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 **Информатика и вычислительная техника**

О Т Ч Е Т

по лабораторной работе № 9

Название: Back-End разработка с использованием фреймворка Echo

Дисциплина: Языки Интернет Программирования

Студент	ИУ6-32Б		А.С.Авдеева
	(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 - оработать сервисы таким образом, чтобы роутинг, обработка запросов, парсинг json, обработка ошибок и логирование осуществлялись на базе фреймворка Echo

1)Доработала сервис Hello

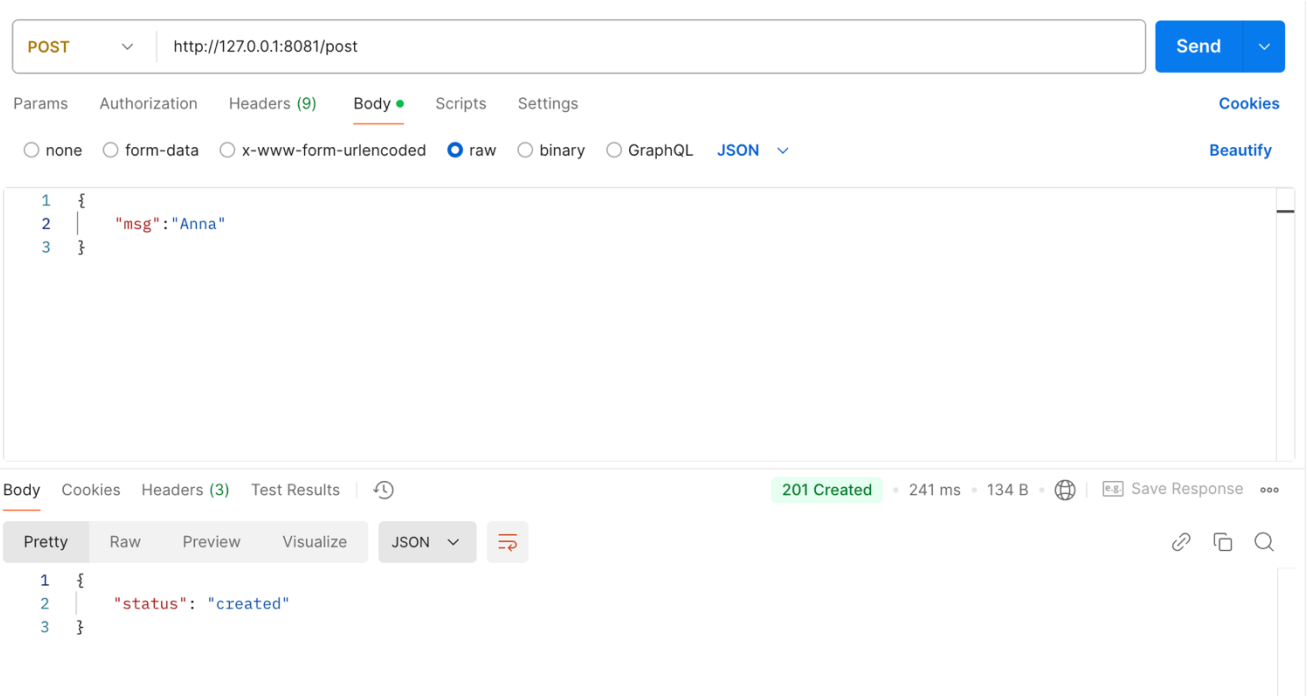
Код к приложению:

```
package main

import (
    "database/sql"
    _ "encoding/json"
    _ "flag"
    "fmt"
    "log"
    "net/http"

    "github.com/labstack/echo/v4"
    _ "github.com/lib/pq"
)

const (
    host     = "localhost"
    port     = 5432
    user     = "postgres"
    password = "123"
    dbname   = "sandbox"
)

type Handlers struct {
    dbProvider DatabaseProvider
}

type DatabaseProvider struct {
    db *sql.DB
}

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": "Hello" + " " + msg
+ "!"})
}

func (h *Handlers) PostHello(c echo.Context) error {
    input := struct {
       Msg string `json:"msg"`
    }{}
    if err := c.Bind(&input); err != nil {
       return c.JSON(http.StatusBadRequest, map[string]string{"error":
err.Error()})
```

```
    }
    if err := h.dbProvider.InsertHello(input.Msg); err != nil {
        return c.JSON(http.StatusInternalServerError, map[string]string{"error":
err.Error()})
    }
    return c.JSON(http.StatusCreated, map[string]string{"status": "created"})
}

func (dp *DatabaseProvider) SelectHello() (string, error) {
    var msg string

    row := dp.db.QueryRow("SELECT message 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 (message) VALUES ($1)", msg)
    if err != nil {
        return err
    }
    return nil
}

func main() {
    psqlInfo := fmt.Sprintf("host=%s port=%d user=%s "+
        "password=%s dbname=%s sslmode=disable",
        host, port, user, password, dbname)

    db, err := sql.Open("postgres", psqlInfo)
    if err != nil {
        log.Fatal(err)
    }
    defer db.Close()

    dp := DatabaseProvider{db: db}
    h := Handlers{dbProvider: dp}

    e := echo.New()

    e.GET("/get", h.GetHello)
    e.POST("/post", h.PostHello)

    address := "127.0.0.1:8081"
    if err := e.Start(address); err != nil {
        log.Fatal(err)
    }
}
```

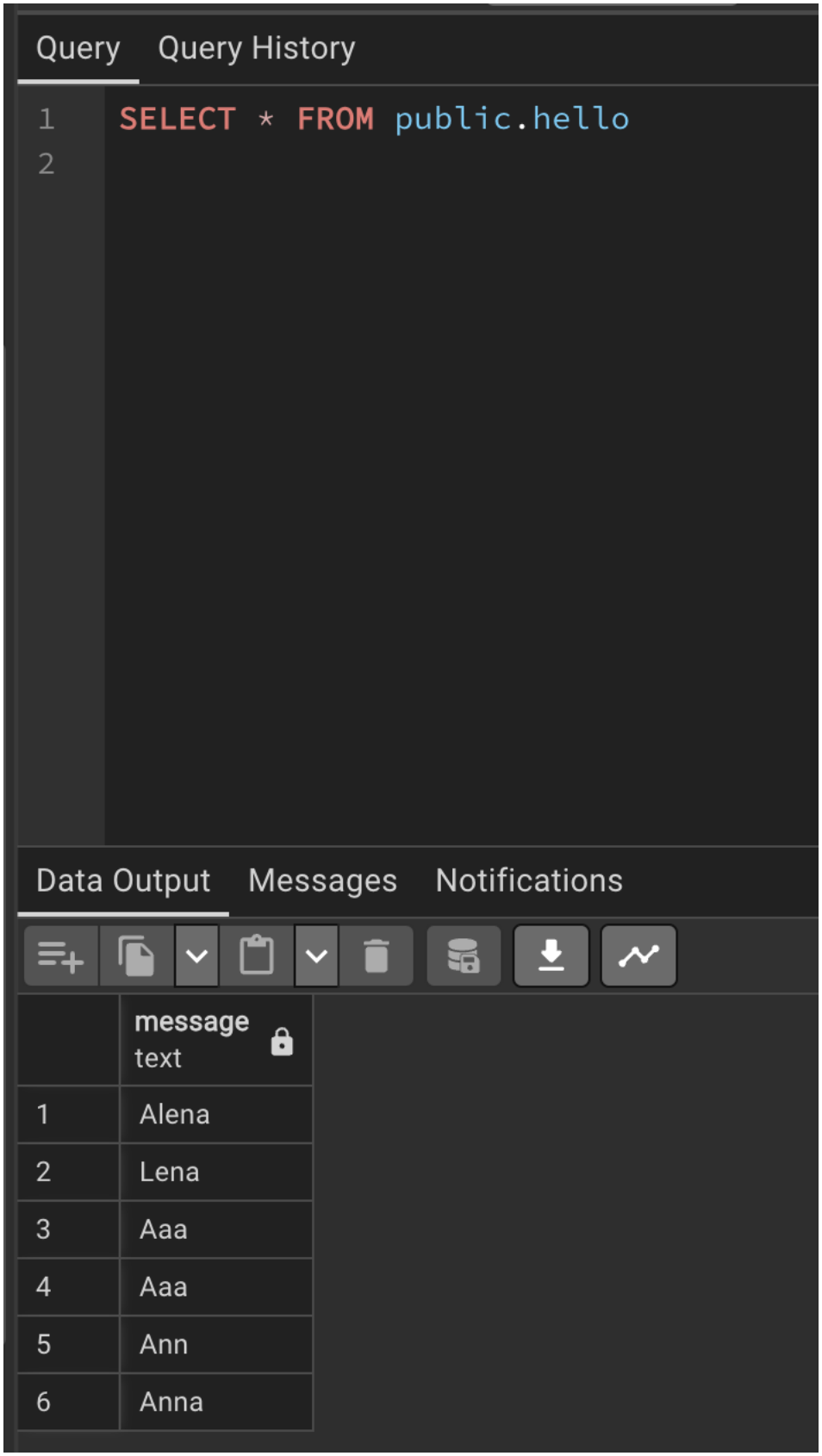
Как работает:

