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**

О Т Ч Е Т

по лабораторной работе № 9

Название: **Back-End разработка с использованием фреймворка Echo**

Дисциплина: **Языки интернет-программирования**

Студент ИУ6-31Б _____ Н.Е. Мамаев
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ В.Д. Шульман
(Подпись, дата) (И.О. Фамилия)

Москва, 2024

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

Задание. Перекопировать код сервисов, полученный в ходе выполнения 8-й лабораторной работы, в соответствующие поддиректории в директории cmd Доработать сервисы таким образом, чтобы роутинг, обработка запросов, парсинг json, обработка ошибок и логирование осуществлялись на базе фреймворка Echo.

Обработчики http-запросов для микросервиса hello изображены на рисунке 1:

```
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!")
}
```

Рисунок 1. Обработчики http-запросов для микросервиса hello.

Обработчики http-запросов для микросервиса counter изображены на рисунке 2:

```go
//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!")
}
```

Рисунок 2. Обработчики http-запросов для микросервиса counter.

Обработчики http-запросов для микросервиса query изображены на рисунке 3:

```go
// Обработчики HTTP-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!")
}
```

Рисунок 3. Обработчики http-запросов для микросервиса query.

Изменение в main изображено на рисунке 4:

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

    e.Use(middleware.Logger())

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

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

Рисунок 4. Изменение функции main.

Пример работы микросервиса counter изображены на рисунках 5-7:

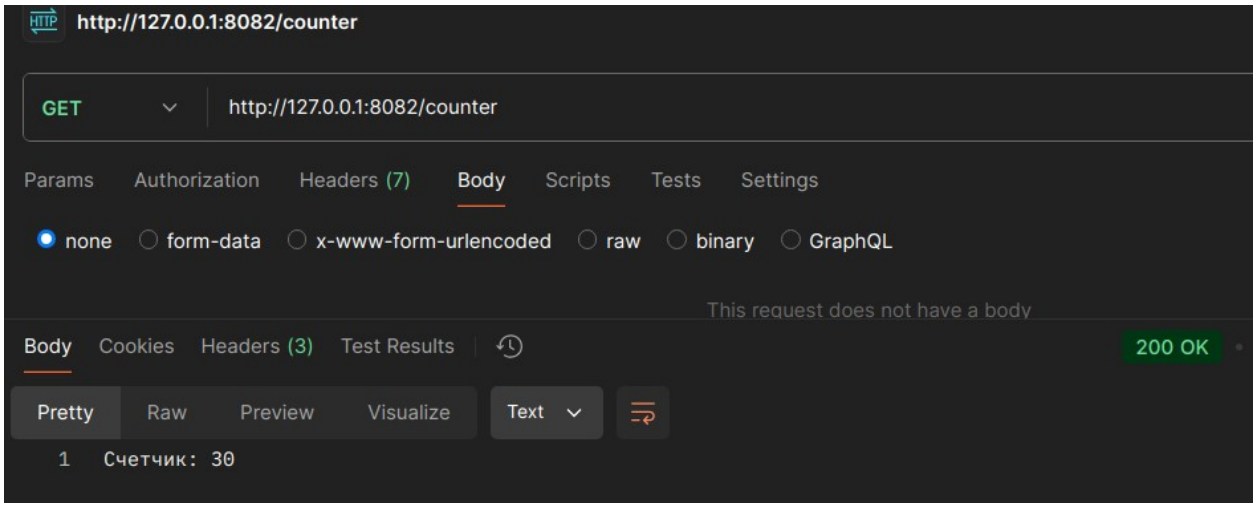

Рисунок 5. Результат get-запроса.

