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)

О т ч е т

по лабораторной работе №8

Название лабораторной работы: Организация клиент-серверного взаимодейсвтия между Golang и PostgreSQL

Дисциплина: Языки интернет-программирования

Студент гр. ИУ6-33Б _____ О.С. Кашу
(Подпись, дата) (И.О. Фамилия)

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

Москва, 2024

1. ВВЕДЕНИЕ

1.1 Цель

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

1.2 Задание

Доработать сервисы из 6 лабораторной работы таким образом, чтобы они использовали для хранения данных СУБД *PostgreSQL*. Каждый сервис должен как добавлять новые данные в БД (*insert*/*update*), так и доставать их для предоставления пользователю (*select*).

2. ХОД РАБОТЫ

2.1 Hello

Эта программа теперь POST-запросом сохраняет поле в таблице базы данных, а GET-запросом получает случайное имя из таблицы (рис. 1, 2)

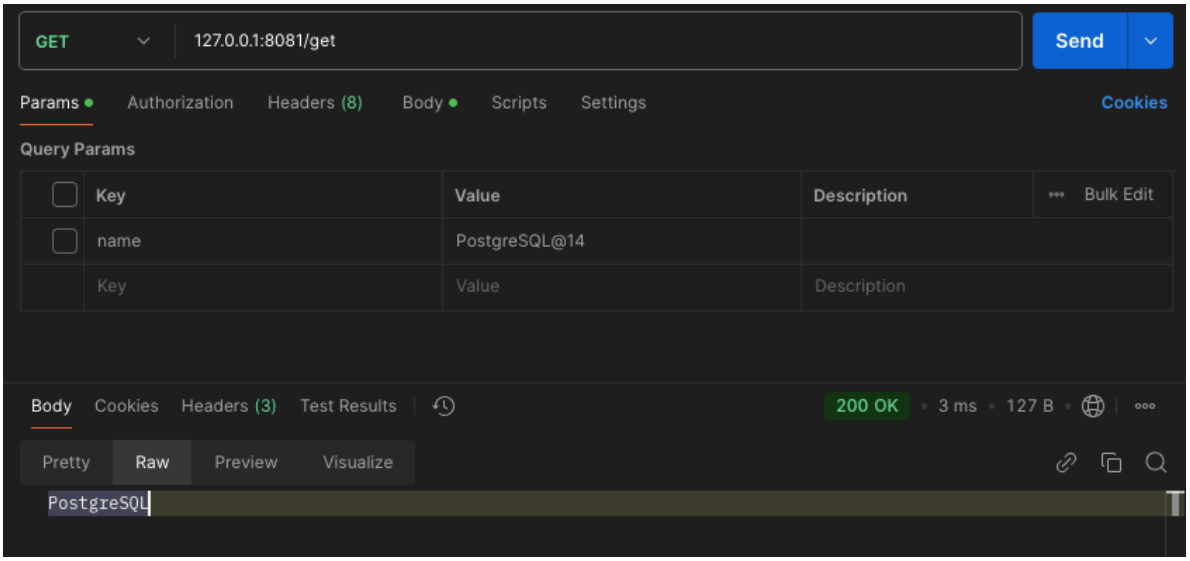

