



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 № 6

Название: Основы Back-End разработки на Golang

Дисциплина: Языки интернет-программирования

Студент	<u>ИУ6-32Б</u> (Группа)	<u>8.10.2024</u> (Подпись, дата)	<u>А.Т.Ахметзянов</u> (И.О. Фамилия)
Преподаватель		<u>8.10.2024</u> (Подпись, дата)	<u>В.Д. Шульман</u> (И.О. Фамилия)

Москва, 2024

Цель работы — изучение основ сетевого взаимодействия и серверной разработки с использованием языка Golang.

Ход работы.

1. Ознакомились с курсом <https://stepik.org/course/54403/info>
2. Сделали форк данного репозитория в GitHub, клонировали получившуюся копию локально, создали от мастера ветку дев и переключились на нее:

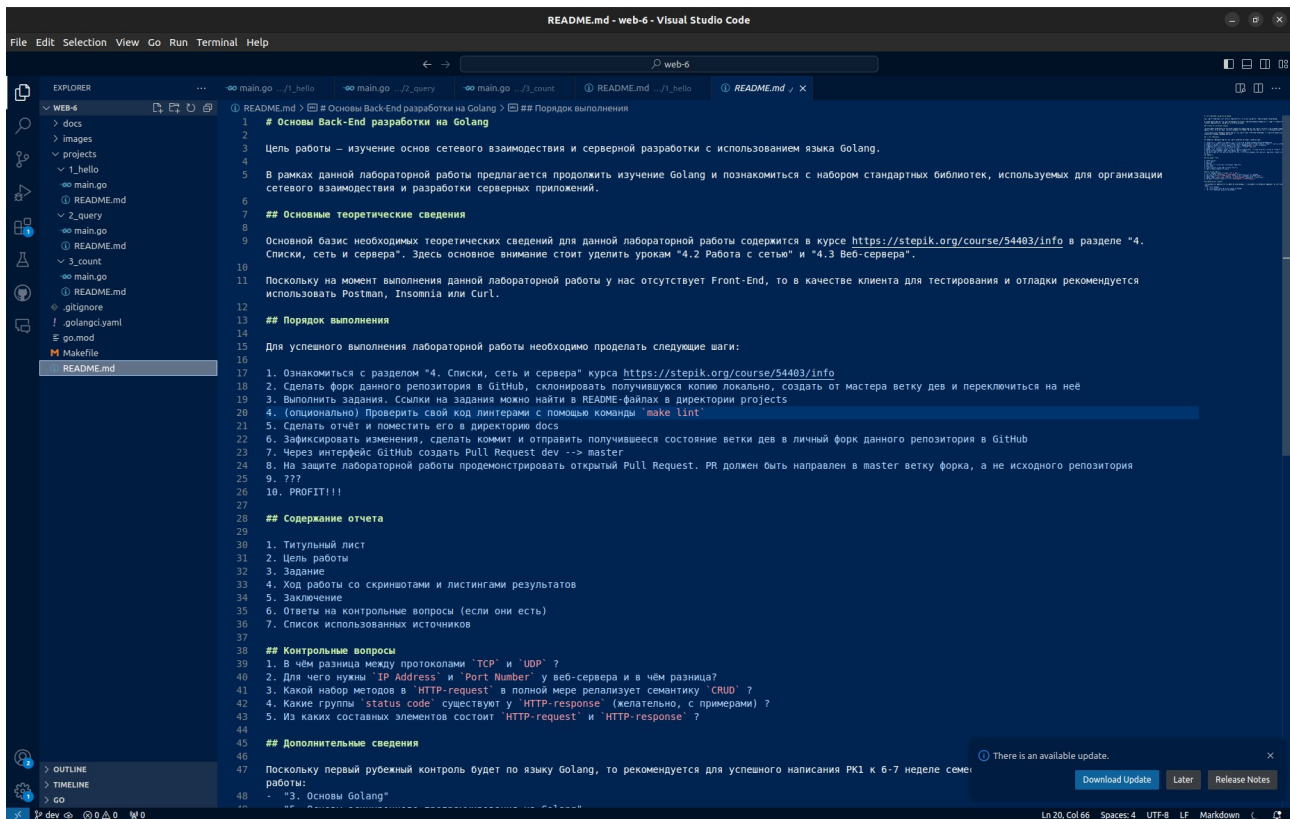


Рисунок 1 — Скопированный репозиторий

3. Написали 3 HTTP сервера на GoLAng. Код серверов и результаты запросов в Postman прикрепили ниже:

