|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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)**

**Отчет**

**по лабораторной работе № 9**

**Дисциплина:** Языки интернет-программирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-32Б |  |  | А. В. Тимохин |
|  | (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-разрабокте на Golang

Задача 1 (hello):

Перекопировать код из соответствующей задачи лабораторной №8, а затем переписать ее через echo.

Воспользуемся базой данных из предыдущего задания

Перепишем код, используя echo.

Вот часть функций, в которых мы используем эхо:

func (h \*Handlers) GetHello(c echo.Context) error {

    msg, err := h.dbProvider.SelectHello()

    if err != nil {

        return c.JSON(http.StatusInternalServerError, map[string]string{"error": err.Error()})

    }

    return c.JSON(http.StatusOK, map[string]string{"message": msg})

}

func (h \*Handlers) PostHello(c echo.Context) error {

    input := struct {

        Msg string `json:"msg"`

    }{}

    // Парсинг JSON-запроса с помощью Echo

    if err := c.Bind(&input); err != nil {

        return c.JSON(http.StatusBadRequest, map[string]string{"error": err.Error()})

    }

    // Вставка данных в базу

    if err := h.dbProvider.InsertHello(input.Msg); err != nil {

        return c.JSON(http.StatusInternalServerError, map[string]string{"error": err.Error()})

    }

    return c.NoContent(http.StatusCreated)

}

func main() {

    address := flag.String("address", "127.0.0.1:8081", "адрес для запуска сервера")

    flag.Parse()

    // Формирование строки подключения PostgreSQL

    psqlInfo := fmt.Sprintf("host=%s port=%d user=%s "+

        "password=%s dbname=%s sslmode=disable",

        host, port, user, password, dbname)

    // Подключение к БД

    db, err := sql.Open("postgres", psqlInfo)

    if err != nil {

        log.Fatal(err)

    }

    defer db.Close()

    // Создаем провайдер базы данных и обработчики

    dp := DatabaseProvider{db: db}

    h := Handlers{dbProvider: dp}

    // Создаем экземпляр Echo

    e := echo.New()

    // Регистрируем маршруты

    e.GET("/get", h.GetHello)

    e.POST("/post", h.PostHello)

    // Запускаем сервер Echo

    fmt.Println("Starting server on:", \*address)

    if err := e.Start(\*address); err != nil {

        log.Fatal(err)

    }

}

Запуск программы:

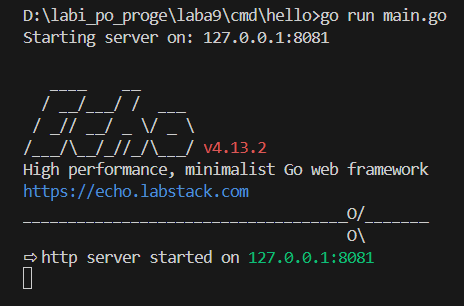


Рисунок 1 – запуск сервера

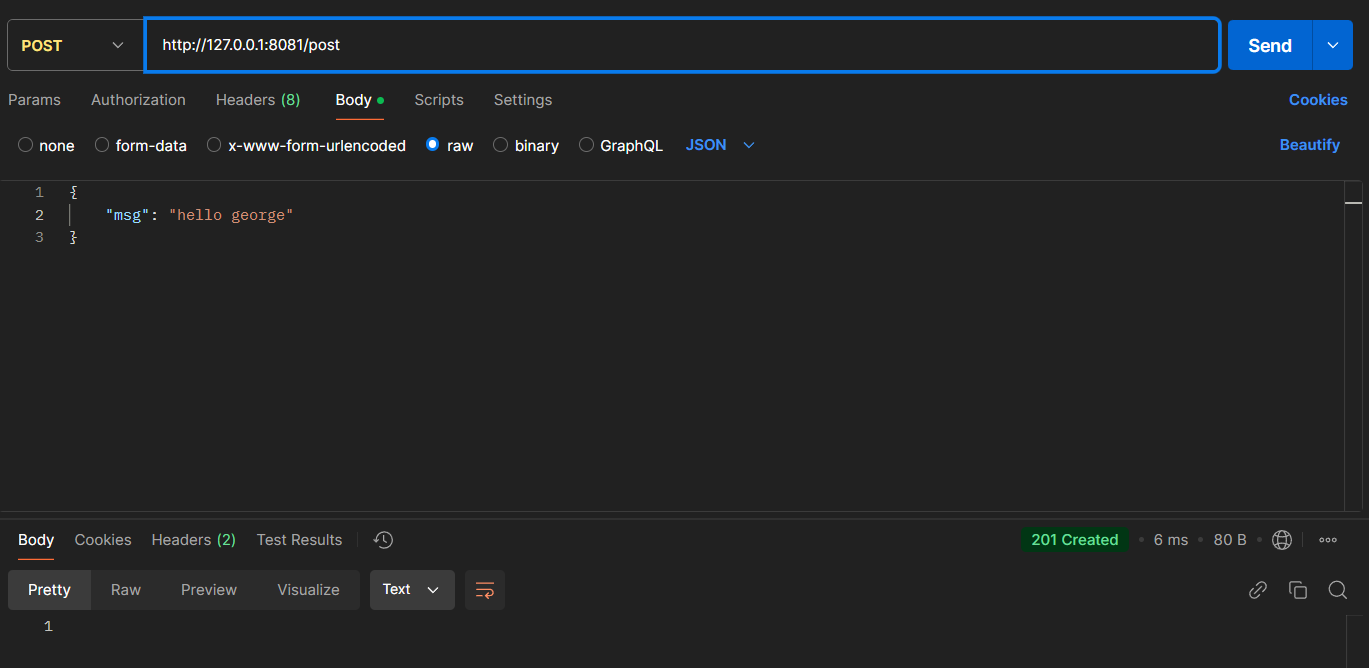


Рисунок 2 – отправка сообщений через postman

Результат отправки сообщений в БД:

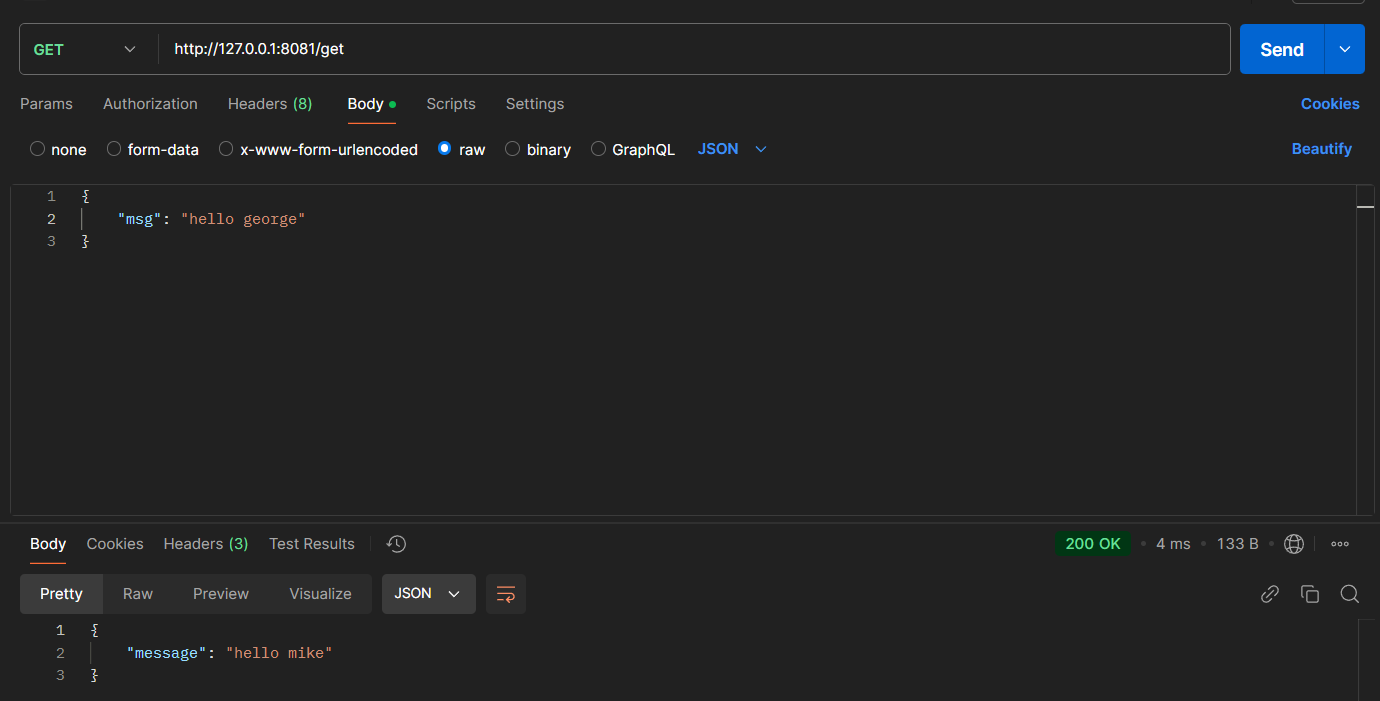
****

Рисунок 3 – результат работы программы

Задача 2 (count):

Перекопировать код из соответствующей задачи лабораторной №8 и реализовать хранение данных в БД PostgreSQL.

Воспользуемся базой данных из предыдущего задания

Перепишем код, используя echo.

Вот часть функций, в которых мы используем эхо:

func (h \*Handlers) GetCount(c echo.Context) error {

    count, err := h.dbProvider.SelectCount()

    if err != nil {

        return c.JSON(http.StatusInternalServerError, map[string]string{"error": err.Error()})

    }

    return c.JSON(http.StatusOK, map[string]int{"count": count})

}

func (h \*Handlers) PostCount(c echo.Context) error {

    input := struct {

        Count int `json:"count"`

    }{}

    // Парсинг JSON-запроса

    if err := c.Bind(&input); err != nil {

        return c.JSON(http.StatusBadRequest, map[string]string{"error": "Invalid input"})

    }

    // Обновление значения в базе

    if err := h.dbProvider.UpdateCount(input.Count); err != nil {

        return c.JSON(http.StatusInternalServerError, map[string]string{"error": err.Error()})

    }

    return c.NoContent(http.StatusCreated)

}

Получим следующий результат:

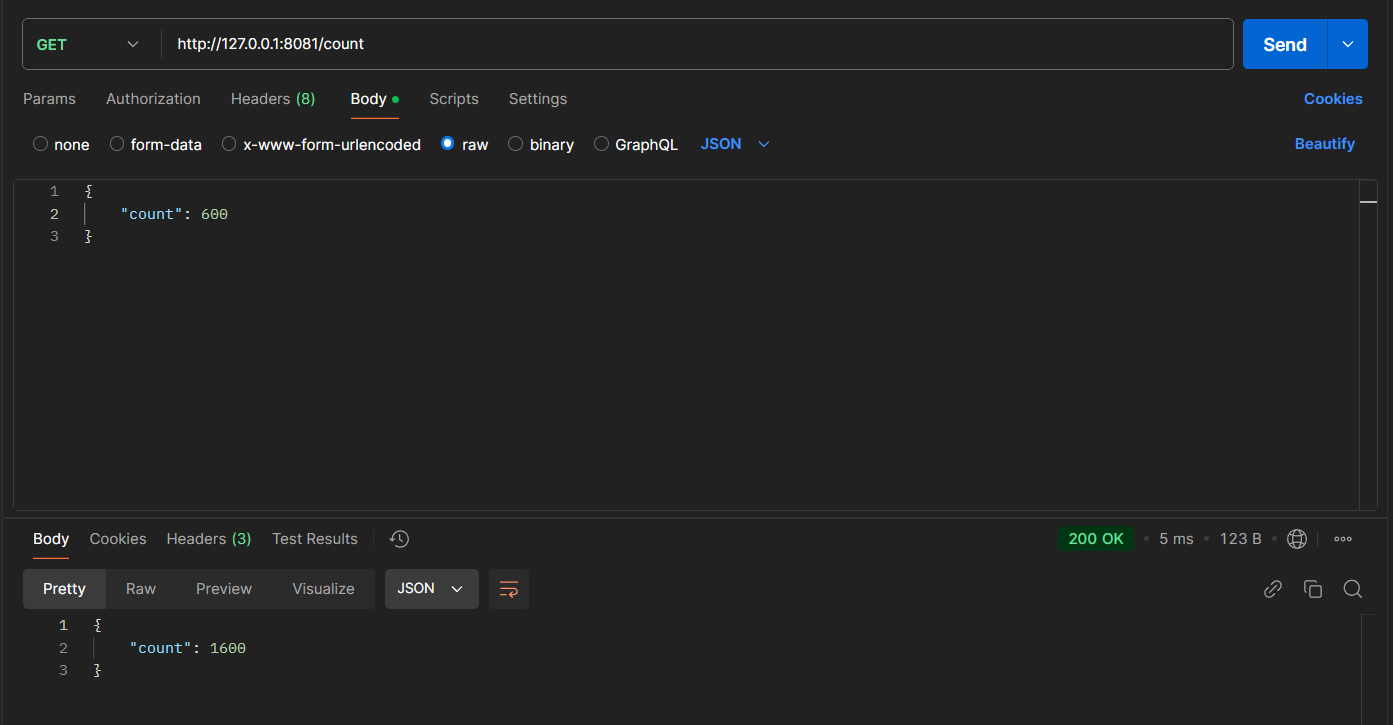


Рисунок 4 - Начальное значение в бд при запросе GET (Оно не равно 0, так я сам уже проводил тесты)

Теперь увеличим его с помощью команды POST:

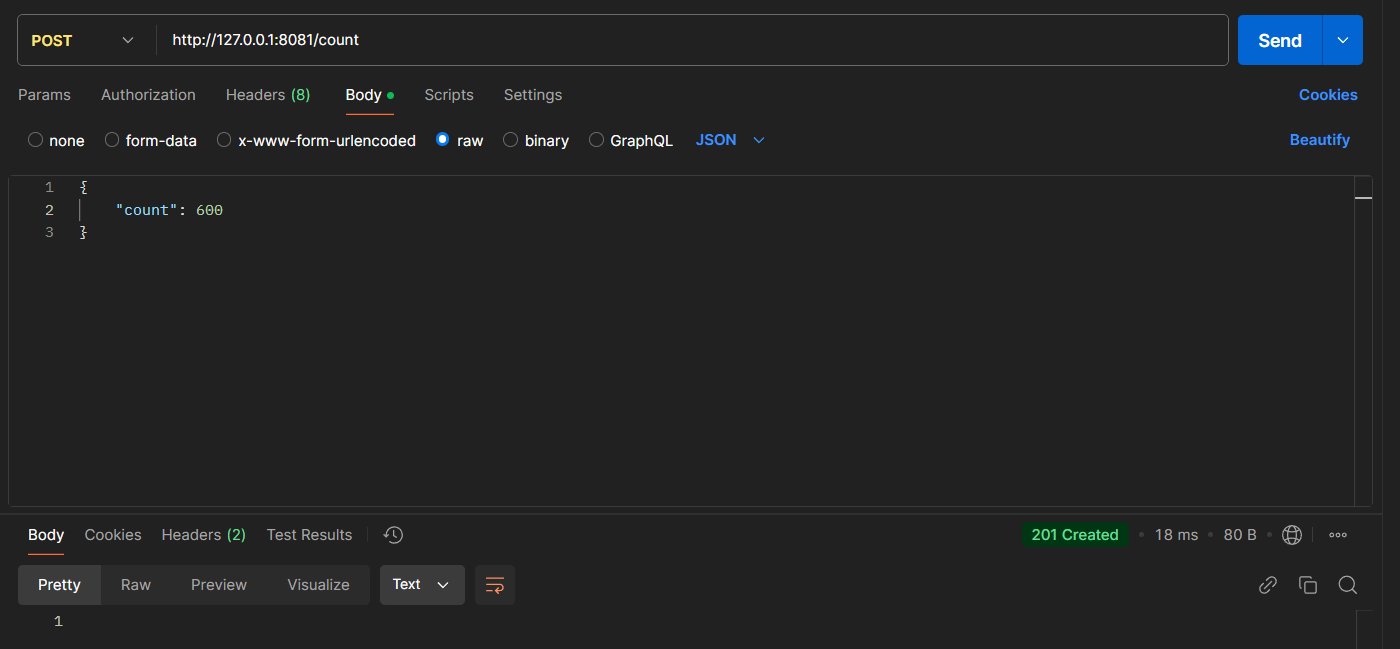


Рисунок 5 – изменение значения

И снова посмотрим на значение с помощью запроса GET:

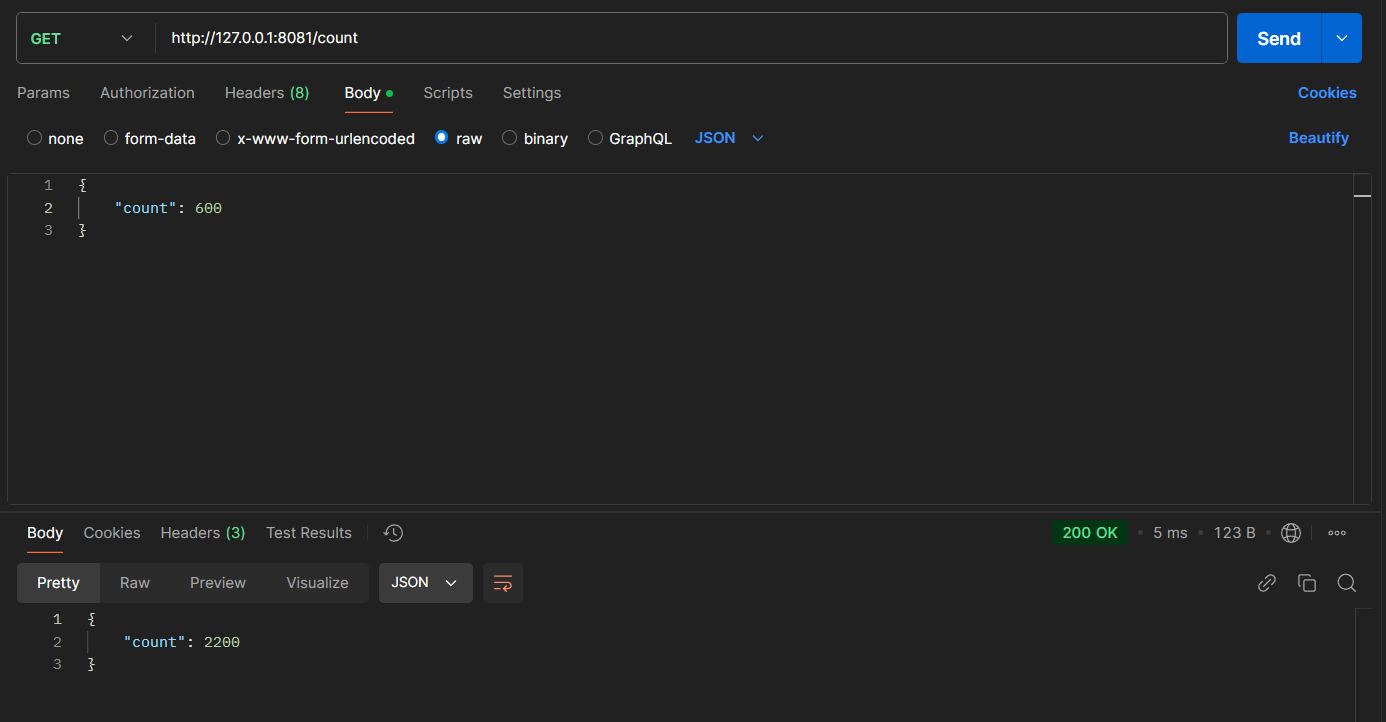


Рисунок 6 – результат работы программы

Как мы видим программа работает исправно.

