

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

Название: Back-End разработка с использованием фреймворка Echo

Дисциплина: Языки Интернет-программирования

Преподаватель _____

(Подпись, дата) (И.О. Фамилия)

Оглавление

Back-End разработка с использованием фреймворка Echo.....	1
Цель работы.....	3
Ход работы.....	3
Задание.....	3
Микросервис hello:.....	3
Микросервис Query.....	8
Условие.....	8
Решение.....	8
Микросервис count.....	9
Вывод.....	13

Цель работы

Получение первичных навыков использования веб-фрейворков в Backend-разработке на Golang.

Ход работы

Задание

Переделать коды из 6 лабораторной работы в микросервисы и организовать клиент-серверное взаимодействие между Golang и PostgreSQL.

Микросервис hello:

```
package main
```

```
import (  
    "database/sql"  
    "flag"  
    "fmt"  
    "log"  
    "net/http"  
  
    "github.com/labstack/echo/v4"  
    "github.com/labstack/echo/v4/middleware"  
    _ "github.com/lib/pq"  
)
```

```
const (  
    host = "localhost"  
    port = 5432  
    user = "ps1"  
    password = "1103"  
    dbname = "lr8"  
)
```

```
type Handlers struct {  
    dbProvider DatabaseProvider  
}
```

```
type DatabaseProvider struct {  
    db *sql.DB  
}
```

```
func (h *Handlers) GetHello(c echo.Context) error {  
    msg, err := h.dbProvider.SelectHello()  
    if err != nil {  
        return c.String(http.StatusInternalServerError, err.Error())  
    }
```

```

}

return c.String(http.StatusOK, "Hello, "+msg+"!")
}

func (h *Handlers) PostHello(c echo.Context) error {
input := struct {
Msg string `json:"msg"`
}{}

err := c.Bind(&input)
if err != nil {
return c.String(http.StatusInternalServerError, err.Error())
}

err = h.dbProvider.InsertHello(input.Msg)
if err != nil {
return c.String(http.StatusInternalServerError, err.Error())
}

return c.String(http.StatusCreated, "Note added")
}

func (dp *DatabaseProvider) SelectHello() (string, error) {
var msg string

row := dp.db.QueryRow("SELECT name_hello FROM hello ORDER BY RANDOM()
LIMIT 1")
err := row.Scan(&msg)
if err != nil {
return "", err
}

return msg, nil
}

func (dp *DatabaseProvider) InsertHello(msg string) error {
_, err := dp.db.Exec("INSERT INTO hello (name_hello) VALUES ($1)", msg)
if err != nil {
return err
}

return nil
}

func main() {

```

```
address := flag.String("address", "127.0.0.1:8081", "server startup address")
flag.Parse()
```

```
psqlInfo := fmt.Sprintf("host=%s port=%d user=%s "+
"password=%s dbname=%s sslmode=disable",
host, port, user, password, dbname)
```

```
db, err := sql.Open("postgres", psqlInfo)
if err != nil {
log.Fatal(err)
}
defer db.Close()
```

```
dp := DatabaseProvider{db: db}
h := Handlers{dbProvider: dp}
```

```
e := echo.New()
```

```
e.Use(middleware.Logger())
```

```
e.GET("/hello", h.GetHello)
e.POST("/hello", h.PostHello)
```

```
e.Logger.Fatal(e.Start(*address))
}
```