Рисунок 1. Пример GET-запроса

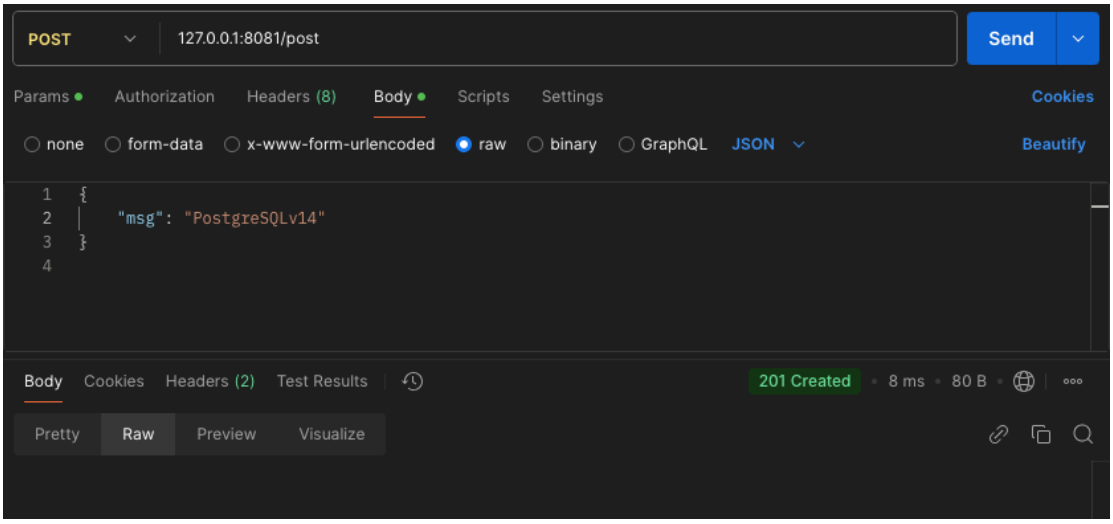

Рисунок 2. Пример POST-запроса

Ниже представлена часть листинга

```
// Методы для работы с базой данных
func (dp *DatabaseProvider) SelectHello() (string, error) {
	var msg string

	// Получаем одно сообщение из таблицы hello, отсортированной в случайном порядке
	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
}
```

2.2 Query

В этой программе реализована схожая с предыдущей функция, только теперь если аргумент *name* отсутствует, берется случайное имя из таблицы, а если указан, то выводится “Hello,*{name}*!” и *name* добавляется в таблицу, если оно там отсутствует (рис. 3)

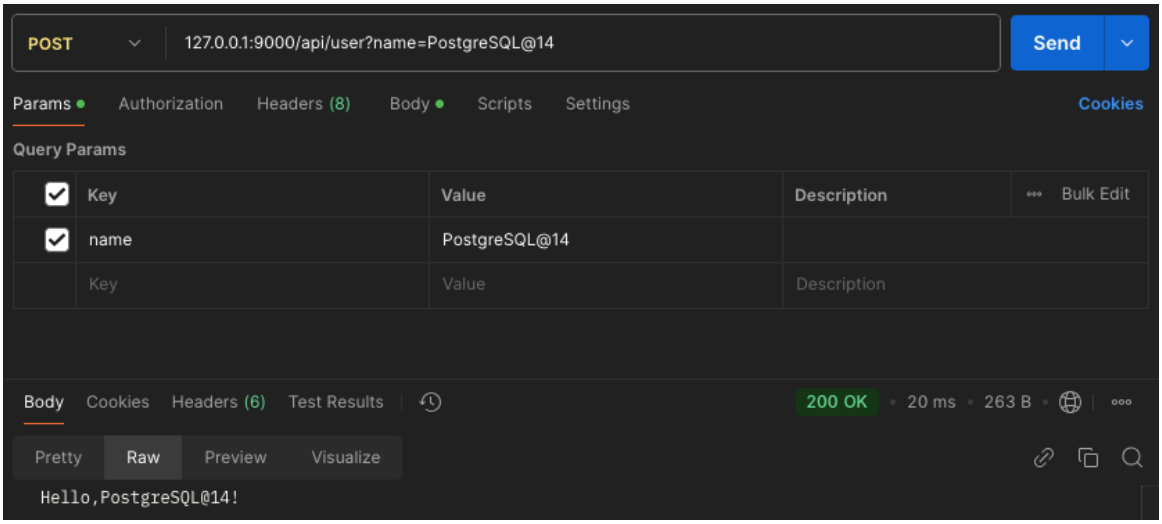

Рисунок 3. Запрос с параметром “name”

2.3 Count

Эта программа при POST-запросе увеличивает число в таблице на параметр count, а при GET-запросе получает его и выводит на экран (рис.)

