "strconv.Atoi error"
        t.Execute(w, data)
        return
    }
    data.Size = make([]bool, num)
    t.Execute(w, data)
}

```

Подгружает файлы «base.html» и «blockchain.html» (1.21).

(1.13) Функция blockchainXPage.

```

func blockchainXPage(w http.ResponseWriter, r *http.Request) {
    t, err := template.ParseFiles(
        TMPL_PATH+"base.html",
        TMPL_PATH+"blockchainX.html",
    )
    if err != nil {
        panic("can't load hmtl files")
    }
    var data struct{
        Error string
        Block *bc.Block
        User *bc.User
    }
    data.User = User
    res := nt.Send(Addresses[0], &nt.Package{
        Option: GET_BLOCK,
        Data: strings.Replace(r.URL.Path, "/blockchain/", "", 1),
    })
    if res == nil || res.Data == "" {
        data.Error = "Receive error"
        t.Execute(w, data)
        return
    }
}

```

```

data.Block = bc.DeserializeBlock(res.Data)
if data.Block == nil {
    data.Error = "Block is nil"
    t.Execute(w, data)
    return
}
t.Execute(w, data)
}

```

Подгружает файлы «base.html» и «blockchainX.html» (1.22).

(1.14) Константа TMPL_PATH.

```

const (
    TMPL_PATH = "templates/"
)

```

Используются следующие пакеты.

```

import (
    "os"
    "fmt"
    nt "./network"
    bc "./blockchain"
    "net/http"
    "strconv"
    "strings"
    "html/template"
    "encoding/json"
)

```

(1.15) Файл base.html.

```
<!DOCTYPE html>
<html>
<head>
  <title>
    {{block "title" .}}
    Title
    {{end}}
  </title>
  <meta charset="utf-8">
  <link rel="stylesheet" type="text/css" href="/static/css/bootstrap.css">
</head>
<body>
  <header>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
      <div class="container">
        <a href="/" class="navbar-brand" exact><h4>Home</h4></a>
        <div class="navbar-collapse">
          <ul class="navbar-nav">
            <a href="/blockchain" class="nav-link"><h5>Blockchain</h5></a>
          </ul>
          <ul class="navbar-nav ml-auto">
            {{ if (not .User) }}
              <a href="/signup" class="nav-link"><h5>Signup</h5></a>
              <a href="/login" class="nav-link"><h5>Login</h5></a>
            {{ else }}
              <a href="/transaction" class="nav-link"><h5>Transaction</h5></a>
              <a href="/account" class="nav-link"><h5>Account</h5></a>
              <a href="/logout" class="nav-link"><h5>Logout</h5></a>
            {{ end }}
          </ul>
        </div>
      </div>
    </nav>
  </header>
  <main>
```



```
    {{block "content" .}}
    Content
    {{end}}
</main>
</body>
</html>
```

(1.16) Файл index.html.

```
{{define "title"}}
Index
{{end}}
{{define "content"}}
<div class="col-md-9 mx-auto">
    <div class="jumbotron">
        <h1 class="text-center">Blockchain</h1>
    </div>
</div>
{{end}}
```

(1.17) Файл login.html.

```
{{define "title"}}
Login
{{end}}
{{define "content"}}
{{ if .Error }}
    <p>{{ .Error }}</p>
{{ end }}
<div class="col-md-8 mx-auto">
    <div class="jumbotron">
        <div class="col-10 mx-auto">
            <form method="POST" action="/login">
```

```

    <div class="form-group">
        <input type="password" class="form-control" name="private" placeholder="Private
Key">
    </div>
    <input type="submit" class="btn btn-success w-100" name="login" value="Login">
</form>
</div>
</div>
</div>
{{end}}

```

(1.18) Файл signup.html.

```

{{define "title"}}
    Signup
{{end}}
{{define "content"}}
    <div class="col-md-8 mx-auto">
        <div class="jumbotron">
            <div class="col-10 mx-auto">
                <form method="POST" action="/signup">
                    <div class="form-group">
                        {{ if .PrivateKey }}
                            <input readonly type="text" class="form-control bg-light" value="{{ .PrivateKey
}}" id="private">
                        {{ end }}
                    </div>
                    <input type="submit" class="btn btn-success w-100" name="generate"
value="Generate Private Key">
                </form>
            </div>
        </div>
    </div>
{{end}}

```

(1.19) Файл account.html.

```
{{define "title"}}
    Account
{{end}}
{{define "content"}}
    <div class="col-md-9 mx-auto">
        <div class="jumbotron">
            <div class="col-12 mx-auto">
                <form>
                    <div class="form-group">
                        <input id="address" readonly class="form-control bg-light" type="text"
name="coins" value="Address: {{ .Address }}">
                    </div>
                    <div class="form-group">
                        <input disabled class="form-control bg-light" type="text" name="coins"
value="Balance: {{ if .Balance }} {{ .Balance }} {{ else }} 0 {{ end }} coins;">
                    </div>
                </form>
            </div>
        </div>
    </div>
{{end}}
```

(1.20) Файл transaction.html.

```
{{define "title"}}
    Transaction
{{end}}
{{define "content"}}
    <div class="col-md-8 mx-auto">
        <div class="jumbotron">
            <div class="col-10 mx-auto">
```

```

<form method="POST" action="/transaction">
  <div class="form-group">
    {{ if .Error }}
      <input disabled class="form-control bg-light" type="text" name="state" value="{{
.Error }};">
    {{ end }}
  </div>
  <div class="form-group">
    <input type="text" class="form-control" name="receiver" placeholder="Receiver">
  </div>
  <div class="form-group">
    <input type="number" class="form-control" name="value" placeholder="Value">
  </div>
  <input type="submit" class="btn btn-success w-100" name="submit" value="Send">
</form>
</div>
</div>
{{end}}

```

(1.21) Файл blockchain.html.

```

{{define "title"}}
  Blockchain
{{end}}
{{define "content"}}
  <div class="col-md-9 mx-auto">
    <div class="jumbotron">
      <div class="col-12 mx-auto">
        <form method="POST" action="/blockchain">
          <div class="form-group">
            {{ if .Address }}
              <input readonly class="form-control bg-light" type="text" name="coins"
value="Address: {{ .Address }}">

```

```

        <input disabled class="form-control bg-light" type="text" name="coins"
value="Balance: {{ .Balance }} coins;">
        {{ end }}
    </div>
    <div class="form-group">
        <input type="text" class="form-control" name="address" placeholder="Address">
    </div>
    <input type="submit" class="btn btn-success w-100" name="submit"
value="Balance">
    </form>
</div>
</div>
<div class="jumbotron">
    {{ if .Error }}
        <p>{{ .Error }}</p>
    {{ else }}
        {{ range $i, $e := .Size }}
            <div class="card">
                <a class="btn btn-info" href="/blockchain/{{ $i }}">[Block {{ $i }}]</a>
            </div>
        {{ end }}
    {{ end }}
</div>
</div>
{{end}}

```

(1.22) Файл blockchainX.html.

```

{{define "title"}}
    Block
{{end}}
{{define "content"}}
    {{ if .Error }}
        <p>{{ .Error }}</p>

```

```
{{ else }}
```

```
<table border="1">
  <tr>
    <th>Nonce</th>
    <td width="100%">{{ .Block.Nonce }}</td>
  </tr>
  <tr>
    <th>Difficulty</th>
    <td width="100%">{{ .Block.Difficulty }}</td>
  </tr>
  <tr>
    <th>CurrHash</th>
    <td width="100%">{{ .Block.CurrHash }}</td>
  </tr>
  <tr>
    <th>PrevHash</th>
    <td width="100%">{{ .Block.PrevHash }}</td>
  </tr>
  <tr>
    <th>Miner</th>
    <td width="100%">{{ .Block.Miner }}</td>
  </tr>
  <tr>
    <th>Signature</th>
    <td width="100%">{{ .Block.Signature }}</td>
  </tr>
  <tr>
    <th>TimeStamp</th>
    <td width="100%">{{ .Block.TimeStamp }}</td>
  </tr>
  <tr>
    <th>Transactions</th>
    <td>
      {{ range .Block.Transactions }}
        <table border="1">
          <tr>
```

```

        <th>RandBytes</th>
        <td width="100%">{{ .RandBytes }}</td>
    </tr>
    <tr>
        <th>PrevBlock</th>
        <td width="100%">{{ .PrevBlock }}</td>
    </tr>
    <tr>
        <th>Sender</th>
        <td width="100%">{{ .Sender }}</td>
    </tr>
    <tr>
        <th>Receiver</th>
        <td width="100%">{{ .Receiver }}</td>
    </tr>
    <tr>
        <th>Value</th>
        <td width="100%">{{ .Value }}</td>
    </tr>
    <tr>
        <th>ToStorage</th>
        <td width="100%">{{ .ToStorage }}</td>
    </tr>
    <tr>
        <th>CurrHash</th>
        <td width="100%">{{ .CurrHash }}</td>
    </tr>
    <tr>
        <th>Signature</th>
        <td width="100%">{{ .Signature }}</td>
    </tr>
</table>
{{ end }}
</td>
</tr>
<tr>

```

```

<th>Mapping</th>
<td>
  {{ range $k, $e := .Block.Mapping }}
    <table border="1">
      <tr>
        <th width="100%">{{ $k }}</th>
        <td>{{ $e }}</td>
      </tr>
    </table>
  {{ end }}
</td>
</tr>
</table>
{{ end }}
{{end}}

```

2.4. «Ethereum»

Для работы с платформой «Ethereum», используя язык Go, необходимо установить стороннюю библиотеку go-ethereum. Далее, перейти в директорию `go/src/github.com/ethereum/go-ethereum/cmd/abigen/`, скомпилировать файл `main.go` и скопировать его в рабочую директорию.

```
$ go get -d github.com/ethereum/go-ethereum
$ cd /home/user/go/src/github.com/ethereum/go-ethereum/cmd/abigen
$ go build main.go
$ mv main abigen
```

2.5. Загрузка контракта

Прежде чем реализовывать интерфейсную составляющую, связанную с платформой «Ethereum», необходимым является создание контракта и подключение нашего приложения к нему.

Контракт. Файл `contract.sol`.

```
pragma solidity ^0.5.0;

contract WorldSkills {

    struct Estate {
        uint estate_id;
        address owner;
        string info;
        uint square;
        uint useful_square;
        address renter_address;
        bool present_status;
```

```

    bool sale_status;
    bool rent_status;
}

struct Present {
    uint estate_id;
    address address_from;
    address address_to;
    bool finished;
}

struct Sale {
    uint estate_id;
    address owner;
    uint price;
    address payable[] customers;
    uint[] prices;
    bool finished;
}

struct Rent {
    uint estate_id;
    address payable owner_address;
    address payable renter_address;
    uint time;
    uint money;
    uint deadline;
    bool finished;
}

Estate[] estates;
Present[] presents;
Sale[] sales;
Rent[] rents;

address admin = msg.sender;

```

```

address payable default_address = 0x0000000000000000000000000000000000000000;

function iam_admin() public view returns(bool) {
    return msg.sender == admin;
}

function get_estates_number() public view returns(uint) {
    return estates.length;
}

function get_presents_number() public view returns(uint) {
    return presents.length;
}

function get_sales_number() public view returns(uint) {
    return sales.length;
}

function get_rents_number() public view returns(uint) {
    return rents.length;
}

function get_estates(uint estate_number) public view returns(uint, address, string
memory, uint, uint, address) {
    return(estates[estate_number].estate_id, estates[estate_number].owner,
estates[estate_number].info, estates[estate_number].square, estates[estate_number].useful_square,
estates[estate_number].renter_address);
}

function get_estates_statuses(uint estate_number) public view returns(bool, bool, bool) {
    return(estates[estate_number].present_status, estates[estate_number].sale_status,
estates[estate_number].rent_status);
}

function get_presents(uint present_number) public view returns(uint, address, address,
bool) {

```

```

        return(presents[present_number].estate_id, presents[present_number].address_from,
presents[present_number].address_to, presents[present_number].finished);
    }

```

```

function get_sales(uint sale_number) public view returns(uint, address, uint, address
payable[] memory, uint[] memory prices, bool) {
    return(sales[sale_number].estate_id, sales[sale_number].owner,
sales[sale_number].price, sales[sale_number].customers, sales[sale_number].prices,
sales[sale_number].finished);
}

```

```

function get_rents(uint rent_number) public view returns(uint, address, address, uint, uint,
uint, bool) {
    return(rents[rent_number].estate_id, rents[rent_number].owner_address,
rents[rent_number].renter_address, rents[rent_number].time, rents[rent_number].money,
rents[rent_number].deadline, rents[rent_number].finished);
}

```

```

modifier status_OK(uint estate_id) {
    require(estates[estate_id].present_status == false);
    require(estates[estate_id].sale_status == false);
    require(estates[estate_id].rent_status == false);
    _;
}

```

```

modifier is_owner(uint estate_id) {
    require(msg.sender == estates[estate_id].owner);
    _;
}

```

```

modifier is_admin {
    require(msg.sender == admin);
    _;
}

```

```

function create_estate(address owner, string memory info, uint square, uint useful_square)
public is_admin{
    estates.push(Estate(estates.length, owner, info, square, useful_square,
0x0000000000000000000000000000000000000000, false, false, false));
}

function create_present(uint estate_id, address address_to) public status_OK(estate_id)
is_owner(estate_id) {
    presents.push(Present(estate_id, msg.sender, address_to, false));
    estates[estate_id].present_status = true;
}

function cancel_present(uint present_number) payable public {
    require(msg.sender == presents[present_number].address_from);
    require(presents[present_number].finished == false);
    estates[presents[present_number].estate_id].present_status = false;
    presents[present_number].finished = true;
}

function confirm_present(uint present_number) payable public {
    require(msg.sender == presents[present_number].address_to);
    require(presents[present_number].finished == false);
    estates[presents[present_number].estate_id].owner =
presents[present_number].address_to;
    estates[presents[present_number].estate_id].present_status = false;
    presents[present_number].finished = true;
}

function create_sale(uint estate_id, uint price) public status_OK(estate_id)
is_owner(estate_id){
    address payable[] memory customers;
    uint[] memory prices;
    sales.push(Sale(estate_id, msg.sender, price, customers, prices, false));
    estates[estate_id].sale_status = true;
}

```