Задача 1(Вывод строки):

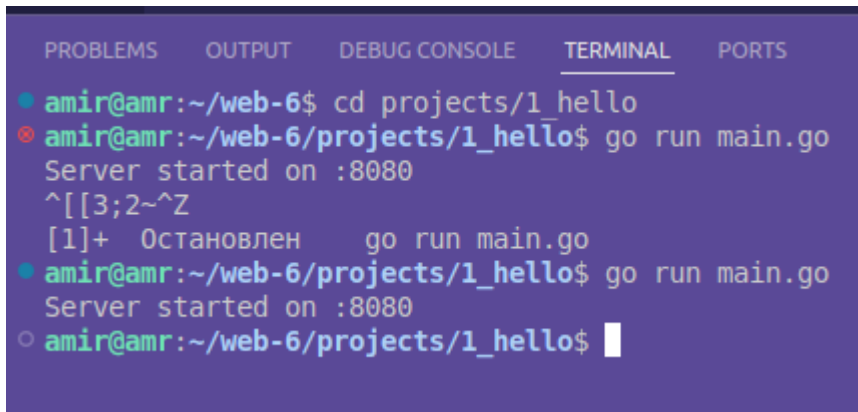
```
package main

import (
    "fmt"
    "net/http"
)

func main() {
    http.HandleFunc("/getMessage", handleGetRequest)
```

```
fmt.Println("Server started on :8080")
http.ListenAndServe(":8080", nil)
}
```

```
func handleGetRequest(w http.ResponseWriter, r *http.Request) {
fmt.Fprint(w, "Hello, web!")
}
```



```
amir@amr:~/web-6$ cd projects/1_hello
amir@amr:~/web-6/projects/1_hello$ go run main.go
Server started on :8080
^[[3;2~^Z
[1]+  Остановлен go run main.go
amir@amr:~/web-6/projects/1_hello$ go run main.go
Server started on :8080
amir@amr:~/web-6/projects/1_hello$
```