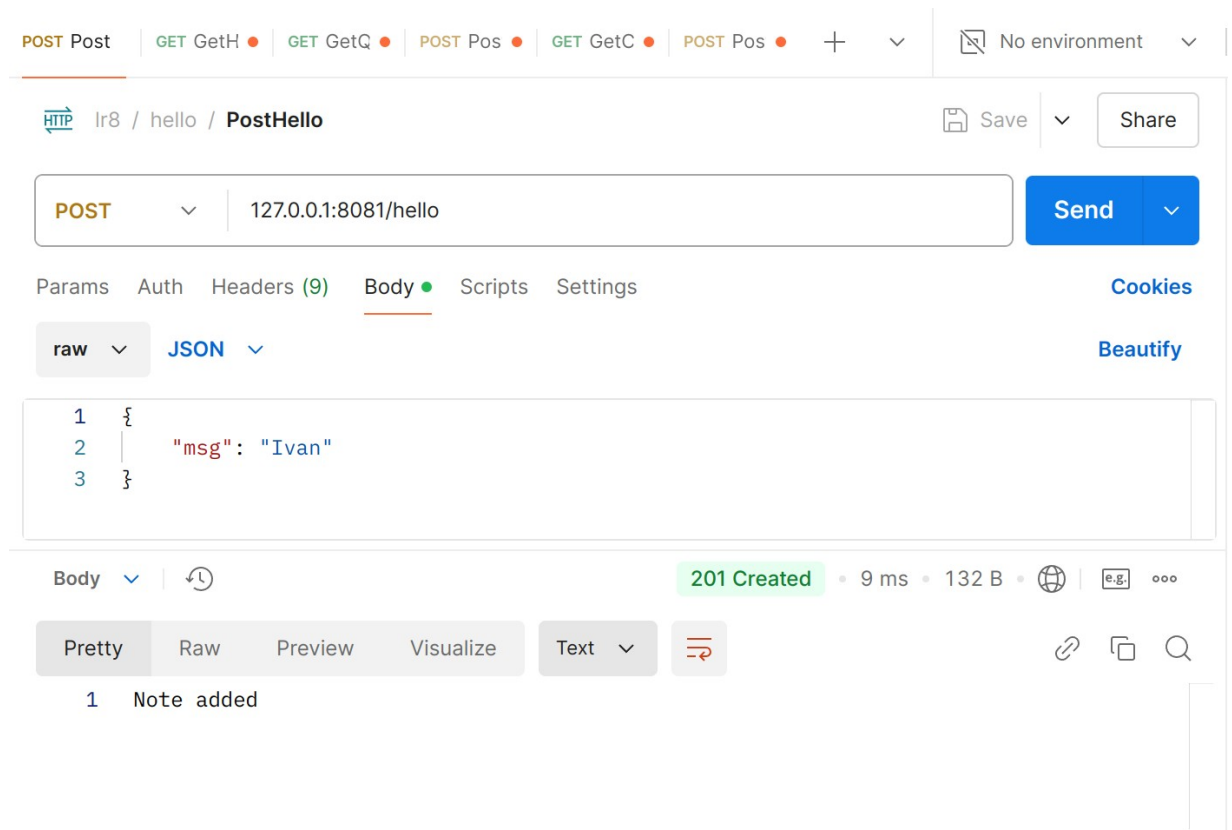


Рисунок 1 — результат Post-запроса hello

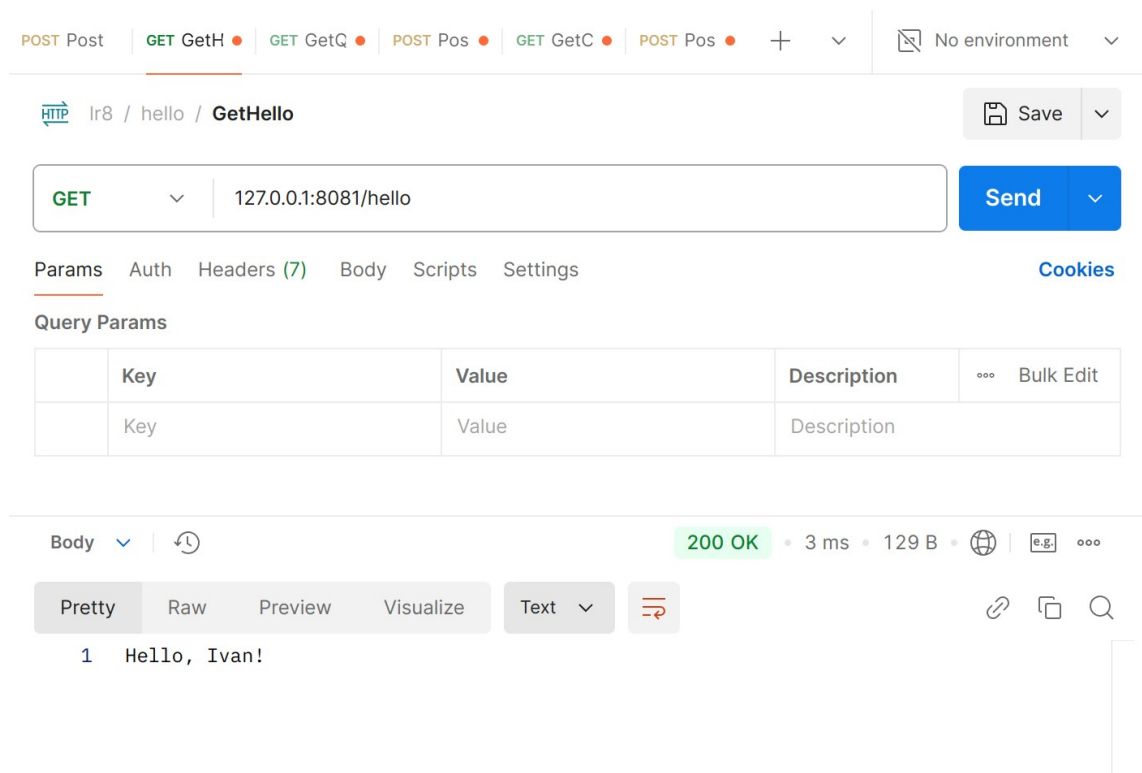


Рисунок 2 - результат выполнения Get-запроса hello

Микросервис Query

```
package main
```

```
import (  
    "database/sql"  
    "flag"  
    "fmt"  
    "log"  
    "net/http"  
  
    "github.com/labstack/echo/v4"  
    "github.com/labstack/echo/v4/middleware"  
    _ "github.com/lib/pq"  
)
```

```
const (  
    host = "localhost"  
    port = 5432  
    user = "ps1"  
    password = "1103"  
    dbname = "lr8"  
)
```

```
type Handlers struct {  
    dbProvider DatabaseProvider  
}
```

```
type DatabaseProvider struct {  
    db *sql.DB  
}
```

```
func (h *Handlers) GetQuery(c echo.Context) error {  
    name := c.QueryParam("msg")
```

```
    if name == "" {  
        return c.String(http.StatusBadRequest, "The parameter is not entered")  
    }
```

```
    test, err := h.dbProvider.SelectQuery(name)  
    if !test && err == nil {  
        return c.String(http.StatusBadRequest, "The note has not been added to DB")  
    } else if !test && err != nil {  
        return c.String(http.StatusInternalServerError, err.Error())  
    }
```

```

return c.String(http.StatusOK, "Hello, "+name+"!")
}

func (h *Handlers) PostQuery(c echo.Context) error {
    name := c.QueryParam("msg")

    if name == "" {
        return c.String(http.StatusBadRequest, "The parameter is not entered")
    }

    test, err := h.dbProvider.SelectQuery(name)
    if test && err == nil {
        return c.String(http.StatusBadRequest, "The note has already been added to DB")
    }

    err = h.dbProvider.InsertQuery(name)
    if err != nil {
        return c.String(http.StatusInternalServerError, err.Error())
    }

    return c.String(http.StatusCreated, "Note added")
}

func (dp *DatabaseProvider) SelectQuery(msg string) (bool, error) {
    var rec string

    row := dp.db.QueryRow("SELECT name_query FROM query WHERE name_query = ($1)", msg)
    err := row.Scan(&rec)
    if err != nil {
        if err == sql.ErrNoRows {
            return false, nil
        }
        return false, err
    }

    return true, nil
}

func (dp *DatabaseProvider) InsertQuery(msg string) error {
    _, err := dp.db.Exec("INSERT INTO query (name_query) VALUES ($1)", msg)
    if err != nil {
        return err
    }

    return nil
}