```

function cancel_sale(uint sale_number) public {
    require(msg.sender == sales[sale_number].owner);
    require(sales[sale_number].finished == false);
    for (uint i = 0; i < sales[sale_number].customers.length; i++){
        (sales[sale_number].customers[i]).transfer(sales[sale_number].prices[i]);
    }
    estates[sales[sale_number].estate_id].sale_status = false;
    sales[sale_number].finished = true;
}

```

```

function check_to_buy(uint sale_number) public payable {
    require(msg.sender != sales[sale_number].owner);
    require(msg.value >= sales[sale_number].price);
    require(sales[sale_number].finished == false);
    uint status = 0;
    for (uint i=0; i < sales[sale_number].customers.length; i++) {
        if (sales[sale_number].customers[i] == msg.sender) {
            status = 1;
            break;
        }
    }
    require(status == 0);
    sales[sale_number].customers.push(msg.sender);
    sales[sale_number].prices.push(msg.value);
}

```

```

function cancel_to_buy(uint sale_number) public payable {
    require(sales[sale_number].finished == false);
    for (uint i=0; i<sales[sale_number].customers.length; i++){
        if (sales[sale_number].customers[i] == msg.sender){
            msg.sender.transfer(sales[sale_number].prices[i]);
            delete sales[sale_number].prices[i];
        }
    }
}

```

```

function confirm_sale(uint sale_number, uint sale_to) public payable {
    require(msg.sender == sales[sale_number].owner);
    require(sales[sale_number].prices[sale_to] != 0);
    require(sales[sale_number].finished == false);
    estates[sales[sale_number].estate_id].owner = sales[sale_number].customers[sale_to];
    msg.sender.transfer(sales[sale_number].prices[sale_to]);
    for (uint i=0; i<sales[sale_number].customers.length; i++){
        if (i != sale_to) {
            sales[sale_number].customers[i].transfer(sales[sale_number].prices[i]);
        }
        else {
            sales[sale_number].prices[sale_to] = 0;
        }
    }
    estates[sales[sale_number].estate_id].sale_status = false;
    sales[sale_number].finished = true;
}

```

```

function create_rent(uint estate_id, uint time, uint money) public is_owner(estate_id)
status_OK(estate_id){
    rents.push(Rent(estate_id, msg.sender, default_address, time, money, 0, false));
    estates[estate_id].rent_status=true;
}

```

```

function to_rent(uint rent_id) public payable{
    require(rents[rent_id].finished == false);
    require(rents[rent_id].renter_address == default_address);
    require(rents[rent_id].owner_address != msg.sender);
    require(rents[rent_id].money == msg.value);
    rents[rent_id].renter_address = msg.sender;
    estates[rents[rent_id].estate_id].renter_address = msg.sender;
    rents[rent_id].deadline = now + rents[rent_id].time*86400;
    rents[rent_id].owner_address.transfer(rents[rent_id].money);
}

```

```

function cancel_rent(uint rent_id) public {
    require(rents[rent_id].finished == false);
    require(rents[rent_id].owner_address == msg.sender);
    require(rents[rent_id].renter_address == default_address);
    estates[rents[rent_id].estate_id].rent_status=false;
    rents[rent_id].finished = true;
}

function finish_rent(uint rent_id) public is_owner(rents[rent_id].estate_id) {
    require(rents[rent_id].finished == false);
    require(rents[rent_id].deadline < now);
    estates[rents[rent_id].estate_id].renter_address = default_address;
    estates[rents[rent_id].estate_id].rent_status=false;
    rents[rent_id].finished = true;
}
}

```

После написания, контракт необходимо скомпилировать. Для этого воспользуемся компилятором solc. В итоге создастся директория build, в которую внесутся два файла WorldSkills.abi и WorldSkills.bin (именование происходит от имени контракта, а не файла).

```
$ solc --overwrite --abi --bin contract.sol -o build
```

Уже после успешной компиляции, необходимо на основе двух созданных файлов сгенерировать библиотеку языка Go, которая будет работать с созданным контрактом.

```

$ mkdir -p contracts
$ ./abigen --bin=./build/WorldSkills.bin --abi=./build/WorldSkills.abi --pkg=contract --
out=./contracts/Contract.go

```

Загрузка контракта. Файл deploy.go.

```
package main

import (
    "os"
    "io/ioutil"
    "context"
    "crypto/ecdsa"
    "fmt"
    "log"
    "math/big"
    "strings"
    "github.com/ethereum/go-ethereum/accounts/abi/bind"
    "github.com/ethereum/go-ethereum/crypto"
    "github.com/ethereum/go-ethereum/ethclient"
    "github.com/ethereum/go-ethereum/common"
    contract "./contracts"
)

type UserType struct {
    Purse string
    AddressHex string
    AddressEth common.Address
    PublicKey *ecdsa.PublicKey
    PrivateKey *ecdsa.PrivateKey
}

var (
    ClientETH = connectToETH("http://127.0.0.1:5555")
    User *UserType
)

func init() {
    if len(os.Args) < 2 {
```

```

        panic("failed: len(os.Args) < 2")
    }
    var (
        userLoadStr = ""
        userLoadExist = false
    )
    for i := 1; i < len(os.Args); i++ {
        arg := os.Args[i]
        switch {
        case strings.HasPrefix(arg, "-loaduser:"):
            userLoadStr = strings.Replace(arg, "-loaduser:", "", 1)
            userLoadExist = true
        }
    }
    if !userLoadExist {
        panic("failed: !userLoadExist")
    }
    if ClientETH == nil {
        panic("failed: connect to ETH")
    }
    User = userLoad(userLoadStr)
    if User == nil {
        panic("failed: load user")
    }
}

// Deploy contract and save address in file.
func main() {
    auth := resetAuth(User)
    address, tx, instance, err := contract.DeployContract(auth, ClientETH)

    if err != nil {
        panic(err)
    }

    fmt.Println(address.Hex())
}

```

```

    fmt.Println(tx.Hash().Hex())
    _ = instance

    contractFile := "contract.address"
    writeFile(contractFile, address.Hex())
}

func writeFile(filename string, data string) error {
    return ioutil.WriteFile(filename, []byte(data), 0644)
}

func userLoad(purse string) *UserType {
    priv, err := crypto.HexToECDSA(purse)
    if err != nil {
        return nil
    }
    pub, ok := priv.Public().(*ecdsa.PublicKey)
    if !ok {
        return nil
    }
    addressHex := crypto.PubkeyToAddress(*pub).Hex()
    addressEth := common.HexToAddress(addressHex)
    return &UserType{
        Purse:    purse,
        AddressHex: addressHex,
        AddressEth: addressEth,
        PublicKey: pub,
        PrivateKey: priv,
    }
}

func connectToETH(address string) *ethclient.Client {
    client, err := ethclient.Dial(address)
    if err != nil {
        return nil
    }
}

```

```

        return client
    }

func resetAuth(user *UserType) *bind.TransactOpts {
    nonce, err := ClientETH.PendingNonceAt(context.Background(), User.AddressEth)
    if err != nil {
        return nil
    }

    gasPrice, err := ClientETH.SuggestGasPrice(context.Background())
    if err != nil {
        return nil
    }

    auth := bind.NewKeyedTransactor(user.PrivateKey)
    auth.Nonce = big.NewInt(int64(nonce))
    auth.Value = big.NewInt(0)

    auth.GasPrice = gasPrice

    return auth
}

func fileIsExist(name string) bool {
    if _, err := os.Stat(name); os.IsNotExist(err) {
        return false
    }
    return true
}

```

Компиляция загрузки контракта.

\$ go build -o deploy deploy.go

Предполагается, что доступ к платформе «Ethereum» лежит через адрес `http://127.0.0.1:5555`. При вызове файла `deploy` необходимым является указание создателя контракта (приватный ключ). Пример:

```
$ ./deploy -loaduser:1a6fd9191c5dc456a7027de4bef197d7b57d8281b886ceceebb1cf7eb97cebf0
```

После успешного исполнения данного файла, в блокчейне платформы «Ethereum» появится блок с контрактом. Таким образом, появляется возможность пользоваться функциями контракта.

Рядом с файлом `deploy` появится также файл `contract.address`, в котором будет находится адрес созданного контракта.

2.6. Консольный интерфейс

Консольный клиент «Ethereum» будет представлять отдельное приложение и потому будет именоваться как «main».

```
package main
```

(2.1) Функция `init`.

```
func init() {  
    if len(os.Args) < 2 {  
        panic("failed: len(os.Args) < 2")  
    }  
    var (  
        userLoadStr = ""  
        userLoadExist = false  
    )  
    for i := 1; i < len(os.Args); i++ {  
        arg := os.Args[i]
```

```

switch {
case strings.HasPrefix(arg, "-loaduser:"):
    userLoadStr = strings.Replace(arg, "-loaduser:", "", 1)
    userLoadExist = true
}
}
if !userLoadExist {
    panic("failed: !userLoadExist")
}
if ClientETH == nil {
    panic("failed: connect to ETH")
}
if Instance == nil {
    panic("failed: instance is nil")
}
User = loadUser(userLoadStr)
if User == nil {
    panic("failed: load user")
}
}

```

Читает с аргументов программы приватный ключ пользователя.

Используются глобальные переменные ClientETH (2.2), Instance (2.3), User (2.4) и функция loadUser (2.5).

(2.2) Глобальная переменная ClientETH.

(2.3) Глобальная переменная Instance.

(2.4) Глобальная переменная User.

```

var (
    User *UserType
    ClientETH = connectToETH("http://127.0.0.1:5555")
    Instance = newContract(
        common.HexToAddress(readFile("contract.address")),
        ClientETH,

```

```
)  
)
```

Используется тип данных UserType (2.6), функции connectToETH (2.7), newContract (2.8) и readFile [1].

(2.5) Функция loadUser.

```
func loadUser(purse string) *UserType {  
    priv, err := crypto.HexToECDSA(purse)  
    if err != nil {  
        return nil  
    }  
    pub, ok := priv.Public().(*ecdsa.PublicKey)  
    if !ok {  
        return nil  
    }  
    addressHex := crypto.PubkeyToAddress(*pub).Hex()  
    addressEth := common.HexToAddress(addressHex)  
    return &UserType{  
        Purse:    purse,  
        AddressHex: addressHex,  
        AddressEth: addressEth,  
        PublicKey: pub,  
        PrivateKey: priv,  
    }  
}
```

(2.6) Структура UserType.

```
type UserType struct {  
    Purse string  
    AddressHex string
```

```
AddressEth common.Address
PublicKey *ecdsa.PublicKey
PrivateKey *ecdsa.PrivateKey
}
```

(2.7) Функция connectToETH.

```
func connectToETH(address string) *ethclient.Client {
    client, err := ethclient.Dial(address)
    if err != nil {
        return nil
    }
    return client
}
```

Подключается к сети «Ethereum» через указанный адрес.

(2.8) Функция newContract.

```
func newContract(contractAddr common.Address, clientEth *ethclient.Client)
*contract.Contract {
    instance, err := contract.NewContract(contractAddr, clientEth)
    if err != nil {
        return nil
    }
    return instance
}
```

Подключается к контракту, используя его адрес и соединение с сетью.

(2.9) Функция main.

```
func main() {
```



```

var (
    message string
    splited []string
)
for {
    message = inputString("> ")
    splited = strings.Split(message, " ")
    switch splited[0] {
    case "/exit":
        os.Exit(0)
    case "/user":
        if len(splited) < 2 {
            fmt.Println("failed: len(user) < 2\n")
            continue
        }
        switch splited[1] {
        case "address":
            userAddress()
        case "purse":
            userPurse()
        case "balance":
            userBalance()
        default:
            fmt.Println("command undefined\n")
        }
    case "/chain":
        if len(splited) < 3 {
            fmt.Println("failed: len(chain) < 3\n")
            continue
        }
        switch splited[1] {
        case "get":
            chainGet(splited[2], splited[2:])
        case "create":
            switch splited[2] {
            case "estate":

```

```

        chainCreateEstate(splited[2:])
    case "present":
        chainCreatePresent(splited[2:])
    case "sale":
        chainCreateSale(splited[2:])
    case "rent":
        chainCreateRent(splited[2:])
    default:
        fmt.Println("command undefined\n")
    }
case "cancel":
    chainCancel(splited[2], splited[2:])
case "confirm":
    switch splited[2] {
    case "present":
        chainConfirmPresent(splited[2:])
    case "sale":
        chainConfirmSale(splited[2:])
    case "rent":
        chainConfirmRent(splited[2:])
    default:
        fmt.Println("command undefined\n")
    }
case "try-buy":
    chainTryBuy(splited[1:])
case "cancel-buy":
    chainCancelBuy(splited[1:])
case "finish-rent":
    finishRent(splited[1:])
    default:
        fmt.Println("command undefined\n")
    }
default:
    fmt.Println("command undefined\n")
}
}

```

```
}
```

Точка входа в программу. Использует функции `inputString` [1], `userAddress` (2.10), `userPurse` (2.11), `userBalance` (2.12), `chainGet` (2.13), `chainCreateEstate` (2.14), `chainCreatePresent` (2.15), `chainCreateSale` (2.16), `chainCreateRent` (2.17), `chainCancel` (2.18), `chainConfirmPresent` (2.19), `chainConfirmSale` (2.20), `chainConfirmRent` (2.21), `chainTryBuy` (2.22), `chainCancelBuy` (2.23), `finishRent` (2.24).

(2.10) Функция `userAddress`.

