МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Систем обработки информации и управления»

ОТЧЕТ

**Лабораторная работа №\_\_5\_\_**

по дисциплине«Технология обработки больших данных»

Тема: «Технология блокчейн»

ИСПОЛНИТЕЛЬ: \_\_Журавлев Н. В.\_\_

ФИО

группа ИУ5-44М \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

"30"\_\_апреля\_\_\_2025 г.

ПРЕПОДАВАТЕЛЬ: \_\_\_\_Сухобоков А.В.\_\_\_\_\_

ФИО

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

"30"\_апреля\_\_\_2025 г.

Москва - 2025

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Цели и задачи

1. Определите структуру Shipment (поля ID, From, To, Status)

2. Внутри main() жёстко задайте набор отправлений:

shipments\_company1 := []Shipment{

{"S1", "Warehouse A", "Client X", "shipped"},

{"S2", "Warehouse B", "Client Y", "in\_transit"},

{"S3", "Warehouse C", "Client Z", "delivered"},

{"S4", "Warehouse A", "Client W", "in\_transit"},

}

shipments\_company2 := []Shipment{

{"S5", "Warehouse D", "Client W", "in\_transit"},

{"S6", "Warehouse E", "Client X", "delivered"},

}

shipments\_company3 := []Shipment{

{"S7", "Warehouse A", "Client X", "shipped"},

{"S8", "Warehouse A", "Client Y", "in\_transit"},

{"S9", "Warehouse A", "Client Z", "in\_transit"},

}

3. Реализовать функцию AddShipments():

AddShipments превращает каждый Shipment из списка в JSON и добавляет в блокчейн.

Задачи:

• Перебор всех отправлений из list.

• Сериализацию через json.Marshal.

• Вызов bc.AddBlock с полученной JSON-строкой и validator.

4. Добавить в блокчейн запись shipments\_company1 используя первого валидатора из AuthorizedValidators, запись shipments\_company2 используя второго валидатора из AuthorizedValidators и запись shipments\_company3 используя неавторизованного валидатора.

5. Вернуть полученную цепочку блоков.

6. (По желанию). Реализовать функцию FindByStatus()

FindByStatus возвращает все блоки, где Shipment.Status == “in\_transit”.

Задачи:

* Перебрать каждый блок в bc.Chain.
* Распаковать JSON данных блока в Shipment.
* Сравнить поле s.Status с аргументом status.
* Вернуть все подходящие блоки со статусом “in\_transit”

# Ход работы

Модифицированный исходный код под выполнения задания представлен в листинге 1.