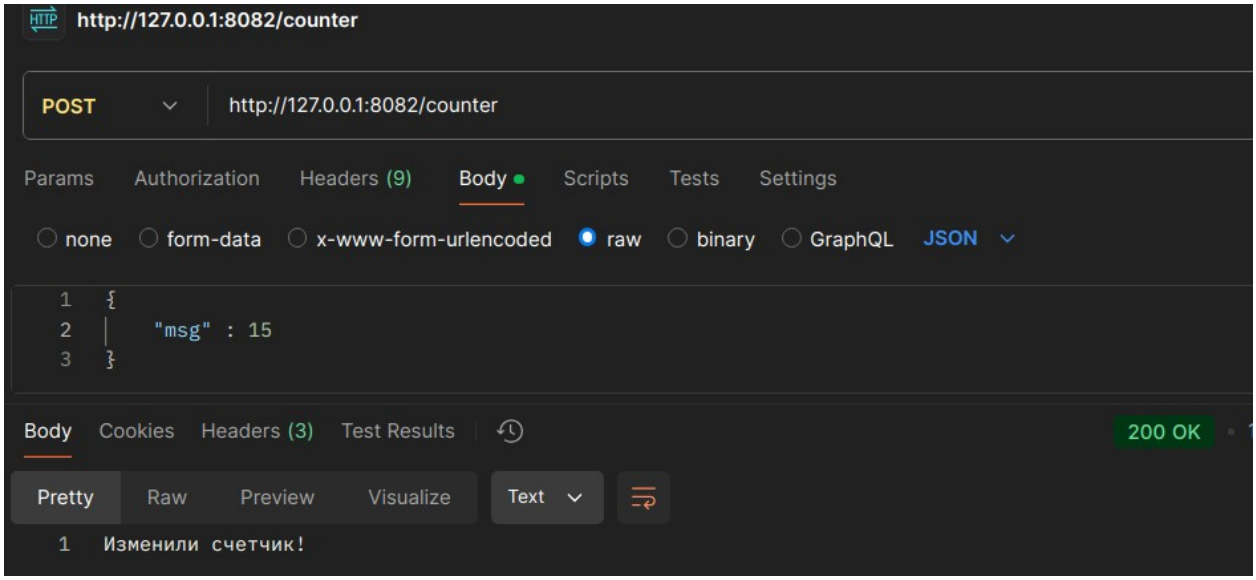

Рисунок 6. Результат post-запроса, изменения счетчика.

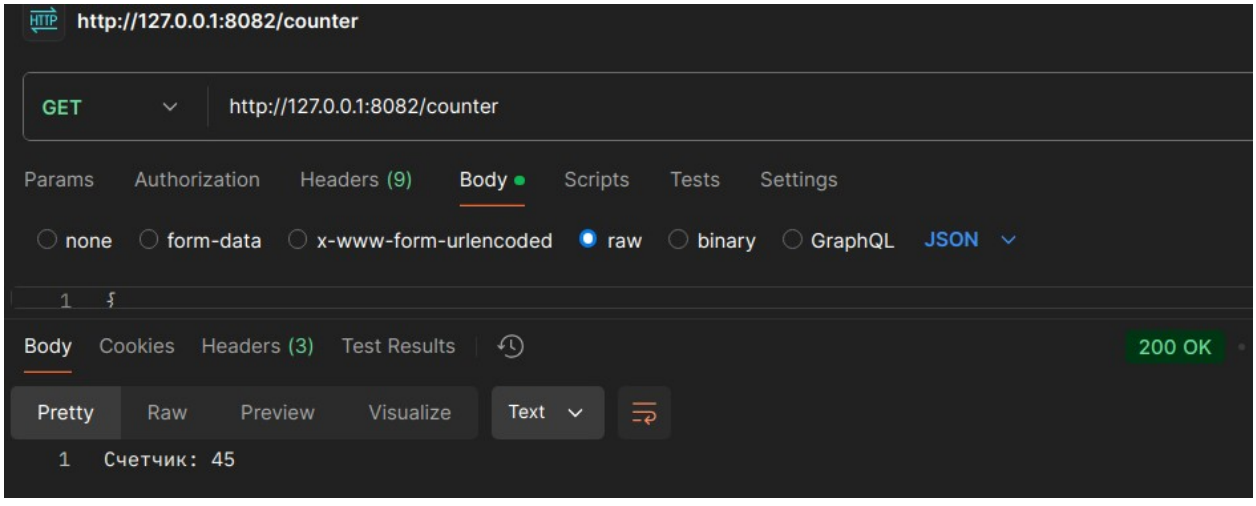

Рисунок 7. Результат get-запроса после изменения счетчика

Вывод: в ходе лабораторной работы были получены навыки использования веб-фрейворка echo в BackEnd-разработке на Golang.