**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по Домашнему заданию

«Разработка телеграмм бота на Go»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-31Б |  | преподаватель каф. ИУ5 |
| Князев А.М. |  | Нардид А.Н. |
|  |  |  |

Москва, 2024 г.

**Описание задания**

В основу Домашнего задания легла лабораторная работа№6(«Разработка телеграмм бота на Go») с некоторыми дополнениями и усложнениями, а именно:

1. Добавить возможность многопоточного использования бота для нескольких пользователей.
2. Реализовать защиту данных
3. Увеличить функционал бота

**Текст программы**

Файл *main.go*

package main

import (

    "crypto/rand"

    "fmt"

    "github.com/go-telegram-bot-api/telegram-bot-api/v5"

    "log"

    "math"

    "math/big"

    "strconv"

    "strings"

    "sync"

)

var userStates = struct {

    sync.RWMutex

    data map[int64]string

}{data: make(map[int64]string)}

func main() {

    bot, err := tgbotapi.NewBotAPI("7557980296:AAHWaavcV85arPbn-erWPAuEy176wm7S4Gg")

    if err != nil {

        log.Panic(err)

    }

    bot.Debug = true

    log.Printf("Authorized on account %s", bot.Self.UserName)

    u := tgbotapi.NewUpdate(0)

    u.Timeout = 60

    updates := bot.GetUpdatesChan(u)

    rootButton := tgbotapi.NewKeyboardButton("Найти корни квадратного уравнения")

    areaButton := tgbotapi.NewKeyboardButton("Найти площадь прямоугольника")

    passwordButton := tgbotapi.NewKeyboardButton("Сгенерировать пароль")

    randomNumberButton := tgbotapi.NewKeyboardButton("Сгенерировать случайное число")

    keyboard := tgbotapi.NewReplyKeyboard(

        tgbotapi.NewKeyboardButtonRow(rootButton),

        tgbotapi.NewKeyboardButtonRow(areaButton),

        tgbotapi.NewKeyboardButtonRow(passwordButton),

        tgbotapi.NewKeyboardButtonRow(randomNumberButton),

    )

    for update := range updates {

        if update.Message == nil {

            continue

        }

        chatID := update.Message.Chat.ID

        msg := tgbotapi.NewMessage(chatID, "")

        switch update.Message.Text {

        case "/start":

            msg.Text = "Выберите функцию:"

            msg.ReplyMarkup = keyboard

            bot.Send(msg)

        case "Найти корни квадратного уравнения":

            setUserState(chatID, "roots")

            msg.Text = "Введите коэффициенты a, b и c через пробел (например: 1 -3 2):"

            bot.Send(msg)

        case "Найти площадь прямоугольника":

            setUserState(chatID, "rectangle")

            msg.Text = "Введите длины сторон прямоугольника a и b через пробел (например: 3 4):"

            bot.Send(msg)

        case "Сгенерировать пароль":

            setUserState(chatID, "password")

            msg.Text = "Введите длину пароля (например: 12):"

            bot.Send(msg)

        case "Сгенерировать случайное число":

            setUserState(chatID, "random")

            msg.Text = "Введите диапазон через пробел (например: 1 100):"

            bot.Send(msg)

        default:

            currentFunction := getUserState(chatID)

            switch currentFunction {

            case "roots":

                input := strings.Fields(update.Message.Text)

                if len(input) == 3 {

                    a, \_ := strconv.ParseFloat(input[0], 64)

                    b, \_ := strconv.ParseFloat(input[1], 64)

                    c, \_ := strconv.ParseFloat(input[2], 64)

                    msg.Text = calculateRoots(a, b, c)

                } else {

                    msg.Text = "Неверный формат. Пожалуйста, введите три коэффициента через пробел."

                }

            case "rectangle":

                input := strings.Fields(update.Message.Text)

                if len(input) == 2 {

                    a, err1 := strconv.ParseFloat(input[0], 64)

                    b, err2 := strconv.ParseFloat(input[1], 64)

                    if err1 == nil && err2 == nil {

                        msg.Text = calculateRectangleArea(a, b)

                    } else {

                        msg.Text = "Неверный формат. Пожалуйста, введите два числа через пробел."

                    }

                } else {

                    msg.Text = "Неверный формат. Пожалуйста, введите два числа через пробел."

                }

            case "password":

                length, err := strconv.Atoi(update.Message.Text)

                if err == nil && length > 0 {

                    msg.Text = generatePassword(length)

                } else {

                    msg.Text = "Неверный формат. Пожалуйста, введите положительное целое число."

                }

            case "random":

                input := strings.Fields(update.Message.Text)

                if len(input) == 2 {

                    min, err1 := strconv.Atoi(input[0])

                    max, err2 := strconv.Atoi(input[1])

                    if err1 == nil && err2 == nil && min < max {

                        msg.Text = generateRandomNumber(min, max)

                    } else {

                        msg.Text = "Неверный формат. Пожалуйста, введите два числа через пробел, где первое меньше второго."

                    }

                } else {

                    msg.Text = "Неверный формат. Пожалуйста, введите два числа через пробел."

                }

            default:

                msg.Text = "Пожалуйста, выберите функцию из предложенных вариантов."

            }

            bot.Send(msg)

        }

    }

}

func setUserState(chatID int64, state string) {

    userStates.Lock()

    defer userStates.Unlock()

    userStates.data[chatID] = state

}

func getUserState(chatID int64) string {

    userStates.RLock()

    defer userStates.RUnlock()

    return userStates.data[chatID]

}

func calculateRoots(a, b, c float64) string {

    d := b\*b - 4\*a\*c

    if a == 0 {

        return "Уравнение не является квадратным"

    }

    if d > 0 {

        x1 := (-b + math.Sqrt(d)) / (2 \* a)

        x2 := (-b - math.Sqrt(d)) / (2 \* a)

        return fmt.Sprintf("Корни уравнения: x1 = %.2f, x2 = %.2f\n", x1, x2)

    } else if d == 0 {

        x := -b / (2 \* a)

        return fmt.Sprintf("Корни уравнения: x = %.2f\n", x)

    } else {

        realPart := -b / (2 \* a)

        imagPart := math.Sqrt(math.Abs(d)) / (2 \* a)

        x1 := complex(realPart, imagPart)

        x2 := complex(realPart, -imagPart)

        return fmt.Sprintf("Уравнение не имеет действительных корней: Корни уравнения: x1 = %.3v, x2 = %.3v\n", x1, x2)

    }

}

func calculateRectangleArea(a, b float64) string {

    if a <= 0 || b <= 0 {

        return "Длины сторон прямоугольника должны быть положительными числами."

    }

    area := a \* b

    return fmt.Sprintf("Площадь прямоугольника: %.2f", area)

}

func generatePassword(length int) string {

    const charset = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&\*()"

    password := make([]byte, length)

    for i := range password {

        charIndex, \_ := rand.Int(rand.Reader, big.NewInt(int64(len(charset))))

        password[i] = charset[charIndex.Int64()]

    }

    return string(password)

}

func generateRandomNumber(min, max int) string {

    rangeValue := max - min + 1

    if rangeValue <= 0 {

        return "Неверный диапазон. Убедитесь, что min меньше max."

    }

    randomValue, \_ := rand.Int(rand.Reader, big.NewInt(int64(rangeValue)))

    return fmt.Sprintf("Случайное число: %d", min+int(randomValue.Int64()))

}

**Экранные формы с примерами выполнения программы**



 