Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 7

по курсу «Компьютерные системы и сети»

«Информационная система мониторинга блоков в блокчеин Etherium»

Студент группы ИУ9-32Б Федуков А. А.

Преподаватель Посевин Д. П.

10 декабря 2024 г.

Цель работы

Реализовать приложение мониторинга состояния заданных блоков блокчейн Etherium. Данные результатов мониторинга должны записываться в Firebase. Аккаунт в infura.io необходимо зарегистрировать свой, также создать собственную Realtime Database.

Реализация

Я создал аккаунты в infura и firebase, получил токены и, используя их, получал блоки и данные о них, после чего записывал результаты в БД.

Код

Листинг 1: Файл 1.go

```
1  // Получение последнего блока
2
3  package main
4
5  import (
6    "context"
7    "fmt"
8    "log"
9    "math/big"
10
11   "github.com/ethereum/go-ethereum/core/types"
12   "github.com/ethereum/go-ethereum/ethclient"
13 )
14
15 func getLatestBlock() (*types.Block, *big.Int) {
16   client, err := ethclient.Dial(ethKey)
17   if err != nil {
18     log.Fatalln(err)
19   }
20
21   header, err := client.HeaderByNumber(context.Background(), nil)
22   if err != nil {
23     log.Fatal(err)
24   }
25
26   fmt.Println(header.Number.String()) // The lastes block in blockchain
       because nil pointer in header
```

```
27
28    blockNumber := big.NewInt(header.Number.Int64())
29    block, err := client.BlockByNumber(context.Background(), blockNumber)
        //get block with this number
30    if err != nil {
31      log.Fatal(err)
32    }
33
34    return block, blockNumber
35  }
```

Листинг 2: Файл 2.go

```
1  // Получение данных из блока по номеру
2
3  package main
4
5  import (
6    "context"
7    "log"
8    "math/big"
9
10   "github.com/ethereum/go-ethereum/core/types"
11   "github.com/ethereum/go-ethereum/ethclient"
12 )
13
14 func getBlock(id int64) *types.Block {
15   client, err := ethclient.Dial(ethKey)
16   if err != nil {
17     log.Fatalln(err)
18   }
19
20   blockNumber := big.NewInt(id)
21   block, err := client.BlockByNumber(context.Background(), blockNumber)
       //get block with this number
22   if err != nil {
23     log.Fatal(err)
24   }
25   return block
26 }
```

Листинг 3: Файл 3.go

```
1  // Получение данных из полей транзакции
2
3  package main
4
```

```
5  import (
6    "context"
7    "fmt"
8    "log"
9    "math/big"
10
11   "github.com/ethereum/go-ethereum/ethclient"
12 )
13
14 func getTransactionsInfo(blockID int64) {
15   client, err := ethclient.Dial(ethKey)
16   if err != nil {
17     log.Fatalln(err)
18   }
19
20   blockNumber := big.NewInt(blockID)
21   block, err := client.BlockByNumber(context.Background(), blockNumber)
       //get block with this number
22   if err != nil {
23     log.Fatal(err)
24   }
25
26   for _, tx := range block.Transactions() {
27     fmt.Println(tx.ChainId())
28     fmt.Println(tx.Hash())
29     fmt.Println(tx.Value())
30     fmt.Println(tx.Cost())
31     fmt.Println(tx.To())
32     fmt.Println(tx.Gas())
33     fmt.Println(tx.GasPrice())
34
35   }
36 }
```

Листинг 4: Файл firebase.go

```
1  package main
2
3  import (
4    "context"
5    "log"
6
7    firebase "firebase.google.com/go/v4"
8    "firebase.google.com/go/v4/db"
9    "google.golang.org/api/option"
10 )
11
```

```
12
13 func connectDB() *db.Ref {
14   opt := option.WithCredentialsFile("key.json")
15   config := &firebase.Config{ProjectID: "lab7-ff266", DatabaseURL: "
       https://lab7-ff266-default-rtdb.europe-west1.firebasedatabase.app/"}
16   app, err := firebase.NewApp(context.Background(), config, opt)
17   if err != nil {
18     log.Fatalf("error initializing app: %v\n", err)
19   }
20
21   // Initialize a database client
22   client, err := app.Database(context.Background())
23   if err != nil {
24     log.Fatalf("Error initializing database client: %v", err)
25   }
26
27   ref := client.NewRef("/")
28   return ref
29 }
30
31 func clearDB(ref *db.Ref) {
32   ref.Delete(context.Background())
33 }
```

