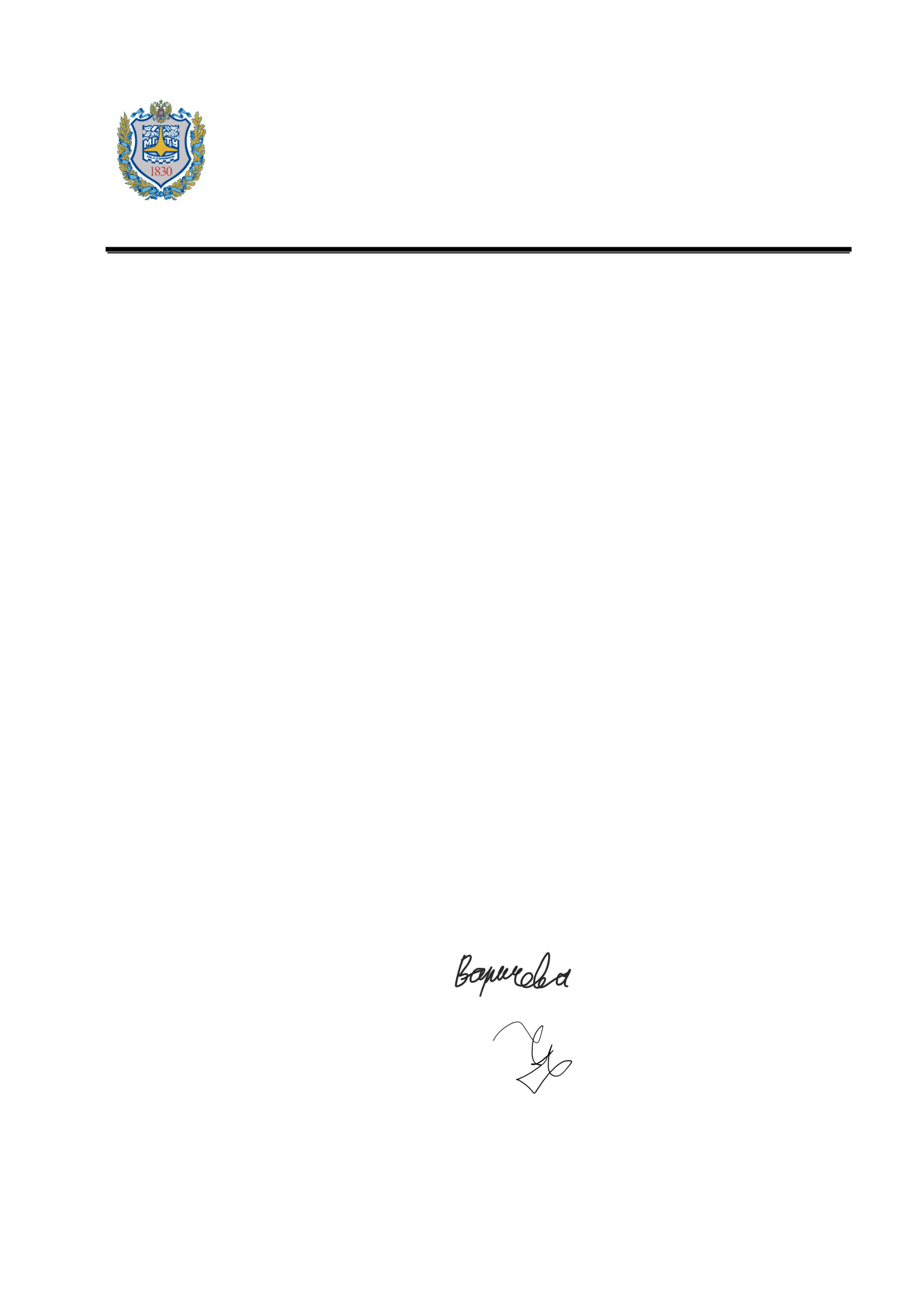
**Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«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 **09.04.01 Информатика и вычислительная техника**

**ОТЧЕТ**

**по лабораторной работе № 8**

**Название:** Организация клиент-серверного взаимодействия между Golang и PostgreSQL

**Дисциплина:** Языки интернет-программирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ6-33Б | 14.12.2024 | |  | Д. А. Варичева |
|  | (Группа) |  | (Подпись, дата) |  | (И.О. Фамилия) |
| Преподаватель |  | 14.12.2024 | |  | В. Д. Шульман |
|  |  |  | (Подпись, дата) |  | (И.О. Фамилия) |