```
func userAddress() {  
    fmt.Println("Address:", User.AddressHex, "\n")  
}
```

(2.11) Функция `userPurse`.

```
func userPurse() {  
    fmt.Println("Purse:", User.Purse, "\n")  
}
```

(2.12) Функция `userBalance`.

```
func userBalance() {  
    balance, err := ClientETH.BalanceAt(context.Background(), User.AddressEth, nil)  
    if err != nil {  
        fmt.Println(err, "\n")  
        return  
    }  
    fmt.Println("Balance:", balance, "\n")  
}
```

(2.13) Функция chainGet.

```
func chainGet(category string, splited []string) {
    if len(splited) != 2 {
        fmt.Println("failed: len(splited) != 2\n")
        return
    }
    var (
        inc = big.NewInt(1)
        err error
        num *big.Int
        jsonData []byte
    )
    switch category {
        case "estates":
            num, err = Instance.GetEstatesNumber(&bind.CallOpts{From:
User.AddressEth})
        case "presents":
            num, err = Instance.GetPresentsNumber(&bind.CallOpts{From:
User.AddressEth})
        case "sales":
            num, err = Instance.GetSalesNumber(&bind.CallOpts{From:
User.AddressEth})
        case "rents":
            num, err = Instance.GetRentsNumber(&bind.CallOpts{From:
User.AddressEth})
        default:
            fmt.Println("undefined category\n")
            return
    }
    if err != nil {
        fmt.Println(err, "\n")
        return
    }
}
```

```

    }
    for index := big.NewInt(0); index.Cmp(num) == -1; index.Add(index, inc) {
        switch category {
            case "estates":
                data := getEstates(index)
                if data == nil {
                    fmt.Println("data is nil\n")
                    return
                }
                if splited[1] == "my" && User.AddressHex !=
data.Owner.Hex() {

                    continue
                }
                if splited[1] != "all" && splited[1] != "my" &&
strings.ToLower(splited[1]) !=
strings.ToLower(data.Owner.Hex()) {

                    continue
                }
                jsonData, err = json.MarshalIndent(data, "", "\t")
            case "presents":
                data := getPresents(index)
                if data == nil {
                    fmt.Println("data is nil\n")
                    return
                }
                if data.Finished {
                    continue
                }
                if splited[1] == "my" &&
(User.AddressHex != data.AddressFrom.Hex() &&
User.AddressHex != data.AddressTo.Hex()){

                    continue
                }
                if splited[1] != "all" && splited[1] != "my" &&
(strings.ToLower(splited[1]) !=
strings.ToLower(data.AddressFrom.Hex()) &&

```

```

                                strings.ToLower(splited[1]) !=
strings.ToLower(data.AddressTo.Hex())) {

                                continue

                                }

                                jsonData, err = json.MarshalIndent(data, "", "\t")
case "sales":
                                data := getSales(index)
                                if data == nil {
                                        fmt.Println("data is nil\n")
                                        return
                                }
                                if data.Finished {
                                        continue
                                }
                                if splited[1] == "my" && User.AddressHex !=
data.Owner.Hex() {

                                        continue

                                }
                                if splited[1] != "all" && splited[1] != "my" &&
                                strings.ToLower(splited[1]) !=
strings.ToLower(data.Owner.Hex()) {

                                        continue

                                }
                                jsonData, err = json.MarshalIndent(data, "", "\t")
case "rents":
                                data := getRents(index)
                                if data == nil {
                                        fmt.Println("data is nil\n")
                                        return
                                }
                                if data.Finished {
                                        continue
                                }
                                if splited[1] == "my" &&
                                (User.AddressHex != data.Owner.Hex() &&
User.AddressHex != data.Renter.Hex()){

```

```

        continue
    }
    if splited[1] != "all" && splited[1] != "my" &&
        (strings.ToLower(splited[1]) !=
strings.ToLower(data.Owner.Hex())) &&
        strings.ToLower(splited[1]) !=
strings.ToLower(data.Renter.Hex())) {
        continue
    }
    jsonData, err = json.MarshalIndent(data, "", "\t")
    default:
        fmt.Println("undefined category\n")
        return
    }
    if err != nil {
        fmt.Println(err, "\n")
        return
    }
    fmt.Println(string(jsonData))
}
fmt.Println()
}

```

Используются функции `getEstates` (2.25), `getPresents` (2.26), `getSales` (2.27), `getRents` (2.38).

(2.14) Функция `chainCreateEstate`.

```

func chainCreateEstate(splited []string) {
    if len(splited) != 5 {
        fmt.Println("failed: len(splited) != 5\n")
        return
    }
    var (
        squere = new(big.Int)
    )
}

```

```

        usefulSquere = new(big.Int)
        ok bool
    )
    squere, ok = squere.SetString(splited[3], 10)
    if !ok {
        fmt.Println("failed: conv(str1) to num\n")
        return
    }
    usefulSquere, ok = usefulSquere.SetString(splited[4], 10)
    if !ok {
        fmt.Println("failed: conv(str2) to num\n")
        return
    }
    var address common.Address
    if splited[1] == "my" {
        address = User.AddressEth
    } else {
        address = common.HexToAddress(splited[1])
    }
    tx, err := Instance.CreateEstate(
        resetAuth(User),
        address,
        splited[2],
        squere,
        usefulSquere,
    )
    if err != nil {
        fmt.Println(err, "\n")
        return
    }
    fmt.Println("Tx:", tx.Hash().Hex(), "\n")
}

```

Используется функция resetAuth (2.29).

(2.15) Функция chainCreatePresent.

```
func chainCreatePresent(splited []string) {
    if len(splited) != 3 {
        fmt.Println("failed: len(splited) != 3\n")
        return
    }
    var (
        estateId = new(big.Int)
        ok bool
    )
    estateId, ok = estateId.SetString(splited[1], 10)
    if !ok {
        fmt.Println("failed: conv(str1) to num\n")
        return
    }
    tx, err := Instance.CreatePresent(
        resetAuth(User),
        estateId,
        common.HexToAddress(splited[2]),
    )
    if err != nil {
        fmt.Println(err, "\n")
        return
    }
    fmt.Println("Tx:", tx.Hash().Hex(), "\n")
}
```

(2.16) Функция chainCreateSale.

```
func chainCreateSale(splited []string) {
    if len(splited) != 3 {
        fmt.Println("failed: len(splited) != 3\n")
        return
    }
}
```

```

var (
    estateId = new(big.Int)
    price    = new(big.Int)
    ok bool
)
estateId, ok = estateId.SetString(splited[1], 10)
if !ok {
    fmt.Println("failed: conv(str1) to num\n")
    return
}
price, ok = price.SetString(splited[2], 10)
if !ok {
    fmt.Println("failed: conv(str2) to num\n")
    return
}
tx, err := Instance.CreateSale(
    resetAuth(User),
    estateId,
    price,
)
if err != nil {
    fmt.Println(err, "\n")
    return
}
fmt.Println("Tx:", tx.Hash().Hex(), "\n")
}

```

(2.17) Функция chainCreateRent.

```

func chainCreateRent(splited []string) {
    if len(splited) != 4 {
        fmt.Println("failed: len(splited) != 4\n")
        return
    }
}

```

```

var (
    estateId = new(big.Int)
    hours    = new(big.Int)
    price    = new(big.Int)
    ok bool
)
estateId, ok = estateId.SetString(splited[1], 10)
if !ok {
    fmt.Println("failed: conv(str1) to num\n")
    return
}
hours, ok = hours.SetString(splited[2], 10)
if !ok {
    fmt.Println("failed: conv(str2) to num\n")
    return
}
price, ok = price.SetString(splited[3], 10)
if !ok {
    fmt.Println("failed: conv(str3) to num\n")
    return
}
tx, err := Instance.CreateRent(
    resetAuth(User),
    estateId,
    hours,
    price,
)
if err != nil {
    fmt.Println(err, "\n")
    return
}
fmt.Println("Tx:", tx.Hash().Hex(), "\n")
}

```

```

func chainCancel(category string, splited []string) {
    if len(splited) != 2 {
        fmt.Println("failed: len(splited) != 2\n")
        return
    }
    var (
        tx *types.Transaction
        err error
        num = new(big.Int)
        ok bool
    )
    num, ok = num.SetString(splited[1], 10)
    if !ok {
        fmt.Println("failed: conv(str1) to num\n")
        return
    }
    switch category {
    case "present":
        tx, err = Instance.CancelPresent(
            resetAuth(User),
            num,
        )
    case "sale":
        tx, err = Instance.CancelSale(
            resetAuth(User),
            num,
        )
    case "rent":
        tx, err = Instance.CancelRent(
            resetAuth(User),
            num,
        )
    }
    if err != nil {
        fmt.Println(err, "\n")
    }
}

```

```

        return
    }
    fmt.Println("Tx:", tx.Hash().Hex(), "\n")
}

```

(2.19) Функция chainConfirmPresent.

```

func chainConfirmPresent(splited []string) {
    if len(splited) != 2 {
        fmt.Println("failed: len(splited) != 2\n")
        return
    }
    var (
        presentNumber = new(big.Int)
        ok bool
    )
    presentNumber, ok = presentNumber.SetString(splited[1], 10)
    if !ok {
        fmt.Println("failed: conv(str1) to num\n")
        return
    }
    tx, err := Instance.ConfirmPresent(
        resetAuth(User),
        presentNumber,
    )
    if err != nil {
        fmt.Println(err, "\n")
        return
    }
    fmt.Println("Tx:", tx.Hash().Hex(), "\n")
}

```

(2.20) Функция chainConfirmSale.

```
func chainConfirmSale(splited []string) {
    if len(splited) != 3 {
        fmt.Println("failed: len(splited) != 3\n")
        return
    }
    var (
        saleNumber = new(big.Int)
        saleTo      = new(big.Int)
        ok bool
    )
    saleNumber, ok = saleNumber.SetString(splited[1], 10)
    if !ok {
        fmt.Println("failed: conv(str1) to num\n")
        return
    }
    saleTo, ok = saleTo.SetString(splited[2], 10)
    if !ok {
        fmt.Println("failed: conv(str2) to num\n")
        return
    }
    tx, err := Instance.ConfirmSale(
        resetAuth(User),
        saleNumber,
        saleTo,
    )
    if err != nil {
        fmt.Println(err, "\n")
        return
    }
    fmt.Println("Tx:", tx.Hash().Hex(), "\n")
}
```

(2.21) Функция chainConfirmRent.

```

func chainConfirmRent(splited []string) {
    if len(splited) != 2 {
        fmt.Println("failed: len(splited) != 2\n")
        return
    }
    var (
        rentNumber = new(big.Int)
        ok bool
    )
    rentNumber, ok = rentNumber.SetString(splited[1], 10)
    if !ok {
        fmt.Println("failed: conv(str1) to num\n")
        return
    }
    rent := getRents(rentNumber)
    if rent == nil {
        fmt.Println("failed: get rent\n")
        return
    }
    auth := resetAuth(User)
    if auth == nil {
        fmt.Println("failed: auth is nil\n")
        return
    }
    auth.Value = rent.Money
    tx, err := Instance.ToRent(
        auth,
        rentNumber,
    )
    if err != nil {
        fmt.Println(err, "\n")
        return
    }
    fmt.Println("Tx:", tx.Hash().Hex(), "\n")
}

```

(2.22) Функция chainTryBuy.

```
func chainTryBuy(splited []string) {
    if len(splited) != 3 {
        fmt.Println("failed: len(splited) != 3\n")
        return
    }
    var (
        saleNumber = new(big.Int)
        value      = new(big.Int)
        ok bool
    )
    saleNumber, ok = saleNumber.SetString(splited[1], 10)
    if !ok {
        fmt.Println("failed: conv(str1) to num\n")
        return
    }
    value, ok = value.SetString(splited[2], 10)
    if !ok {
        fmt.Println("failed: conv(str2) to num\n")
        return
    }
    auth := resetAuth(User)
    if auth == nil {
        fmt.Println("failed: auth is nil\n")
        return
    }
    auth.Value = value
    tx, err := Instance.CheckToBuy(
        auth,
        saleNumber,
    )
    if err != nil {
        fmt.Println(err, "\n")
        return
    }
}
```



```
    }  
    fmt.Println("Tx:", tx.Hash().Hex(), "\n")  
}
```

(2.23) Функция chainCancelBuy.

```
func chainCancelBuy(splited []string) {  
    if len(splited) != 2 {  
        fmt.Println("failed: len(splited) != 2\n")  
        return  
    }  
    var (  
        saleNumber = new(big.Int)  
        ok bool  
    )  
    saleNumber, ok = saleNumber.SetString(splited[1], 10)  
    if !ok {  
        fmt.Println("failed: conv(str1) to num\n")  
        return  
    }  
    tx, err := Instance.CancelToBuy(  
        resetAuth(User),  
        saleNumber,  
    )  
    if err != nil {  
        fmt.Println(err, "\n")  
        return  
    }  
    fmt.Println("Tx:", tx.Hash().Hex(), "\n")  
}
```

(2.24) Функция finishRent.

```

func finishRent(splited []string) {
    if len(splited) != 2 {
        fmt.Println("failed: len(splited) != 2\n")
        return
    }
    var (
        rentNumber = new(big.Int)
        ok bool
    )
    rentNumber, ok = rentNumber.SetString(splited[1], 10)
    if !ok {
        fmt.Println("failed: conv(str1) to num\n")
        return
    }
    tx, err := Instance.FinishRent(
        resetAuth(User),
        rentNumber,
    )
    if err != nil {
        fmt.Println(err, "\n")
        return
    }
    fmt.Println("Tx:", tx.Hash().Hex(), "\n")
}

```

(2.25) Функция getEstates.