Листинг . Исходный код

|  |
| --- |
| package main  import (  "crypto/sha256"  "encoding/hex"  "encoding/json"  "errors"  "fmt"  "time"  )  type Shipment struct {  ID string `json:"id"`  From string `json:"from"`  To string `json:"to"`  Status string `json:"status"`  }  type Block struct {  Index uint64 `json:"index"`  Timestamp int64 `json:"timestamp"`  ShipmentData string `json:"shipment\_data"`  PrevHash string `json:"prev\_hash"`  Validator string `json:"validator"`  Hash string `json:"hash"`  }  func (b \*Block) ComputeHash() string {  record := fmt.Sprintf("%d%d%s%s%s", b.Index, b.Timestamp, b.ShipmentData, b.PrevHash, b.Validator)  sum := sha256.Sum256([]byte(record))  return hex.EncodeToString(sum[:])  }  var AuthorizedValidators = []string{"validator1\_pubkey", "validator2\_pubkey"}  func isAuthorized(val string) bool {  for \_, v := range AuthorizedValidators {  if v == val {  return true  }  }  return false  }  type Blockchain struct {  Chain []Block  }  func NewBlockchain() \*Blockchain {  genesis := Block{  Index: 0,  Timestamp: time.Now().Unix(),  ShipmentData: "genesis",  PrevHash: "0",  Validator: "system",  }  genesis.Hash = genesis.ComputeHash()  return &Blockchain{Chain: []Block{genesis}}  }  func (bc \*Blockchain) AddBlock(data, validator string) error {  if !isAuthorized(validator) {  return errors.New("validator not authorized")  }  prev := bc.Chain[len(bc.Chain)-1]  blk := Block{  Index: prev.Index + 1,  Timestamp: time.Now().Unix(),  ShipmentData: data,  PrevHash: prev.Hash,  Validator: validator,  }  blk.Hash = blk.ComputeHash()  bc.Chain = append(bc.Chain, blk)  return nil  }  func (bc \*Blockchain) IsValid() bool {  for i := 1; i < len(bc.Chain); i++ {  curr, prev := bc.Chain[i], bc.Chain[i-1]  if curr.PrevHash != prev.Hash || curr.Hash != curr.ComputeHash() {  return false  }  if !isAuthorized(curr.Validator) {  return false  }  }  return true  }  func (bc \*Blockchain) AddShipments(shipments []Shipment, validator string) error {  for \_, shipment := range shipments {  jsonData, err := json.Marshal(shipment)  if err != nil {  return err  }  err = bc.AddBlock(string(jsonData), validator)  if err != nil {  return err  }  }  return nil  }  func (bc \*Blockchain) FindByStatus(status string) []Block {  var result []Block  for \_, block := range bc.Chain {  if block.ShipmentData == "genesis" {  continue  }  var shipment Shipment  err := json.Unmarshal([]byte(block.ShipmentData), &shipment)  if err != nil {  continue  }  if shipment.Status == status {  result = append(result, block)  }  }  return result  }  func main() {  shipments\_company1 := []Shipment{  {"S1", "Warehouse A", "Client X", "shipped"},  {"S2", "Warehouse B", "Client Y", "in\_transit"},  {"S3", "Warehouse C", "Client Z", "delivered"},  {"S4", "Warehouse A", "Client W", "in\_transit"},  }  shipments\_company2 := []Shipment{  {"S5", "Warehouse D", "Client W", "in\_transit"},  {"S6", "Warehouse E", "Client X", "delivered"},  }  shipments\_company3 := []Shipment{  {"S7", "Warehouse A", "Client X", "shipped"},  {"S8", "Warehouse A", "Client Y", "in\_transit"},  {"S9", "Warehouse A", "Client Z", "in\_transit"},  }  bc := NewBlockchain()  fmt.Println("Genesis:", bc.Chain[0])  // Добавляем shipments\_company1 с первым валидатором  err := bc.AddShipments(shipments\_company1, AuthorizedValidators[0])  if err != nil {  fmt.Println("Error adding company1 shipments:", err)  }  // Добавляем shipments\_company2 со вторым валидатором  err = bc.AddShipments(shipments\_company2, AuthorizedValidators[1])  if err != nil {  fmt.Println("Error adding company2 shipments:", err)  }  // Добавляем shipments\_company3 с неавторизованным валидатором  err = bc.AddShipments(shipments\_company3, "bad\_validator")  if err != nil {  fmt.Println("Error adding company3 shipments:", err) // Ожидаемая ошибка  }  // вывод всей цепочки  out, \_ := json.MarshalIndent(bc.Chain, "", " ")  fmt.Println(string(out))  // Проверка валидности  fmt.Println("\nChain valid?", bc.IsValid())  // 6. Поиск блоков со статусом "in\_transit"  inTransitBlocks := bc.FindByStatus("in\_transit")  fmt.Println("\nBlocks with 'in\_transit' status:")  for \_, block := range inTransitBlocks {  fmt.Printf("Block %d: %s\n", block.Index, block.ShipmentData)  }  } |

Цепочка блоков после выполнения задания представлена на рис. 1 и листинг 2.