Москва, 2024

**Цель работы:**

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

**Ход работы:**

**Реализация Back-End-части приложения на Golang**

1. **Сервис Count**

**Файл main.go**

package main

import (

"database/sql"

"encoding/json"

"flag"

"fmt"

"log"

"net/http"

"strconv"

\_ "github.com/lib/pq"

)

const (

host = "localhost"

port = 5432

user = "RA"

password = "postgres"

dbname = "sandbox"

)

type DatabaseProvider struct { // Структура с полем, которое хранит ссылку на СУБД db \*sql.DB

}

type Handlers struct {

DProvider DatabaseProvider

}

func (h \*Handlers) GetCount(writer http.ResponseWriter, r \*http.Request) { answer, err := h.DProvider.SelectCount()

if err != nil {

writer.WriteHeader(500)

writer.Write([]byte(err.Error()))

return

}

writer.WriteHeader(200)

writer.Write([]byte("Значение счётчика: " + answer))

}

func (h \*Handlers) SetCount(writer http.ResponseWriter, r \*http.Request) { input := struct {

Massage string `json:massage`

}{}

decoder := json.NewDecoder(r.Body)

err := decoder.Decode(&input)

if err != nil {

writer.WriteHeader(400)

writer.Write([]byte(err.Error()))

return

}

value, err := strconv.Atoi(input.Massage)

if err != nil {

writer.WriteHeader(400)

writer.Write([]byte("Было введено не число или присутствуют пробелы в за-писи числа"))

return

}

err = h.DProvider.UpdateCount(value)

if err != nil {

writer.WriteHeader(500)

writer.Write([]byte(err.Error()))

return

}

writer.WriteHeader(201)

if value > 0 {

writer.Write([]byte("Значение счётчика было изменено на +" + strconv.Itoa(value)))

} else {

writer.Write([]byte("Значение счётчика было изменено на " + strconv.Itoa(value)))

}

}

func (Dp \*DatabaseProvider) SelectCount() (string, error) { var dbAnswer string

row := Dp.db.QueryRow("SELECT value FROM count")

err := row.Scan(&dbAnswer) // Проверка на то, есть ли искомые данные в БД if err != nil {

return "", err

}

return dbAnswer, nil

}

func (Dp \*DatabaseProvider) UpdateCount(n int) error {

\_, err := Dp.db.Exec("UPDATE count SET value = value + ($1)", n) if err != nil {

return err

}

return nil

}

func main() {

address := flag.String("address", "127.0.0.1:8083", "адрес для запуска сер-вера")

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{DProvider: dp}

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

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

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

if err != nil {

fmt.Print("error: server does not start")

}

}

1. **Сервис Query**

**Файл main.go**

package main

import (

"database/sql"

"flag"

"fmt"

"log"

"net/http"

\_ "github.com/lib/pq"

)

const (

host = "localhost"

port = 5432

user = "RA"

password = "postgres"

dbname = "sandbox"

)

type DatabaseProvider struct {

db \*sql.DB

}

type Handlers struct {

DProvider DatabaseProvider

}

func (h \*Handlers) SetName(writer http.ResponseWriter, r \*http.Request) { str := r.URL.Query().Get("name")

if str == "" {

writer.Write([]byte("Попробуй ввести своё имя через query-параметр 'name'"))

return

}

err := h.DProvider.AddName(str)

if err != nil {

writer.WriteHeader(500)

writer.Write([]byte(err.Error()))

return

}

writer.WriteHeader(201)

writer.Write([]byte("Имя было изменено на " + str))

}

func (h \*Handlers) GetName(writer http.ResponseWriter, r \*http.Request) { answer, err := h.DProvider.SelectName()

if err != nil {

writer.WriteHeader(500)

writer.Write([]byte(err.Error()))

return

}

writer.WriteHeader(200)

writer.Write([]byte("Привет, " + answer + "!"))

}

func (Db \*DatabaseProvider) AddName(name string) error {

\_, err := Db.db.Exec("UPDATE query SET name = ($1)", name)

if err != nil {

return err

}

return nil

}

func (Db \*DatabaseProvider) SelectName() (string, error) { var answer string

row := Db.db.QueryRow("SELECT name FROM query LIMIT 1")

err := row.Scan(&answer)

if err != nil {

return "", err

}

return answer, nil

}

func main() {

address := flag.String("address", "127.0.0.1:8082", "адрес для запуска сер-вера")

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{DProvider: dp}

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

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

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

if err != nil {

fmt.Print("error: server does not start")

}

}

**Вывод:**

При выполнении лабораторной работы удалось получить первичные навыки в организации долгосрочного хранения данных с использованием

PostgreSQL и Golang.