1)Добавим в базу данных имя Anna(рис.1):



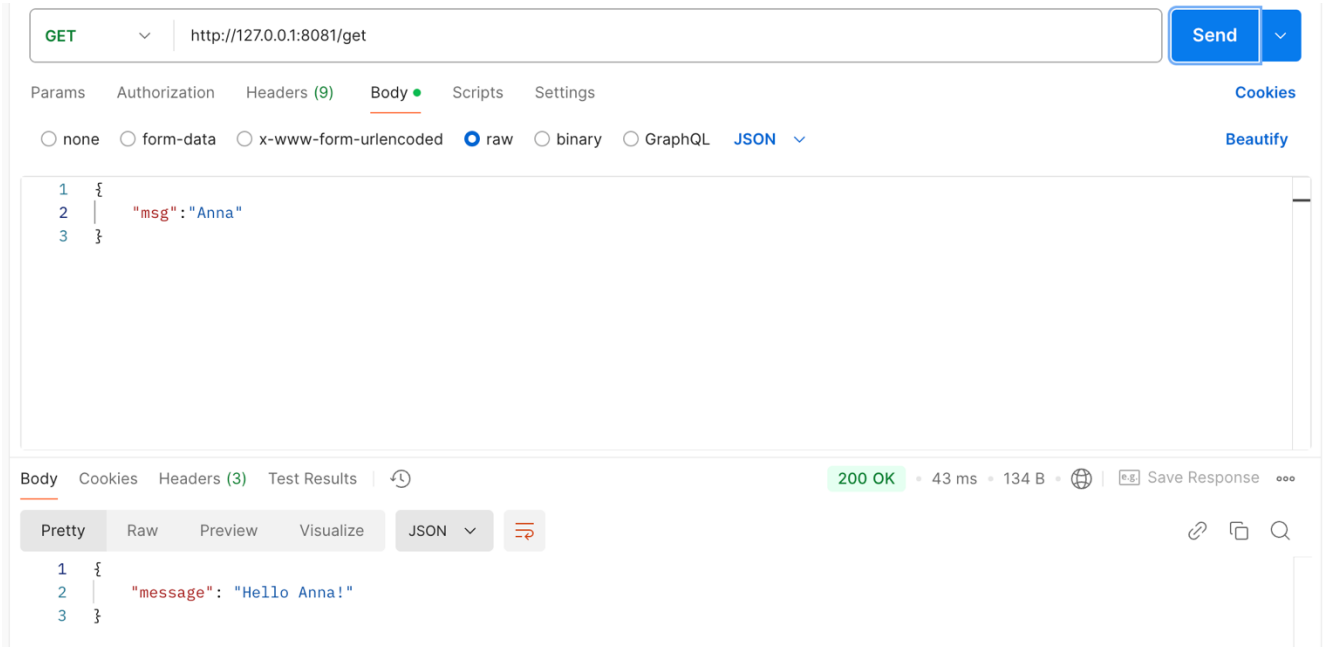
(рис.1)

2)Смотрим что имя Anna добавилось в нашу таблицу через pgAdmin4(рис.2):



(рис.2)-Имя Anna добавлено

3)Выведем имя Anna, чтобы он поздоровался с ней(рис.3):



(рис.3)

2)Задание на Query – параметр

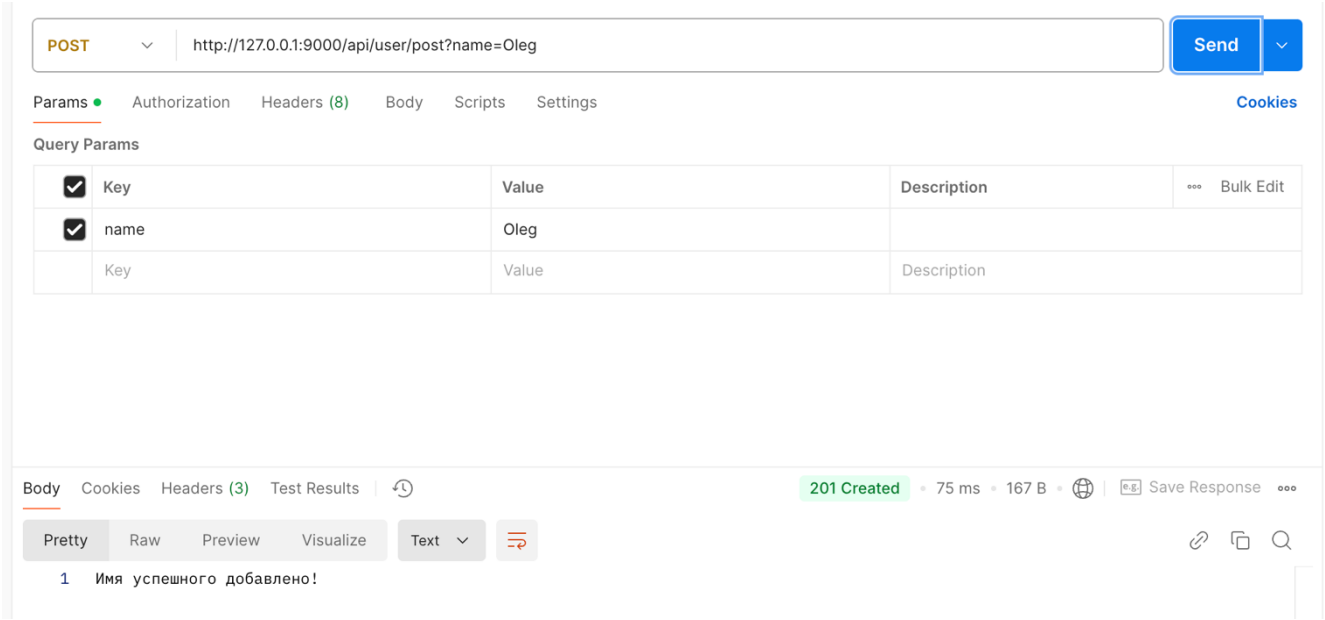
Код к приложению:

```
package main

import (
    "database/sql"
    "fmt"
    "log"
    "net/http"

    "github.com/labstack/echo/v4"
    _ "github.com/lib/pq"
)

const (
    host     = "localhost"
    port     = 5432
    user     = "postgres"
    password = "123"
    dbname   = "db1_query"
)

type Handlers struct {
    dbProvider DatabaseProvider
}

type DatabaseProvider struct {
    db *sql.DB
}

func (h *Handlers) GetQuery(c echo.Context) error {
    name := c.QueryParam("name")
    if name == "" {
        return c.String(http.StatusBadRequest, "Введите с кем здороваться!")
    }
```

```
    name, err := h.dbProvider.SelectQuery(name)
    if err != nil {
       if err == sql.ErrNoRows {
          return c.String(http.StatusInternalServerError, "Такого человека нет в
базе данных")
       }
       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, "Введите имя для добавления!")
    }
    err := h.dbProvider.InsertQuery(name)
    if err != nil {
       return c.String(http.StatusInternalServerError, err.Error())
    }
    return c.String(http.StatusCreated, "Имя успешного добавлено!")
}

func (dp *DatabaseProvider) SelectQuery(name string) (string, error) {
    var nam string
    row := dp.db.QueryRow("SELECT name FROM table1 WHERE name = ($1)", name)
    err := row.Scan(&nam)
    if err != nil {
       return "", err
    }
    return nam, nil
}

func (dp *DatabaseProvider) InsertQuery(name string) error {
    _, err := dp.db.Exec("INSERT INTO table1 (name) VALUES ($1)", name)

    if err != nil {
       return err
    }
    return nil
}

func main() {

    psqlInfo := fmt.Sprintf("host=%s port=%d user=%s "+
       "password=%s dbname=%s sslmode=disable",
       host, port, user, password, dbname)

    db, err := sql.Open("postgres", psqlInfo)
    if err != nil {
       log.Fatal(err)
    }
    defer db.Close()

    dp := DatabaseProvider{db: db}
    h := Handlers{dbProvider: dp}

    e := echo.New()
    e.GET("/api/user/get", h.GetQuery)
    e.POST("/api/user/post", h.PostQuery)

    address := "127.0.0.1:9000"
    log.Printf("Сервер запущен на %s", address)
    if err := e.Start(address); err != nil {
       log.Fatal(err)
```

