



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 КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 Компьютерные системы и сети

О Т Ч Е Т

по лабораторной работе № 10

Название: Архитектура микросервисов на Golang

Дисциплина: Языки Интернет-Программирования

Студент

ИУ6-31Б

(Группа)

(Подпись, дата)

К.Д. Коротаев

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

И.О. Фамилия

(И.О. Фамилия)

Москва, 2024

Цель работы: получение первичных навыков организации кодовой базы проекта на Golang

Задание:

1. Перекопировать код, полученный в ходе выполнения предыдущей лабораторной работы
2. Ознакомиться с теоретическими сведениями и образцом реализации в лице сервиса hello
3. Модернизировать код сервисов count и query в соответствии с рекомендациями из теоретических сведений и/или опираясь на образец в лице сервиса hello
4. Сделать отчёт и поместить его в директорию docs
5. Защитить лабораторную работу

Ход работы:

Общий код для трех сервисов:

Hello_example.yaml

```
ip: "127.0.0.1"
port_hello: 8081
port_count: 8082
port_query: 8083
api:
  max_message_size: 32
usecase:
  default_message: "hello, world"
  default_message_count: 0
  default_message_query: "Hello, friend!"
db:
  host: "localhost"
  port: 5432
  user: "postgres"
  dbname: "sandbox9"
```

Provider.go

```
package provider
```

```
import (
    "database/sql"
    "fmt"
    "log"
)
```

```
type Provider struct {
    conn *sql.DB
}
```

```
func NewProvider(host string, port int, user, dbName string) *Provider {
    psqlInfo := fmt.Sprintf("host=%s port=%d user=%s "+
        "dbname=%s sslmode=disable",
        host, port, user, dbName)

    // Создание соединения с сервером postgres
    conn, err := sql.Open("postgres", psqlInfo)
    if err != nil {
        log.Fatal(err)
    }

    return &Provider{conn: conn}
}
```

usecase.go

package usecase

```
type Usecase struct {  
    defaultMsg string  
  
    p Provider  
}
```

```
func NewUsecase(defaultMsg string, p Provider) *Usecase {  
    return &Usecase{  
        defaultMsg: defaultMsg,  
        p:          p,  
    }  
}
```

config.go

package config

```
type Config struct {  
    IP   string `yaml:"ip"`  
    Port int    `yaml:"port_query"`  
  
    API   api    `yaml:"api"`  
    Usecase usecase `yaml:"usecase"`  
    DB    db     `yaml:"db"`  
}
```

```
type api struct {  
    MaxMessageSize int `yaml:"max_message_size"`  
}
```

```
type usecase struct {  
    DefaultMessageQuery string `yaml:"default_message_query"`  
}
```

```
type db struct {  
    Host   string `yaml:"host"`  
    Port   int    `yaml:"port"`  
    User   string `yaml:"user"`  
    DBname string `yaml:"dbname"`  
}
```

load.go

package config

```
import (  
    "os"  
    "path/filepath"  
  
    "gopkg.in/yaml.v3"  
)  
  
func LoadConfig(pathToFile string) (*Config, error) {  
    filename, err := filepath.Abs(pathToFile)  
    if err != nil {  
        return nil, err  
    }  
  
    yamlFile, err := os.ReadFile(filename)  
    if err != nil {  
        return nil, err  
    }  
  
    var cfg Config  
  
    err = yaml.Unmarshal(yamlFile, &cfg)  
    if err != nil {  
        return nil, err  
    }  
  
    return &cfg, nil  
}
```

