|  |  |
| --- | --- |
| 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 № 8**

**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 в организации долгосрочного хранения данных с использованием PostgreSQL и Golang.

Задача 1 (hello):

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

Сначала создадим бд через инструменты pgAdmin и консоль с помощью следующих строк:

create table hello

(

    message text not null

);

Затем напишем код в vs studio для обработки и отправки запросов.

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

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 "+

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

        host, port, user, password, 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}

    // Регистрируем обработчики

    http.HandleFunc("/get", h.GetHello)

    http.HandleFunc("/post", h.PostHello)

    // Запускаем веб-сервер на указанном адресе

    fmt.Println("Starting server on: 127.0.0.1:8081")

    err = http.ListenAndServe(\*address, nil)

    if err != nil {

        log.Fatal(err)

    }

}

Запуск программы:

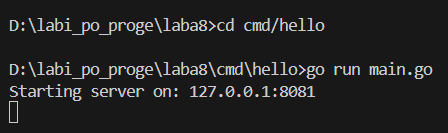


Рисунок 1 – запуск сервера

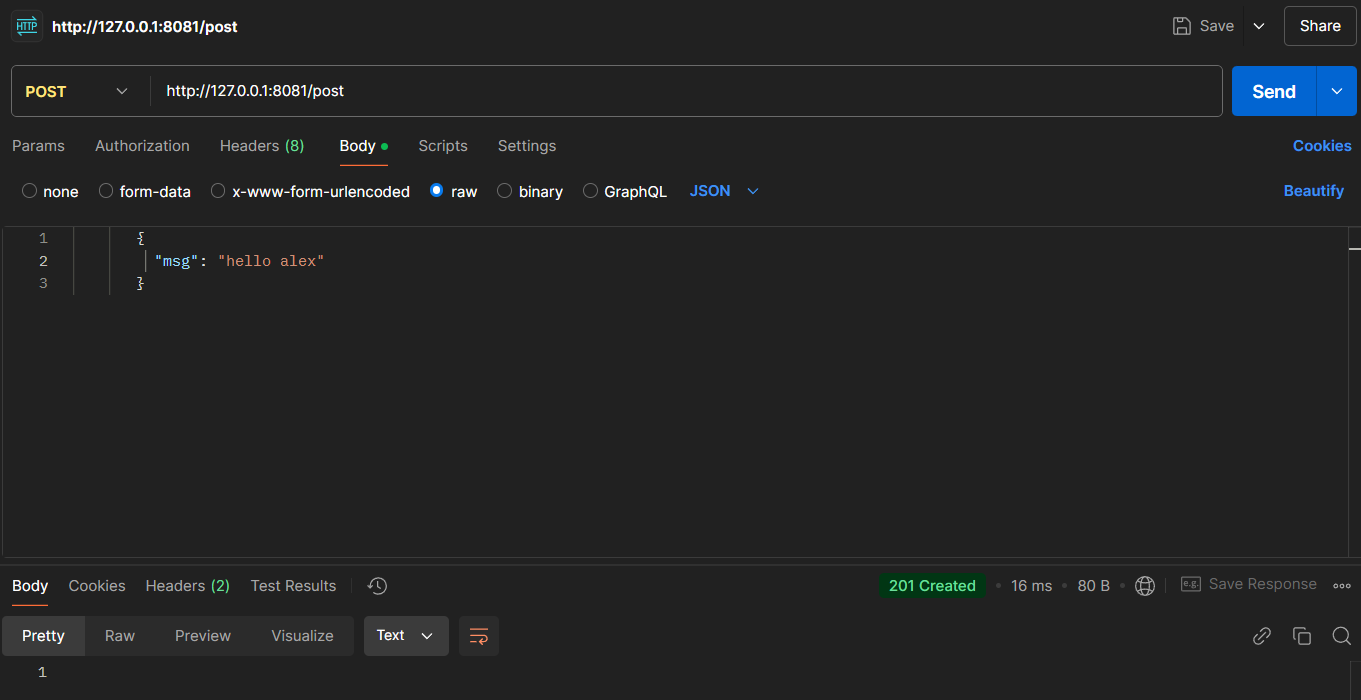


Рисунок 2 – отправка сообщений через postman

Результат отправки сообщений в БД:

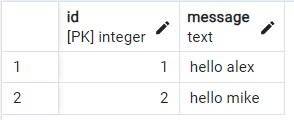


Рисунок 3 – результат работы программы

Задача 2 (count):

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

Создадим таблицу с помощью этого кода:

CREATE TABLE laba8\_count (

number INTEGER

);

Далее добавим начальное значение для number:

INSERT INTO counts (number) VALUES (0);

Затем напишем код в vs studio для обработки и отправки запросов.