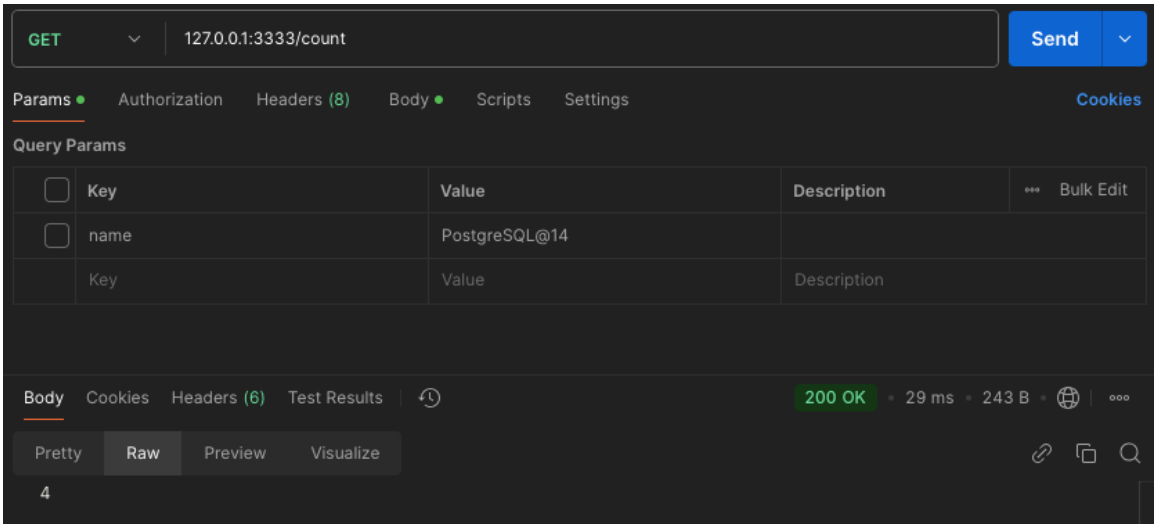

Рисунок 4. Получение счетчика

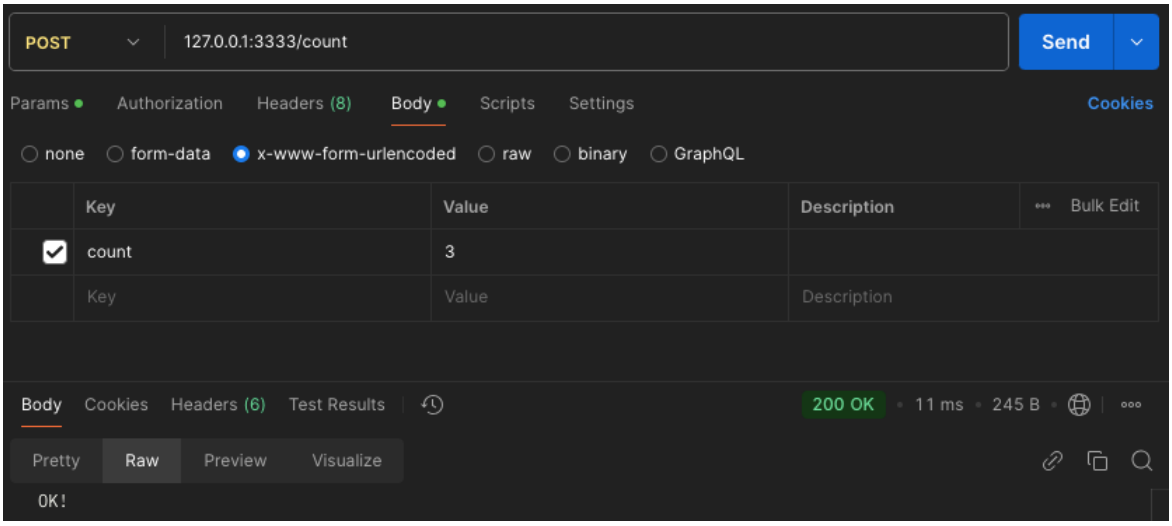

Рисунок 5. Увеличение счетчика

Ниже представлена часть листинга кода

```
func (dp *DatabaseProvider) GetCount() (int, error) {
	var value int

	// Получаем одно сообщение из таблицы hello, отсортированной в случайном порядке
	row := dp.db.QueryRow("SELECT COALESCE(count, 0) FROM count WHERE name=$1", "key1")
	err := row.Scan(&value)
	if err != nil {
		return 0, err
	}

	return value, nil
}

func (dp *DatabaseProvider) AddCount(a int) error {
	_, err := dp.db.Exec("INSERT INTO count (name, count) VALUES ($2, $1) ON CONFLICT (name) DO UPDATE SET count = count.count + $1", a, "key1")
	if err != nil {
		return err
	}

	return nil
}
```

3. ВЫВОД

Изучен набор стандартных библиотек Go для организации клиент-серверного взаимодействия между Golang и PostgreSQL.