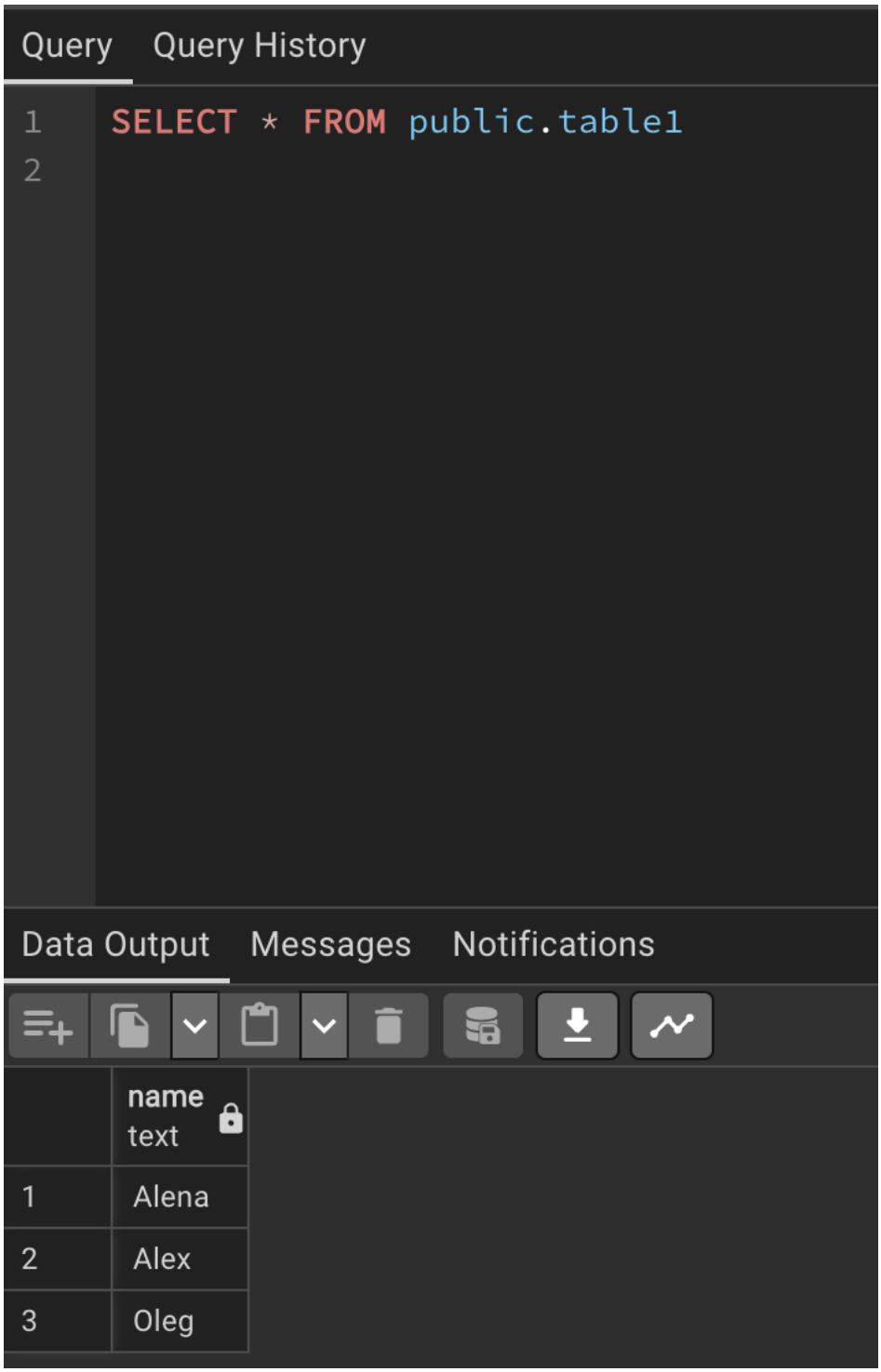
```
    }
}
```

1)Добавим в базу данных имя Oleg(рис.1):



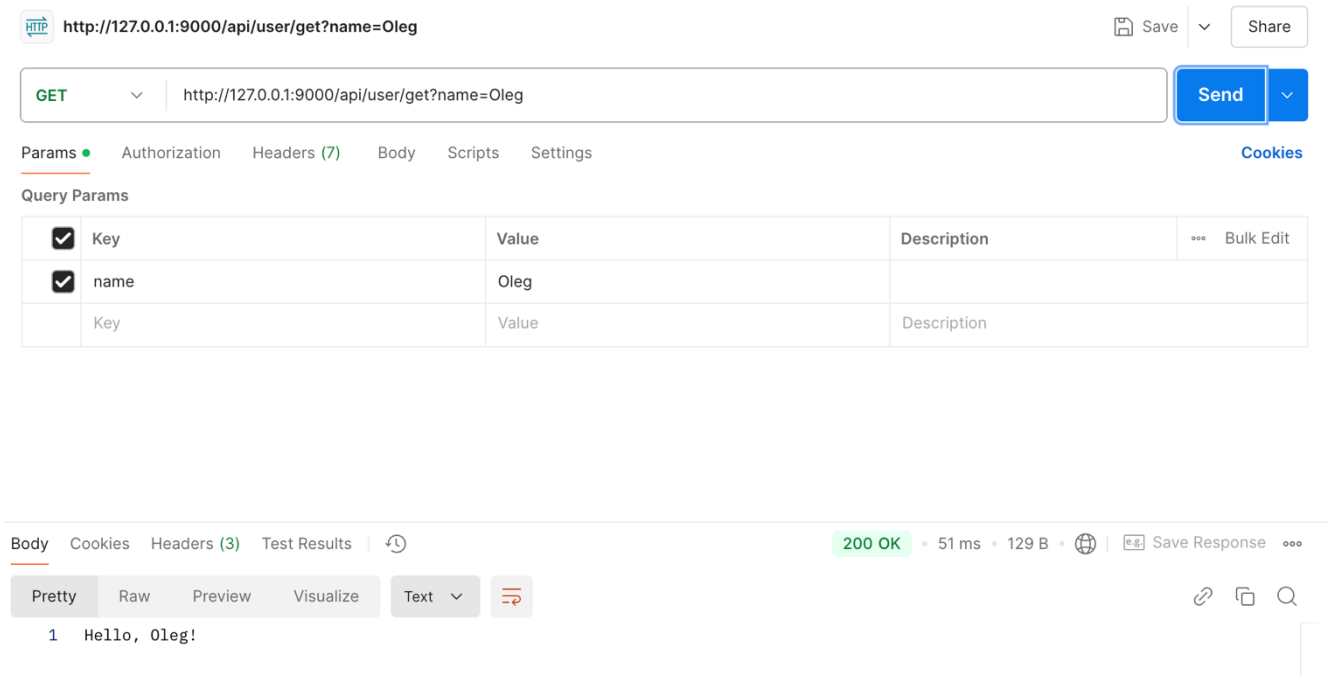
(рис.1)