```

func getEstates(index *big.Int) *Estate {
    // (*big.Int, common.Address, string, *big.Int, *big.Int, common.Address, error)
    id, owner, info, square, usefulsquare, renteraddress, err :=
Instance.GetEstates(&bind.CallOpts{From: User.AddressEth}, index)
    if err != nil {
        return nil
    }
}

```

```

        presentS, saleS, rentS, err := Instance.GetEstatesStatuses(&bind.CallOpts{From:
User.AddressEth}, index)
        if err != nil {
            return nil
        }
        return &Estate{
            Id: id,
            Owner: owner,
            Info: info,
            Squire: squire,
            UsefulSquire: usefulsquire,
            RenterAddress: renteraddress,
            PresentStatus: presentS,
            SaleStatus: saleS,
            RentStatus: rentS,
        }
    }
}

```

(2.26) Функция getPresents.

```

func getPresents(index *big.Int) *Present {
    // (*big.Int, common.Address, common.Address, bool, error)
    id, from, to, finished, err := Instance.GetPresents(&bind.CallOpts{From:
User.AddressEth}, index)
    if err != nil {
        return nil
    }
    return &Present{
        Id: index,
        EstateId: id,
        AddressFrom: from,
        AddressTo: to,
        Finished: finished,
    }
}

```

```
}
```

(2.27) Функция getSales.

```
func getSales(index *big.Int) *Sale {
    // (*big.Int, common.Address, *big.Int, []common.Address, []*big.Int, bool, error)
    id, owner, price, customers, prices, finished, err :=
Instance.GetSales(&bind.CallOpts{From: User.AddressEth}, index)
    if err != nil {
        return nil
    }
    return &Sale{
        Id: index,
        EstateId: id,
        Owner: owner,
        Price: price,
        Customers: customers,
        Prices: prices,
        Finished: finished,
    }
}
```

(2.28) Функция getRents.

```
func getRents(index *big.Int) *Rent {
    // (*big.Int, common.Address, common.Address, *big.Int, *big.Int, *big.Int, bool,
error)
    id, owner, renter, time, money, deadline, finished, err :=
Instance.GetRents(&bind.CallOpts{From: User.AddressEth}, index)
    if err != nil {
        return nil
    }
    return &Rent{
```

```

        Id: index,
        EstateId: id,
        Owner: owner,
        Renter: renter,
        Time: time,
        Money: money,
        DeadLine: deadline,
        Finished: finished,
    }
}

```

(2.29) Функция resetAuth.

```

func resetAuth(user *UserType) *bind.TransactOpts {
    nonce, err := ClientETH.PendingNonceAt(context.Background(),
crypto.PubkeyToAddress(*user.PublicKey))
    if err != nil {
        return nil
    }

    gasPrice, err := ClientETH.SuggestGasPrice(context.Background())
    if err != nil {
        return nil
    }

    auth := bind.NewKeyedTransactor(user.PrivateKey)
    auth.Nonce = big.NewInt(int64(nonce))
    auth.Value = big.NewInt(0)

    auth.GasLimit = uint64(3000000)
    auth.GasPrice = gasPrice

    return auth
}

```

Используемые пакеты.

```
import (  
    "os"  
    "fmt"  
    "bufio"  
    "context"  
    "strings"  
    "math/big"  
    "encoding/json"  
    "io/ioutil"  
    "crypto/ecdsa"  
    contract "./contracts"  
    "github.com/ethereum/go-ethereum/core/types"  
    "github.com/ethereum/go-ethereum/common"  
    "github.com/ethereum/go-ethereum/accounts/abi/bind"  
    "github.com/ethereum/go-ethereum/crypto"  
    "github.com/ethereum/go-ethereum/ethclient"  
)
```

2.7. Графический интерфейс «Ethereum»

Графический клиент «Ethereum» будет представлять отдельное приложение и потому будет именоваться как «main».

```
package main
```

(3.1) Функция init.

```
func init() {  
    if ClientETH == nil {  
        panic("failed: connect to ETH")  
    }  
    if Instance == nil {  
        panic("failed: instance is nil")  
    }  
}
```

Используются глобальные переменные ClientETH (2.2), Instance (2.3).

(3.2) Функция main.

```
func main() {  
    fmt.Println("Server is running ...")  
  
    http.Handle("/static/", http.StripPrefix(  
        "/static/",  
        handleFileServer(http.Dir(STTC_PATH))),  
    )  
  
    http.HandleFunc("/", indexPage)  
    http.HandleFunc("/login", loginPage)  
    http.HandleFunc("/logout", logoutPage)
```

```

http.HandleFunc("/account", accountPage)

http.HandleFunc("/blockchain", blockchainPage)
http.HandleFunc("/blockchain/estates", blockchainEstatesPage)
http.HandleFunc("/blockchain/presents", blockchainPresentsPage)
http.HandleFunc("/blockchain/sales", blockchainSalesPage)
http.HandleFunc("/blockchain/rents", blockchainRentsPage)

http.HandleFunc("/blockchain/estates/", blockchainEstatesXPage)
http.HandleFunc("/blockchain/presents/", blockchainPresentsXPage)
http.HandleFunc("/blockchain/sales/", blockchainSalesXPage)
http.HandleFunc("/blockchain/rents/", blockchainRentsXPage)

http.HandleFunc("/blockchain/presents/do/", blockchainPresentsDoPage)
http.HandleFunc("/blockchain/sales/do/", blockchainSalesDoPage)
http.HandleFunc("/blockchain/rents/do/", blockchainRentsDoPage)

http.ListenAndServe(":7545", nil)
}

```

Точка входа в программу. Используется константа STTC_PATH (3.3) и функции handleFileServer (1.5), indexPage (3.4), loginPage (3.5), logoutPage (3.6), accountPage (3.7), blockchainPage (3.8), blockchainEstatesPage (3.9), blockchainPresentsPage (3.10), blockchainSalesPage (3.11), blockchainRentsPage (3.12), blockchainEstatesXPage (3.13), blockchainPresentsXPage (3.14), blockchainSalesXPage (3.15), blockchainRentsXPage (3.16), blockchainPresentsDoPage (3.17), blockchainSalesDoPage (3.18), blockchainRentsDoPage (3.19).

(3.3) Константа STTC_PATH.

```

const (
    STTC_PATH = "static/"
)

```

(3.4) Функция indexPage.

```
func indexPage(w http.ResponseWriter, r *http.Request) {
    t, err := template.ParseFiles(
        TMPL_PATH+"base.html",
        TMPL_PATH+"index.html",
    )
    if err != nil {
        panic("can't load hmtl files")
    }
    var data struct{
        User *UserType
    }
    data.User = User
    t.Execute(w, data)
}
```

Используется глобальная переменная TMPL_PATH (3.20) и структура UserType (2.6). Базируется на файлах base.html (3.21) и index.html (3.22).

(3.5) Функция loginPage.

```
func loginPage(w http.ResponseWriter, r *http.Request) {
    t, err := template.ParseFiles(
        TMPL_PATH+"base.html",
        TMPL_PATH+"login.html",
    )
    if err != nil {
        panic("can't load hmtl files")
    }
    var data struct{
        User *UserType
    }
```

```

        Error string
    }
    if r.Method == "POST" {
        r.ParseForm()
        User = loadUser(r.FormValue("private"))
        if User == nil {
            data.Error = "Load Private Key Error"
        } else {
            http.Redirect(w, r, "/", 302)
            return
        }
    }
    data.User = User
    t.Execute(w, data)
}

```

Используется функция loadUser (2.5). Базируется на файлах base.html и login.html (3.23).

(3.6) Функция logoutPage.

```

func logoutPage(w http.ResponseWriter, r *http.Request) {
    User = nil
    http.Redirect(w, r, "/", 302)
}

```

(3.7) Функция accoutPage.

```

func accountPage(w http.ResponseWriter, r *http.Request) {
    t, err := template.ParseFiles(
        TMPL_PATH+"base.html",
        TMPL_PATH+"account.html",
    )
}

```

```

    if err != nil {
        panic("can't load html files")
    }
    var data struct{
        User *UserType
        Address string
        Balance string
    }
    data.User = User
    if data.User != nil {
        data.Address = User.AddressHex
        balance, err := ClientETH.BalanceAt(context.Background(),
User.AddressEth, nil)
        if err == nil {
            data.Balance = balance.String()
        }
    } else {
        http.Redirect(w, r, "/", 302)
        return
    }
    t.Execute(w, data)
}

```

Базируется на файлах base.html и account.html (3.24).

(3.8) Функция blockchainPage.

```

func blockchainPage(w http.ResponseWriter, r *http.Request) {
    t, err := template.ParseFiles(
        TMPL_PATH+"base.html",
        TMPL_PATH+"blockchain.html",
    )
    if err != nil {
        panic("can't load html files")
    }
}

```

```

if User == nil {
    http.Redirect(w, r, "/login", 302)
    return
}
var data struct{
    User *UserType
    IsAdmin bool
    Error string
}
data.User = User
iamAdmin, err := Instance.IamAdmin(&bind.CallOpts{From: User.AddressEth})
if err != nil {
    data.Error = err.Error()
    t.Execute(w, data)
    return
}
if iamAdmin {
    data.IsAdmin = true
}
if r.Method == "POST" {
    r.ParseForm()
    var (
        squire = new(big.Int)
        usefulSquire = new(big.Int)
        ok bool
    )
    squire, ok = squire.SetString(r.FormValue("squire"), 10)
    if !ok {
        data.Error = "strconv error 1"
        t.Execute(w, data)
        return
    }
    usefulSquire, ok = usefulSquire.SetString(r.FormValue("usefulsquire"), 10)
    if !ok {
        data.Error = "strconv error 2"
        t.Execute(w, data)
    }
}

```

```

        return
    }
    _, err := Instance.CreateEstate(
        resetAuth(User),
        User.AddressEth,
        r.FormValue("info"),
        squire,
        usefulSquire,
    )
    if err != nil {
        data.Error = err.Error()
        t.Execute(w, data)
        return
    }
    data.Error = "Success created"
}
t.Execute(w, data)
}

```

Базируется на файлах base.html и blockchain.html (3.25).

(3.9) Функция blockchainEstatesPage.

```

func blockchainEstatesPage(w http.ResponseWriter, r *http.Request) {
    t, err := template.ParseFiles(
        TMPL_PATH+"base.html",
        TMPL_PATH+"estates.html",
    )
    if err != nil {
        panic("can't load hmtl files")
    }
    if User == nil {
        http.Redirect(w, r, "/login", 302)
        return
    }
}

```

```

var data struct{
    Error string
    Blocks []uint64
    Address string
    User *UserType
}
data.User = User
data.Address = User.AddressHex
if r.Method == "POST" {
    data.Address = r.FormValue("address")
}
var inc = big.NewInt(1)
num, err := Instance.GetEstatesNumber(&bind.CallOpts{From: User.AddressEth})
if err != nil {
    data.Error = err.Error()
    t.Execute(w, data)
    return
}
for index := big.NewInt(0); index.Cmp(num) == -1; index.Add(index, inc) {
    block := getEstates(index)
    if block == nil {
        data.Error = "data is nil"
        t.Execute(w, data)
        return
    }
    if data.Address != "all" && strings.ToLower(data.Address) !=
strings.ToLower(block.Owner.Hex()) {
        continue
    }
    data.Blocks = append(data.Blocks, index.Uint64())
}
t.Execute(w, data)
}

```

Базируется на файлах base.html и estates.html (3.26).

(3.10) Функция blockchainPresentsPage.

```
func blockchainPresentsPage(w http.ResponseWriter, r *http.Request) {
    t, err := template.ParseFiles(
        TMPL_PATH+"base.html",
        TMPL_PATH+"presents.html",
    )
    if err != nil {
        panic("can't load hmtl files")
    }
    if User == nil {
        http.Redirect(w, r, "/login", 302)
        return
    }
    var data struct{
        Error string
        Blocks []uint64
        Address string
        User *UserType
    }
    data.User = User
    data.Address = User.AddressHex
    if r.Method == "POST" {
        data.Address = r.FormValue("address")
    }
    var inc = big.NewInt(1)
    num, err := Instance.GetPresentsNumber(&bind.CallOpts{From: User.AddressEth})
    if err != nil {
        data.Error = err.Error()
        t.Execute(w, data)
        return
    }
    for index := big.NewInt(0); index.Cmp(num) == -1; index.Add(index, inc) {
        block := getPresents(index)
        if block == nil {
            data.Error = "data is nil"
        }
    }
}
```

```

        t.Execute(w, data)
        return
    }
    if block.Finished {
        continue
    }
    if data.Address != "all" &&
        strings.ToLower(data.Address) !=
strings.ToLower(block.AddressFrom.Hex()) &&
        strings.ToLower(data.Address) !=
strings.ToLower(block.AddressTo.Hex()) {
        continue
    }
    data.Blocks = append(data.Blocks, index.Uint64())
}
t.Execute(w, data)
}

```

Базируется на файлах base.html и presents.html (3.27).

(3.11) Функция blockchainSalesPage.