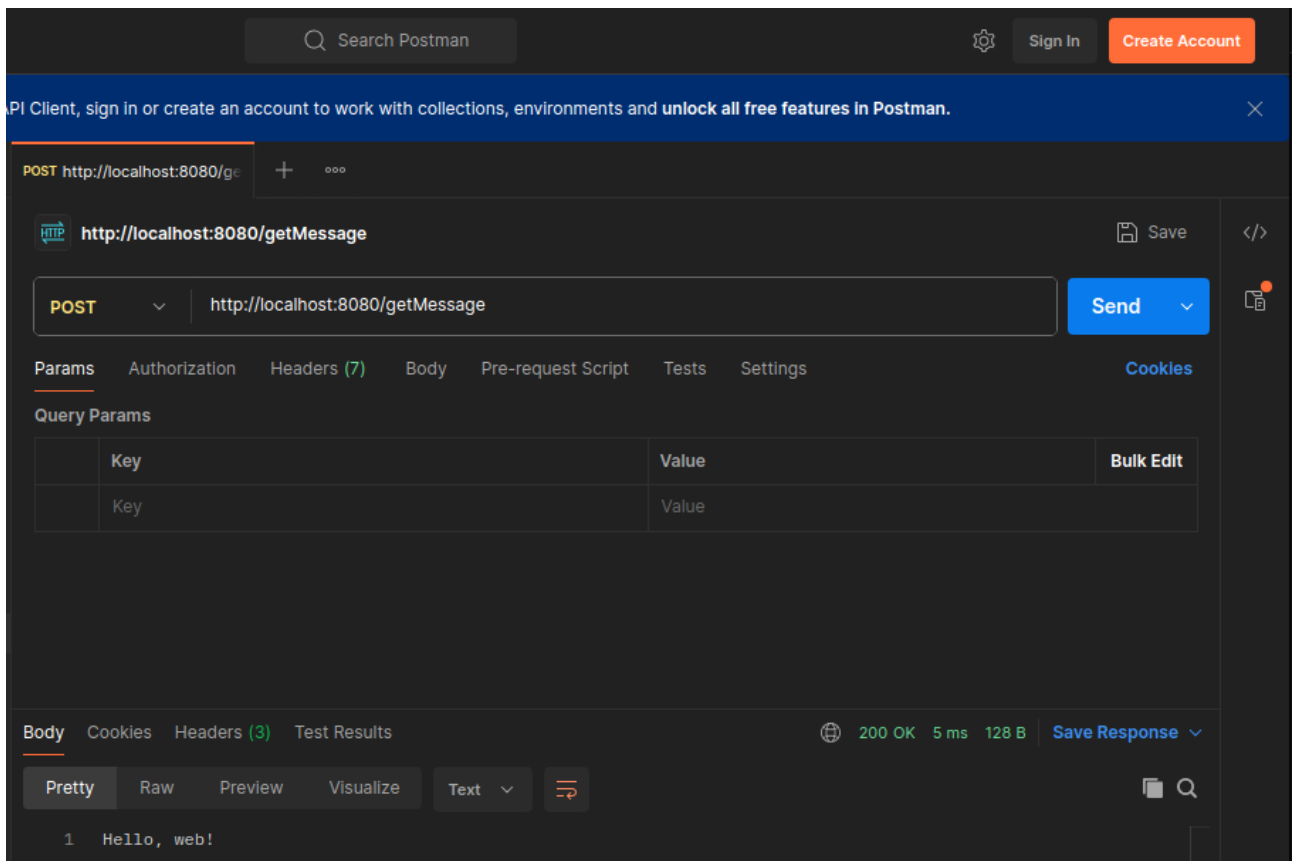


Рисунок 2 — Вывод задачи 1

Задача 2(Вывод строки с ключом):

```
package main
```

```
import (
    "fmt"
    "net/http"
)

func main() {
    http.HandleFunc("/api/user", handleUserRequest)
    fmt.Println("Starting server on :8080")
    http.ListenAndServe(":8080", nil)
}

func handleUserRequest(w http.ResponseWriter, r *http.Request) {
    name := r.URL.Query().Get("name")
    if name == "" {
        http.Error(w, "Name parameter is required", http.StatusBadRequest)
        return
    }

    fmt.Fprintf(w, "Hello, %s!", name)
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
• amir@amr:~/web-6$ cd projects/2_query
○ amir@amr:~/web-6/projects/2_query$ go run main.go
Starting server on :8080
□
```

POST http://localhost:8080/api/user?name=AMR

Save </>

POST http://localhost:8080/api/user?name=AMR Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Bulk Edit
<input checked="" type="checkbox"/>	name	AMR	
	Key	Value	

Body Cookies Headers (3) Test Results 200 OK 2 ms 128 B Save Response

Pretty Raw Preview Visualize Text

1 Hello, AMR!

Рисунок 3 — Вывод задачи 2

Задача 3(Store):

```
package main

import (
    "fmt"
    "net/http"
    "strconv"
)

var counter int = 0

func main() {
    http.HandleFunc("/count", handleCount)
    fmt.Println("Сервер запущен на порту :3333")
    http.ListenAndServe(":3333", nil)
}

func handleCount(w http.ResponseWriter, r *http.Request) {
    switch r.Method {
    case "GET":
        fmt.Fprintf(w, "%d", counter)
    case "POST":
        count, err := strconv.Atoi(r.FormValue("count"))
        if err != nil {
            http.Error(w, "это не число", http.StatusBadRequest)
            return
        }
        counter += count
        fmt.Fprintf(w, "Success")
    default:
        http.Error(w, "Неизвестный метод", http.StatusMethodNotAllowed)
    }
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
• amir@amr:~/web-6$ cd projects/3_count
○ amir@amr:~/web-6/projects/3_count$ go run main.go
Сервер запущен на порту :3333
```

