Программирование интерфейса

Коваленко Г. А.

СОДЕРЖАНИЕ

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

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

2.1. Консольный интерфейс

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

2.2. 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 hmtl 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 authorizated"
                        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>
    </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>
```

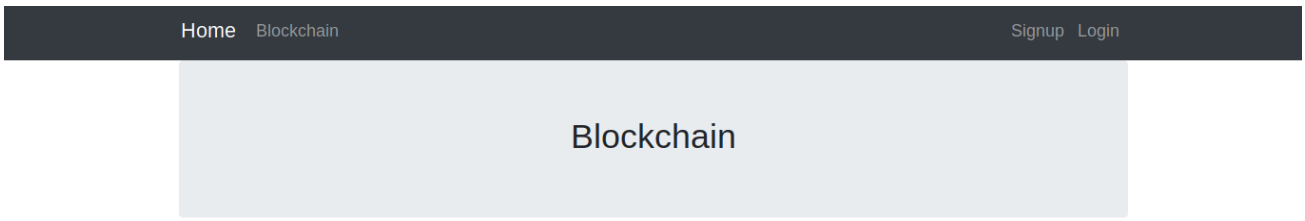
```
                <th>RandBytes</th>
                <td width="100%">{{ .RandBytes }}</td>
            </tr>
            <tr>
                <th>PrevBlock</th>
