

**МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**

**ІМЕНІ ІГОРЯ СІКОРСЬКОГО”**

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп’ютерних систем**

**Лабораторна робота №3**

**з дисципліни “ Основи web-програмування ”**

**тема “РЕАЛІЗАЦІЯ ВЕБ-СЕРВЕРА З ШАБЛОНІЗАТОРОМ”**

|  |  |  |
| --- | --- | --- |
| **Виконав**  **студент II курсу**  **групи КП-83**  **Мортіков Владислав Євгенович** |  | **Перевірив**  **“\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.**  **викладач**  **Гадиняк Руслан Анатолійович**  **(прізвище, ім’я, по батькові)** |

**Київ 2019**

**Мета роботи**

Навчитись створювати простий веб-сервер, що надає доступ до статичних документів.  
Ознайомитись із шаблонізаторами веб-сторінок для генерації контенту.

**Постановка завдання**

### **Завдання**

1. Створити простий веб-сервер за допомогою фреймворку [ExpressJS](https://expressjs.com/).
   1. Налаштувати у сервері шлях для отримання статичних файлів із директорії public (див. Додатки). В результаті, якщо сервер запущений на порті 3000, у браузері має стати доступний файл стилю по шляху http://localhost:3000/stylesheets/style.css та всі інші файли з дерева директорії public.  
      Оновити у HTML-сторінках URL-шляхи до CSS файлів та статичних зображень.
   2. Реалізувати на сервері обробники запитів для отримання відповідних web-документів із директорії views. Наприклад:

GET

/ -> index.html

/users -> users.html

/users/1 -> users/1.html

/{entities} -> {entities}.html

/{entities}/1 -> {entities}/1.html

/about -> about.html

Для включення цієї можливості використайте код налаштування із Додатків

1. Підключити до сервера моделі із [ЛР№2](https://progbase.herokuapp.com/modules/webprogbase/tasks/lab2).
2. JSON API користувачів
   1. Описати обробники GET запитів, що повертають список користувачів або одного користувача за ідентифікатором у форматі JSON (з відповідним Content-type):

/api/users -> [{User}, {User}]

/api/users/:id -> {User}

У випадку помилки (неіснування користувача з таким id) повертати HTTP код статуса 404.

1. Шаблонізація документів
   1. Параметризувати URL шляхи отримання сутностей за ідентифікатором в обробниках сервера:

/users/1, /users/2 -> /users/:id

/{entities}/1, /{entities}/2 -> /{entities}/:id

Якщо ористувач чи сутність з таким id не існує - повертати HTTP код статуса 404.

* 1. Виділити із HTML-сторінок [ЛР№1](https://progbase.herokuapp.com/modules/webprogbase/tasks/lab1) шаблони для вставки у них об’єктів моделей за допомогою обраного шаблонізатора (наприклад, [Mustache](https://github.com/janl/mustache.js)). Повинні з'явитися наступні шаблони:

views/

index.mst

users.mst

user.mst

{entities}.mst

{entity}.mst

about.mst

* 1. Виділити 3 часткові шаблони і помістити їх у views/partials/:
     + head.mst - винести сюди все спільне з тегу head шаблонів
     + header.mst - винести спільну розмітку "шапки" із тегу body шаблонів
     + footer.mst - винести спільну розмітку "підвалу" із тегу body шаблонів
  2. Включити часткові шаблони у всі інші шаблони.
  3. Змінити двигун відображень з html на mustache та видалити лишні HTML-документи.

**Тексти коду програм**

|  |
| --- |
| **app.js** |
| const express = require('express');  const path = require('path');  const User = require("./models/user.js");  const Items = require("./models/item.js");  const mustache = require("mustache-express");  const app = express();  const viewsDir = path.join(\_\_dirname, 'views');  app.engine("mst", mustache(path.join(viewsDir, "partials")));  app.set('views', viewsDir);  app.set('view engine', 'mst');  // will open public/ directory files for http requests  app.use(express.static('public'));  app.get('/', function (req, res) {  res.render('index');  });  app.get('/users', function (req, res) {  let userList = {};  userList.users = User.getAll();  res.render('users', userList);  });  app.get('/users/:id', function (req, res) {  let id = parseInt(req.params.id);  const user = User.getById(id);  if (typeof (user) === "undefined") {  res.sendStatus(404);  }  else {  res.render('user', user);  }  });  app.get('/items', function (req, res) {  let itemsList = {};  itemsList.items = Items.getAll();  res.render('items', itemsList);  });  app.get('/items/:id', function (req, res) {  let id = parseInt(req.params.id);  const item = Items.getById(id);  if (typeof (item) === "undefined") {  res.sendStatus(404);  }  else {  res.render('item', item);  }  });  app.get('/philosophy', function (req, res) {  res.render('philosophy');  });  app.get('/handbook', function (req, res) {  res.render('handbook');  });  app.get('/enrolment', function (req, res) {  res.render('enrolment');  });  app.get('/api/users', function (req, res) {  const users = User.getAll();  res.send(users);  });  app.get("/api/user/:id", function (req, res) {  const self = User.getById(parseInt(req.params.id));  if (typeof (self) === "undefined") {  res.sendStatus(404);  }  else {  res.send(self);  }  });  app.listen(3000, function () {  console.log('Example app listening on port 3000');  }); |

**Приклади результатів**

|  |
| --- |
|  |
|  |
|  |

**Висновки**

Виконавши дану лабораторну роботу ми навчилися створювати простий веб-сервер, що надає доступ до статичних документів.  
Ознайомилися із шаблонізаторами веб-сторінок для генерації контенту.

Компіляція всього коду відбувалася за допомогою node app.js .

Після цього на локальному сервері доступ можна отримати за посиланням <http://localhost:3000/>. Бажано користуватися браузером Google Chrome.