Листинг 5: Файл main.go

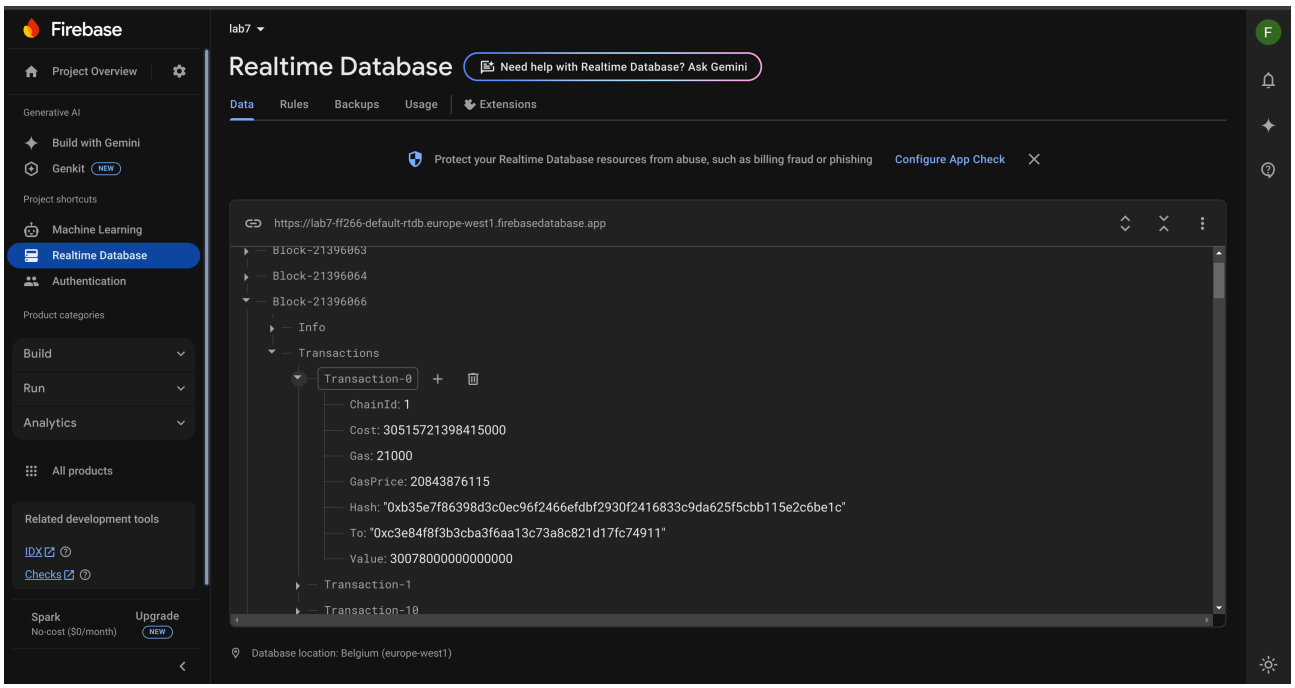
```
1  package main
2
3  import (
4    "context"
5    "fmt"
6    "math/big"
7    "time"
8
9    "firebase.google.com/go/v4/db"
10   "github.com/ethereum/go-ethereum/common"
11   "github.com/ethereum/go-ethereum/core/types"
12 )
13
14 var ethKey = "https://mainnet.infura.io/v3/
       ce09af264e6f4ef4a930b4037c908465"
15 var DB *db.Ref
16
17 func sendData(data interface{}) {
18   DB.Set(context.Background(), data)
19   fmt.Printf("Sended data: [%v]", data)
20 }
```

```
21
22 func hasBlock(blockID string) bool {
23   blocks, err := DB.OrderByKey().GetOrdered(context.Background())
24   if err != nil {
25     fmt.Errorf("Bad blocks: %v", err)
26   }
27
28   for _, v := range blocks {
29     if v.Key() == blockID {
30       return true
31     }
32   }
33
34   return false
35 }
36
37 type Block struct {
38   BlockID           uint64 `json:"BlockID"`
39   Time              uint64 `json:"Time"`
40   Difficulty        uint64 `json:"Difficulty"`
41   Hash              string `json:"Hash"`
42   TransactionNumber int    `json:"TransactionNumber"`
43 }
44
45 type Transaction struct {
46   ChainId  *big.Int        `json:"ChainId"`
47   Hash     common.Hash     `json:"Hash"`
48   Value    *big.Int        `json:"Value"`
49   Cost     *big.Int        `json:"Cost"`
50   To       *common.Address `json:"To"`
51   Gas      uint64          `json:"Gas"`
52   GasPrice *big.Int        `json:"GasPrice"`
53 }
54
55 func getInfoFromBlock(block *types.Block) {
56   // all info about block
57   blockID := block.Number().Uint64()
58   DB = DB.Child(fmt.Sprintf("Block-%v", blockID)).Child("Info")
59
60   sendData(Block{
61     BlockID:           blockID,
62     Time:              block.Time(),
63     Difficulty:        block.Difficulty().Uint64(),
64     Hash:              block.Hash().Hex(),
65     TransactionNumber: len(block.Transactions()),
66   })
```

```
67
68    DB = DB.Parent().Child("Transactions")
69
70    for i, tx := range block.Transactions() {
71      DB = DB.Child(fmt.Sprintf("Transaction-%v", i))
72      sendData(Transaction{
73        ChainId:  tx.ChainId(),
74        Hash:     tx.Hash(),
75        Value:    tx.Value(),
76        Cost:     tx.Cost(),
77        To:       tx.To(),
78        Gas:      tx.Gas(),
79        GasPrice: tx.GasPrice(),
80      })
81
82      DB = DB.Parent()
83    }
84
85    DB = DB.Parent().Parent()
86  }
87
88  func main() {
89    DB = connectDB()
90    clearDB(DB)
91    for {
92      fmt.Println("Check blocks")
93      block, id := getLatestBlock()
94      if hasBlock(id.String()) {
95        fmt.Println("Has block!")
96      } else {
97        fmt.Println("Add block")
98        getInfoFromBlock(block)
99      }
100
101     time.Sleep(time.Second * 2)
102   }
103 }
```

Вывод программы

Программа успешно получала данные о новых блоках блокчейн сети Ethereum и при необходимости загружала информацию о них в базу данных Firebase.



Вывод

В этот лабораторной я познакомился с концепцией блокчена, а также научился пользоваться realtime database на примере Firebase.