Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені І. Сікорського»
Кафедра інженерії програмного забезпечення в енергетиці

Лабораторна робота №3

з курсу:

«Програмування вебзастосунків»

Виконав:

студент 4-го курсу, групи ТВ-11

Дзюбан Даніїл Олександрович

Посилання на GitHub репозиторій:

https://github.com/01xniel/KPI-Golang

Перевірив:

Недашківський О.Л.

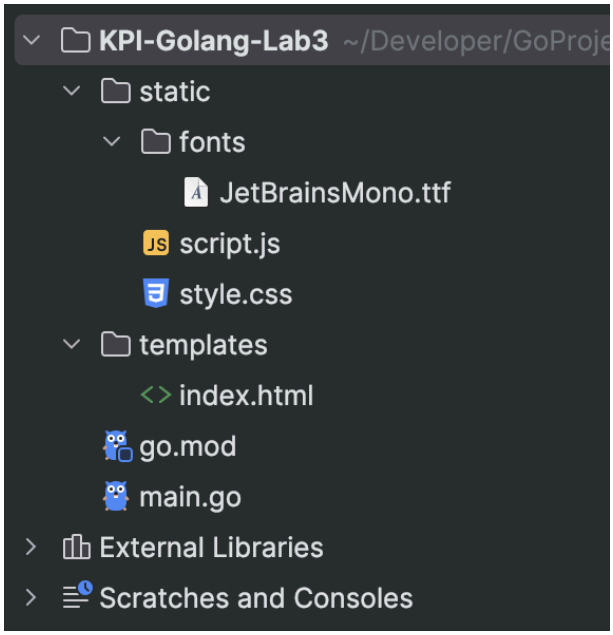
Київ 2025

Завдання:

Створити веб-застосунок (калькулятор) для розрахунку прибутку від сонячних електростанцій з встановленою системою прогнозування сонячної потужності.

Хід роботи:

Структура проєкту:



Файл index.html:

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Лабораторна робота №3</title>
    <link rel="stylesheet" href="../static/style.css">
</head>
<body>
    <div id="app-container">
        <div id="info-section">
            <h1>Калькулятор</h1>
            <p class="description">Калькулятор розрахунку прибутку сонячних електростанцій
                з встановленою системою прогнозування сонячної потужності</p>
        </div>
```

```
        <form id="calculator-form">
            <div class="input-block">
                <label for="average_daily_capacity">Середньодобова потужність (МВт)</label>
                <input id="average_daily_capacity" name="average_daily_capacity"
                       type="number" step="any" value="5.0" required>
            </div>
            <div class="input-block">
                <label for="electricity_cost">Вартість електроенергії (грн/кВт·год)</label>
                <input id="electricity_cost" name="electricity_cost"
                       type="number" step="any" value="7.0" required>
            </div>
            <div class="input-block">
                <label for="standard_deviation">Середньоквадратичне відхилення (МВт)</label>
                <input id="standard_deviation" name="standard_deviation"
                       type="number" step="any" value="1.0" required>
            </div>
            <button type="submit" class="process-button">Розрахувати</button>
        </form>
        <script src="../static/script.js"></script>
    </div>
</body>
</html>
```

Файл script.js:

```
document.addEventListener( type: "DOMContentLoaded", listener: function () {
    const form = document.getElementById( elementId: "calculator-form");

    form.addEventListener( type: "submit", listener: async function (event) {
        event.preventDefault();

        const formData = new FormData(form);

        const response = await fetch( input: "/evaluate", init: {
            method: "POST",
            body: formData
        });

        if (!response.ok) {
            if (response.status === 400) {
                alert("Помилка у введених даних...");
            } else if (response.status === 500) {
                alert("Помилка сервера...");
            } else {
                alert("Невідома помилка...");
            }
            return;
        }