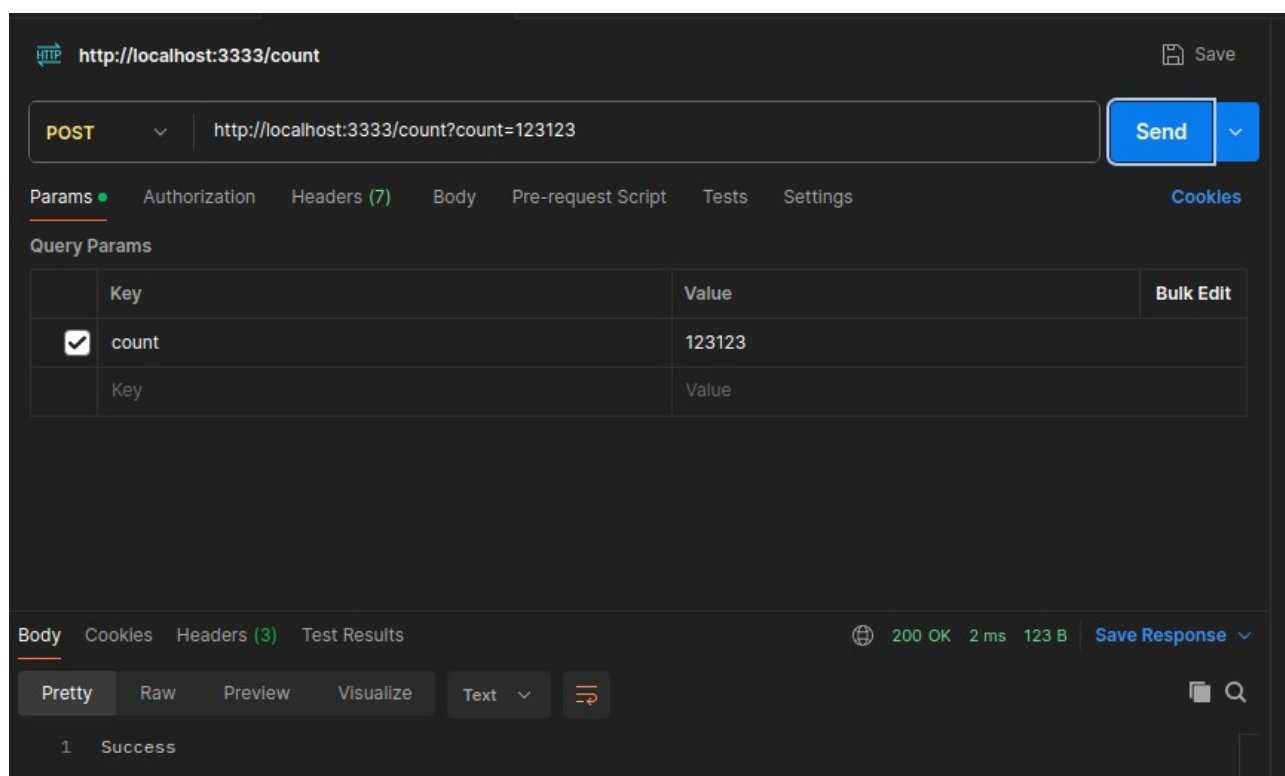


Рисунок 4 — Post запрос

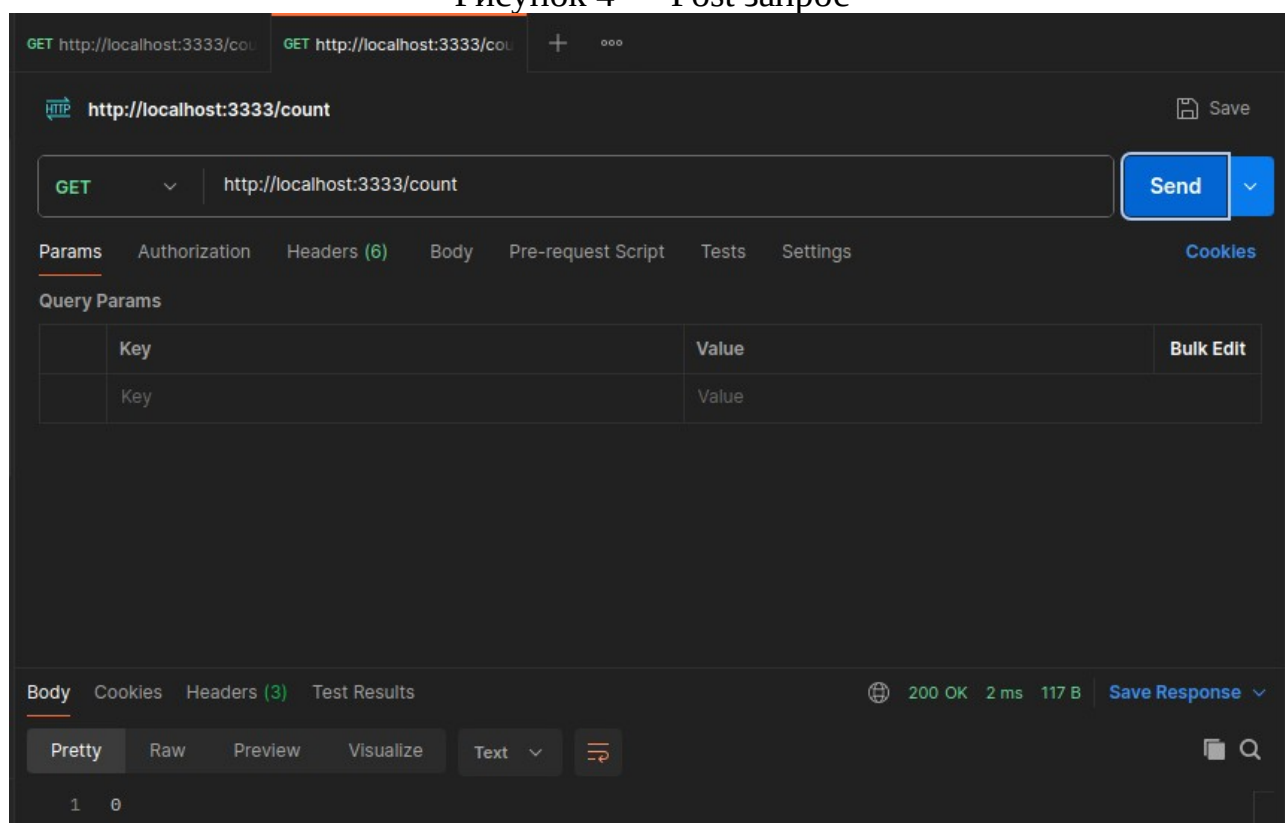


Рисунок 5 — Get запрос

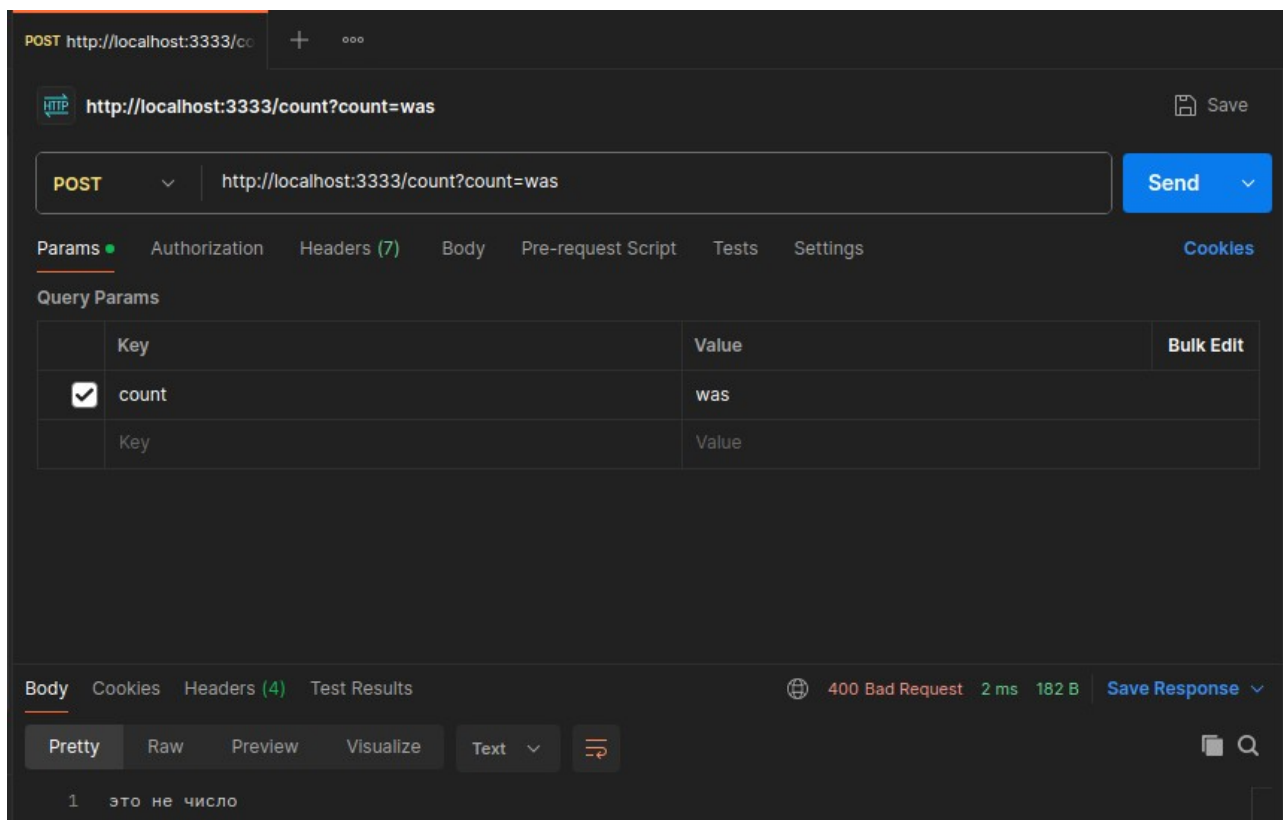


Рисунок 5 — Обработка исключения к задаче 3

4. Зафиксировали изменения, сделали коммит и отправили полученное состояние ветки дев в удаленный репозиторий GitHub. Через интерфейс GitHub создали Pull Request dev --> master

Заключение: в ходе лабораторной работы изучили основы сетевого взаимодействия и серверной разработки с использованием языка Golang. Освоили работу с Postman.

Контрольные вопросы:

1. Разница между протоколами TCP и UDP:

- TCP (Transmission Control Protocol) - надежный, ориентированный на соединение протокол. Он обеспечивает гарантированную доставку данных, контроль потока и порядка пакетов.