```



```
}
```

```
func main() {  
    address := flag.String("address", "127.0.0.1:8083", "server startup address")  
    flag.Parse()
```

```
    psqlInfo := fmt.Sprintf("host=%s port=%d user=%s "+  
        "password=%s dbname=%s sslmode=disable",  
        host, port, user, password, dbname)
```

```
    db, err := sql.Open("postgres", psqlInfo)  
    if err != nil {  
        log.Fatal(err)  
    }  
    defer db.Close()
```

```
    dp := DatabaseProvider{db: db}  
    h := Handlers{dbProvider: dp}
```

```
    e := echo.New()
```

```
    e.Use(middleware.Logger())
```

```
    e.GET("/query", h.GetQuery)  
    e.POST("/query", h.PostQuery)
```

```
    e.Logger.Fatal(e.Start(*address))  
}
```

Navigation: < GET GetH • GET GetQt | **POST Post** | GET GetC • POST Pos • > + ▾ No environment ▾

HTTP Ir8 / query / **PostQuery** Save ▾ Share

POST ▾ 127.0.0.1:8083/query?msg=Egor Send ▾

Params • Auth Headers (8) Body Scripts Settings Cookies

Query Params

<input checked="" type="checkbox"/>	Key	Value	Description	⋮ Bulk Edit
<input checked="" type="checkbox"/>	msg	Egor		
	Key	Value	Description	

Body ▾ | ↺ 201 Created • 8 ms • 132 B • 🌐 | e.g. ⋮

Pretty Raw Preview Visualize Text ▾ 🔗 📄 🔍

1 Note added

Рисунок 3 - Post-запрос query

Navigation: POST PostHello | GET GetHello | **GET GetQuery** • POST PostQuer, • + ▾ No environment ▾

HTTP Ir8 / query / **GetQuery** Save ▾

GET ▾ 127.0.0.1:8083/get?name=Ivan Send ▾

Params • Auth Headers (7) **Body** Scripts Settings Cookies

raw ▾ **JSON** ▾ Beautify

1

Body ▾ | ↺ 200 OK • 3 ms • 128 B • 🌐 | e.g. ⋮

Pretty Raw Preview Visualize Text ▾ 🔗 📄 🔍

1 Hello, Ivan!

Рисунок 4 - Get-запрос query

Микросервис count

```
package main
```

```
import (  
    "database/sql"  
    "flag"  
    "fmt"  
    "log"  
    "net/http"  
    "strconv"  
  
    "github.com/labstack/echo/v4"  
    "github.com/labstack/echo/v4/middleware"  
    _ "github.com/lib/pq"  
)
```

```
const (  
    host = "localhost"  
    port = 5432  
    user = "ps1"  
    password = "1103"  
    dbname = "lr8"  
)
```

```
type Handlers struct {  
    dbProvider DatabaseProvider  
}
```

```
type DatabaseProvider struct {  
    db *sql.DB  
}
```

```
func (h *Handlers) GetCounter(c echo.Context) error {  
    msg, err := h.dbProvider.SelectCounter()  
    if err != nil {  
        return c.String(http.StatusInternalServerError, err.Error())  
    }
```

```
    return c.String(http.StatusOK, "Counter: "+strconv.Itoa(msg))  
}
```

```
func (h *Handlers) PostCounter(c echo.Context) error {  
    input := struct {  
        Msg int `json:"msg"`  
    }{}  
}
```

```

err := c.Bind(&input)
if err != nil {
return c.String(http.StatusInternalServerError, err.Error())
}

err = h.dbProvider.UpdateCounter(input.Msg)
if err != nil {
return c.String(http.StatusInternalServerError, err.Error())
}

return c.String(http.StatusOK, "Counter changed")
}

func (dp *DatabaseProvider) SelectCounter() (int, error) {
var msg int

row := dp.db.QueryRow("SELECT number FROM counter WHERE id_number = 1")
err := row.Scan(&msg)
if err != nil {
return -1, err
}

return msg, nil
}

func (dp *DatabaseProvider) UpdateCounter(msg int) error {
_, err := dp.db.Exec("UPDATE counter SET number = number + $1 WHERE
id_number = 1", msg)
if err != nil {
return err
}

return nil
}

func main() {
address := flag.String("address", "127.0.0.1:8081", "server startup adress")
flag.Parse()

psqlInfo := fmt.Sprintf("host=%s port=%d user=%s "+
"password=%s dbname=%s sslmode=disable",
host, port, user, password, dbname)

db, err := sql.Open("postgres", psqlInfo)
if err != nil {

