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 лабораторной работе №3

«Система управления пользователями и заметками с использованием сессий на языке Go»

Выполнил:
студент группы ИУ5-35Б
Бердников Н.О.
Подпись и дата:

Проверил:
Гапанюк Ю. Е.

Подпись и дата:

Москва, 2024 г

Описание программы:

Программа представляет собой веб-приложение, разработанное на языке

программирования Go, с использованием библиотеки Gorilla sessions для управления сессиями. Основная цель программы — предоставление пользователям возможности регистрации, авторизации и управления личными заметками. Программа имеет следующие функциональные модули:

Регистрация пользователей: Новые пользователи могут зарегистрироваться, указав имя пользователя и пароль. Эти данные сохраняются в текстовый файл для последующего использования.

Авторизация пользователей: Зарегистрированные пользователи могут войти в систему, введя свои учетные данные. Если введенные данные совпадают с сохраненными, начинается пользовательская сессия.

Создание и хранение заметок: После успешной авторизации пользователи могут создавать текстовые заметки, которые сохраняются вместе с именем автора. Заметки хранятся в текстовом файле и могут быть просмотрены только тем пользователем, который их создал.

Управление сессиями: Для управления сеансами авторизации используются куки-файлы с использованием библиотеки Gorilla sessions. Пользователь остается авторизованным в течение одной сессии или до выхода из системы.

Тестирование: Включены функции для тестирования различных аспектов работы программы, таких как авторизация с правильными и неправильными учетными данными, регистрация нового пользователя, отображение заметок для авторизованного пользователя и выход из системы.

Текст программы:

Main.go:

```
package main

import (
	"bufio"
	"fmt"
	"html/template"
	"net/http"
	"os"
	"strings"

	"github.com/gorilla/sessions"
)

var (
	store = sessions.NewCookieStore([]byte("secret-key"))
)

var tmpl = template.Must(template.ParseFiles("templates/login.html",
"templates/register.html", "templates/notes.html"))

type Note struct {
	Title   string
	Content string
	Created string
}

func loginHandler(w http.ResponseWriter, r *http.Request) {
	session, _ := store.Get(r, "session-name")
	if auth, _ := session.Values["authenticated"].(bool); auth {
		http.Redirect(w, r, "/notes", http.StatusSeeOther)
		return
	}
	if r.Method == http.MethodPost {
```

```
            username := r.FormValue("username")
            password := r.FormValue("password")
            if findUser(username, password) {
                  session.Values["authenticated"] = true
                  session.Values["username"] = username
                  session.Options = &sessions.Options{
                        Path:     "/",
                        MaxAge:   3600,
                        HttpOnly: true,
                  }
                  session.Save(r, w)
                  http.Redirect(w, r, "/notes", http.StatusSeeOther)
                  return
            } else {
                  fmt.Fprintf(w, "Invalid credentials")
            }
            return
      }
      tmpl.ExecuteTemplate(w, "login.html", nil)
}

func logoutHandler(w http.ResponseWriter, r *http.Request) {
      session, _ := store.Get(r, "session-name")
      session.Values["authenticated"] = false
      session.Save(r, w)
      http.Redirect(w, r, "/", http.StatusSeeOther)
}

func registerHandler(w http.ResponseWriter, r *http.Request) {
      if r.Method == http.MethodPost {
            username := r.FormValue("username")
            password := r.FormValue("password")
            if writeUser(username, password) {
                  http.Redirect(w, r, "/", http.StatusSeeOther)
                  return
            }
      }
      tmpl.ExecuteTemplate(w, "register.html", nil)
}

func notesHandler(w http.ResponseWriter, r *http.Request) {
      session, _ := store.Get(r, "session-name")

      if auth, ok := session.Values["authenticated"].(bool); !ok || !auth {
            http.Redirect(w, r, "/", http.StatusSeeOther)
            return
      }
      created := session.Values["username"].(string)
      if r.Method == http.MethodPost {
            title := r.FormValue("title")
            content := r.FormValue("content")
            writeNote(Note{title, content, created})
            http.Redirect(w, r, "/notes", http.StatusSeeOther)
            return
      }
      notes := getNotes(created)
      tmpl.ExecuteTemplate(w, "notes.html", notes)
}

func writeNote(note Note) {
      file, err := os.OpenFile("notes.txt", os.O_APPEND|os.O_CREATE|os.O_WRONLY, 0644)
      if err != nil {
            fmt.Println("Error opening file:", err)
            return
      }
      defer file.Close()
```

```
	_, err = file.WriteString(fmt.Sprintf("%s %s %s\n", note.Title, note.Content,
note.Created))
	if err != nil {
		fmt.Println("Error writing to file:", err)
	}
}

func writeUser(username string, password string) bool {
	file, err := os.OpenFile("users.txt", os.O_APPEND|os.O_RDWR, 0644)
	if err != nil {
		fmt.Println("Error opening file:", err)
		return false
	}
	defer file.Close()
	loginUsed := false
	scanner := bufio.NewScanner(file)
	for scanner.Scan() {
		line := scanner.Text()
		values := strings.Split(line, " ")
		if username == values[0] {
			loginUsed = true
			break
		}
	}
	if !loginUsed {
		_, err := file.WriteString(username + " " + password + "\n")
		if err != nil {
			fmt.Println("Error writing to file:", err)
			return false
		}
	}
	if err := scanner.Err(); err != nil {
		fmt.Println("Error reading file:", err)
	}

	return !loginUsed
}

func findUser(username string, password string) bool {
	file, err := os.Open("users.txt")
	if err != nil {
		fmt.Println("Error opening file:", err)
		return false
	}
	status := false
	defer file.Close()
	scanner := bufio.NewScanner(file)
	for scanner.Scan() {
		line := scanner.Text()
		values := strings.Split(line, " ")
		if username == values[0] && password == values[1] {
			status = true
		}
	}

	if err := scanner.Err(); err != nil {
		fmt.Println("Error reading file:", err)
	}
	return status
}

func getNotes(username string) []Note {
	file, err := os.Open("notes.txt")
	if err != nil {
		fmt.Println("Error opening file:", err)
		return make([]Note, 0)
	}
```

```
	defer file.Close()
	notes := make([]Note, 0)
	scanner := bufio.NewScanner(file)
	for scanner.Scan() {
		line := scanner.Text()
		values := strings.Split(line, " ")
		if username == values[2] {
			notes = append(notes, Note{values[0], values[1], values[2]})
		}
	}

	if err := scanner.Err(); err != nil {
		fmt.Println("Error reading file:", err)
	}
	return notes
}

func main() {
	http.HandleFunc("/", loginHandler)
	http.HandleFunc("/register", registerHandler)
	http.HandleFunc("/logout", logoutHandler)
	http.HandleFunc("/notes", notesHandler)
	fmt.Println("Starting server at :8080")
	err := http.ListenAndServe(":8080", nil)
	if err != nil {
		panic(err)
	}
}
```