Код для работы с сервисом Hello:

```
main.go X
C: > Users > admin > .ssh > web-10 > cmd > hello > main.go > ...

1 package main
2
3 import (
4     "flag"
5     "log"
6
7     "github.com/ValeryBMSTU/web-10/internal/hello/api"
8     "github.com/ValeryBMSTU/web-10/internal/hello/config"
9     "github.com/ValeryBMSTU/web-10/internal/hello/provider"
10    "github.com/ValeryBMSTU/web-10/internal/hello/usecase"
11    _ "github.com/lib/pq"
12 )
13
14 func main() {
15     // Считываем аргументы командной строки
16     configPath := flag.String("config-path", "C:/Users/admin/.ssh/web-10/configs/hello_example.yaml", "путь к файлу конфигурации")
17     flag.Parse()
18
19     cfg, err := config.LoadConfig(*configPath)
20     if err != nil {
21         log.Fatal(err)
22     }
23
24     prv := provider.NewProvider(cfg.DB.Host, cfg.DB.Port, cfg.DB.User, cfg.DB.DBname)
25     use := usecase.NewUsecase(cfg.Usecase.DefaultMessage, prv)
26     srv := api.NewServer(cfg.IP, cfg.Port, cfg.API.MaxMessageSize, use)
27
28     srv.Run()
29 }
```

```
api.go X
C: > Users > admin > .ssh > web-10 > internal > hello > api > api.go > ...

1 package api
2
3 import (
4     "fmt"
5
6     "github.com/labstack/echo/v4"
7 )
8
9 type Server struct {
10     maxSize int
11
12     server *echo.Echo
13     address string
14
15     uc Usecase
16 }
17
18 func NewServer(ip string, port int, maxSize int, uc Usecase) *Server {
19     api := Server{
20         maxSize: maxSize,
21         uc:      uc,
22     }
23
24     api.server = echo.New()
25     api.server.GET("/hello", api.GetHello)
26     api.server.POST("/hello", api.PostHello)
27
28     api.address = fmt.Sprintf("%s:%d", ip, port)
29
30     return &api
31 }
32
33 func (api *Server) Run() {
34     api.server.Logger.Fatal(api.server.Start(api.address))
35 }
36
```

```

handler.go
C: > Users > admin > .ssh > web-10 > internal > hello > api > handler.go > ...
1 package api
2 import ("errors" ; "net/http" ; "github.com/ValeryBMSTU/web-10/pkg/vars" ; "github.com/labstack/echo/v4")
3 // GetHello возвращает случайное приветствие пользователю
4 func (srv *Server) GetHello(e echo.Context) error {
5     msg, err := srv.uc.FetchHelloMessage()
6     if err != nil {
7         return e.String(http.StatusInternalServerError, err.Error())
8     }
9
10    return e.JSON(http.StatusOK, msg)
11 }
12 // PostHello Помещает новый вариант приветствия в БД
13 func (srv *Server) PostHello(e echo.Context) error {
14     input := struct {
15         Msg *string `json:"msg"`
16     }{}
17     err := e.Bind(&input)
18     if err != nil {
19         return e.String(http.StatusInternalServerError, err.Error())
20     }
21     if input.Msg == nil {
22         return e.String(http.StatusBadRequest, "msg is empty")
23     }
24     if len([]rune(*input.Msg)) > srv.maxSize {
25         return e.String(http.StatusBadRequest, "hello message too large")
26     }
27     err = srv.uc.SetHelloMessage(*input.Msg)
28     if err != nil {
29         if errors.Is(err, vars.ErrAlreadyExist) {
30             return e.String(http.StatusConflict, err.Error())
31         }
32         return e.String(http.StatusInternalServerError, err.Error())
33     }
34
35     return e.String(http.StatusCreated, "OK")
36 }

```

```

interface.go X handler.go
C: > Users > admin > .ssh > web-10 > internal > hello > api > interface.go
1 package api
2
3 type Ucase interface {
4     FetchHelloMessage() (string, error)
5     SetHelloMessage(msg string) error
6 }
7