2)Проверим, что имя сохранилось в базу данных(рис.2)



(рис.2)-Имя Oleg добавлено

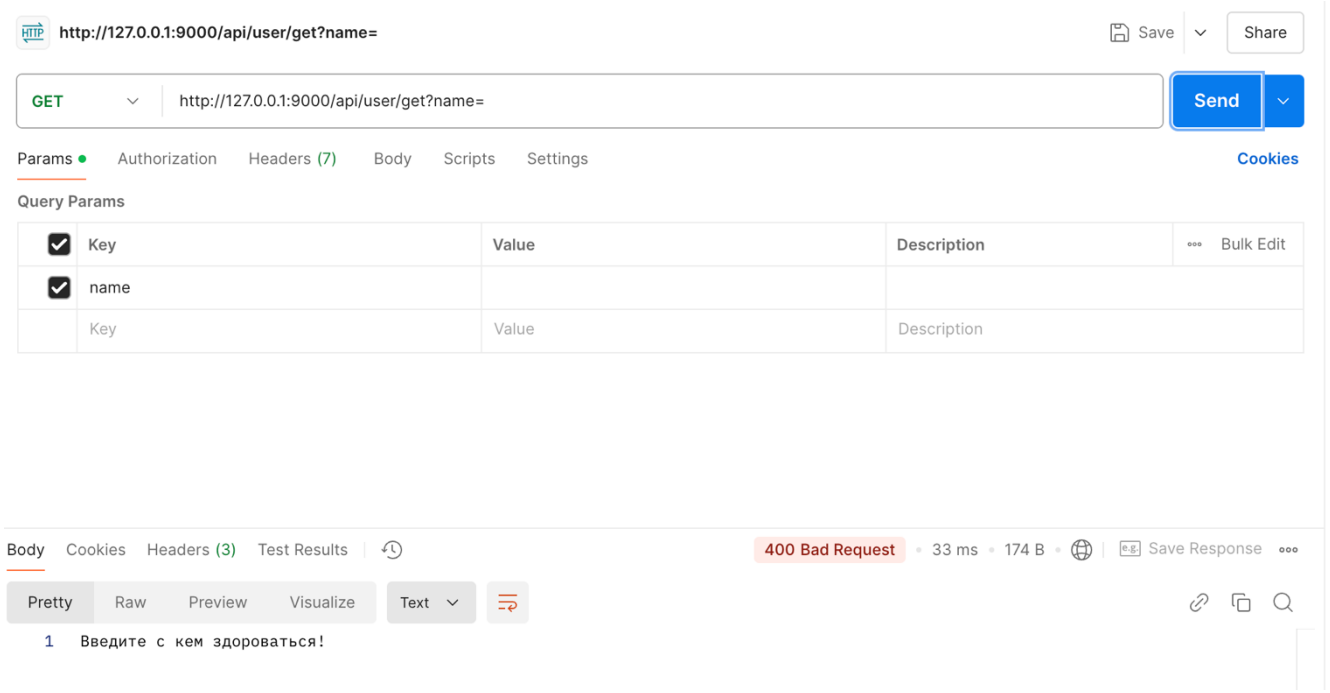
3)Выведем приветствие с Олегом(рис.3)



(рис.3)

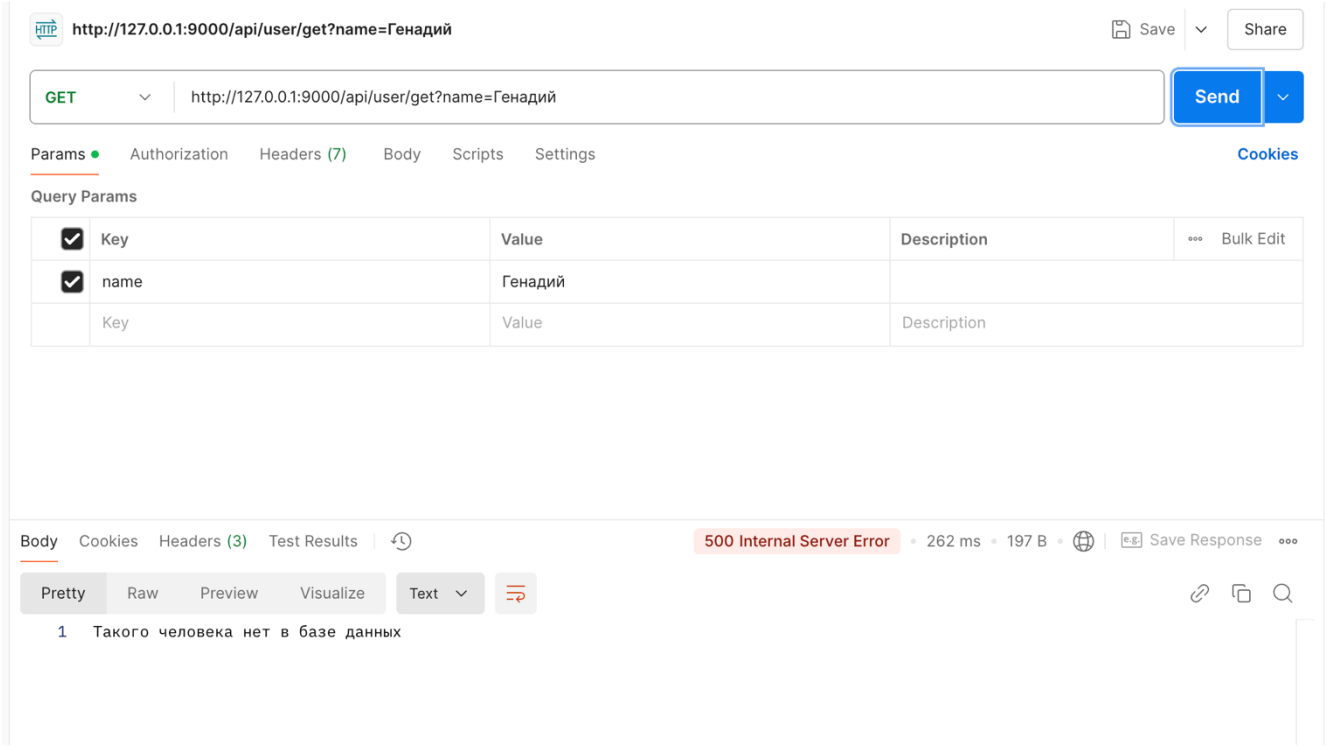
Проверим работу ошибок, прописанных в коде:

1)На ввод пустой строки(рис.4):



(рис.4)

2)На ввод человека, которого нет в базе данных(рис.5):



(рис. 5)

2)Задание на count:

Код к приложению:

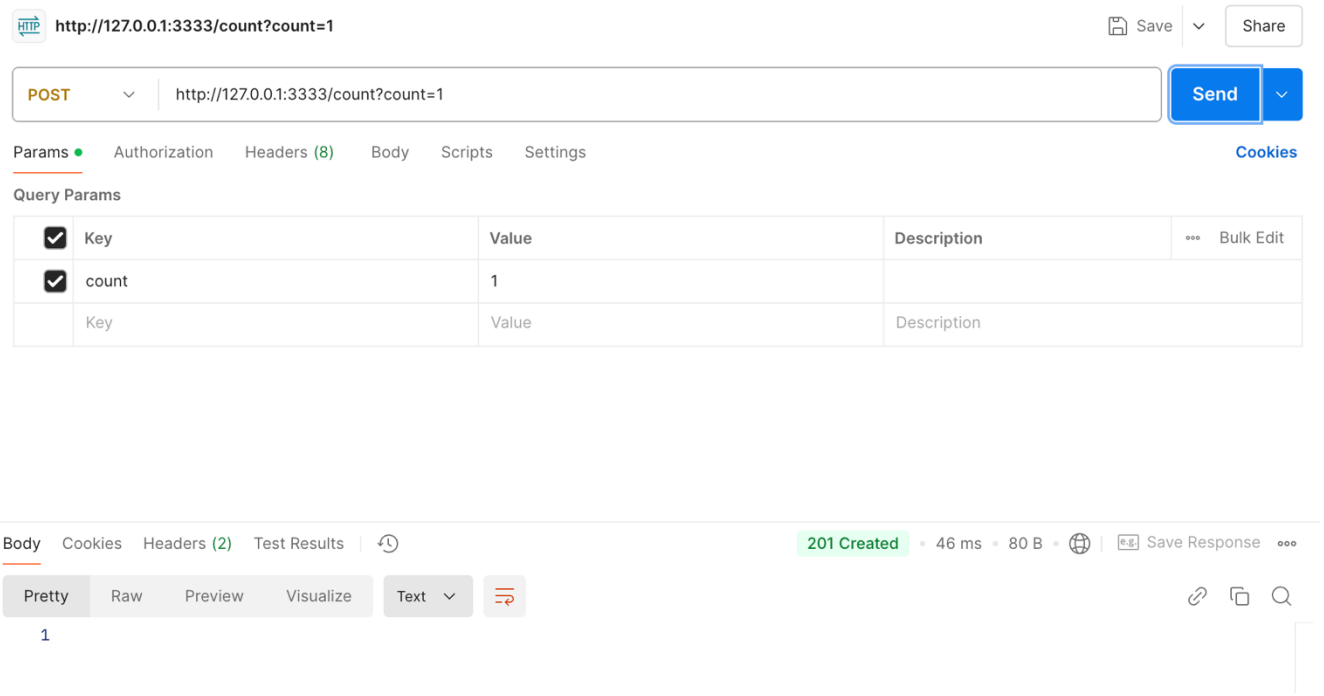
```
package main

import (
    "database/sql"
    "flag"
    "fmt"
    "log"
    "net/http"
    "strconv"

    "github.com/labstack/echo/v4"
    _ "github.com/lib/pq"
)

const (
    host     = "localhost"
    port     = 5432
    user     = "postgres"
    password = "123"
    dbname   = "db2_count"
)

type Handlers struct {
    dbProvider DatabaseProvider
}
```

```
type DatabaseProvider struct {
    db *sql.DB
}

func (h *Handlers) GetCount(c echo.Context) error {
    count, err := h.dbProvider.SelectQuery()
    if err != nil {
       return c.JSON(http.StatusInternalServerError, map[string]string{"error":
err.Error()})
    }
    return c.String(http.StatusOK, strconv.Itoa(count))
}

func (h *Handlers) PostCount(c echo.Context) error {
    countStr := c.FormValue("count")
    count, err := strconv.Atoi(countStr)
    if err != nil {
       return c.JSON(http.StatusBadRequest, map[string]string{"error": "Это не
число!"})
    }
    err = h.dbProvider.InsertQuery(count)
    if err != nil {
       return c.JSON(http.StatusInternalServerError, map[string]string{"error":
err.Error()})
    }
    return c.NoContent(http.StatusCreated)
}

func (dp *DatabaseProvider) SelectQuery() (int, error) {
    var count int
    row := dp.db.QueryRow("SELECT count FROM table4")
    err := row.Scan(&count)
    if err != nil {
       return 0, err
    }
    return count, nil
}

func (dp *DatabaseProvider) InsertQuery(count int) error {
    _, err := dp.db.Exec("UPDATE table4 SET count = count + $1", count) // Changed
table and column name, using ON CONFLICT

    if err != nil {
       return err
    }
    return nil
}

func main() {
    address := flag.String("address", "127.0.0.1:3333", "адрес для запуска
сервера")
    flag.Parse()

    psqlInfo := fmt.Sprintf("host=%s port=%d user=%s "+
       "password=%s dbname=%s sslmode=disable",
       host, port, user, password, dbname)

    db, err := sql.Open("postgres", psqlInfo)
    if err != nil {
       log.Fatal(err)
    }
    defer db.Close()

    dp := DatabaseProvider{db: db}
    h := Handlers{dbProvider: dp}
```

```
    e := echo.New()

    e.GET("/count", h.GetCount)
    e.POST("/count", h.PostCount)

    log.Printf("Сервер запущен на %s", *address)
    if err := e.Start(*address); err != nil {
        log.Fatal(err)
    }
}
```