```

func blockchainSalesPage(w http.ResponseWriter, r *http.Request) {
    t, err := template.ParseFiles(
        TMPL_PATH+"base.html",
        TMPL_PATH+"sales.html",
    )
    if err != nil {
        panic("can't load hmtl files")
    }
    if User == nil {
        http.Redirect(w, r, "/login", 302)
        return
    }
    var data struct{

```



```

        Error string
        Blocks []uint64
        Address string
        User *UserType
    }
    data.User = User
    data.Address = User.AddressHex
    if r.Method == "POST" {
        data.Address = r.FormValue("address")
    }
    var inc = big.NewInt(1)
    num, err := Instance.GetSalesNumber(&bind.CallOpts{From: User.AddressEth})
    if err != nil {
        data.Error = err.Error()
        t.Execute(w, data)
        return
    }
    for index := big.NewInt(0); index.Cmp(num) == -1; index.Add(index, inc) {
        block := getSales(index)
        if block == nil {
            data.Error = "data is nil"
            t.Execute(w, data)
            return
        }
        if block.Finished {
            continue
        }
        if data.Address != "all" &&
            strings.ToLower(data.Address) !=
strings.ToLower(block.Owner.Hex()) {
            continue
        }
        data.Blocks = append(data.Blocks, index.Uint64())
    }
    t.Execute(w, data)
}

```

Базируется на файлах base.html и sales.html (3.28).

(3.12) Функция blockchainRentsPage.

```
func blockchainRentsPage(w http.ResponseWriter, r *http.Request) {
    t, err := template.ParseFiles(
        TMPL_PATH+"base.html",
        TMPL_PATH+"rents.html",
    )
    if err != nil {
        panic("can't load hmtl files")
    }
    if User == nil {
        http.Redirect(w, r, "/login", 302)
        return
    }
    var data struct{
        Error string
        Blocks []uint64
        Address string
        User *UserType
    }
    data.User = User
    data.Address = User.AddressHex
    if r.Method == "POST" {
        data.Address = r.FormValue("address")
    }
    var inc = big.NewInt(1)
    num, err := Instance.GetRentsNumber(&bind.CallOpts{From: User.AddressEth})
    if err != nil {
        data.Error = err.Error()
        t.Execute(w, data)
        return
    }
}
```

```

for index := big.NewInt(0); index.Cmp(num) == -1; index.Add(index, inc) {
    block := getRents(index)
    if block == nil {
        data.Error = "data is nil"
        t.Execute(w, data)
        return
    }
    if block.Finished {
        continue
    }
    if data.Address != "all" &&
        strings.ToLower(data.Address) !=
strings.ToLower(block.Owner.Hex()) &&
        strings.ToLower(data.Address) !=
strings.ToLower(block.Renter.Hex()) {
        continue
    }
    data.Blocks = append(data.Blocks, index.Uint64())
}

t.Execute(w, data)
}

```

Базируется на файлах base.html и rents.html (3.29).

(3.13) Функция blockchainEstatesXPage.

```

func blockchainEstatesXPage(w http.ResponseWriter, r *http.Request) {
    t, err := template.ParseFiles(
        TMPL_PATH+"base.html",
        TMPL_PATH+"estatesX.html",
    )
    if err != nil {
        panic("can't load hmtl files")
    }
}

```

```

if User == nil {
    http.Redirect(w, r, "/login", 302)
    return
}
var data struct{
    User *UserType
    Block *EstateStr
    Error string
}
data.User = User
var (
    index = new(big.Int)
    ok bool
)
num := strings.Replace(r.URL.Path, "/blockchain/estates/", "", 1)
index, ok = index.SetString(num, 10)
if !ok {
    data.Error = "strconv error"
    t.Execute(w, data)
    return
}
estate := getEstates(index)
if estate == nil {
    data.Error = "estate is nil"
    t.Execute(w, data)
    return
}
data.Block = estatesToString(estate)
t.Execute(w, data)
}

```

Базируется на файлах base.html и estatesX.html (3.30).

(3.14) Функция blockchainPresentsXPage.

```

func blockchainPresentsXPage(w http.ResponseWriter, r *http.Request) {
    t, err := template.ParseFiles(
        TMPL_PATH+"base.html",
        TMPL_PATH+"presentsX.html",
    )
    if err != nil {
        panic("can't load hmtl files")
    }
    if User == nil {
        http.Redirect(w, r, "/login", 302)
        return
    }
    var data struct{
        User *UserType
        Block *PresentStr
        Error string
    }
    data.User = User
    var (
        index = new(big.Int)
        ok bool
    )
    num := strings.Replace(r.URL.Path, "/blockchain/presents/", "", 1)
    index, ok = index.SetString(num, 10)
    if !ok {
        data.Error = "strconv error"
        t.Execute(w, data)
        return
    }
    present := getPresents(index)
    if present == nil {
        data.Error = "present is nil"
        t.Execute(w, data)
        return
    }
    data.Block = presentsToString(present)
}

```

```

if r.Method == "POST" {
    r.ParseForm()
    if r.FormValue("cancel") != "" {
        _, err := Instance.CancelPresent(
            resetAuth(User),
            index,
        )
        if err != nil {
            data.Error = err.Error()
            t.Execute(w, data)
            return
        }
        data.Error = "Success cancel"
    }
    if r.FormValue("confirm") != "" {
        _, err := Instance.ConfirmPresent(
            resetAuth(User),
            index,
        )
        if err != nil {
            data.Error = err.Error()
            t.Execute(w, data)
            return
        }
        data.Error = "Success confirm"
    }
}
t.Execute(w, data)
}

```

Базируется на файлах base.html и presentsX.html (3.31).

(3.15) Функция blockchainSalesXPage.

```

func blockchainSalesXPage(w http.ResponseWriter, r *http.Request) {

```

```

t, err := template.ParseFiles(
    TMPL_PATH+"base.html",
    TMPL_PATH+"salesX.html",
)
if err != nil {
    panic("can't load hmtl files")
}
if User == nil {
    http.Redirect(w, r, "/login", 302)
    return
}
var data struct{
    User *UserType
    Block *SaleStr
    Error string
    InCustomers bool
}
data.User = User
var (
    index = new(big.Int)
    ok bool
)
num := strings.Replace(r.URL.Path, "/blockchain/sales/", "", 1)
index, ok = index.SetString(num, 10)
if !ok {
    data.Error = "strconv error"
    t.Execute(w, data)
    return
}
sale := getSales(index)
if sale == nil {
    data.Error = "sale is nil"
    t.Execute(w, data)
    return
}
data.Block = salesToString(sale)

```

```

for _, cust := range data.Block.Customers {
    if cust == User.AddressHex {
        data.InCustomers = true
        break
    }
}

if r.Method == "POST" {
    r.ParseForm()
    if r.FormValue("cancel") != "" {
        _, err := Instance.CancelSale(
            resetAuth(User),
            index,
        )
        if err != nil {
            data.Error = err.Error()
            t.Execute(w, data)
            return
        }
        data.Error = "Success cancel"
    }
    if r.FormValue("confirm") != "" {
        var (
            saleTo = new(big.Int)
            ok bool
        )
        saleTo, ok = saleTo.SetString(r.FormValue("addrnum"), 10)
        if !ok {
            data.Error = "strconv error"
            t.Execute(w, data)
            return
        }
        _, err := Instance.ConfirmSale(
            resetAuth(User),
            index,
            saleTo,
        )
    }
}

```



```

        if err != nil {
            data.Error = err.Error()
            t.Execute(w, data)
            return
        }
        data.Error = "Success confirm"
    }
    if r.FormValue("buy") != "" {
        var (
            price = new(big.Int)
            ok bool
        )
        price, ok = price.SetString(r.FormValue("price"), 10)
        if !ok {
            data.Error = "strconv error"
            t.Execute(w, data)
            return
        }
        auth := resetAuth(User)
        if auth == nil {
            data.Error = "failed: auth is nil"
            t.Execute(w, data)
            return
        }
        auth.Value = price
        _, err := Instance.CheckToBuy(
            auth,
            index,
        )
        if err != nil {
            data.Error = err.Error()
            t.Execute(w, data)
            return
        }
        data.Error = "Success buy request"
    }
}

```

```

        if r.FormValue("close") != "" {
            _, err := Instance.CancelToBuy(
                resetAuth(User),
                index,
            )
            if err != nil {
                data.Error = err.Error()
                t.Execute(w, data)
                return
            }
            data.Error = "Success close buy request"
        }
    }
    t.Execute(w, data)
}

```

Базируется на файлах base.html и salesX.html (3.32).

(3.16) Функция blockchainRentsXPage.

```

func blockchainRentsXPage(w http.ResponseWriter, r *http.Request) {
    t, err := template.ParseFiles(
        TMPL_PATH+"base.html",
        TMPL_PATH+"rentsX.html",
    )
    if err != nil {
        panic("can't load html files")
    }
    if User == nil {
        http.Redirect(w, r, "/login", 302)
        return
    }
    var data struct{
        User *UserType
        Block *RentStr
    }
}

```

```

        Error string
        Confirmed bool
    }
    data.User = User
    var (
        index = new(big.Int)
        ok bool
    )
    num := strings.Replace(r.URL.Path, "/blockchain/rents/", "", 1)
    index, ok = index.SetString(num, 10)
    if !ok {
        data.Error = "strconv error"
        t.Execute(w, data)
        return
    }
    rent := getRents(index)
    if rent == nil {
        data.Error = "rent is nil"
        t.Execute(w, data)
        return
    }
    data.Block = rentsToString(rent)
    if data.Block.Renter != "0x0000000000000000000000000000000000000000" {
        data.Confirmed = true
    }
    if r.Method == "POST" {
        r.ParseForm()
        if r.FormValue("cancel") != "" {
            _, err := Instance.CancelRent(
                resetAuth(User),
                index,
            )
            if err != nil {
                data.Error = err.Error()
                t.Execute(w, data)
                return
            }
        }
    }

```

```

    }
    data.Error = "Success cancel"
}
if r.FormValue("confirm") != "" {
    auth := resetAuth(User)
    if auth == nil {
        data.Error = "failed: auth is nil"
        t.Execute(w, data)
        return
    }
    auth.Value = rent.Money
    _, err := Instance.ToRent(
        auth,
        index,
    )
    if err != nil {
        data.Error = err.Error()
        t.Execute(w, data)
        return
    }
    data.Error = "Success confirm"
}
if r.FormValue("finish") != "" {
    _, err := Instance.FinishRent(
        resetAuth(User),
        index,
    )
    if err != nil {
        data.Error = err.Error()
        t.Execute(w, data)
        return
    }
    data.Error = "Success finish"
}
}
t.Execute(w, data)

```

```
}
```

Базируется на файлах base.html и rentsX.html (3.33).

(3.17) Функция blockchainPresentsDoPage.

```
func blockchainPresentsDoPage(w http.ResponseWriter, r *http.Request) {
    t, err := template.ParseFiles(
        TMPL_PATH+"base.html",
        TMPL_PATH+"presentsDo.html",
    )
    if err != nil {
        panic("can't load hmtl files")
    }
    if User == nil {
        http.Redirect(w, r, "/login", 302)
        return
    }
    var data struct{
        User *UserType
        Block *EstateStr
        Error string
    }
    data.User = User
    var (
        index = new(big.Int)
        ok bool
    )
    num := strings.Replace(r.URL.Path, "/blockchain/presents/do/", "", 1)
    index, ok = index.SetString(num, 10)
    if !ok {
        data.Error = "strconv error"
        t.Execute(w, data)
        return
    }
}
```

```

estate := getEstates(index)
if estate == nil {
    data.Error = "estate is nil"
    t.Execute(w, data)
    return
}
data.Block = estatesToString(estate)
if r.Method == "POST" {
    r.ParseForm()
    _, err := Instance.CreatePresent(
        resetAuth(User),
        index,
        common.HexToAddress(r.FormValue("address")),
    )
    if err != nil {
        data.Error = err.Error()
        t.Execute(w, data)
        return
    }
    data.Error = "Success created"
}
t.Execute(w, data)
}

```

Базируется на файлах base.html и presentsDo.html (3.34).

(3.18) Функция blockchainSalesDoPage.

```

func blockchainSalesDoPage(w http.ResponseWriter, r *http.Request) {
    t, err := template.ParseFiles(
        TMPL_PATH+"base.html",
        TMPL_PATH+"salesDo.html",
    )
    if err != nil {
        panic("can't load hmtl files")
    }
}

```

```

}
if User == nil {
    http.Redirect(w, r, "/login", 302)
    return
}
var data struct{
    User *UserType
    Block *EstateStr
    Error string
}
data.User = User
var (
    index = new(big.Int)
    ok bool
)
num := strings.Replace(r.URL.Path, "/blockchain/sales/do/", "", 1)
index, ok = index.SetString(num, 10)
if !ok {
    data.Error = "strconv error"
    t.Execute(w, data)
    return
}
estate := getEstates(index)
if estate == nil {
    data.Error = "estate is nil"
    t.Execute(w, data)
    return
}
data.Block = estatesToString(estate)
if r.Method == "POST" {
    r.ParseForm()
    var (
        price = new(big.Int)
        ok bool
    )
    price, ok = price.SetString(r.FormValue("price"), 10)

```

```

        if !ok {
            data.Error = "strconv error"
            t.Execute(w, data)
            return
        }
        _, err := Instance.CreateSale(
            resetAuth(User),
            index,
            price,
        )
        if err != nil {
            data.Error = err.Error()
            t.Execute(w, data)
            return
        }
        data.Error = "Success created"
    }
    t.Execute(w, data)
}

```

Базируется на файлах base.html и salesDo.html (3.35).

(3.19) Функция blockchainRentsDoPage.

