|  |  |
| --- | --- |
| **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 № 9**

**Название:** Back-End разработка с использованием фреймворка Echo

**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

# **Back-End разработка с использованием фреймворка Echo**

**Цель работы:** получение первичных навыков использования веб-фрейворков в BackEnd-разрабокте на Golang

**Задание:** В рамках данной лабораторной работы предлагается продолжить изучение Golang и познакомиться с веб-фреймворком Echo, используемым для организации обработки клиентских запросов. Echo упрощает работу и заменяет собой такие пакеты из std-библиотеки, как: "net/http", "encoding/json" и другие...

**Решение:** Рассмотрим на примере calculator

**package** main

**import** (

"database/sql"

"encoding/json"

"net/http"

"github.com/labstack/echo/v4"

\_ "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(c echo.Context) **error** {

**var** count **int**

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

**var** res **struct** {

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

Err **string** `json:"error"`

}

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

res.Err = err.Error()

**return** c.JSON(http.StatusInternalServerError, res)

}

res.Count = count

**return** c.JSON(http.StatusOK, res)

}

**func** (h \*Handlers) ServePost(c echo.Context) **error** {

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

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

}

**var** res **struct** {

Err **string** `json:"error"`

}

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

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

res.Err = err.Error()

**return** c.JSON(http.StatusBadRequest, res)

}

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

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

res.Err = err.Error()

**return** c.JSON(http.StatusInternalServerError, res)

}

**return** c.JSON(http.StatusOK, res)

}

**func** main() {

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

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

**return**

}

**defer** db.Close()

h := Handlers{db: db}

e := echo.New()

e.GET("/get", h.ServeGet)

e.POST("/post", h.ServePost)

e.Logger.Fatal(e.Start("localhost:8080"))

}

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

**Использованные источники:**

**https://golangdocs.com/golang-postgresql-example**

**https://echo.labstack.com/docs/**