        const data = await response.json();
```

```
        showResultSection( htmlContent: `
            <h2>Результат</h2>
            <p class="result-paragraph"><b>${data.Profit.toFixed( fractionDigits: 1)} тис. грн.</b>
                - дохід від генерації енергії без небалансів <b>(
                ${data.ElectricityNoImbalance.toFixed( fractionDigits: 1)} МВт·год)</b></p>
            <p class="result-paragraph"><b>${data.Penalty.toFixed( fractionDigits: 1)} тис. грн.</b>
                - штраф за генерацію енергії з небалансами <b>(
                ${data.ElectricityImbalance.toFixed( fractionDigits: 1)} МВт·год)</b></p>
            <p class="result-paragraph">${data.NetProfit >= 0 ? "Прибуток" : "Збитки"}:
                <b>${Math.abs( x: data.NetProfit.toFixed( fractionDigits: 1))} тис. грн.</b></p>
        `)
    });
});

function showResultSection(htmlContent) {  Show usages
    let resultSection = document.getElementById( elementId: "result-section");

    if (!resultSection) {
        resultSection = document.createElement( tagName: "div");
        resultSection.id = "result-section";
        document.getElementById( elementId: "app-container").appendChild( node: resultSection);
    }

    resultSection.innerHTML = htmlContent;
}
```

Файл main.go:

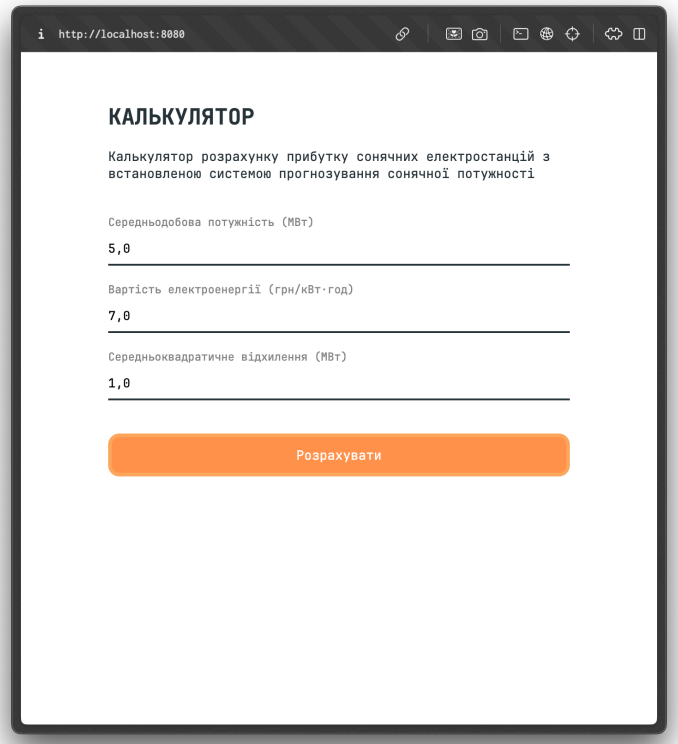
```
package main

import (
    "encoding/json"
    "html/template"
    "log"
    "math"
    "net/http"
    "strconv"
)

Implement interface
type ResultData struct {  3 usages
    Profit                float64
    ElectricityNoImbalance float64
    Penalty               float64
    ElectricityImbalance  float64
    NetProfit             float64
}

Implement interface
type CapacityRange struct {  2 usages
    Min float64
    Max float64
}
```

```go
func normalPowerDistributionLaw(  1 usage
    p float64,
    averageDailyCapacity float64,
    standardDeviation float64,
) float64 {
    normalizationFactor := 1 / (standardDeviation * math.Sqrt(2*math.Pi))
    exponentTerm := math.Pow(p-averageDailyCapacity, 2.0) / (2 * math.Pow(standardDeviation, 2.0))
    return normalizationFactor * math.Exp(-exponentTerm)
}

func gaussLegendreIntegration(  1 usage
    averageDailyCapacity float64,
    standardDeviation float64,
    capacityRange CapacityRange,
) float64 {
    t := [5]float64{-0.90617985, -0.53846931, 0.0, 0.53846931, 0.90617985}
    coefA := [5]float64{0.23692688, 0.47862868, 0.56888889, 0.47862868, 0.23692688}