```

```

log.Fatal(err)
}
defer db.Close()

dp := DatabaseProvider{db: db}
h := Handlers{dbProvider: dp}

e := echo.New()

e.Use(middleware.Logger())

e.GET("/counter", h.GetCounter)
e.POST("/counter", h.PostCounter)

e.Logger.Fatal(e.Start(*address))
}

```

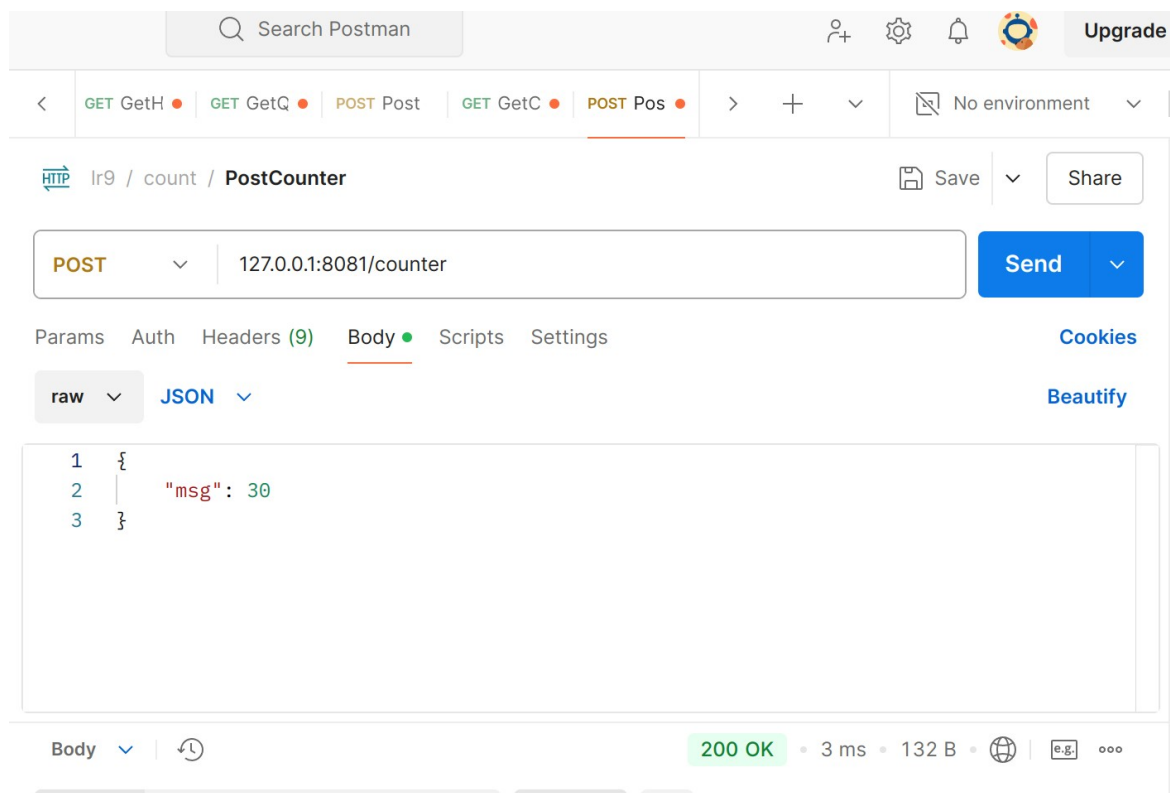


Рисунок 5 - Post-запрос count

The screenshot shows a REST client interface with a tab for 'GET GetC'. The request is a GET to '127.0.0.1:8081/counter'. The response is '200 OK' with a body of 'Counter: 30'.

GET 127.0.0.1:8081/counter

Params Auth Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body 200 OK • 2 ms • 128 B • e.g. ...

Pretty Raw Preview Visualize Text

```
1 Counter: 30
```

Рисунок 6 - Get-запрос count

Вывод

При выполнении заданий лабораторной были получены навыки по работе с фреймворком echo.