

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

ОТЧЕТ

по лабораторной работе № 9			
Название: <u>Echo</u>	Back-End pa3pa	ботка с использованием (фреймворка_
Дисциплина: <u>Языки Интернет-Программирования</u>			
Студент		(Подпись, дата)	К.Д. Коротаев (И.О. Фамилия)
Преподавателн	, , ,	(подинев, дата)	И.О. Фамилия
		(Подпись, дата)	(И.О. Фамилия)

Цель работы: получение первичных навыков использования веб-фрейворков в BackEnd-разрабокте на Golang

Задание:

- 1. Ознакомиться с теоретическими сведениями
- 2. Сделать форк данного репозитория в GitHub, склонировать получившуюся копию локально, создать от мастера ветку dev и переключиться на неё
- 3. Перекопировать код сервисов, полученный в ходе выполнения 8-й лабораторной работы, в соответствующие поддиректории в директории cmd 4. Доработать сервисы таким образом, чтобы роутинг, обработка запросов, парсинг json, обработка ошибок и логирование осуществлялись на базе фреймворка Echo
- 5. Проверить свой код линтерами с помощью команды make lint
- 6. Сделать отчёт и поместить его в директорию docs
- 7.Зафиксировать изменения, сделать коммит и отправить получившееся состояние ветки дев в личный форк данного репозитория в GitHub
- 8. Через интерфейс GitHub создать Pull Request dev --> master
- 9. На защите лабораторной продемонстрировать корректность работы обновленных сервисов на Golang

Ход работы:

}{}

```
Код для работы с сервисом Hello:
package main
import (
      "database/sql"
      "flag"
      "fmt"
      "log"
      "net/http"
      "github.com/labstack/echo"
      "github.com/labstack/echo/middleware"
      _ "github.com/lib/pq"
)
const (
      host = "localhost"
      port = 5432
      user = "postgres"
      dbname = "sandbox9"
)
type Handlers struct {
      dbProvider DatabaseProvider
}
type DatabaseProvider struct {
      db *sql.DB
}
// Обработчики НТТР-запросов
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, 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.StatusBadRequest, err.Error())
      }
      err = h.dbProvider.InsertHello(input.Msg)
      if err != nil {
            return c.String(http.StatusInternalServerError, err.Error())
      return c.String(http.StatusCreated, "Добавили запись!")
}
// Методы для работы с базой данных
func (dp *DatabaseProvider) SelectHello() (string, error) {
      var msg string
      // Получаем одно сообщение из таблицы hello, отсортированной в
случайном порядке
      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() {
      // Считываем аргументы командной строки
      address := flag.String("address", "127.0.0.1:8081", "адрес для запуска сервера")
      flag.Parse()
      // Формирование строки подключения для postgres
      psqlInfo := fmt.Sprintf("host=%s port=%d user=%s "+
```

```
"dbname=%s sslmode=disable",
            host, port, user, dbname)
      // Создание соединения с сервером postgres
      db, err := sql.Open("postgres", psqlInfo)
      if err != nil {
            log.Fatal(err)
      defer db.Close()
      // Создаем провайдер для БД с набором методов
      dp := DatabaseProvider{db: db}
      // Создаем экземпляр структуры с набором обработчиков
      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))
}
Код для работы с сервисом 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 = "postgres"
      dbname = "sandbox9"
```

```
)
type Handlers struct {
      dbProvider DatabaseProvider
type DatabaseProvider struct {
      db *sql.DB
// обработчики http-запросов
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, "Счетчик: "+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, "Изменили счетчик!")
// методы для работы с базой данных
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
```

