ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Кафедра інженерії програмного забезпечення

ЛАБОРАТОРНА РОБОТА №1

з дисципліни «Веб-технології»

.(назва дисципліни)

на тему: «Створення простого застосунку в Go»

ЛРІПЗс.2401075.01.07.ЛР

(шифр, назва)

Рівень вищої освіти                     Перший (бакалаврський)

Галузь знань                      12 «Інформаційні технології»

Спеціальність     121 «Інженерія програмного забезпечення»

Освітня програма   «Інженерія програмного забезпечення»

Студента. І  курсу, група . ІПЗс-24. \_\_\_\_\_\_\_\_\_\_\_ М.О. Лапко.

            (шифр)                         (підпис)     (ініціали, прізвище)

Викладач канд. техн. наук, доцент  .   \_\_\_\_\_\_\_\_\_\_\_ О.М. Яшина

              (посада, вчене звання, науковий ступінь)         (підпис)   (ініціали, прізвище)

Кількість балів \_\_\_\_\_\_\_

Оцінка за шкалою:

національною\_\_\_\_\_\_\_\_\_/ЄКТС\_\_\_\_

Хмельницький, 2024

ЗВІТ

**Варіант 7**

**Мета:** Створити простий застосунок на Go, що спрямований дати початкове розуміння роботи з htmx templates.

**ХІД РОБОТИ**

**Завдання №1.** Обрати із тему курсового проекту.

Була обрана тема: Social network Y

**Завдання №2.** Розробити технічне завдання до проекту.

## 1. ****Загальна інформація****

**Назва проекту**: Social network Y  
**Мета проекту**: Розробка соціальної мережі з можливістю публікації постів, редагування, видалення, модерації та керування правами користувачів і адміністраторів.

## 2. ****Опис продукту****

**Функціонал для користувачів**:

* Реєстрація та аутентифікація.
* Публікація, редагування та видалення постів.
* Лайкання постів.
* Перегляд постів.
* Повідомлення про порушення.
* Зміна пароля, аватарки, нікнейму, username (login) та email.
* Видалення акаунта.
* Пошук користувачів.
* Пошук постів по тегам.

**Функціонал для адміністрації**:

* Модерація постів користувачів (редагування, видалення).
* Перегляд та обробка повідомлень про порушення.
* Керування правами користувачів та інших адміністраторів.
* Зміна аватарок, нікнеймів, username (login) та email користувачів.
* Блокування користувачів.
* Видалення акаунтів користувачів (тільки для адміністраторів рівня 3).

## 3. ****Функціональні вимоги****

### 3.1. ****Користувачі****

1. **Реєстрація**:
   * Реєстрація за email або username з перевіркою унікальності.
2. **Аутентифікація**:
   * Вхід користувачів з отриманням JWT токенів для подальших дій у системі.
3. **Пости**:
   * Створення, редагування, видалення та перегляд власних постів.
   * Лайкання постів.
4. **Зміна пароля, аватарки, нікнейму, username та email**:
   * Користувач може змінювати свій пароль, аватар, нікнейм, username (login) та email.
   * Адміністратор може змінювати ці дані для будь-якого користувача.
5. **Видалення акаунта**:
   * Користувач може видалити свій акаунт.
   * Адміністратор рівня 3 може видаляти акаунти будь-яких користувачів.
6. **Повідомлення про порушення**:
   * Повідомлення про порушення публікацій.
7. **Пошук користувачів, тегів**:
   * Можливість пошуку користувачів за їхнім ім'ям або нікнеймом.
   * Можливість пошуку постів за тегами.

### 3.2. ****Адміністратори****

1. **Ролі та рівні доступу**:
   * **Рівень 1-3**: перегляд повідомлень про порушення.
   * **Рівень 2-3**: призначення ролей користувачам та адміністраторам.
   * **Рівень 3**: модерація інших адміністраторів та видалення акаунтів користувачів.
2. **Модерація постів**:
   * Видалення та редагування постів інших користувачів.
3. **Блокування користувачів**:
   * Блокування порушників.
4. **Зміна пароля, аватарок, нікнеймів, username та email**:
   * Адміністратори можуть змінювати пароль, аватар, нікнейм, username (login) та email користувачів.

## 4. ****Нефункціональні вимоги****

### 4.1. ****Продуктивність****

Час відповіді сервера не більше 500 мс при навантаженні до 1000 користувачів.

### 4.2. ****Безпека****

* Захищені JWT токени для аутентифікації.
* Хешування паролів.
* Захист від атак (SQL-ін'єкції, XSS, CSRF).

### 4.3. ****Масштабованість****

Система повинна підтримувати можливість розширення функціоналу та масштабування мікросервісів для обслуговування великої кількості користувачів.