```

func blockchainRentsDoPage(w http.ResponseWriter, r *http.Request) {
    t, err := template.ParseFiles(
        TMPL_PATH+"base.html",
        TMPL_PATH+"rentsDo.html",
    )
    if err != nil {
        panic("can't load hmtl files")
    }
    if User == nil {
        http.Redirect(w, r, "/login", 302)
        return
    }
}

```



```

}
var data struct{
    User *UserType
    Block *EstateStr
    Error string
}
data.User = User
var (
    index = new(big.Int)
    ok bool
)
num := strings.Replace(r.URL.Path, "/blockchain/rents/do/", "", 1)
index, ok = index.SetString(num, 10)
if !ok {
    data.Error = "strconv error"
    t.Execute(w, data)
    return
}
rent := getEstates(index)
if rent == nil {
    data.Error = "rent is nil"
    t.Execute(w, data)
    return
}
data.Block = estatesToString(rent)
if r.Method == "POST" {
    r.ParseForm()
    var (
        days = new(big.Int)
        price = new(big.Int)
        ok bool
    )
    days, ok = days.SetString(r.FormValue("days"), 10)
    if !ok {
        data.Error = "strconv error"
        t.Execute(w, data)
    }
}

```

```

        return
    }
    price, ok = price.SetString(r.FormValue("price"), 10)
    if !ok {
        data.Error = "strconv error"
        t.Execute(w, data)
        return
    }
    _, err := Instance.CreateRent(
        resetAuth(User),
        index,
        days,
        price,
    )
    if err != nil {
        data.Error = err.Error()
        t.Execute(w, data)
        return
    }
    data.Error = "Success created"
}
t.Execute(w, data)
}

```

Базируется на файлах base.html и rentsDo.html (3.36).

(3.20) Константа TMPL_PATH.

```

const (
    TMPL_PATH = "templates_eth/"
)

```

Используемые пакеты.

```

import (
    "os"
    "fmt"
    "strings"
    "context"
    "net/http"
    "math/big"
    "html/template"
    "io/ioutil"
    "crypto/ecdsa"
    contract "./contracts"
    "github.com/ethereum/go-ethereum/accounts/abi/bind"
    "github.com/ethereum/go-ethereum/common"
    "github.com/ethereum/go-ethereum/crypto"
    "github.com/ethereum/go-ethereum/common"
    "github.com/ethereum/go-ethereum/ethclient"
)

```

(3.21) Файл base.html.

```

<!DOCTYPE html>
<html>
<head>
    <title>
        {{block "title" .}}
        Title
        {{end}}
    </title>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css" href="/static/css/bootstrap.css">
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
            <div class="container">

```

```

<a href="/" class="navbar-brand" exact><h4>Home</h4></a>
<div class="navbar-collapse">
  <ul class="navbar-nav">
    <a href="/blockchain" class="nav-link"><h5>Blockchain</h5></a>
  </ul>
  <ul class="navbar-nav ml-auto">
    {{ if (not .User) }}
      <a href="/login" class="nav-link"><h5>Login</h5></a>
    {{ else }}
      <a href="/account" class="nav-link"><h5>Account</h5></a>
      <a href="/logout" class="nav-link"><h5>Logout</h5></a>
    {{ end }}
  </ul>
</div>
</div>
</nav>
</header>
<main>
  <div class="col-md-9 mx-auto">
    {{block "content" .}}
    Content
    {{end}}
  </div>
</main>
</body>
</html>

```

(3.22) Файл index.html.

```

{{define "title"}}
  Index
{{end}}
{{define "content"}}
  <div class="jumbotron">

```

```
<h1 class="text-center">Blockchain</h1>
</div>
{{end}}
```

(3.23) Файл login.html.

```
{{define "title"}}
    Login
{{end}}
{{define "content"}}
    {{ if .Error }}
        <div class="jumbotron">
            <p>{{ .Error }}</p>
        </div>
    {{ else }}
        <div class="jumbotron">
            <div class="col-10 mx-auto">
                <form method="POST" action="/login">
                    <div class="form-group">
                        <input type="password" class="form-control" name="private" placeholder="Private
Key">
                    </div>
                    <input type="submit" class="btn btn-success w-100" name="login" value="Login">
                </form>
            </div>
        </div>
    {{ end }}
{{end}}
```

(3.24) Файл account.html.

```
{{define "title"}}
    Account
```

```

{{end}}
{{define "content"}}
  <div class="jumbotron">
    <div class="col-12 mx-auto">
      <form>
        <div class="form-group">
          <input id="address" readonly class="form-control bg-light" type="text" name="coins"
value="Address: {{ .Address }}">
        </div>
        <div class="form-group">
          <input readonly class="form-control bg-light" type="text" name="coins"
value="Balance: {{ if .Balance }} {{ .Balance }} {{ else }} 0 {{ end }} coins;">
        </div>
      </form>
    </div>
  </div>
{{end}}

```

(3.25) Файл blockchain.html.

```

{{define "title"}}
  Chain
{{end}}
{{define "content"}}
  {{ if .Error }}
    <div class="jumbotron">
      <p>{{ .Error }}</p>
    </div>
  {{ end }}
  {{ if .IsAdmin }}
    <div class="jumbotron">
      <div class="col-10 mx-auto">
        <form method="POST" action="/blockchain">
          <div class="form-group">
            <input class="form-control" type="text" name="info" placeholder="Information">

```

```

    </div>
    <div class="form-group">
        <input class="form-control" type="number" name="square" placeholder="Square">
    </div>
    <div class="form-group">
        <input class="form-control" type="number" name="usefulsquare"
placeholder="Useful Square">
    </div>
    <input type="submit" class="btn btn-success w-100" name="submit" value="Create">
</form>
</div>
</div>
{{ end }}
<div class="jumbotron">
    <div class="card">
        <a class="btn btn-info" href="/blockchain/estates">Estates</a>
    </div>
    <div class="card">
        <a class="btn btn-info" href="/blockchain/presents">Presents</a>
    </div>
    <div class="card">
        <a class="btn btn-info" href="/blockchain/sales">Sales</a>
    </div>
    <div class="card">
        <a class="btn btn-info" href="/blockchain/rents">Rents</a>
    </div>
</div>
{{end}}

```

(3.26) Файл estates.html.

```

{{define "title"}}
    Estates
{{end}}
{{define "content"}}

```

```

<div class="jumbotron">
  <div class="col-12 mx-auto">
    <form method="POST" action="/blockchain/estates">
      <div class="form-group">
        {{ if .Address }}
          <input readonly class="form-control bg-light" type="text" name="coins"
value="Address: {{ .Address }}">
        {{ end }}
      </div>
      <div class="form-group">
        <input type="text" class="form-control" name="address" placeholder="Address">
      </div>
      <input type="submit" class="btn btn-success w-100" name="submit" value="Get
estates">
    </form>
  </div>
</div>
<div class="jumbotron">
  {{ if .Error }}
    {{ .Error }}
  {{ else }}
    {{ range $i, $e := .Blocks }}
      <div class="card">
        <a class="btn btn-info" href="/blockchain/estates/{{ $e }}">[Estate {{ $e }}]</a>
      </div>
    {{ end }}
  {{ end }}
</div>
{{end}}

```

(3.27) Файл presents.html.

```

{{define "title"}}
  Presents
{{end}}

```



```

{{define "content"}}
<div class="jumbotron">
  <div class="col-12 mx-auto">
    <form method="POST" action="/blockchain/presents">
      <div class="form-group">
        {{ if .Address }}
          <input readonly class="form-control bg-light" type="text" name="coins"
value="Address: {{ .Address }}">
        {{ end }}
      </div>
      <div class="form-group">
        <input type="text" class="form-control" name="address" placeholder="Address">
      </div>
      <input type="submit" class="btn btn-success w-100" name="submit" value="Get
presents">
    </form>
  </div>
</div>
<div class="jumbotron">
  {{ if .Error }}
    <p>{{ .Error }}</p>
  {{ else }}
    {{ range $i, $e := .Blocks }}
      <div class="card">
        <a class="btn btn-info" href="/blockchain/presents/{{ $e }}">[Presents {{ $e }}]</a>
      </div>
    {{ end }}
  {{ end }}
</div>
{{end}}

```

(3.28) Файл sales.html.

```

{{define "title"}}

```

```

Sales
{{end}}
{{define "content"}}
<div class="jumbotron">
  <div class="col-12 mx-auto">
    <form method="POST" action="/blockchain/sales">
      <div class="form-group">
        {{ if .Address }}
          <input readonly class="form-control bg-light" type="text" name="coins"
value="Address: {{ .Address }}">
        {{ end }}
      </div>
      <div class="form-group">
        <input type="text" class="form-control" name="address" placeholder="Address">
      </div>
      <input type="submit" class="btn btn-success w-100" name="submit" value="Get sales">
    </form>
  </div>
</div>
<div class="jumbotron">
  {{ if .Error }}
    <p>{{ .Error }}</p>
  {{ else }}
    {{ range $i, $e := .Blocks }}
      <div class="card">
        <a class="btn btn-info" href="/blockchain/sales/{{ $e }}">[Sales {{ $e }}]</a>
      </div>
    {{ end }}
  {{ end }}
</div>
{{end}}

```

(3.29) Файл rents.html.

```

{{define "title"}}
    Rents
{{end}}
{{define "content"}}
    <div class="jumbotron">
        <div class="col-12 mx-auto">
            <form method="POST" action="/blockchain/rents">
                <div class="form-group">
                    {{ if .Address }}
                        <input readonly class="form-control bg-light" type="text" name="coins"
value="Address: {{ .Address }}">
                    {{ end }}
                </div>
                <div class="form-group">
                    <input type="text" class="form-control" name="address" placeholder="Address">
                </div>
                <input type="submit" class="btn btn-success w-100" name="submit" value="Get rents">
            </form>
        </div>
    </div>
    <div class="jumbotron">
        {{ if .Error }}
            Error: {{ .Error }}
        {{ else }}
            {{ range $i, $e := .Blocks }}
                <div class="card">
                    <a class="btn btn-info" href="/blockchain/rents/{{ $e }}">[Rents {{ $e }}]</a>
                </div>
            {{ end }}
        {{ end }}
    </div>
{{end}}

```

```

{{define "title"}}
    Estate
{{end}}
{{define "content"}}
    {{ if .Error }}
        <div class="jumbotron">
            <p>{{ .Error }}</p>
        </div>
    {{ else }}
        {{ if (and (eq .Block.Owner .User.AddressHex) (not .Block.PresentStatus) (not
.Block.SaleStatus) (not .Block.RentStatus)) }}
            <div class="jumbotron">
                <div class="card">
                    <a class="btn btn-info" href="/blockchain/presents/do/{{ .Block.Id }}">Do
present</a>
                </div>
                <div class="card">
                    <a class="btn btn-info" href="/blockchain/sales/do/{{ .Block.Id }}">Do sales</a>
                </div>
                <div class="card">
                    <a class="btn btn-info" href="/blockchain/rents/do/{{ .Block.Id }}">Do rents</a>
                </div>
            </div>
        {{ end }}
        <table border="1">
            <tr>
                <th>Id</th>
                <td width="100%">{{ .Block.Id }}</td>
            </tr>
            <tr>
                <th>Owner</th>
                <td width="100%">{{ .Block.Owner }}</td>
            </tr>
            <tr>
                <th>Info</th>
                <td width="100%">{{ .Block.Info }}</td>
            </tr>
        </table>
    {{ end }}

```

```

</tr>
<tr>
  <th>Square</th>
  <td width="100%">{{ .Block.Square }}</td>
</tr>
<tr>
  <th>UsefulSquare</th>
  <td width="100%">{{ .Block.UsefulSquare }}</td>
</tr>
<tr>
  <th>RenterAddress</th>
  <td width="100%">{{ .Block.RenterAddress }}</td>
</tr>
<tr>
  <th>PresentStatus</th>
  <td width="100%">{{ .Block.PresentStatus }}</td>
</tr>
<tr>
  <th>SaleStatus</th>
  <td width="100%">{{ .Block.SaleStatus }}</td>
</tr>
<tr>
  <th>RentStatus</th>
  <td width="100%">{{ .Block.RentStatus }}</td>
</tr>
</table>
{{ end }}
{{end}}

```

(3.31) Файл presentsX.html.

```

{{define "title"}}
  Present
{{end}}
{{define "content"}}

```

```

{{ if .Error }}
  <div class="jumbotron">
    <p>{{ .Error }}</p>
  </div>
{{ else }}
  {{ if (eq .Block.AddressFrom .User.AddressHex )}}
    <div class="jumbotron">
      <div class="col-10 mx-auto">
        <form method="POST" action="/blockchain/presents/{{ .Block.Id }}">
          <input type="submit" class="btn btn-success w-100" name="cancel"
value="Cancel">
        </form>
      </div>
    </div>
  {{ end }}
  {{ if (eq .Block.AddressTo .User.AddressHex )}}
    <div class="jumbotron">
      <div class="col-10 mx-auto">
        <form method="POST" action="/blockchain/presents/{{ .Block.Id }}">
          <input type="submit" class="btn btn-success w-100" name="confirm"
value="Confirm">
        </form>
      </div>
    </div>
  {{ end }}
  <table border="1">
    <tr>
      <th>Id</th>
      <td width="100%">{{ .Block.Id }}</td>
    </tr>
    <tr>
      <th>EstateId</th>
      <td width="100%">{{ .Block.EstateId }}</td>
    </tr>
    <tr>
      <th>AddressFrom</th>

```

```

        <td width="100%">{{ .Block.AddressFrom }}</td>
    </tr>
    <tr>
        <th>AddressTo</th>
        <td width="100%">{{ .Block.AddressTo }}</td>
    </tr>
    <tr>
        <th>Finished</th>
        <td width="100%">{{ .Block.Finished }}</td>
    </tr>
</table>

{{ end }}
{{end}}

```

(3.32) Файл salesX.html.