```

```
sql.go handler.go
C: > Users > admin > .ssh > web-10 > internal > hello > provider > sql.go > ...

1 package provider
2 import (
3     "database/sql"
4     "errors"
5 )
6 func (p *Provider) SelectRandomHello() (string, error) {
7     var msg string
8     // Получаем одно сообщение из таблицы hello, отсортированной в случайном порядке
9     err := p.conn.QueryRow("SELECT name_hello FROM hello ORDER BY RANDOM() LIMIT 1").Scan(&msg)
10    if err != nil {
11        if errors.Is(err, sql.ErrNoRows) {
12            return "", nil
13        }
14        return "", err
15    }
16    return msg, nil
17 }
18 func (p *Provider) CheckHelloExitByMsg(msg string) (bool, error) {
19     // Получаем одно сообщение из таблицы hello
20     err := p.conn.QueryRow("SELECT name_hello FROM hello WHERE name_hello = $1 LIMIT 1", msg).Scan(&msg)
21     if err != nil {
22         if errors.Is(err, sql.ErrNoRows) {
23             return false, nil
24         }
25         return false, err
26     }
27
28     return true, nil
29 }
30 func (p *Provider) InsertHello(msg string) error {
31     _, err := p.conn.Exec("INSERT INTO hello (name_hello) VALUES ($1)", msg)
32     if err != nil {
33         return err
34     }
35
36     return nil
37 }
```



```
sql.go  hello.go  handler.go
C: > Users > admin > .ssh > web-10 > internal > hello > usecase > -go hello.go > ...
1 package usecase
2
3 func (u *Usecase) FetchHelloMessage() (string, error) {
4     msg, err := u.p.SelectRandomHello()
5     if err != nil {
6         return "", err
7     }
8
9     if msg == "" {
10        return u.defaultMsg, nil
11    }
12
13    return msg, nil
14 }
15
16 func (u *Usecase) SetHelloMessage(msg string) error {
17     isExist, err := u.p.CheckHelloExitByMsg(msg)
18     if err != nil {
19         return err
20     }
21
22     if isExist {
23         return nil
24     }
25
26     err = u.p.InsertHello(msg)
27     if err != nil {
28         return err
29     }
30
31     return nil
32 }
33
```

```
interface.go X
C: > Users > admin > .ssh > web-10 > internal > hello > usecase > -go interface.go
1 package usecase
2
3 type Provider interface {
4     SelectRandomHello() (string, error)
5     CheckHelloExitByMsg(string) (bool, error)
6     InsertHello(string) error
7 }
```

Код для работы с сервисом Count:

```
C: > Users > admin > .ssh > web-10 > cmd > count > -o main.go
1 package main
2
3 import (
4     "flag"
5     "log"
6
7     "github.com/ValeryBMSTU/web-10/internal/count/api"
8     "github.com/ValeryBMSTU/web-10/internal/count/config"
9     "github.com/ValeryBMSTU/web-10/internal/count/provider"
10    "github.com/ValeryBMSTU/web-10/internal/count/usecase"
11
12    _ "github.com/lib/pq"
13 )
14
15 func main() {
16     // Считываем аргументы командной строки
17     configPath := flag.String("config-path", "../configs/hello_example.yaml", "путь к файлу конфигурации")
18     flag.Parse()
19
20     cfg, err := config.LoadConfig(*configPath)
21     if err != nil {
22         log.Fatal(err)
23     }
24
25     prv := provider.NewProvider(cfg.DB.Host, cfg.DB.Port, cfg.DB.User, cfg.DB.DBname)
26     use := usecase.NewUsecase(cfg.Usecase.DefaultMessageCount, prv)
27     srv := api.NewServer(cfg.IP, cfg.Port, cfg.API.MaxMessageSize, use)
28
29     srv.Run()
30 }
```

```
api.go
C: > Users > admin > .ssh > web-10 > internal > count > api > -o api.go > ...
1 package api
2
3 import (
4     "fmt"
5
6     "github.com/labstack/echo/v4"
7 )
8
9 type Server struct {
10     maxSize int
11
12     server *echo.Echo
13     address string
14
15     uc Usecase
16 }
17
18 func NewServer(ip string, port int, maxSize int, uc Usecase) *Server {
19     api := Server{
20         maxSize: maxSize,
21         uc:      uc,
22     }
23
24     api.server = echo.New()
25     api.server.GET("/count", api.GetCount)
26     api.server.POST("/count", api.PostCount)
27
28     api.address = fmt.Sprintf("%s:%d", ip, port)
29
30     return &api
31 }
32
33 func (api *Server) Run() {
34     api.server.Logger.Fatal(api.server.Start(api.address))
35 }
```

```

handler.go
C: > Users > admin > .ssh > web-10 > internal > count > api > handler.go > (*Server).PostCount
1  package api
2
3  import (
4      "errors"
5      "net/http"
6      "github.com/ValeryBMSTU/web-10/pkg/vars"
7      "github.com/labstack/echo/v4"
8  )
9  func (srv *Server) GetCount(e echo.Context) error {
10     msg, err := srv.uc.FetchCount()
11     if err != nil {
12         return e.String(http.StatusInternalServerError, err.Error())
13     }
14     return e.JSON(http.StatusOK, msg)
15 }
16 func (srv *Server) PostCount(e echo.Context) error {
17     input := struct {
18         Msg *int `json:"count"`
19     }{}
20     err := e.Bind(&input)
21     if err != nil {
22         return e.String(http.StatusInternalServerError, err.Error())
23     }
24     if input.Msg == nil {
25         return e.String(http.StatusBadRequest, "msg is empty")
26     }
27     err = srv.uc.IncrementCount(*input.Msg)
28     if err != nil {
29         if errors.Is(err, vars.ErrAlreadyExist) {
30             return e.String(http.StatusConflict, err.Error())
31         }
32         return e.String(http.StatusInternalServerError, err.Error())
33     }
34     return e.String(http.StatusCreated, "OK")
35 }
36