Задача 3 (query):

Перекопировать код из соответствующей задачи лабораторной №6 и реализовать хранение данных в БД PostgreSQL.

Создадим таблицу с помощью этого кода:

CREATE TABLE usernames (

name\_user text

);

Часть кода программы (представлен только код функции main(), остальной код в соответствующем файле в папке cmd:

func (h \*Handlers) GreetGet(c echo.Context) error {

    name := c.QueryParam("name")

    if name == "" {

        return c.String(http.StatusOK, "Hello, stranger!")

    }

    // Проверяем наличие имени в базе

    exists, err := h.dbProvider.SelectHello(name)

    if err != nil {

        return c.JSON(http.StatusInternalServerError, map[string]string{"error": err.Error()})

    }

    if !exists {

        return c.String(http.StatusOK, "Such user does not exist!")

    }

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

}

func (h \*Handlers) GreetPost(c echo.Context) error {

    name := c.QueryParam("name")

    if name == "" {

        return c.String(http.StatusOK, "Hello, stranger!")

    }

    // Добавляем имя в базу данных

    err := h.dbProvider.InsertHello(name)

    if err != nil {

        return c.JSON(http.StatusInternalServerError, map[string]string{"error": err.Error()})

    }

    return c.NoContent(http.StatusCreated)

}

При вводе post запроса получим:

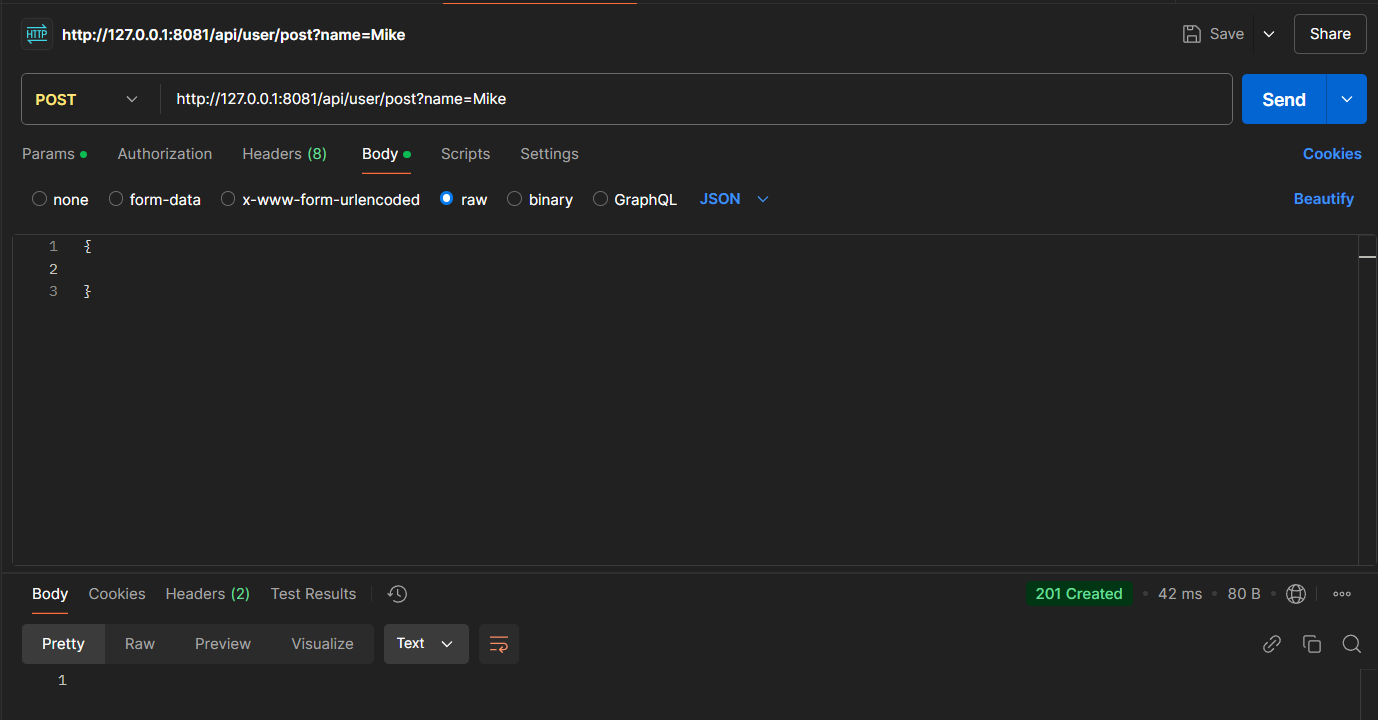


Рисунок 7 – результат post запроса

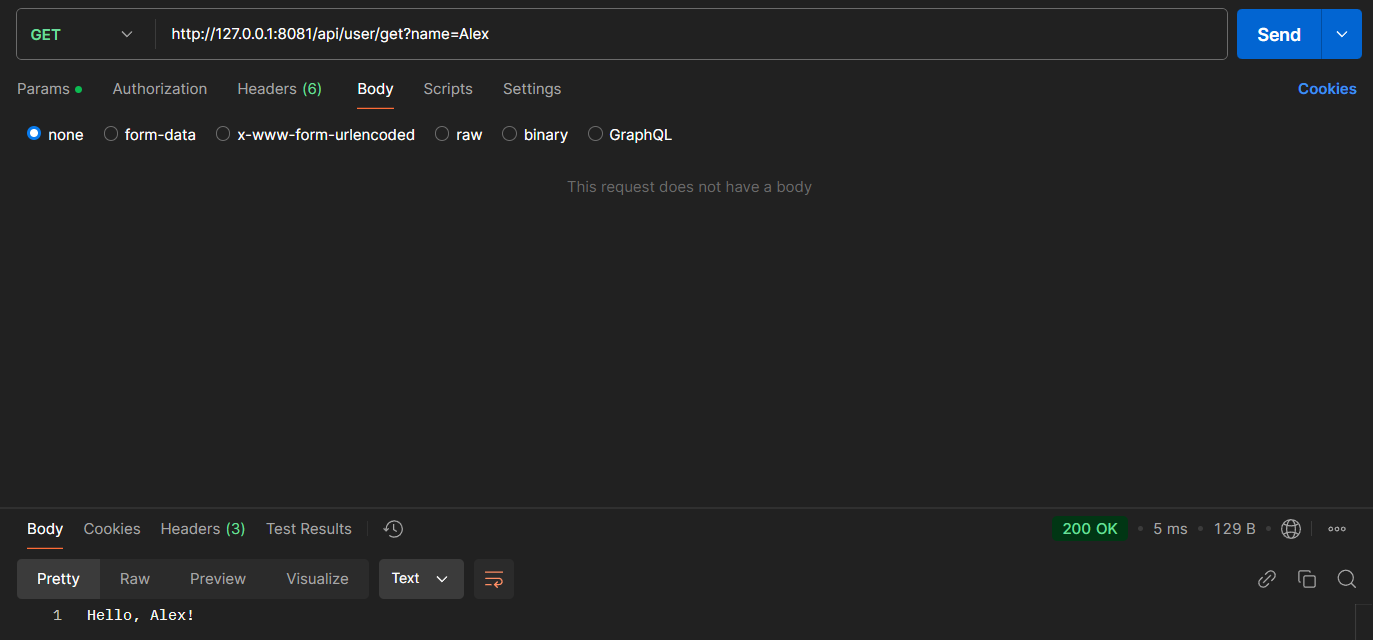


Рисунок 8 – результат get запроса

Вывод: в ходе выполнения лабораторной работы были получены базовые знания и навыки использования веб-фреймворков на языке программирования Go для BackEnd-разработки.