```

{{define "title"}}
    Sale
{{end}}
{{define "content"}}
    {{ if .Error }}
        <div class="jumbotron">
            <p>{{ .Error }}</p>
        </div>
    {{ else }}
        {{ if (eq .Block.Owner .User.AddressHex) }}
            <div class="jumbotron">
                <div class="col-10 mx-auto">
                    <form method="POST" action="/blockchain/sales/{{ .Block.Id }}">
                        <input type="submit" class="btn btn-success w-100" name="cancel"
value="Cancel">
                    </form>
                </div>
            </div>
        {{ else }}
            <div class="jumbotron">
                <div class="col-10 mx-auto">
                    <form method="POST" action="/blockchain/sales/{{ .Block.Id }}">
                        <input type="submit" class="btn btn-success w-100" name="cancel"
value="Cancel">
                    </form>
                </div>
            </div>
        {{ end }}
    {{ end }}

```

```

    <form id="my_form" method="POST" action="/blockchain/sales/{{ .Block.Id }}">
    </form>
{{ else }}
    {{ if .InCustomers }}
        <div class="jumbotron">
            <div class="col-10 mx-auto">
                <form method="POST" action="/blockchain/sales/{{ .Block.Id }}">
                    <input type="submit" class="btn btn-success w-100" name="close"
value="Close">
                </form>
            </div>
        </div>
    {{ else }}
        <div class="jumbotron">
            <div class="col-10 mx-auto">
                <form method="POST" action="/blockchain/sales/{{ .Block.Id }}">
                    <div class="form-group">
                        <input type="number" class="form-control" name="price" value="{{
.Block.Price }}">
                    </div>
                    <input type="submit" class="btn btn-success w-100" name="buy" value="Buy">
                </form>
            </div>
        </div>
    {{ end }}
{{ end }}
<table border="1">
<tr>
    <th>Id</th>
    <td width="100%">{{ .Block.Id }}</td>
</tr>
<tr>
    <th>EstateId</th>
    <td width="100%">{{ .Block.EstateId }}</td>
</tr>
<tr>

```



```

    <th>Owner</th>
    <td width="100%">{{ .Block.Owner }} - {{ .User.AddressHex }} </td>
</tr>
<tr>
    <th>Price</th>
    <td width="100%">{{ .Block.Price }}</td>
</tr>
<tr>
    <th>Customers</th>
    <td>
        {{ $owner := .Block.Owner }}
        {{ $useraddr := .User.AddressHex }}
        {{ $prices := .Block.Prices }}
        {{ range $i, $e := .Block.Customers }}
            <table border="1">
                <tr>
                    <th>Customer</th>
                    <td width="100%">{{ $e }}</td>
                </tr>
                <tr>
                    <th>Price</th>
                    <td width="100%">{{ index $prices $i }}</td>
                </tr>
                {{ if (eq $owner $useraddr) }}
                    <tr>
                        <th>Confirm</th>
                        <td width="100%">
                            <input form="my_form" type="hidden" name="addrnum" value="{{ $i
}}} ">
                                <input form="my_form" type="submit" class="btn btn-success w-100"
name="confirm" value="Confirm">
                            </td>
                        </tr>
                    {{ end }}
                </table>
            {{ end }}
        </td>
    </tr>
    {{ end }}
</table>
{{ end }}

```

```

        </td>
    </tr>
    <tr>
        <th>Finished</th>
        <td width="100%">{{ .Block.Finished }}</td>
    </tr>
</table>
{{ end }}
{{end}}

```

(3.33) Файл rentsX.html.

```

{{define "title"}}
    Rent
{{end}}
{{define "content"}}
    {{ if .Error }}
        <div class="jumbotron">
            <p>{{ .Error }}</p>
        </div>
    {{ else }}
        {{ if (eq .Block.Owner .User.AddressHex) }}
            {{ if (not .Confirmed) }}
                <div class="jumbotron">
                    <div class="col-10 mx-auto">
                        <form method="POST" action="/blockchain/rents/{{ .Block.Id }}">
                            <input type="submit" class="btn btn-success w-100" name="cancel"
value="Cancel">
                        </form>
                    </div>
                </div>
            {{ else }}
                <div class="jumbotron">
                    <div class="col-10 mx-auto">

```

```

        <form method="POST" action="/blockchain/rents/{{ .Block.Id }}">
            <input type="submit" class="btn btn-success w-100" name="finish"
value="Finish">
        </form>
    </div>
</div>
{{ end }}
{{ else }}
    {{ if (eq .Block.Renter "0x0000000000000000000000000000000000000000") }}
        <div class="jumbotron">
            <div class="col-10 mx-auto">
                <form method="POST" action="/blockchain/rents/{{ .Block.Id }}">
                    <input type="submit" class="btn btn-success w-100" name="confirm"
value="Confirm">
                </form>
            </div>
        </div>
    {{ end }}
{{ end }}
<table border="1">
<tr>
    <th>Id</th>
    <td width="100%">{{ .Block.Id }}</td>
</tr>
<tr>
    <th>EstateId</th>
    <td width="100%">{{ .Block.EstateId }}</td>
</tr>
<tr>
    <th>Owner</th>
    <td width="100%">{{ .Block.Owner }}</td>
</tr>
<tr>
    <th>Renter</th>
    <td width="100%">{{ .Block.Renter }}</td>
</tr>

```

```

<tr>
  <th>Time</th>
  <td width="100%">{{ .Block.Time }}</td>
</tr>
<tr>
  <th>Money</th>
  <td width="100%">{{ .Block.Money }}</td>
</tr>
<tr>
  <th>DeadLine</th>
  <td width="100%">{{ .Block.DeadLine }}</td>
</tr>
<tr>
  <th>Finished</th>
  <td width="100%">{{ .Block.Finished }}</td>
</tr>
</table>
{{ end }}
{{end}}

```

(3.34) Файл presentsDo.html.

```

{{define "title"}}
  Present
{{end}}
{{define "content"}}
  {{ if .Error }}
    <div class="jumbotron">
      <p>{{ .Error }}</p>
    </div>
  {{ else }}
    {{ if (eq .Block.Owner .User.AddressHex) }}
      <div class="jumbotron">
        <div class="col-10 mx-auto">

```

```

    <form method="POST" action="/blockchain/presents/do/{{ .Block.Id }}">
      <div class="form-group">
        <input type="text" class="form-control" name="address" placeholder="Address">
      </div>
      <input type="submit" class="btn btn-success w-100" name="submit" value="Send
present">
    </form>
  </div>
</div>
{{ end }}

<table border="1">
<tr>
  <th>Id</th>
  <td width="100%">{{ .Block.Id }}</td>
</tr>
<tr>
  <th>Owner</th>
  <td width="100%">{{ .Block.Owner }}</td>
</tr>
<tr>
  <th>Info</th>
  <td width="100%">{{ .Block.Info }}</td>
</tr>
<tr>
  <th>Square</th>
  <td width="100%">{{ .Block.Square }}</td>
</tr>
<tr>
  <th>UsefulSquare</th>
  <td width="100%">{{ .Block.UsefulSquare }}</td>
</tr>
<tr>
  <th>RenterAddress</th>
  <td width="100%">{{ .Block.RenterAddress }}</td>
</tr>
<tr>

```

```

        <th>PresentStatus</th>
        <td width="100%">{{ .Block.PresentStatus }}</td>
    </tr>
    <tr>
        <th>SaleStatus</th>
        <td width="100%">{{ .Block.SaleStatus }}</td>
    </tr>
    <tr>
        <th>RentStatus</th>
        <td width="100%">{{ .Block.RentStatus }}</td>
    </tr>
</table>
{{ end }}
{{end}}

```

(3.35) Файл salesDo.html.

```

{{define "title"}}
    Sale
{{end}}
{{define "content"}}
    {{ if .Error }}
        <div class="jumbotron">
            <p>{{ .Error }}</p>
        </div>
    {{ else }}
        {{ if (eq .Block.Owner .User.AddressHex )}}
            <div class="jumbotron">
                <div class="col-10 mx-auto">
                    <form method="POST" action="/blockchain/sales/do/{{ .Block.Id }}">
                        <div class="form-group">
                            <input type="number" class="form-control" name="price" placeholder="Price">
                        </div>

```

```

        <input type="submit" class="btn btn-success w-100" name="submit" value="Do
sale">

        </form>

    </div>

</div>
{{ end }}

<table border="1">

<tr>

    <th>Id</th>

    <td width="100%">{{ .Block.Id }}</td>

</tr>

<tr>

    <th>Owner</th>

    <td width="100%">{{ .Block.Owner }}</td>

</tr>

<tr>

    <th>Info</th>

    <td width="100%">{{ .Block.Info }}</td>

</tr>

<tr>

    <th>Square</th>

    <td width="100%">{{ .Block.Square }}</td>

</tr>

<tr>

    <th>UsefulSquare</th>

    <td width="100%">{{ .Block.UsefulSquare }}</td>

</tr>

<tr>

    <th>RenterAddress</th>

    <td width="100%">{{ .Block.RenterAddress }}</td>

</tr>

<tr>

    <th>PresentStatus</th>

    <td width="100%">{{ .Block.PresentStatus }}</td>

</tr>

<tr>

```

```

        <th>SaleStatus</th>
        <td width="100%">{{ .Block.SaleStatus }}</td>
    </tr>
    <tr>
        <th>RentStatus</th>
        <td width="100%">{{ .Block.RentStatus }}</td>
    </tr>
</table>
{{ end }}
{{end}}

```

(3.36) Файл rentsDo.html.

```

{{define "title"}}
    Rent
{{end}}
{{define "content"}}
    {{ if .Error }}
        <div class="jumbotron">
            <p>{{ .Error }}</p>
        </div>
    {{ else }}
        {{ if (eq .Block.Owner .User.AddressHex) }}
            <div class="jumbotron">
                <div class="col-10 mx-auto">
                    <form method="POST" action="/blockchain/rents/do/{{ .Block.Id }}">
                        <div class="form-group">
                            <input type="number" class="form-control" name="days" placeholder="Days">
                        </div>
                        <div class="form-group">
                            <input type="number" class="form-control" name="price" placeholder="Price">
                        </div>
                        <input type="submit" class="btn btn-success w-100" name="submit" value="Do
rent">

```



```

        </form>
    </div>
</div>
{{ end }}
<table border="1">
<tr>
    <th>Id</th>
    <td width="100%">{{ .Block.Id }}</td>
</tr>
<tr>
    <th>Owner</th>
    <td width="100%">{{ .Block.Owner }}</td>
</tr>
<tr>
    <th>Info</th>
    <td width="100%">{{ .Block.Info }}</td>
</tr>
<tr>
    <th>Square</th>
    <td width="100%">{{ .Block.Square }}</td>
</tr>
<tr>
    <th>UsefulSquare</th>
    <td width="100%">{{ .Block.UsefulSquare }}</td>
</tr>
<tr>
    <th>RenterAddress</th>
    <td width="100%">{{ .Block.RenterAddress }}</td>
</tr>
<tr>
    <th>PresentStatus</th>
    <td width="100%">{{ .Block.PresentStatus }}</td>
</tr>
<tr>
    <th>SaleStatus</th>
    <td width="100%">{{ .Block.SaleStatus }}</td>