Листинг 2. Результат добавления перевозок

|  |
| --- |
| Error adding company3 shipments: validator not authorized  [  {  "index": 0,  "timestamp": 1745778385,  "shipment\_data": "genesis",  "prev\_hash": "0",  "validator": "system",  "hash": "a8575caac2a87285855dad35b4a300c37ab2e5d637753592de660eed2525450f"  },  {  "index": 1,  "timestamp": 1745778385,  "shipment\_data": "{\"id\":\"S1\",\"from\":\"Warehouse A\",\"to\":\"Client X\",\"status\":\"shipped\"}",  "prev\_hash": "a8575caac2a87285855dad35b4a300c37ab2e5d637753592de660eed2525450f",  "validator": "validator1\_pubkey",  "hash": "6296a6fa6f3da933710df00a4e2b4f5b93573541a31ca5c804e74796805fd8c0"  },  {  "index": 2,  "timestamp": 1745778385,  "shipment\_data": "{\"id\":\"S2\",\"from\":\"Warehouse B\",\"to\":\"Client Y\",\"status\":\"in\_transit\"}",  "prev\_hash": "6296a6fa6f3da933710df00a4e2b4f5b93573541a31ca5c804e74796805fd8c0",  "validator": "validator1\_pubkey",  "hash": "ec4686b0448c5f65d45bef2c1788a6ed413001094b77e9a465740cd140c8dbfd"  },  {  "index": 3,  "timestamp": 1745778385,  "shipment\_data": "{\"id\":\"S3\",\"from\":\"Warehouse C\",\"to\":\"Client Z\",\"status\":\"delivered\"}",  "prev\_hash": "ec4686b0448c5f65d45bef2c1788a6ed413001094b77e9a465740cd140c8dbfd",  "validator": "validator1\_pubkey",  "hash": "687c90168a6e0b902fe967e598ab889c026d1ffa6559dcb6d706ce3b4e94ec21"  },  {  "index": 4,  "timestamp": 1745778385,  "shipment\_data": "{\"id\":\"S4\",\"from\":\"Warehouse A\",\"to\":\"Client W\",\"status\":\"in\_transit\"}",  "prev\_hash": "687c90168a6e0b902fe967e598ab889c026d1ffa6559dcb6d706ce3b4e94ec21",  "validator": "validator1\_pubkey",  "hash": "bf6b5e30ad50d6aa2d8bca3acdb563a358415487fc37308ae440eb4caa268049"  },  {  "index": 5,  "timestamp": 1745778385,  "shipment\_data": "{\"id\":\"S5\",\"from\":\"Warehouse D\",\"to\":\"Client W\",\"status\":\"in\_transit\"}",  "prev\_hash": "bf6b5e30ad50d6aa2d8bca3acdb563a358415487fc37308ae440eb4caa268049",  "validator": "validator2\_pubkey",  "hash": "d93253fe0ec1642481d062d0cb4564150dbd38bddf3557a56cb23479e820161a"  },  {  "index": 6,  "timestamp": 1745778385,  "shipment\_data": "{\"id\":\"S6\",\"from\":\"Warehouse E\",\"to\":\"Client X\",\"status\":\"delivered\"}",  "prev\_hash": "d93253fe0ec1642481d062d0cb4564150dbd38bddf3557a56cb23479e820161a",  "validator": "validator2\_pubkey",  "hash": "dd2b7c67a806d39cab04b4a8d5b70ee34ffee0d931901c2218f3d4374ffeff7d"  }  ]  Chain valid? true |

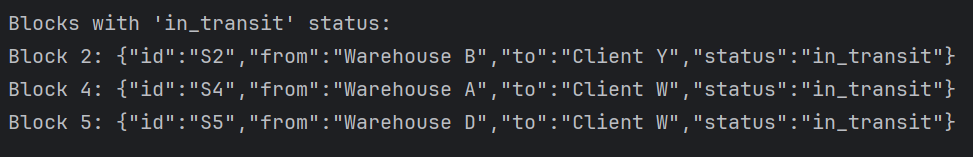


Рисунок . Результат выполнения поиска

# Вывод

В ходе выполнения лабораторной работы были получены навыки взаимодействия с технологии блокчейн и программного взаимодействия с ней.