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 № 8

Название: Организация клиент-серверного взаимодейсвтия между Golang и PostgreSQL

Дисциплина: Языки интернет-программирования

Студент	ИУ6-31Б	13.12.2024	А.В. Палий
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель		13.12.2024	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.

Ход работы.

1. Сделали форк данного репозитория в GitHub, склонировали получившуюся копию локально, создали от мастера ветку дев и переключились на нее. Скопировали наши микросервисы из шестой лабораторной работы:

2. Установили postgreSQL и создали новую базу данных.

```
saolia@saolia-VirtualBox:~$ cd web-8
saolia@saolia-VirtualBox:~/web-8$ CREATE DATABASE sandbox;
```

Рисунок 1 – Создание базы данных

Дописали наши микросервисы для работы с базой данных (рис. 2-4)

```go
// Методы для работы с базой данных
func (dp *DatabaseProvider) SelectQuery() (int, error) {
    var c int
    row := dp.db.QueryRow("SELECT c FROM counter LIMIT 1")
    err := row.Scan(&c)
    if err != nil {
        return 0, err
    }

    return c, nil
}
func (dp *DatabaseProvider) InsertQuery(c int) error {
    _, err := dp.db.Exec("INSERT INTO counter (c) VALUES ($1)", c)
    if err != nil {
        return err
    }

    return nil
}
func (dp *DatabaseProvider) SetQuery(c int) error {
    _, err := dp.db.Exec("UPDATE counter SET c=$1", c)
    if err != nil {
        return err
    }

    return nil
}

func (dp *DatabaseProvider) ClearQuery() error {
    _, err := dp.db.Exec("DELETE FROM counter")
    if err != nil {
```

Рисунок 2 – Методы для работы с БД

```go
// Методы для работы с базой данных
func (dp *DatabaseProvider) SelectQuery() (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) InsertQuery(msg string) error {
    _, err := dp.db.Exec("INSERT INTO hello (message) VALUES ($1)", msg)
    if err != nil {
        return err
    }

    return nil
}

func main() {
    // Считываем аргументы командной строки
    address := flag.String("address", "127.0.0.1:8081", "адрес для запуска сервиса Hello")
    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)
```

Рисунок 3 – Методы для работы с БД

```go
// Методы для работы с базой данных
func (dp *DatabaseProvider) SelectQuery() (string, int, error) {
    var name string
    var age int

    // Получаем одно сообщение из таблицы query, отсортированной в случайном порядке
    row := dp.db.QueryRow("SELECT name, age FROM query ORDER BY RANDOM() LIMIT 1")
    err := row.Scan(&name, &age)
    if err != nil {
        return "", 0, err
    }

    return name, age, nil
}
func (dp *DatabaseProvider) InsertQuery(name string, age int) error {
    _, err := dp.db.Exec("INSERT INTO query (name, age) VALUES ($1, $2)", name, age)
    if err != nil {
        return err
    }

    return nil
}

func (dp *DatabaseProvider) ClearQuery() error {
    _, err := dp.db.Exec("DELETE FROM query")
    if err != nil {
        return err
    }

    return nil
}
```

Рисунок 4 – Методы для работы с БД

3. Сделали проверку работы всех методов (рис. 5)

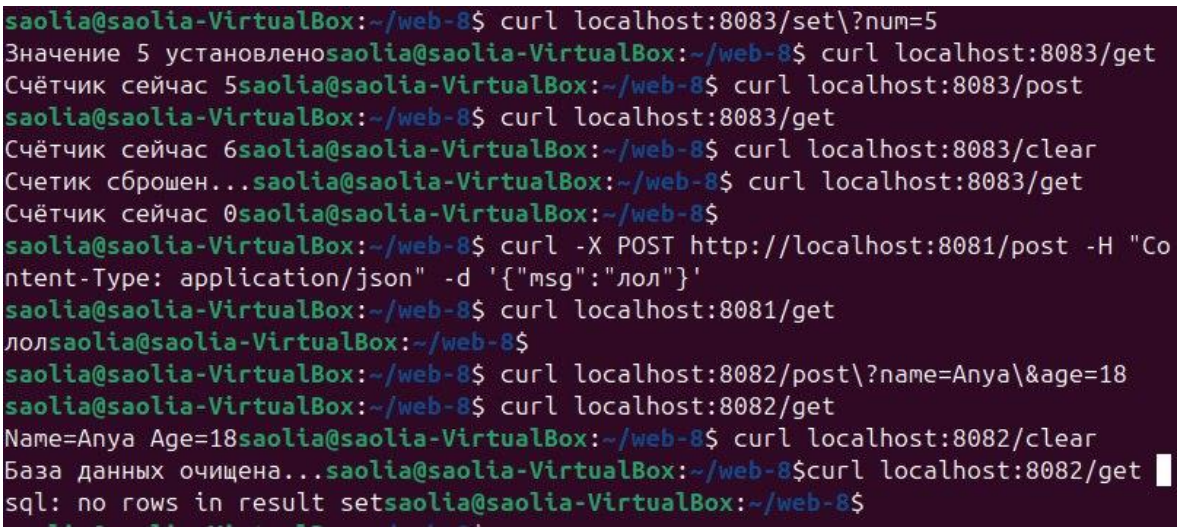

Рисунок 5 - Проверка

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 получили первичные навыки в организации долгосрочного хранения данных с использованием PostgreSQL и Golang.