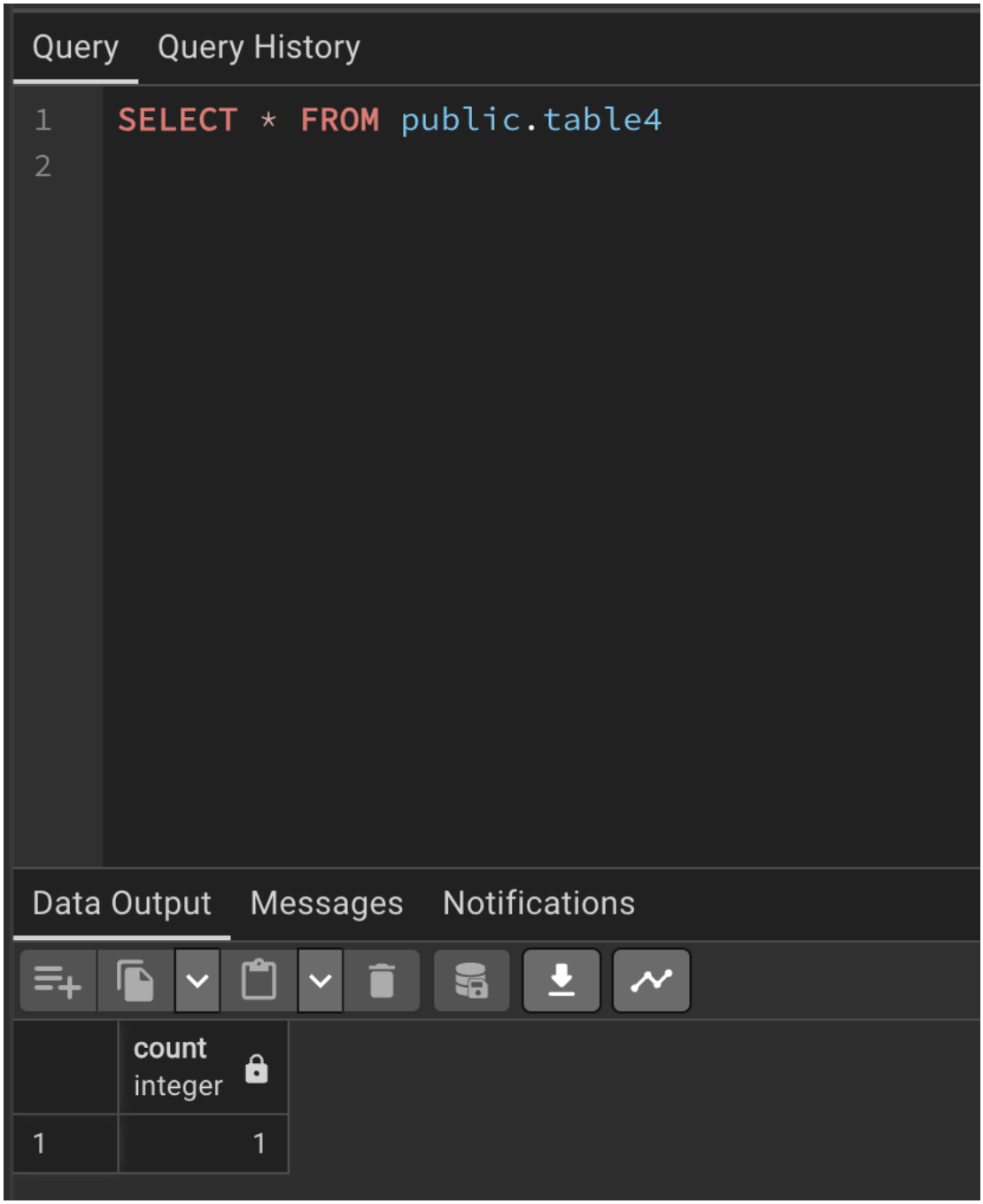
Как работает:

1)Так как у нас HTML форма, забьём значение count = 1 через Postman(рис.1):



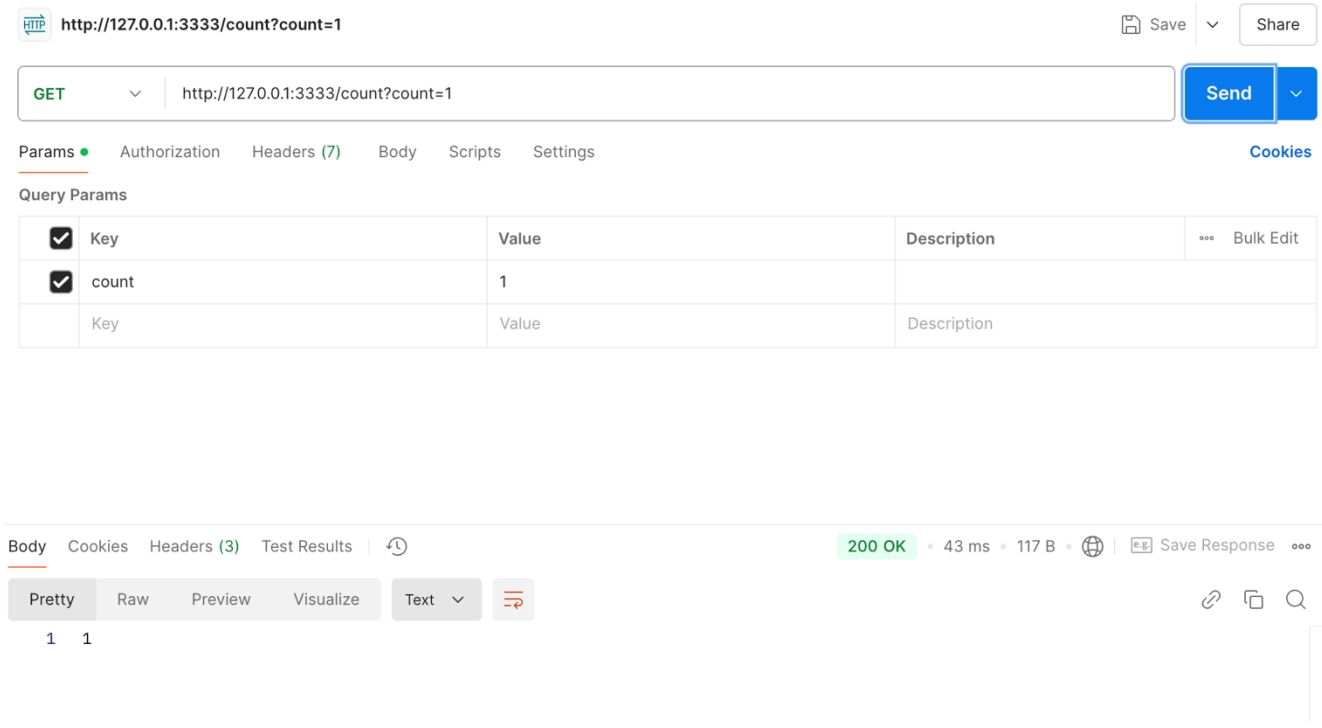
(рис.1)

2)Посмотрим, что данные были добавлены в базу данных через PgAdmin4(рис.2):



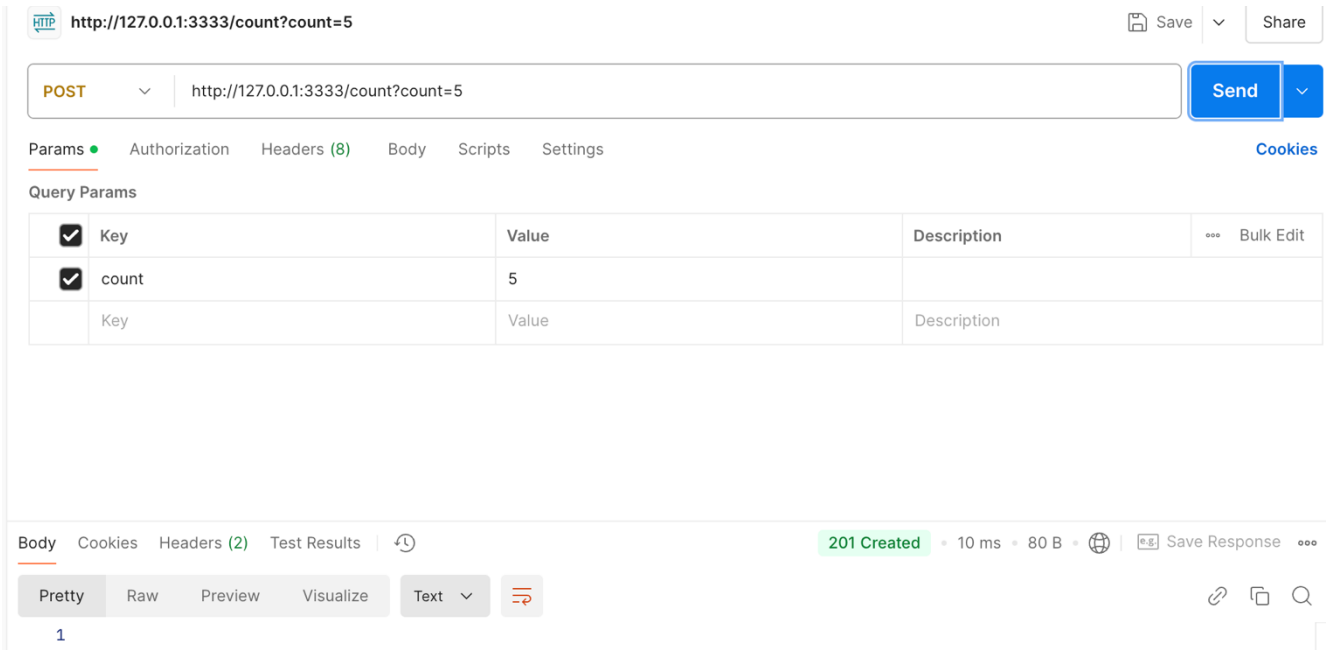
(рис.2)-Данные успешно добавились!