```

```

interface.go X handler.go
C: > Users > admin > .ssh > web-10 > internal > count > api > interface.go
1  package api
2
3  type Usecase interface {
4      FetchCount() (int, error)
5      IncrementCount(count int) error
6  }
7

```

```

interface.go  sql.go  handler.go
C > Users > admin > .ssh > web-10 > internal > count > provider > sql.go > (*Provider).CheckCountExist

1  package provider
2
3  import (
4      "database/sql"
5      "errors"
6  )
7  func (p *Provider) FetchCount() (int, error) {
8      var msg int
9      err := p.conn.QueryRow("SELECT number FROM counter").Scan(&msg)
10     if err != nil {
11         if errors.Is(err, sql.ErrNoRows) {
12             return 0, nil
13         }
14         return 0, err
15     }
16     return msg, nil
17 }
18 func (p *Provider) CheckCountExist() (bool, error) {
19     msg := 1
20     err := p.conn.QueryRow("SELECT number FROM counter WHERE id_number = $1", msg).Scan(&msg)
21     if err != nil {
22         if errors.Is(err, sql.ErrNoRows) {
23             return false, nil
24         }
25         return false, err
26     }
27
28     return true, nil
29 }
30 func (p *Provider) UpdateCount(count int) error {
31     _, err := p.conn.Exec("UPDATE counter SET number = number + $1 WHERE id_number = 1", count)
32     if err != nil {
33         return err
34     }
35
36     return nil
37 }

```

```

count.go  X
C > Users > admin > .ssh > web-10 > internal > count > usecase > count.go

1  package usecase
2
3  func (u *Usecase) FetchCount() (int, error) {
4      msg, err := u.p.FetchCount()
5      if err != nil {
6          return 0, err
7      }
8
9      if msg == 0 {
10         return u.defaultMsg, nil
11     }
12
13     return msg, nil
14 }
15
16 func (u *Usecase) IncrementCount(count int) error {
17     isExist, err := u.p.CheckCountExist()
18     if err != nil {
19         return err
20     }
21
22     if !isExist {
23         return nil
24     }
25
26     err = u.p.UpdateCount(count)
27     if err != nil {
28         return err
29     }
30
31     return nil
32 }
33

```

interface.go X

C: > Users > admin > .ssh > web-10 > internal > count > usecase > interface.go

```
1 package usecase
2
3 type Provider interface {
4     FetchCount() (int, error)
5     UpdateCount(int) error
6     CheckCountExist() (bool, error)
7 }
8
```