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

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 "+

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

        host, port, user, password, 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}

    // Регистрируем обработчики

    http.HandleFunc("/count", h.Count)

    // Запускаем веб-сервер на указанном адресе

    err = http.ListenAndServe(\*address, nil)

    if err != nil {

        log.Fatal(err)

    }

}

Получим следующий результат:

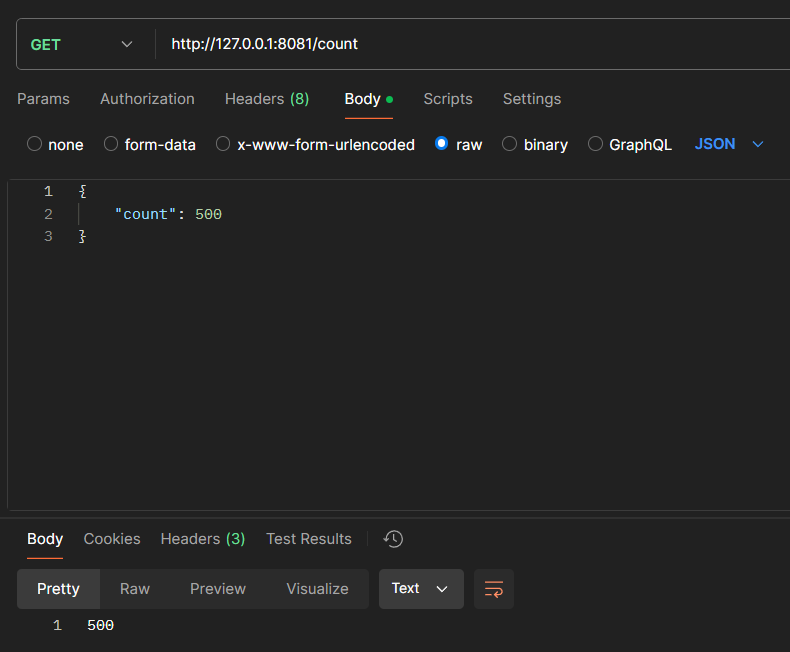


Рисунок 4 - Начальное значение в бд при запросе GET (Оно не равно 0, так я сам уже проводил тесты)

Теперь увеличим его с помощью команды POST:

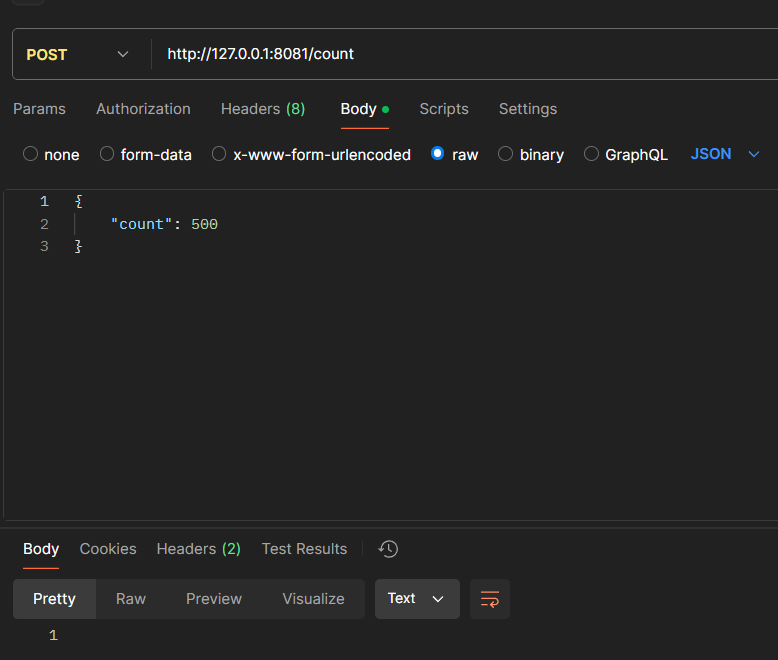


Рисунок 5 – изменение значения

И снова посмотрим на значение с помощью запроса GET:

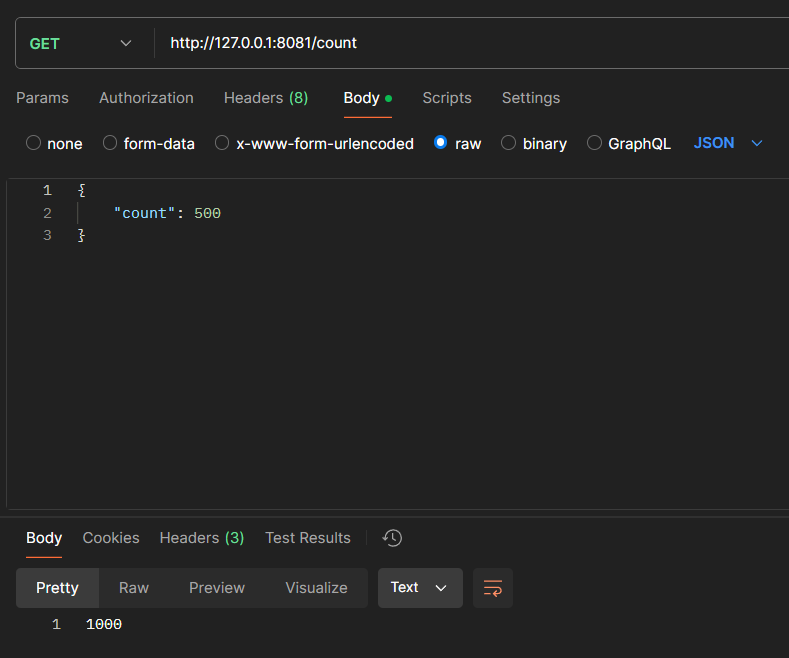


Рисунок 6 – результат работы программы

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 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 "+

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

        host, port, user, password, 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}

    // Регистрируем обработчики

    http.HandleFunc("/api/user/get", h.GreetGet)

    http.HandleFunc("/api/user/post", h.GreetPost)

    // Запускаем веб-сервер на указанном адресе

    err = http.ListenAndServe(\*address, nil)

    if err != nil {

        log.Fatal(err)

    }

}

При вводе post запроса получим:

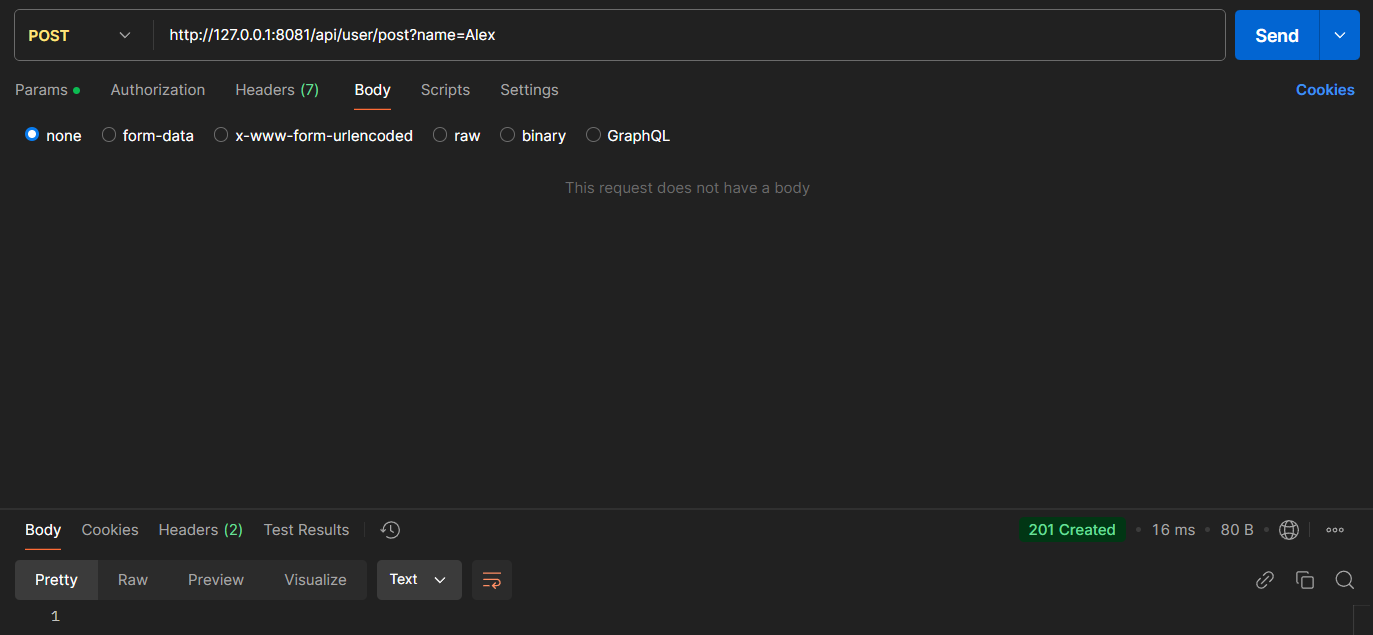


Рисунок 7 – результат post запроса

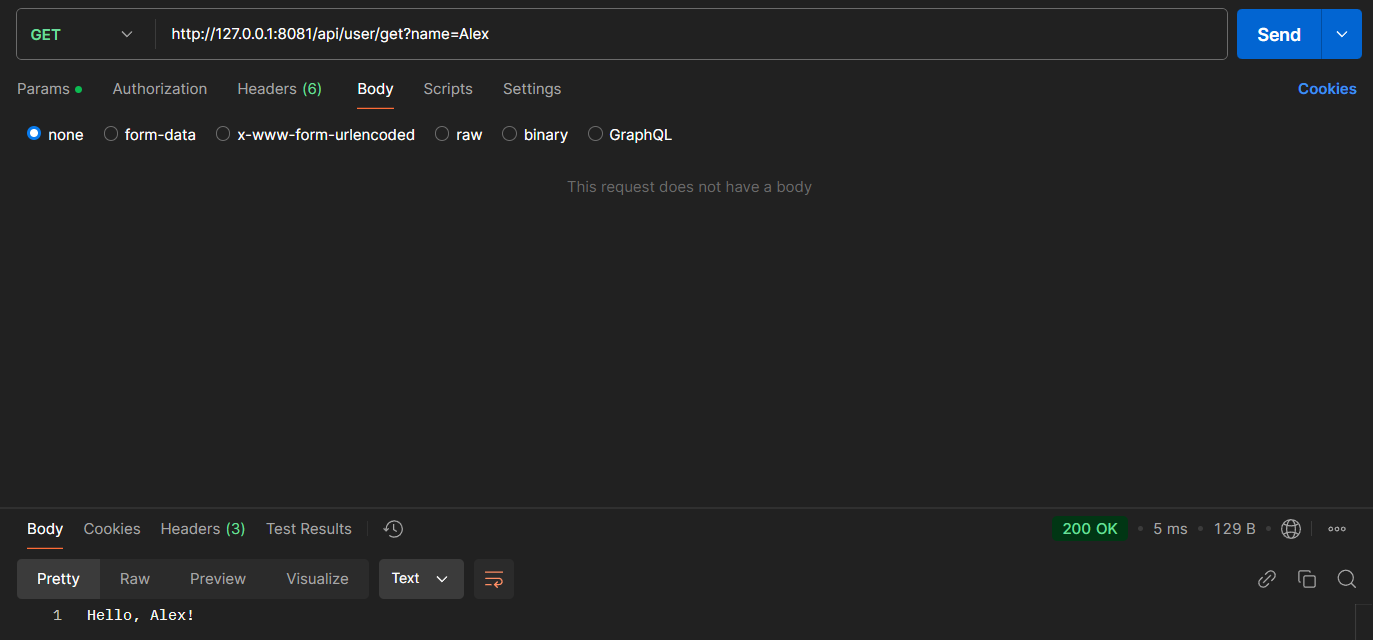


Рисунок 8 – результат get запроса

Вывод: в ходе лабораторной работы был достигнут основной итог: формирование первичных навыков в организации долгосрочного хранения данных с использованием PostgreSQL и языка программирования Golang.