|  |  |
| --- | --- |
| **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**

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

**Дисциплина:** Языки интернет программирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-32Б |  |  | А.Н. Абрамов |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | И.О. Фамилия |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2024

# **Организация клиент-серверного взаимодейсвтия между Golang и PostgreSQL**

**Цель работы:** получение первичных навыков в организации долгосрочного хранения данных с использованием PostgreSQL и Golang.

**Задание:** В рамках данной лабораторной работы предлагается продолжить изучение Golang и познакомиться с набором стандартных библиотек, используемых для организации клиент-серверного взаимодействия между Golang и Postgresql, где в роли клиента выступает сервис Golang, а в роли сервера СУБД Postgresql.

**Решение:** Рассмотрим на примере calculator

**package** main

**import** (

"database/sql"

"encoding/json"

"fmt"

"net/http"

\_ "github.com/lib/pq"

)

**const** connectionString = "host=localhost port=5432 user=postgres dbname=sandbox password=postgres"

**type** Handlers **struct** {

db \*sql.DB

}

**func** (h \*Handlers) ServeGet(w http.ResponseWriter, r \*http.Request) {

**var** count **int**

row := h.db.QueryRow("SELECT count FROM count\_table LIMIT 1")

err := row.Scan(&count)

**if** err != **nil** {

fmt.Println(err)

w.WriteHeader(http.StatusInternalServerError)

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

**return**

}

err = json.NewEncoder(w).Encode(**struct** {

Count **int** `json:"count"`

}{Count: count})

**if** err != **nil** {

fmt.Println(err)

w.WriteHeader(http.StatusInternalServerError)

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

**return**

}

w.WriteHeader(http.StatusOK)

}

**func** (h \*Handlers) ServePost(w http.ResponseWriter, r \*http.Request) {

**var** dcount **struct** {

Count **int** `json:"count"`

}

err := json.NewDecoder(r.Body).Decode(&dcount)

**if** err != **nil** {

fmt.Println(err)

w.WriteHeader(http.StatusBadRequest)

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

**return**

}

\_, err = h.db.Exec("UPDATE count\_table SET count = count + $1", dcount.Count)

**if** err != **nil** {

fmt.Println(err)

w.WriteHeader(http.StatusInternalServerError)

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

**return**

}

w.WriteHeader(http.StatusOK)

}

**func** main() {

db, err := sql.Open("postgres", connectionString)

**if** err != **nil** {

fmt.Println(err)

**return**

}

**defer** db.Close()

handlers := Handlers{db: db}

http.HandleFunc("/get", handlers.ServeGet)

http.HandleFunc("/post", handlers.ServePost)

err = http.ListenAndServe(":8080", **nil**)

**if** err != **nil** {

fmt.Println(err)

}

}

**Заключение:** Были получены первичныу навыки в организации долгосрочного хранения данных с использованием PostgreSQL и Golang.

**Использованные источники:**

[**https://golangdocs.com/golang-postgresql-example**](https://golangdocs.com/golang-postgresql-example)