Код для работы с сервисом Query:

```
C:\> Users > admin > .ssh > web-10 > cmd > query > -> main.go > ...
1 package main
2
3 import (
4     "flag"
5     "log"
6
7     "github.com/ValeryBMSTU/web-10/internal/query/usecase"
8     "github.com/ValeryBMSTU/web-10/internal/query/api"
9     "github.com/ValeryBMSTU/web-10/internal/query/config"
10    "github.com/ValeryBMSTU/web-10/internal/query/provider"
11    _ "github.com/lib/pq"
12 )
13
14
15 func main() {
16     // Считываем аргументы командной строки
17     configPath := flag.String("config-path", "C:/Users/admin/.ssh/web-10/configs/hello_example.yaml", "путь к файлу конфигурации")
18     flag.Parse()
19
20     cfg, err := config.LoadConfig(*configPath)
21     if err != nil {
22         log.Fatal(err)
23     }
24
25     prv := provider.NewProvider(cfg.DB.Host, cfg.DB.Port, cfg.DB.User, cfg.DB.DBname)
26     use := usecase.NewUsecase(cfg.Usecase.DefaultMessageQuery, prv)
27     srv := api.NewServer(cfg.IP, cfg.Port, cfg.API.MaxMessageSize, use)
28
29     srv.Run()
30 }
```

```
api.go
C:\> Users > admin > .ssh > web-10 > internal > query > api > -> api.go
1 package api
2
3 import (
4     "fmt"
5
6     "github.com/labstack/echo/v4"
7 )
8
9 type Server struct {
10     maxSize int
11
12     server *echo.Echo
13     address string
14
15     uc Usecase
16 }
17
18 func NewServer(ip string, port int, maxSize int, uc Usecase) *Server {
19     api := Server{
20         maxSize: maxSize,
21         uc:      uc,
22     }
23
24     api.server = echo.New()
25     api.server.GET("/query", api.GetQuery)
26     api.server.POST("/query", api.PostQuery)
27
28     api.address = fmt.Sprintf("%s:%d", ip, port)
29
30     return &api
31 }
32
33 func (api *Server) Run() {
34     api.server.Logger.Fatal(api.server.Start(api.address))
35 }
```

```

handler.go X
C: > Users > admin > .ssh > web-10 > internal > query > api > handler.go
1 package api
2
3 import (
4     "net/http"
5
6     "github.com/labstack/echo/v4"
7 )
8
9 func (srv *Server) GetQuery(e echo.Context) error {
10     // get query name
11     name := e.QueryParam("name")
12     msg, err := srv.uc.FetchQuery(name)
13     if err != nil {
14         return e.String(http.StatusInternalServerError, err.Error())
15     }
16     return e.JSON(http.StatusOK, msg)
17 }
18
19 func (srv *Server) PostQuery(e echo.Context) error {
20     // post query name
21     name := e.QueryParam("name")
22     err := srv.uc.InsertQuery(name)
23     if err != nil {
24         return e.String(http.StatusInternalServerError, err.Error())
25     }
26     return e.NoContent(http.StatusOK)
27 }
28

```

```

interface.go X
C: > Users > admin > .ssh > web-10 > internal > query > api > interface.go
1 package api
2
3 type Usecase interface {
4     FetchQuery(name string) (string, error)
5     InsertQuery(name string) error
6 }

```

```

1 package provider
2 import (
3     "database/sql"
4     "errors"
5     "fmt"
6 )
7 func (p *Provider) FetchQuery(name string) (string, error) {
8     var msg string
9     err := p.conn.QueryRow("SELECT record FROM query WHERE record = $1", name).Scan(&msg)
10    if err != nil {
11        if errors.Is(err, sql.ErrNoRows) {
12            return "", nil
13        }
14        return "", err
15    }
16    return "Hello, " + msg + "!", nil
17 }
18 func (p *Provider) CheckQueryExist(name string) (bool, error) {
19     var msg string
20     err := p.conn.QueryRow("SELECT record FROM query WHERE record = $1", name).Scan(&msg)
21     if err != nil {
22         if errors.Is(err, sql.ErrNoRows) {
23             return false, nil
24         }
25         return false, err
26     }
27     return true, nil
28 }
29 func (p *Provider) InsertQuery(name string) error {
30     _, err := p.conn.Exec("INSERT INTO query (record) VALUES ($1)", name)
31     if err != nil {
32         return err
33     }
34     fmt.Println("Inserted successfully!")
35     return nil
36 }
37

```

```

1 package usecase
2
3 type Provider interface {
4     FetchQuery(name string) (string, error)
5     InsertQuery(name string) error
6     CheckQueryExist(name string) (bool, error)
7 }
8

```



```
1 package usecase
2
3 func (u *Usecase) FetchQuery(name string) (string, error) {
4     msg, err := u.p.FetchQuery(name)
5     if err != nil {
6         return "", err
7     }
8
9     if msg == "" {
10        return u.defaultMsg, nil
11    }
12
13    return msg, nil
14 }
15
16 func (u *Usecase) InsertQuery(name string) error {
17     isExist, err := u.p.CheckQueryExist(name)
18     if err != nil {
19         return err
20     }
21
22     if isExist {
23         return nil
24     }
25
26     err = u.p.InsertQuery(name)
27     if err != nil {
28         return err
29     }
30
31     return nil
32 }
33
```

Тестирование к сервису Hello:

```
1 SELECT * FROM hello;
```

	id [PK] integer	name_hello character varying (255)
1	4	Hello, World!
2	5	Hello, aboba!
3	7	Hello, golang