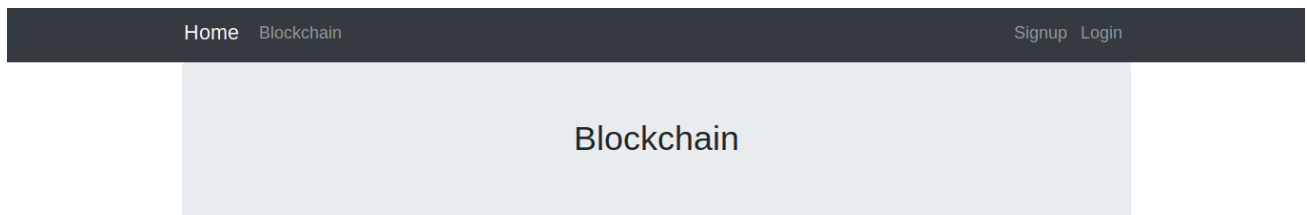
```

```
</tr>
<tr>
  <th>RentStatus</th>
  <td width="100%">{{ .Block.RentStatus }}</td>
</tr>
</table>
{{ end }}
{{end}}
```

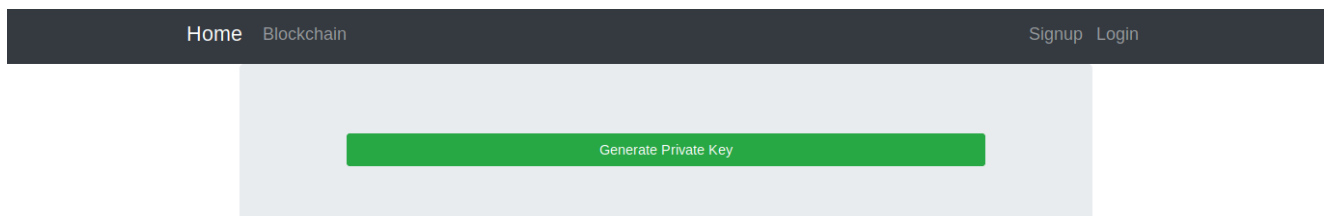
3. Скриншоты

3.1. «Программирование узла блокчейн»

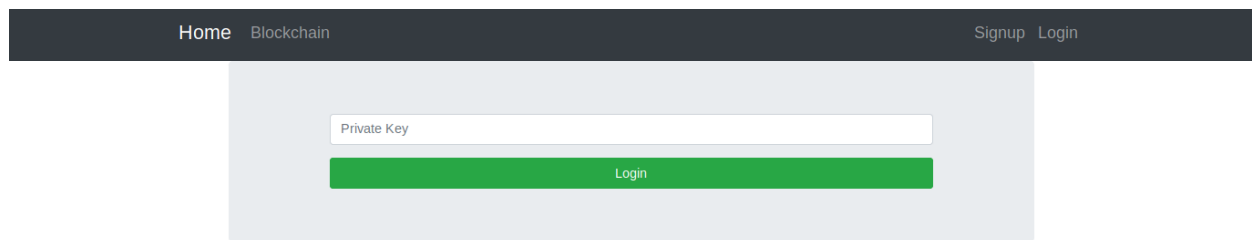
(1) Главная страница.



(2) Страница регистрации (генерации приватного ключа).

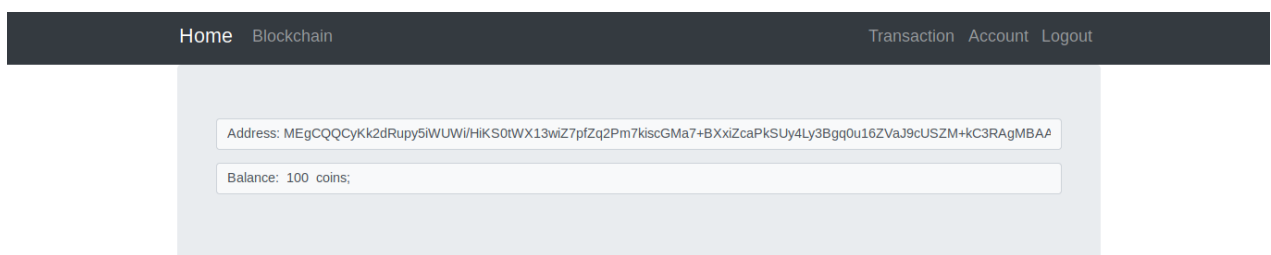


(3) Страница авторизации (ввод приватного ключа).



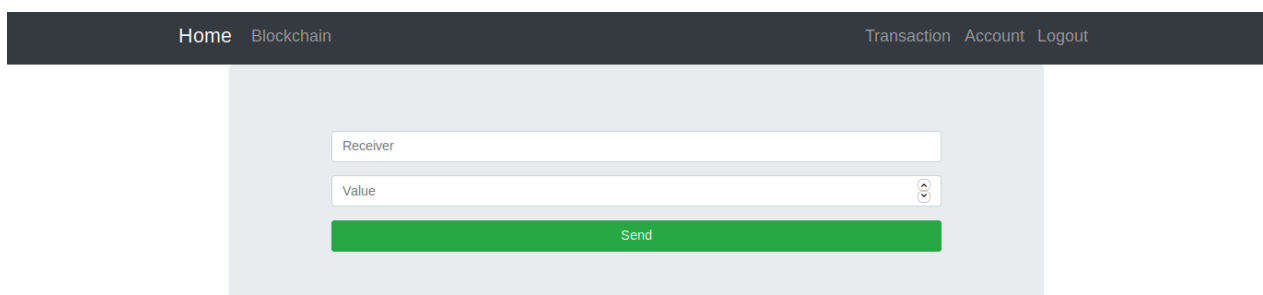
The screenshot shows a web application interface for authorization. At the top, a dark navigation bar contains the links 'Home' and 'Blockchain' on the left, and 'Signup' and 'Login' on the right. The main content area has a light gray background and features a white input field labeled 'Private Key'. Below this field is a prominent green button labeled 'Login'.

(4) Аккаунт пользователя.



The screenshot displays a user account page. The top navigation bar includes 'Home' and 'Blockchain' on the left, and 'Transaction', 'Account', and 'Logout' on the right. The main content area is light gray and contains two white boxes. The first box displays the text 'Address: MEgCQQCyKk2dRupy5iWUWi/iHiKS0tWX13wiZ7pfZq2Pm7kiscGma7+BXxiZcaPkSUy4Ly3Bgq0u16ZVaJ9cUSZM+kC3RAgMBA^'. The second box displays 'Balance: 100 coins;'.

(5) Формирование транзакций.



The screenshot shows a page for creating transactions. The top navigation bar has 'Home' and 'Blockchain' on the left, and 'Transaction', 'Account', and 'Logout' on the right. The main content area is light gray and contains two white input fields. The first field is labeled 'Receiver' and the second is labeled 'Value'. Below these fields is a green button labeled 'Send'.

(6) Информация о блокчейне.

HomeBlockchainTransactionAccountLogout

Address

Balance

[Block 0]

[Block 1]

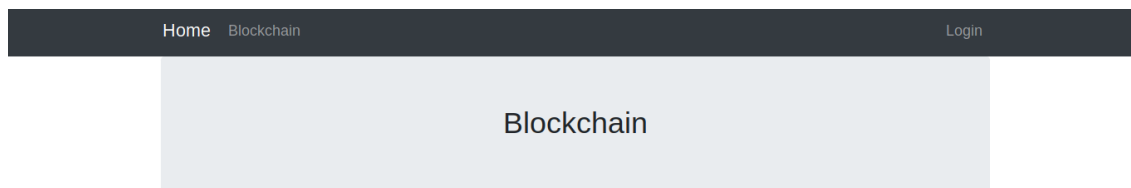
[Block 2]

(7) Информация о блоке.

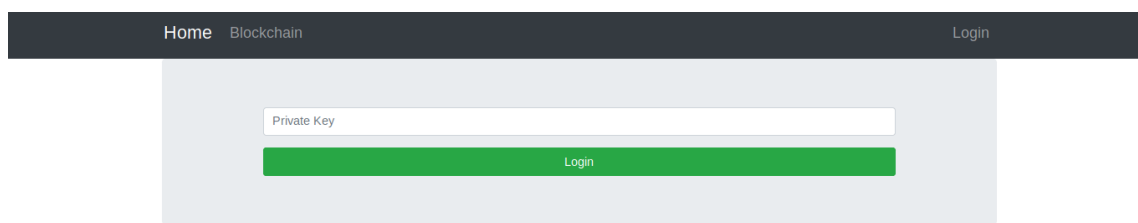
	HomeBlockchainTransactionAccountLogout
Nonce	2011578523
Difficulty	20
CurrHash	[12 78 69 132 168 51 73 67 181 253 226 230 229 122 31 176 133 228 105 139 152 132 1 178 137 0 58 192 248 28 14 106]
PrevHash	[71 69 78 69 83 73 83 45 66 76 79 67 75]
Miner	MEgCQQCyKk2dRupy5iWUWiHiKS0tWX13wiZ7pfZq2Pm7kiscGMA7+BXxiZcaPkSUy4Ly3Bgq0u16ZVaJ9cUSZM+kC3RAgMBAAE=
Signature	[56 96 127 186 233 60 205 107 206 104 193 147 50 85 173 202 230 226 23 14 152 246 188 231 238 5 118 62 26 90 221 146 121 181 253 84 128 8 92 100 16 96 49 46 184 143 208 236 32 51 250 136 173 72 145 22 78 247 24 126 13 119 122 80]
TimeStamp	2020-07-26T13:39:43-04:00
Transactions	RandBytes[230 177 129 140 220 158 16 90 53 168 99 98 228 246 179 53 121 99 24 159 206 110 173 222 251 204 68 23 245 81 111 140]
	PrevBlock[71 69 78 69 83 73 83 45 66 76 79 67 75]
	SenderMEgCQQCyKk2dRupy5iWUWiHiKS0tWX13wiZ7pfZq2Pm7kiscGMA7+BXxiZcaPkSUy4Ly3Bgq0u16ZVaJ9cUSZM+kC3RAgMBAAE=
	Receiveraaa
	Value2
	ToStorage0
	CurrHash[170 177 242 187 158 125 138 20 57 107 173 237 31 41 202 233 137 198 103 72 128 240 56 234 101 90 170 71 24 205 2 43]
	Signature[33 37 142 176 32 50 162 120 252 107 35 127 253 176 146 119 126 117 198 149 63 121 224 13 40 227 152 8 115 97 54 221 129 138 7 98 68 244 202 166 227 67 29 227 238 52 253 239 143 135 15 204 125 154 198 241 127 40 156 164 83 155 177 180]
	RandBytes[33 62 225 108 153 150 24 222 120 139 12 53 63 61 208 128 237 42 214 80 139 33 225 121 92 40 141 76 8 40 123 241]
	PrevBlock[71 69 78 69 83 73 83 45 66 76 79 67 75]
	SenderMEgCQQCyKk2dRupy5iWUWiHiKS0tWX13wiZ7pfZq2Pm7kiscGMA7+BXxiZcaPkSUy4Ly3Bgq0u16ZVaJ9cUSZM+kC3RAgMBAAE=
	Receiverbbb
	Value3
	ToStorage0

3.2. «Ethereum»

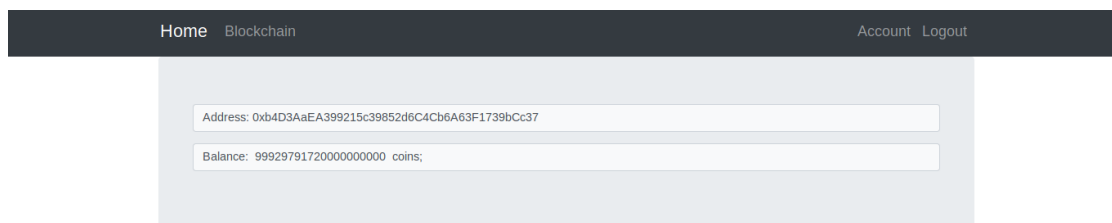
(1) Главная страница.



(2) Страница авторизации (ввод приватного ключа).



(3) Аккаунт пользователя.



(4) Информация о блокчейне. Если пользователь является создателем контракта, тогда у него появляется возможность создавать объекты (недвижимости).

The screenshot shows a web interface with a dark header containing 'Home', 'Blockchain', 'Account', and 'Logout'. The main content area has a light gray background. It features a form with three input fields: 'Information', 'Squere' (with a clear button), and 'Userful Squere' (with a clear button). Below these fields is a green 'Create' button. Underneath the form is a list of four teal buttons: 'Estates', 'Presents', 'Sales', and 'Rents'.

(5) Страница недвижимости. По-умолчанию отображаются все объекты, которые относятся к пользователю. В адресное поле можно ввести адрес другого пользователя, чтобы просмотреть информацию о нём. Если ввести в поле «all», тогда отобразятся все недвижимости в блокчейне. Такой же механизм работает с подарками, продажами и арендами.

The screenshot shows a web interface with a dark header containing 'Home', 'Blockchain', 'Account', and 'Logout'. The main content area has a light gray background. It features a form with two input fields: the first contains the address '0xb4D3AaEA399215c39852d6C4Cb6A63F1739bCc37' and is labeled 'Address:'; the second is empty and labeled 'Address'. Below these fields is a green 'Get estates' button. Underneath the form is a list of three teal buttons: '[Estate 0]', '[Estate 1]', and '[Estate 2]'.

(6) Информация о недвижимости. Если пользователь владеет недвижимостью, тогда у него появляются функции с дарением, продажей и арендой.

Home
Blockchain
Account
Logout

Do present
Do sales
Do rents

Id	0
Owner	0xb4D3AaEA399215c39852d6C4Cb6A63F1739bCc37
Info	one
Squere	10
UsefulSquere	5
RenterAddress	0x00
PresentStatus	false
SaleStatus	false
RentStatus	false

(7) Страница дарения недвижимости.

Home
Blockchain
Account
Logout

Address
Send present

Id	0
Owner	0xb4D3AaEA399215c39852d6C4Cb6A63F1739bCc37
Info	one
Squere	10
UsefulSquere	5
RenterAddress	0x00
PresentStatus	false
SaleStatus	false
RentStatus	false

(8) Страница продажи недвижимости.

Home
Blockchain
Account
Logout

Price
Do sale

Id	0
Owner	0xb4D3AaEA399215c39852d6C4Cb6A63F1739bCc37
Info	one
Squere	10
UsefulSquere	5
RenterAddress	0x00
PresentStatus	false
SaleStatus	false
RentStatus	false

(9) Страница аренды недвижимости.

[Home](#) [Blockchain](#) [Account](#) [Logout](#)

id	0
Owner	0xb4D3AaEA399215c39852d6C4Cb6A63F1739bCc37
Info	one
Square	10
UsefulSquare	5
RenterAddress	0x00
PresentStatus	false
SaleStatus	false
RentStatus	false

(10) Страница принятия подарка.

[Home](#) [Blockchain](#) [Account](#) [Logout](#)

id	0
EstateId	0
AddressFrom	0xb4D3AaEA399215c39852d6C4Cb6A63F1739bCc37
AddressTo	0xe2dB725A7Cb0EAB36a3B3943C7b5d99187ff5013
Finished	false

(11) Страница покупки недвижимости.

[Home](#) [Blockchain](#) [Account](#) [Logout](#)

id	0
EstateId	0
Owner	0xe2dB725A7Cb0EAB36a3B3943C7b5d99187ff5013 - 0xcC7dE97BED30A5eAc13c58CB363cd4A8A1658593
Price	10000000000000000
Customers	
Finished	false

(12) Страница подтверждения продажи.

HomeBlockchainAccountLogout

Cancel

Id	0
EstateId	0
Owner	0xe2dB725A7Cb0EAB36a3B3943C7b5d99187ff5013 - 0xe2dB725A7Cb0EAB36a3B3943C7b5d99187ff5013
Price	1000000000000000000
Customers	Customer 0xcC7dE97BED30A5eAc13c58CB363cd4A8A1658593
	Price 1000000000000000000
	Confirm Confirm
Finished	false

(13) Страница принятия аренды.

HomeBlockchainAccountLogout

Confirm

Id	0
EstateId	0
Owner	0xcC7dE97BED30A5eAc13c58CB363cd4A8A1658593
Renter	0x00
Time	3
Money	1000000000000000000
DeadLine	0
Finished	false

4. Литература

- [1] Коваленко, Г. Программирование узла блокчейн / Г. Коваленко. 2020. - 104 с.
- [2] Керниган, Б. У., Донован, Ф. Ф. Язык программирования Go / Б. У. Керниган, А. А. Донован. - М.: ООО «И.Д. Вильямс», 2018. - 432 с.
- [3] Мота М. Ethereum development with Go [Электронный ресурс] // URL: <https://goethereumbook.org/ethereum-development-with-go.pdf> (дата обращения: 10.08.2020).

5. Исходный код

<https://github.com/Number571/Blockchain>