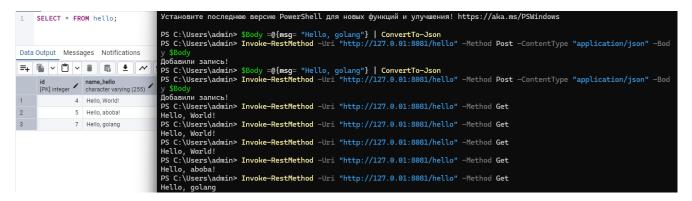
```
return msg, nil
}
func (dp *DatabaseProvider) UpdateCounter(msg int) error {
      _, err := dp.db.Exec("UPDATE counter SET number = number + $1 WHERE
id_number = 1", msg)
      if err != nil {
            return err
      return nil
}
func main() {
      // Считываем аргументы командной строки
      address := flag.String("address", "127.0.0.1:8082", "адрес для запуска сервера")
      flag.Parse()
      // Формирование строки подключения для postgres
      psqlInfo := fmt.Sprintf("host=%s port=%d user=%s "+
            "dbname=%s sslmode=disable",
            host, port, user, dbname)
      // Создание соединения с сервером postgres
      db, err := sql.Open("postgres", psqlInfo)
      if err != nil {
            log.Fatal(err)
      defer db.Close()
      // Создаем провайдер для БД с набором методов
      dp := DatabaseProvider{db: db}
     // Создаем экземпляр структуры с набором обработчиков
      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))
}
Код для работы с сервисом 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 = "postgres"
      dbname = "sandbox9"
)
type Handlers struct {
      dbProvider DatabaseProvider
}
type DatabaseProvider struct {
      db *sql.DB
}
// Обработчики НТТР-запросов
func (h *Handlers) GetQuery(c echo.Context) error {
      name := c.QueryParam("name")
      if name == "" {
            return c.String(http.StatusBadRequest, "Не введен параметр!")
      test, err := h.dbProvider.SelectQuery(name)
      if !test && err == nil {
            return c.String(http.StatusBadRequest, "Запись не добавлена в БД!")
      } 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("name")
```

```
if name == "" {
             return c.String(http.StatusBadRequest, "Не введен параметр!")
      }
      test, err := h.dbProvider.SelectQuery(name)
      if test && err == nil {
            return c.String(http.StatusBadRequest, "Запись уже добавлена в БД!")
      }
      err = h.dbProvider.InsertQuery(name)
      if err != nil {
            return c.String(http.StatusInternalServerError, err.Error())
      }
      return c.String(http.StatusCreated, "Добавили запись!")
}
// Методы для работы с базой данных
func (dp *DatabaseProvider) SelectQuery(msg string) (bool, error) {
      var rec string
      row := dp.db.QueryRow("SELECT record FROM query WHERE record = ($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 (record) VALUES ($1)", msg) if err != nil {
            return err
      }
      return nil
}
func main() {
      // Считываем аргументы командной строки
      address := flag.String("address", "127.0.0.1:8083", "адрес для запуска сервера")
```

```
flag.Parse()
     // Формирование строки подключения для postgres
     psqlInfo := fmt.Sprintf("host=%s port=%d user=%s "+
            "dbname=%s sslmode=disable",
           host, port, user, dbname)
     // Создание соединения с сервером postgres
     db, err := sql.Open("postgres", psqlInfo)
     if err != nil {
           log.Fatal(err)
     defer db.Close()
     // Создаем провайдер для БД с набором методов
     dp := DatabaseProvider{db: db}
     // Создаем экземпляр структуры с набором обработчиков
     h := Handlers {dbProvider: dp}
     e := echo.New()
     e.Use(middleware.Logger())
     e.GET("/query", h.GetQuery)
     e.POST("/query1", h.PostQuery)
     e.Logger.Fatal(e.Start(*address))
}
```

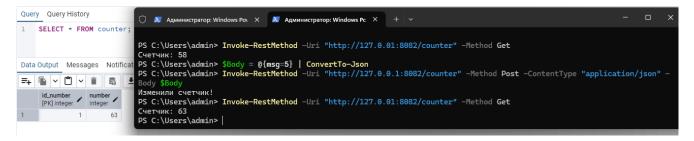
Тестирование к сервису Hello:



Тестирование к сервису Query:



Тестирование к сервису Count:



Заключение: получены первичные навыки использования веб-фрейворков в BackEnd-разрабокте на Golang