- UDP (User Datagram Protocol) - ненадежный, без установления соединения протокол. Он не гарантирует доставку пакетов и не контролирует их порядок. Применяется для передачи данных, где важна скорость, а не надежность (например, потоковое видео).

2. IP-адрес и номер порта веб-сервера:

- IP-адрес (Internet Protocol address) - уникальный идентификатор устройства в сети Интернет. Он позволяет маршрутизировать трафик до нужного устройства.
- Номер порта (Port number) - номер логического "канала" на хосте, используемый для идентификации приложения, принимающего и

отправляющего сетевые пакеты. Это позволяет нескольким приложениям на одном хосте обмениваться данными независимо.

3. Методы HTTP, реализующие CRUD:

- Create (POST)
- Read (GET)
- Update (PUT/PATCH)
- Delete (DELETE)

4. Группы кодов состояния HTTP-ответов:

- 1xx (Informational) - запрос принят, продолжается обработка
- 2xx (Success) - запрос успешно обработан (например, 200 OK)
- 3xx (Redirection) - клиенту требуется выполнить дополнительные действия (например, 301 Moved Permanently)
- 4xx (Client Error) - ошибка на стороне клиента (например, 404 Not Found)
- 5xx (Server Error) - ошибка на стороне сервера (например, 500 Internal Server Error)

5. Элементы HTTP-запроса и HTTP-ответа:

HTTP-запрос:

- Метод (GET, POST, PUT, DELETE, etc.)
- URL
- Заголовки (Headers)
- Тело (Body)

HTTP-ответ:

- Версия протокола
- Код состояния (Status Code)
- Заголовки (Headers)
- Тело (Body)