3)Теперь выведем значение count = 1 через ручку get(рис.3):



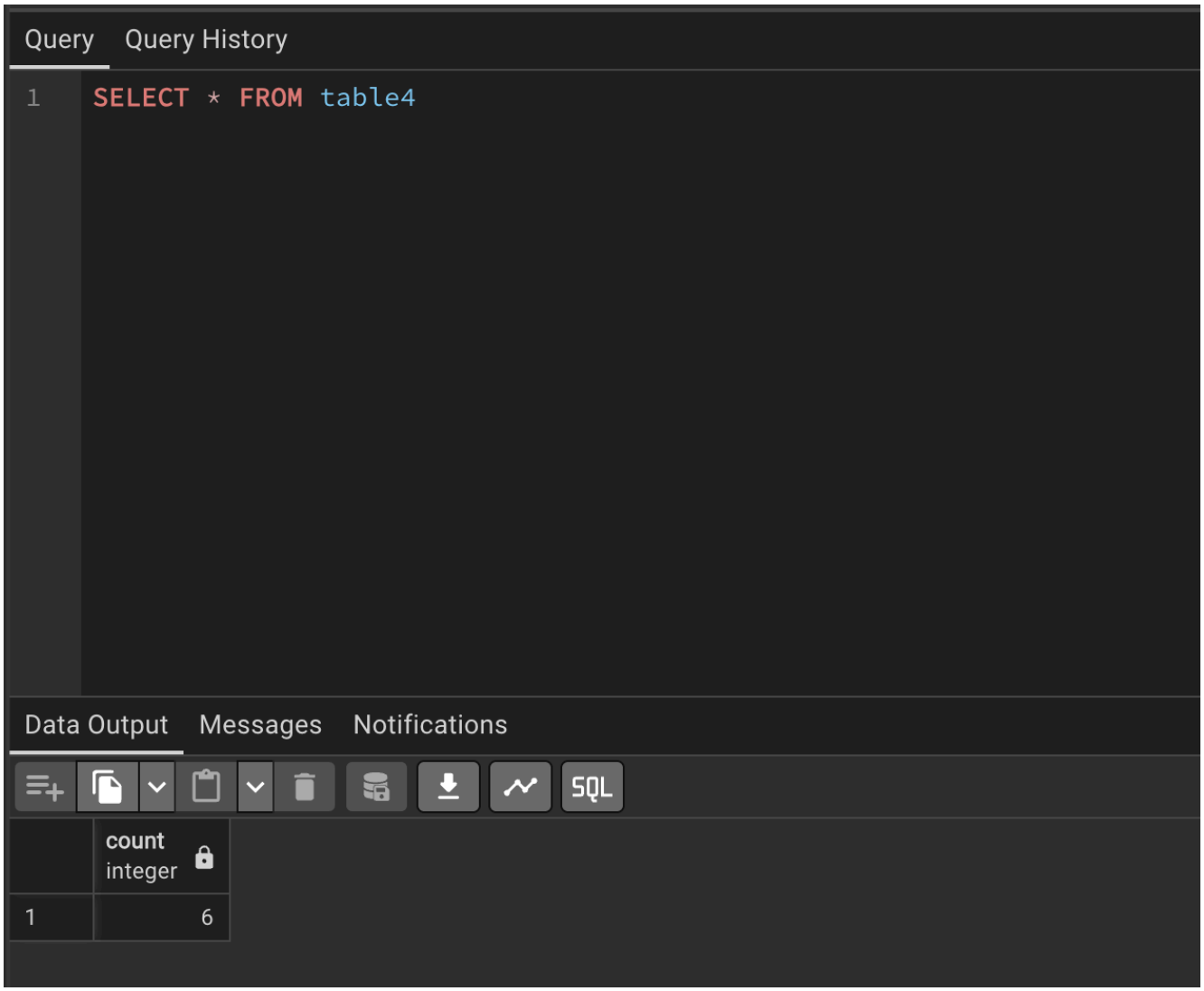
(рис.3)

4)Добавим ещё значение 5 через ручку POST(рис.4):



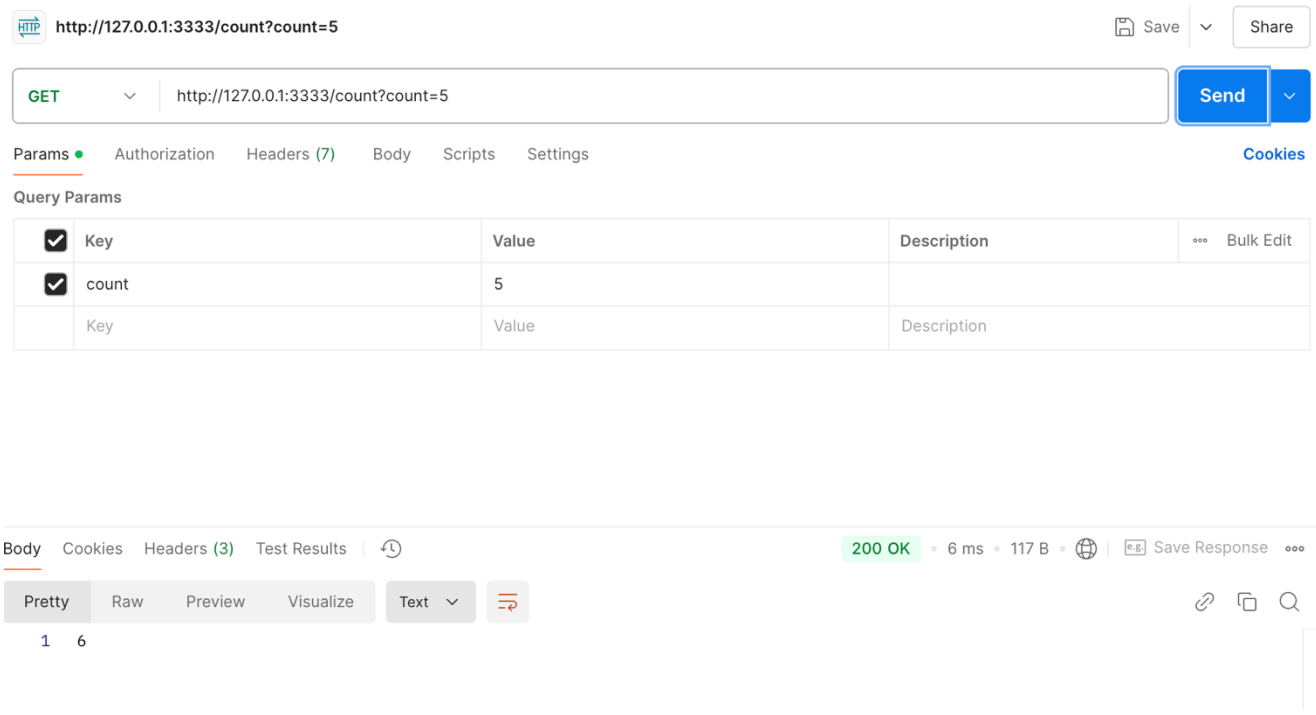
(рис.4)

5)Посмотрим, что сохранилось в базу данных(рис.5):