## 5. ****Інтерфейс користувача****

### 5.1. ****Головна сторінка****

Перелік постів з можливістю лайкання та повідомлення про порушення.

### 5.2. ****Сторінка профілю****

* Інформація про користувача.
* Можливість редагування та видалення постів.
* Зміна аватарки, нікнейму, пароля, username (login) та email.
* Видалення акаунта.

### 5.3. ****Пошук****

* Інтерфейс для пошуку користувачів за ім'ям або нікнеймом.
* Інтерфейс для пошуку постів за тегами.

### 5.4. ****Адмін панель****

* Інтерфейс для перегляду і модерації постів.
* Можливість перегляду повідомлень про порушення.
* Керування ролями користувачів та адміністраторів.
* Можливість блокування користувачів.
* Можливість видалення акаунтів користувачів.

## 6. ****Технологічні вимоги****

* **Backend**: Go (Golang), GORM, MySQL.
* **Frontend**: HTML/CSS/JavaScript/React.
* **Сервіси**: мікросервіси (auth, post, report, user).
* **API**: REST/gRPC для взаємодії фронтенду з бекендом.

## 7. ****Опціональний функціонал (може бути додано у майбутньому)****

1. **Чат** між користувачами.
2. **Система повідомлень** (сповіщення про нові пости, лайки, коментарі).
3. **Інтеграція зі сторонніми сервісами** (наприклад, для автентифікації через Google або Facebook).
4. **Статистика та аналітика** для користувачів і постів (кількість лайків, коментарів тощо).

**Завдання №3.** Створити простий застосунок із визначенням головної сторінки, на якій вказати дані про студента:

1. Прізвище та ім’я. Maksim Lapko

2. Номер групи та назва вузу. IPZs-24 HNU

3. Тему курсового проекту. Social Network Y

Для виконання цієї лабораторної роботи використовуватимемо бібліотеку echo ([посилання](github.com/labstack/echo/v4)), яка забезпечує високопродуктивний веб-фреймворк для мови програмування **Go**, який відрізняється простотою у використанні та багатим набором можливостей для створення веб-додатків і API. Основні особливості Echo:

* **Мінімалізм**: невелика кількість залежностей і високий рівень продуктивності.
* **Мультиплексування маршрутів**: Echo підтримує зручне визначення маршрутів для різних HTTP-запитів (GET, POST, PUT тощо).
* **Мідлвари (middleware)**: дає змогу легко додавати проміжні обробники, які виконуються перед або після основного обробника запиту, що спрощує роботу з аутентифікацією, логуванням, стисненням і т.д.
* **Шаблони (templates)**: Echo підтримує роботу з шаблонами, що дозволяє створювати динамічні HTML-сторінки.

### Використання шаблонів у Echo

Я планую використовувати функціональність шаблонів для створення HTML-сторінок у лабораторній роботі. Echo дозволяє інтегрувати різні шаблонізатори, зокрема **HTML шаблони** на основі стандартної бібліотеки Go. Це дозволить створювати динамічні сторінки з можливістю підставляти дані з сервера в HTML.

Для початку необхідно встановити необхідні пакети для проекту. Основна бібліотека, яку ми будемо використовувати — це **Echo**. Її можна додати до проекту за допомогою команди:

|  |
| --- |
| go get github.com/labstack/echo/v4 |

Проект матиме таку структуру:

|  |
| --- |
| - main.go  - public/  - static/  - styles.css  - views/  - base.html  - content.html |

Файл main.go є основним файлом нашого проекту, який відповідає за ініціалізацію сервера Echo, налаштування рендерингу шаблонів і визначення маршрутів.

Код файлу:

|  |
| --- |
| package main  import (      "html/template"      "io"      "net/http"      "github.com/labstack/echo/v4"  )  type Template struct {      templates \*template.Template  }  func (t \*Template) Render(w io.Writer, name string, data interface{}, c echo.Context) error {      return t.templates.ExecuteTemplate(w, name, data)  }  func main() {      e := echo.New()      e.Renderer = &Template{          templates: template.Must(template.ParseGlob("public/views/\*.html")),      }      e.Static("/static", "public/static")      e.GET("/", func(c echo.Context) error {          return c.Render(http.StatusOK, "base", map[string]interface{}{              "Name":       "Maksim Lapko",              "Group":      "IPZs-24",              "University": "HNU",              "Project":    "Social Network Y",          })      })      e.Logger.Fatal(e.Start(":8080"))  } |

Ми використовуємо два HTML-шаблони для рендерингу контенту.