    integrationResult := 0.0

    for i := 0; i < 5; i++ {
        x := 0.5*(capacityRange.Max+capacityRange.Min) + 0.5*(capacityRange.Max-capacityRange.Min)*t[i]
        y := normalPowerDistributionLaw(x, averageDailyCapacity, standardDeviation)
        coefC := 0.5 * (capacityRange.Max - capacityRange.Min) * coefA[i]
        integrationResult += coefC * y
    }

    return integrationResult
}

var tmpl *template.Template  2 usages

func init() {
    var err error
    tmpl, err = template.ParseFiles("templates/index.html")
    if err != nil {
        log.Fatal(err)
    }
}

func renderTemplate(w http.ResponseWriter, data ResultData) {  1 usage
    err := tmpl.Execute(w, data)
    if err != nil {
        http.Error(w, err.Error(), http.StatusInternalServerError)
    }
}

func handleCalculator(w http.ResponseWriter, r *http.Request) {  1 usage
    renderTemplate(w, ResultData{})
}
```

```go
func handleCalculations(w http.ResponseWriter, r *http.Request) {  1 usage
    if r.Method != "POST" {
        http.Redirect(w, r, "/", http.StatusSeeOther)
        return
    }

    averageDailyCapacityStr := r.FormValue("average_daily_capacity")
    electricityCostStr := r.FormValue("electricity_cost")
    standardDeviationStr := r.FormValue("standard_deviation")

    averageDailyCapacity, err1 := strconv.ParseFloat(averageDailyCapacityStr, 64)
    electricityCost, err2 := strconv.ParseFloat(electricityCostStr, 64)
    standardDeviation, err3 := strconv.ParseFloat(standardDeviationStr, 64)

    if err1 != nil || err2 != nil || err3 != nil {
        http.Error(w, "Invalid input data", http.StatusBadRequest)
        return
    }

    forecastError := averageDailyCapacity * 0.05
    capacityRange := CapacityRange{
        Min: averageDailyCapacity - forecastError,
        Max: averageDailyCapacity + forecastError,
    }

    electricityNoImbalanceRelativeVal := gaussLegendreIntegration(averageDailyCapacity, standardDeviatio
    electricityImbalanceRelativeVal := 1 - electricityNoImbalanceRelativeVal

    electricityNoImbalance := averageDailyCapacity * 24 * electricityNoImbalanceRelativeVal
    electricityImbalance := averageDailyCapacity * 24 * electricityImbalanceRelativeVal

    profit := electricityNoImbalance * electricityCost
    penalty := electricityImbalance * electricityCost
    netProfit := profit - penalty

    w.Header().Set("Content-Type", "application/json")
    err := json.NewEncoder(w).Encode(ResultData{
        Profit:                profit,
        ElectricityNoImbalance: electricityNoImbalance,
        Penalty:               penalty,
        ElectricityImbalance:   electricityImbalance,
        NetProfit:             netProfit,
    })
    if err != nil {
        http.Error(w, "Failed to encode response", http.StatusInternalServerError)
    }
}

func main() {
    http.HandleFunc("/", handleCalculator)
    http.HandleFunc("/evaluate", handleCalculations)
    http.Handle("/static/", http.StripPrefix("/static", http.FileServer(http.Dir("static"))))
```

```
    log.Println("Server running on http://localhost:8080")
    log.Fatal(http.ListenAndServe(":8080", nil))
}
```

Перевірка роботи калькулятора (контрольний приклад):



Висновки:

У рамках виконання даної практичної роботи було розроблено веб-застосунок (калькулятор) для розрахунку прибутку від сонячних електростанцій із встановленою системою прогнозування сонячної потужності.

Під час виконання роботи було здобуто нові навички створення сучасних веб-застосунків із використанням мови програмування Go, зокрема:

- реалізація маршрутизації та обробки HTTP-запитів за допомогою стандартного пакета net/http;
- робота з формами введення та обробка введених користувачем даних на сервері;

Крім цього, у процесі розробки було досліджено та застосовано метод чисельного інтегрування Гауса-Лежандра для обчислення частки енергії, що генерується без

небалансів, на основі функції нормального розподілу потужності.