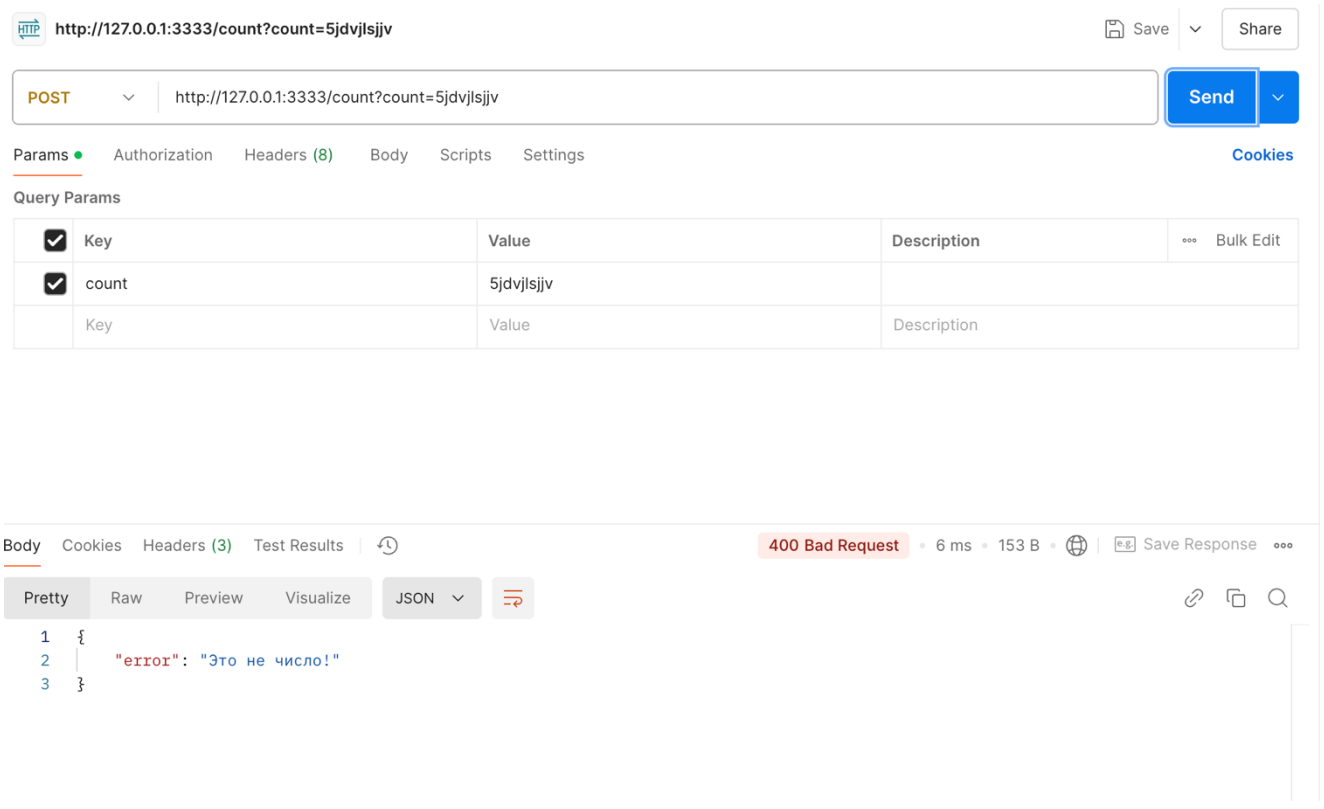
(рис. 5)-Всё верно, должна сохраняться сумма первого добавленного числа и второго

6)Посмотрим, что выведет Postman через ручку Get(рис. 6):



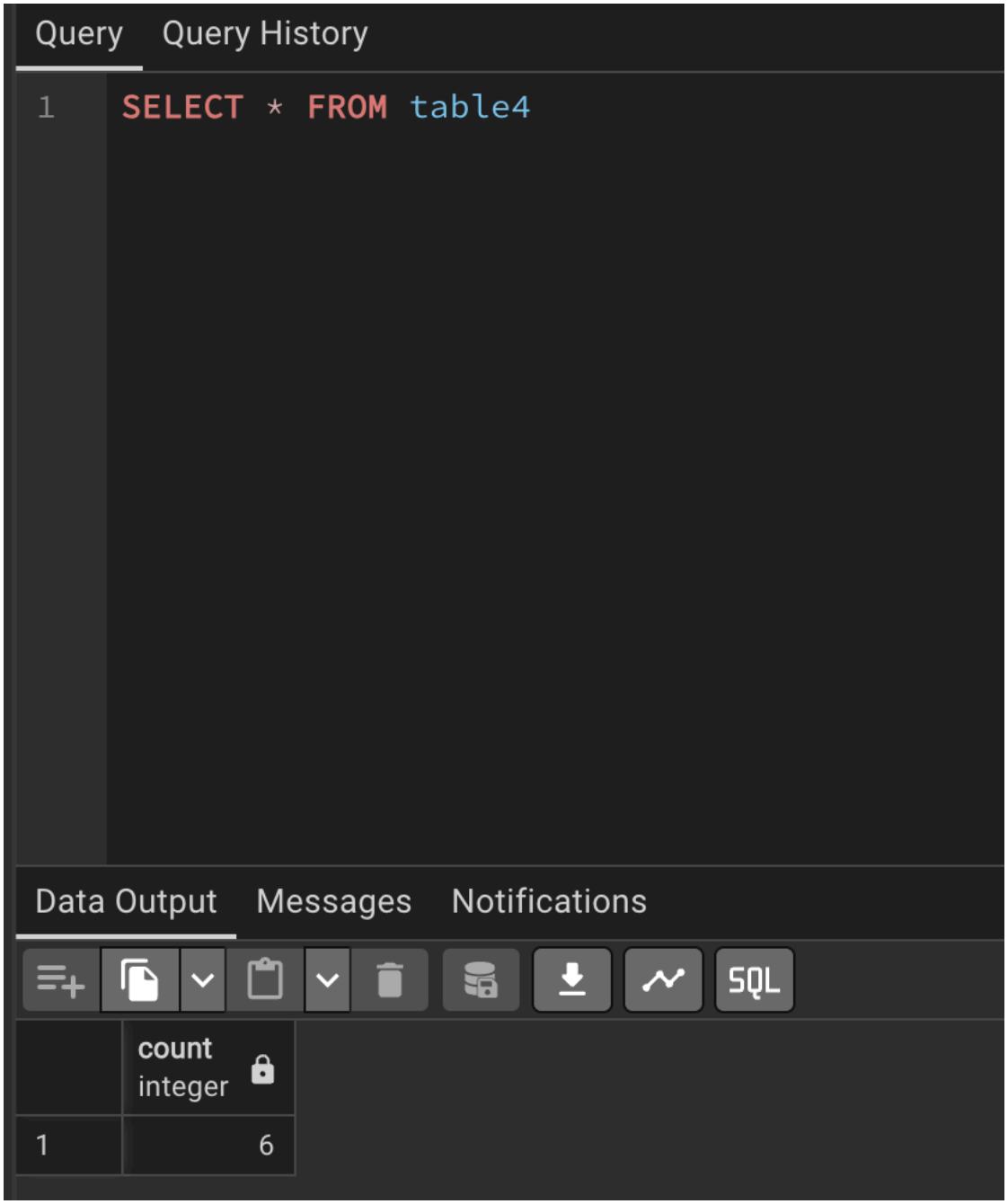
(рис. 6)-Всё верно!

7)Проверим ошибки, прописанные в коде. Проверим, что будет, если введем не число, а строку (рис. 7):



(рис. 7)-Всё верно!

8)Проверим, что эта строка ни в коем случае не сохранилась в базу данных(рис.8):



(рис.8)-Всё верно! Всё работает!

Заключение – проделана успешная работа в понимании работы с echo!