Файл **base.html**

Цей шаблон є основним, і до нього буде підключатися інший шаблон content.html для відображення контенту. Файл містить підключення до стилів Bootstrap і кастомних стилів:

|  |
| --- |
| {{define "base"}}  <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>PR 1</title>      <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">      <link href="/static/styles.css" rel="stylesheet">  </head>  <body>      <div class="container">          {{template "content" .}}      </div>  </body>  </html>  {{end}} |

Файл **content.html**

Шаблон для відображення інформації про студента:

|  |
| --- |
| {{define "content"}}  <div class="card">      <div class="card-header text-center">          <h1>{{.Name}}</h1>      </div>      <div class="card-body">          <p class="text-center">{{.Group}} {{.University}}</p>          <p class="text-center">{{.Project}}</p>      </div>  </div>  {{end}} |

Файл **styles.css** для стилізації картки з інформацією:

|  |
| --- |
| body {      background-color: #f8f9fa;  }  .card {      margin-top: 50px;      box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  }  .card-header {      background-color: #007bff;      color: white;  } |

Щоб запустити проект, використайте команду:

|  |
| --- |
| go run main.go |

Проект доступний за адресою [http://localhost:8080](http://localhost:8080/)

Таким чином результати роботи відповідного функціоналу наявний на

**Рисунку 1.**

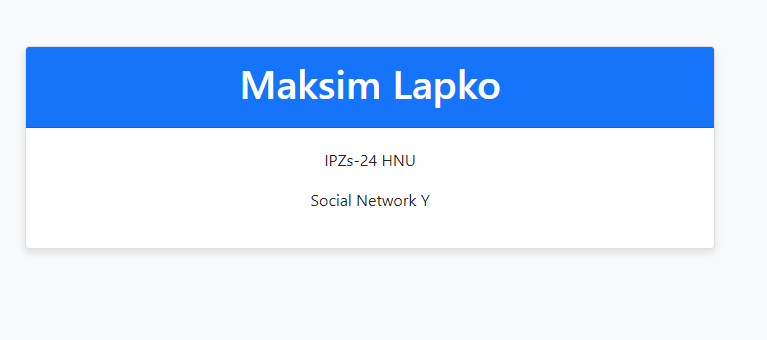


Рисунок 1 — Головна сторінка

**Контрольні запитання**

**Що таке Go?**

* **Go**, також відомий як **Golang**, — це мова програмування, розроблена компанією Google. Вона характеризується простотою синтаксису, швидкістю компіляції, а також вбудованою підтримкою паралелізму через горутини (goroutines) та канали (channels). Go ідеально підходить для розробки високопродуктивних додатків, зокрема серверних додатків і мікросервісів.

**Що таке пакет у Go?**

* **Пакет** (package) — це основна одиниця організації коду у Go. Пакети дозволяють розділити код на окремі модулі для полегшення управління і повторного використання. Наприклад, стандартна бібліотека Go містить пакети для роботи з HTTP, файловою системою та іншими функціями.

**Що таке горутина?**

* **Горутина** (goroutine) — це легкий потік виконання, який управляється середовищем виконання Go. Горутини використовуються для реалізації паралелізму у Go. Вони дозволяють виконувати декілька завдань одночасно, і їх створення дуже дешеве з точки зору ресурсів.

**Як Go обробляє помилки?**

* У Go помилки обробляються за допомогою значень типу **error**. Функції, які можуть повернути помилку, зазвичай повертають значення типу **error** як останній результат. Якщо помилка відбулася, функція повертає ненульове значення error, інакше — nil.

**Як виглядає базова структура проекту Go?**

* Якщо легкий проект можна всі файли можуть бути розміщені в одній директорії.
* Базова структура проекту Go зазвичай виглядає так:
  + cmd/: каталог для команд (програми з точки входу).
  + pkg/: каталог для бібліотек та загальних модулів.
  + internal/: каталог для внутрішніх пакетів, які не можуть бути використані за межами проекту.
  + api/: каталог для описів API та протоколів.
  + config/: каталог для конфігураційних файлів.
  + go.mod і go.sum: файли для управління залежностями.

**Що являє собою вебзастосунок?**

* **Вебзастосунок** (веб-додаток) — це програмний додаток, що працює на сервері і доступний через браузер. Вебзастосунки можуть виконувати різні функції, від простих інформаційних сайтів до складних платформ для взаємодії з користувачем, таких як соціальні мережі, інтернет-магазини або системи управління контентом.

**Що таке вебсервер?**

* **Вебсервер** — це серверне програмне забезпечення (або апаратне забезпечення), яке обробляє HTTP-запити від клієнтів (зазвичай веббраузерів) і повертає їм веб-сторінки або інший контент (зображення, файли). Вебсервер, як-от **IIS**, **Apache** або **NGINX**, отримує запит, виконує відповідну програму (вебзастосунок) і надсилає відповідь клієнту.