                <td width="100%">{{ .PrevBlock }}</td>
            </tr>
            <tr>
                <th>Sender</th>
                <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}}
```

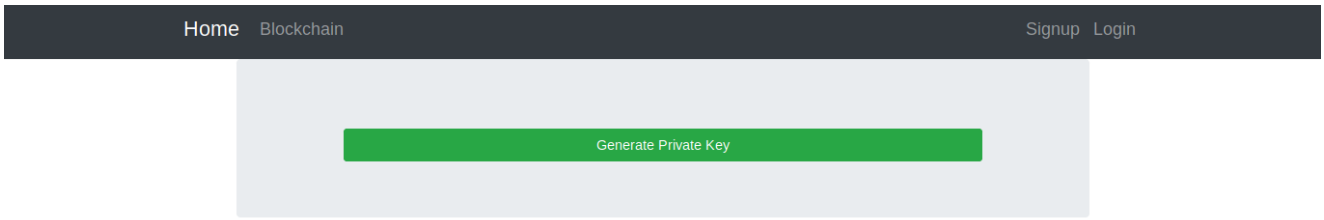
---

3. Скриншоты

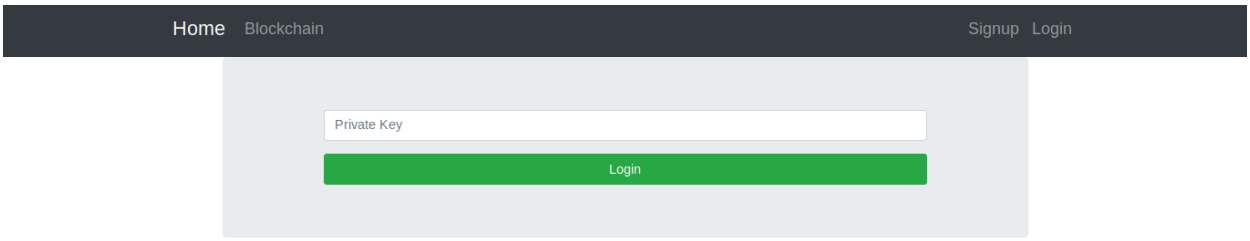
(2.1) Главная страница.



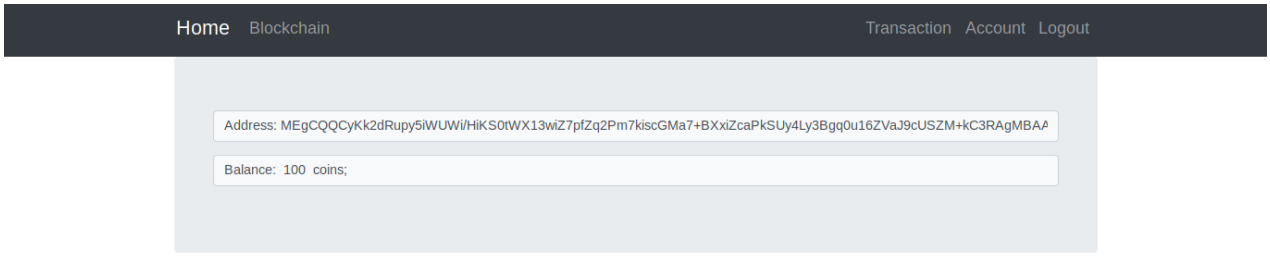
(2.2) Страница регистрации (генерации приватного ключа).



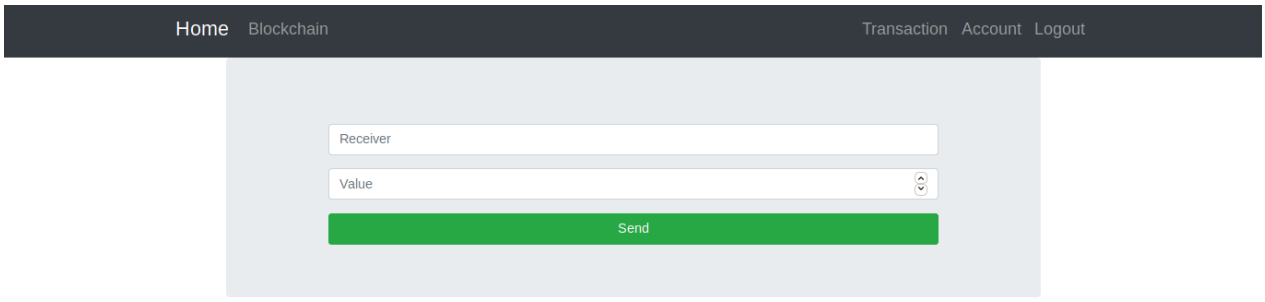
(2.3) Страница авторизации (ввод приватного ключа).



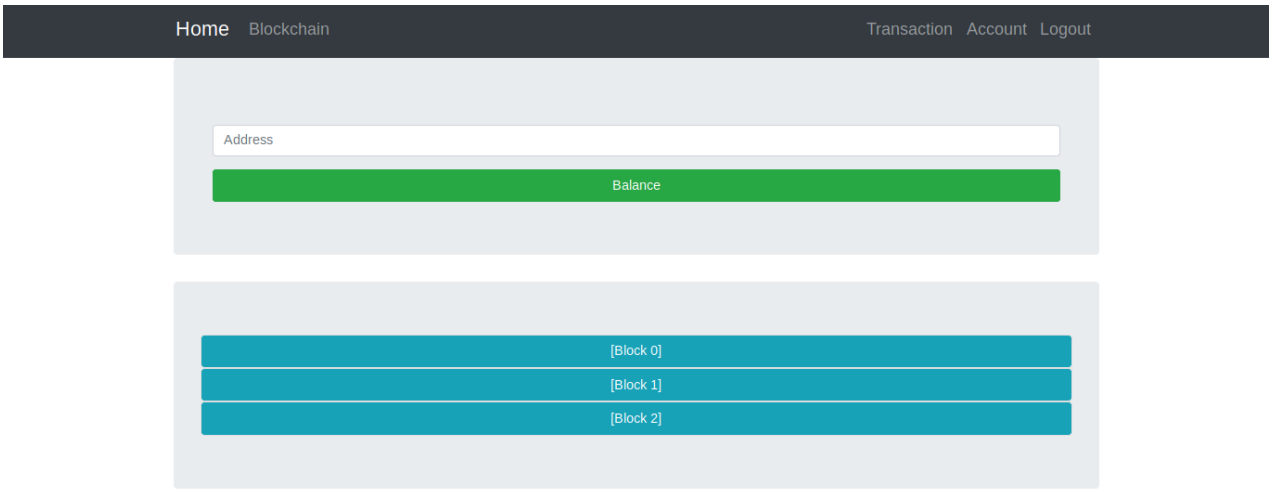
(2.4) Аккаунт пользователя.



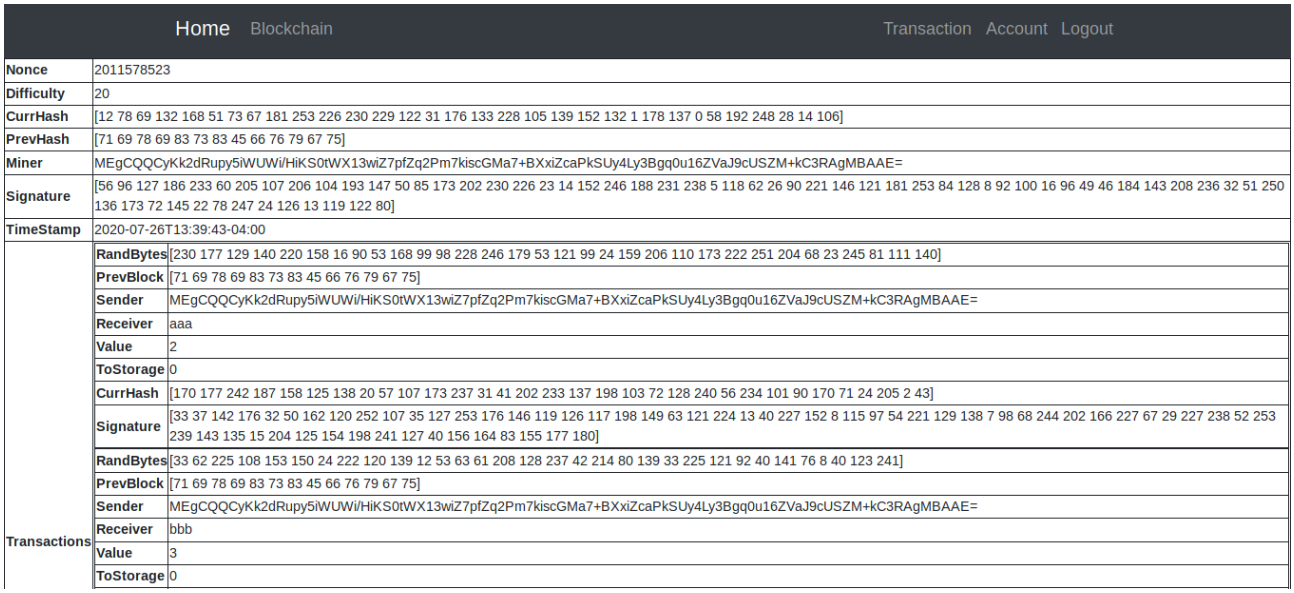
(2.5) Формирование транзакций.



(2.6) Информация о блокчейне.



(2.7) Информация о блоке.



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	MEgCQQCyKk2dRupy5iWUWi/HiKS0tWX13wiZ7pfZq2Pm7kiscGMa7+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]
	Sender	MEgCQQCyKk2dRupy5iWUWi/HiKS0tWX13wiZ7pfZq2Pm7kiscGMa7+BXxiZcaPkSUy4Ly3Bgq0u16ZVaJ9cUSZM+kC3RAgMBAAE=
	Receiver	aaa
	Value	2
	ToStorage	0
	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]
	Sender	MEgCQQCyKk2dRupy5iWUWi/HiKS0tWX13wiZ7pfZq2Pm7kiscGMa7+BXxiZcaPkSUy4Ly3Bgq0u16ZVaJ9cUSZM+kC3RAgMBAAE=
	Receiver	bbb
	Value	3
	ToStorage	0

4. Литература

[1] Коваленко, Г., Программирование узла блокчейн / Г. Коваленко. 2020. - 104 с.

[2] Керниган, Б. У., Донован, Ф. Ф. Язык программировани Go / Б. У. Керниган, А. А. Донован. - М.: ООО «И.Д. Вильямс», 2018. - 432 с.

5. Исходный код

https://github.com/Number571/Blockchain