Установите последнюю версию PowerShell для новых функций и улучшения! <https://aka.ms/PSWindows>

```
PS C:\Users\admin> $Body=@{msg= "Hello, golang"} | ConvertTo-Json
PS C:\Users\admin> Invoke-RestMethod -Uri "http://127.0.0.1:8081/hello" -Method Post -ContentType "application/json" -Body $Body
Добавили запись!
PS C:\Users\admin> $Body=@{msg= "Hello, golang"} | ConvertTo-Json
PS C:\Users\admin> Invoke-RestMethod -Uri "http://127.0.0.1:8081/hello" -Method Post -ContentType "application/json" -Body $Body
Добавили запись!
PS C:\Users\admin> Invoke-RestMethod -Uri "http://127.0.0.1:8081/hello" -Method Get
Hello, World!
PS C:\Users\admin> Invoke-RestMethod -Uri "http://127.0.0.1:8081/hello" -Method Get
Hello, World!
PS C:\Users\admin> Invoke-RestMethod -Uri "http://127.0.0.1:8081/hello" -Method Get
Hello, World!
PS C:\Users\admin> Invoke-RestMethod -Uri "http://127.0.0.1:8081/hello" -Method Get
Hello, aboba!
PS C:\Users\admin> Invoke-RestMethod -Uri "http://127.0.0.1:8081/hello" -Method Get
Hello, golang
```

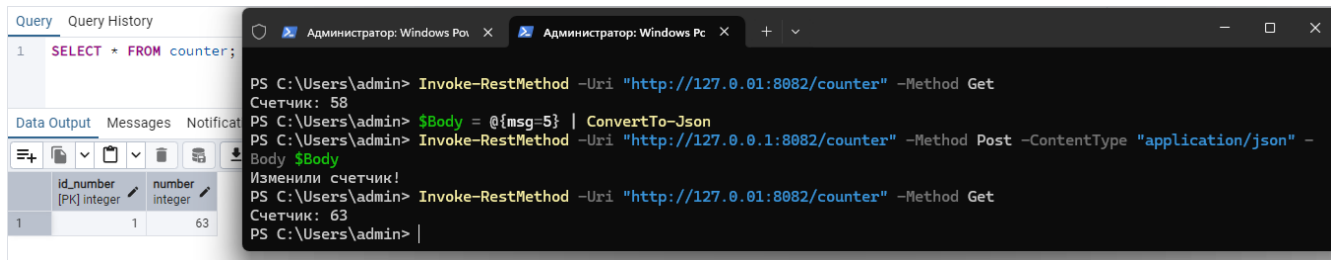
Тестирование к сервису Query:

```
1 SELECT * FROM query;
```

	id [PK] integer	record character varying (255)
1	1	Привет, мир!
2	2	Привет, golang!
3	3	aboba!
4	5	afawe
5	6	afawe1
6	7	afawe2
7	8	privet

```
PS C:\Users\admin> Invoke-RestMethod -Uri "http://127.0.0.1:8083/query1?name=privet" -Method Post -ContentType "application/json"
Добавили запись!
PS C:\Users\admin> Invoke-RestMethod -Uri "http://127.0.0.1:8083/query?name=privet" -Method Get
Hello privet!
PS C:\Users\admin>
```

Тестирование к сервису Count:



The screenshot displays two windows side-by-side. The left window is a database client showing a query result for the 'counter' table. The right window is a PowerShell terminal running REST client commands to interact with a service at 'http://127.0.0.1:8082/counter'.

Database Client (Left):

- Query: `SELECT * FROM counter;`
- Data Output table:

	id_number [PK] integer	number integer
1	1	63

PowerShell Terminal (Right):

```
PS C:\Users\admin> Invoke-RestMethod -Uri "http://127.0.0.1:8082/counter" -Method Get
Счетчик: 58
PS C:\Users\admin> $Body = @{msg=5} | ConvertTo-Json
PS C:\Users\admin> Invoke-RestMethod -Uri "http://127.0.0.1:8082/counter" -Method Post -ContentType "application/json" -
Body $Body
Изменили счетчик!
PS C:\Users\admin> Invoke-RestMethod -Uri "http://127.0.0.1:8082/counter" -Method Get
Счетчик: 63
PS C:\Users\admin> |
```

Заключение: получены первичные навыки организации кодовой базы проекта на Golang