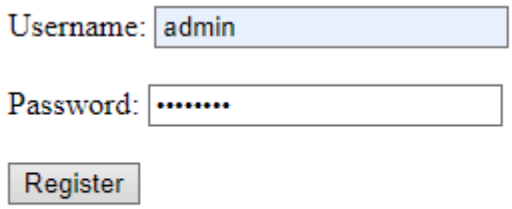
Main_test.go:

```
package main

import (
	"net/http"
	"net/http/httptest"
	"strings"
	"testing"
)

func TestLoginHandler_ValidCredentials(t *testing.T) {
	req := httptest.NewRequest("POST", "/",
strings.NewReader("username=testuser&password=testpass"))
	req.Header.Set("Content-Type", "application/x-www-form-urlencoded")
	w := httptest.NewRecorder()

	loginHandler(w, req)

	if status := w.Code; status != http.StatusOK {
		t.Errorf("Expected status code 200, got %v", status)
	}
}

func TestLoginHandler_InvalidCredentials(t *testing.T) {
	req := httptest.NewRequest("POST", "/",
strings.NewReader("username=wronguser&password=wrongpass"))
	req.Header.Set("Content-Type", "application/x-www-form-urlencoded")
	w := httptest.NewRecorder()

	loginHandler(w, req)

	if !strings.Contains(w.Body.String(), "Invalid credentials") {
		t.Errorf("Expected 'Invalid credentials', got %v", w.Body.String())
	}
}
func TestRegisterHandler(t *testing.T) {
	req := httptest.NewRequest("POST", "/register",
strings.NewReader("username=newuser&password=newpass"))
	req.Header.Set("Content-Type", "application/x-www-form-urlencoded")
```

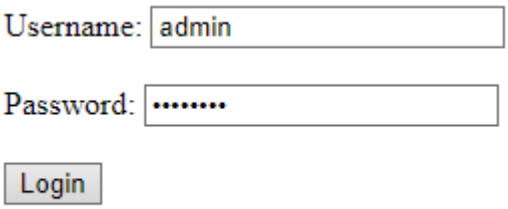
```
	w := httptest.NewRecorder()

	registerHandler(w, req)

	if status := w.Code; status != http.StatusOK {
		t.Errorf("Expected status code 200, got %v", status)
	}
}
func TestNotesHandler_Authenticated(t *testing.T) {
	req := httptest.NewRequest("GET", "/notes", nil)
	w := httptest.NewRecorder()

	session, _ := store.Get(req, "session-name")
	session.Values["authenticated"] = true
	session.Values["username"] = "testuser"
	session.Save(req, w)

	notesHandler(w, req)

	if status := w.Code; status != http.StatusOK {
		t.Errorf("Expected status code 200, got %v", status)
	}
}
func TestLogoutHandler(t *testing.T) {
	req := httptest.NewRequest("GET", "/logout", nil)
	w := httptest.NewRecorder()

	session, _ := store.Get(req, "session-name")
	session.Values["authenticated"] = true
	session.Save(req, w)

	logoutHandler(w, req)

	if status := w.Code; status != http.StatusSeeOther {
		t.Errorf("Expected status code 303, got %v", status)
	}
}
```

Результат вывода:

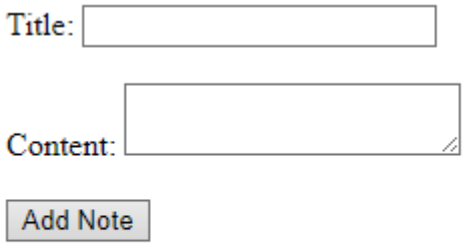
Страница регистрации /register



Страница авторизации /



Странице создания и просмотра заметок /notes



Использованные источники:

1. [Электронный ресурс] Курс ПиКЯП (Парадигмы и конструкции языков программирования). URL:https://github.com/ugapanyuk/courses_content/wiki/COURSE_PCPL_MAIN