МІНІСТЕРСТВО ОСВІТИ 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 програмування”
тема “**Багатосторінковий веб-сайт**”

Виконав
студент 2 курсу
групи КП-91
Климчук Нікіта Олегович

Перевірив
“14” “травня” 2020р.
викладач
Гадиняк Руслан Анатолійович

Київ 2021

Мета роботи

Навчитись створювати веб-сайт, що надає CRUD доступ до ресурсів веб-сервера.

Ознайомитись із шаблонізаторами веб-сторінок для генерації контенту.

Навчитись створювати HTML-форми для взаємодії користувача із веб-сайтом та обробляти на сервері внесені користувачем дані.

Підготовка

Знати:

1. Шаблонізацію тексту за допомогою модуля mustache.
2. HTML форми, їх призначення, принципи роботи та обмеження.
3. Типи елементів вводу у формах для різних типів даних.
4. Елемент форми для завантаження файлів.

Потрібно:

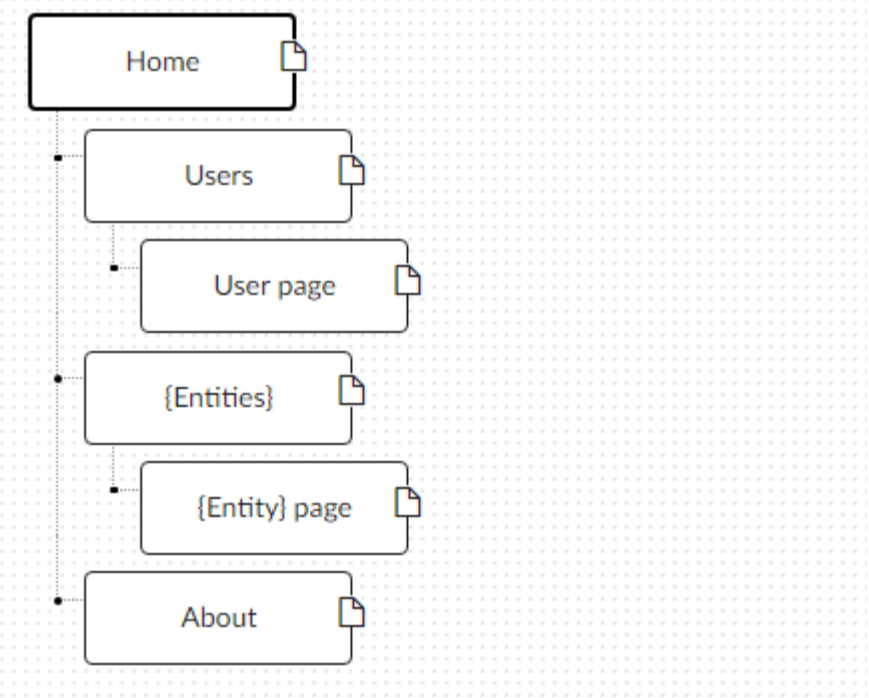
1. Ініціалізація проекту завдання
    1. Створити у навчальному репозиторії директорію проекту labs/lab3 і перейти до неї у редакторі.
    2. Скопіювати у корінь проекту код, налаштування та дані з ЛР№2.
2. Створити у корені проекту директорії public і views.
3. Встановити:
    1. Локальні пакети проекту: npm i consolidate swig mustache busboy-body-parser

Замітки:

1. Можна обрати інший шаблонізатор (*template engine*) замість **Mustache**.

Завдання

Реалізувати багатосторінковий веб-сайт із заданою структурою розділів:



Забезпечити можливість пошуку, пагінації, додавання та видалення сутностей за варіантом.

Вказівки до виконання

Якщо ви будете використовувати nodemon, додайте у корінь вашого проекту файл nodemon.json для ігнорування змін у директорії data:

```
{
  "verbose": true,
  "ignore": ["data/*"]
}
```

Підключіть до сервера модуль morgan для відслідковування всіх запитів:

```
const app = express();
//...
const morgan = require('morgan');
app.use(morgan('dev'));
```

Можливі помилки:

- Сервер не повинен виконувати запити сам до себе.
- У коді і розмітці не має бути захардкоджених localhost (хостнейм) та 3000 (номер порта) окрім одного місця в app.js (або в налаштуваннях). Сайт має коректно працювати після зміни номера порта на будь-який інший. Для цього всі URL мають починатись на / або бути відносними.

- У коді обробників/контроллерах не повинно бути конкатенації рядків з HTML розміткою. Вся розмітка розміщується в HTML файлах та/чи шаблонах. Шаблону можна передати булеві змінні чи інші дані, на основі яких шаблонними конструкціями змінювати вид згенерованої сторінки.

При винекненні проблем використовуйте Debug в VSCode та вкладку Networking в Developer Tools вашого браузера

Частина 1. HTML та CSS

1. Всі HTML файли розміщувати у директорії views.
2. Реалізувати основні розділи сайту:
    1. **Home**. Додати на головну сторінку (index.html) назву сайту і додаткову інформацію про сайт і його призначення (параграф опису).
    2. **Users**. Сторінки користувачів:
        1. Створити файл users.html зі списком користувачів.
        2. Створити файли для одного користувача (наприклад, user.html). Додати на сторінку інформацію про будь-якого користувача:
            - Логін.
            - Повне ім'я.
            - Дату реєстрації на сайті.
            - Параграф із біографією.
        3. Зв'язати сторінки так, щоби з index можна було перейти до users, а з users до сторінки з користувача (user.html).
    3. **{Entities}** (**назву замінити**). Сторінки сутностей за варіантом:
        1. Створити файл зі списом сутностей (наприклад, {entities}.html) кожен елемент списку - посилання на сутність.
        2. Створити файли для однієї сутності (наприклад, {entity}.html). У файлі розмістити детальну інформацію про будь-яку сутність.
        3. Зв'язати сторінки так, щоби з index можна було перейти до списку сутностей, а зі списку сутностей до конкретної сутності.
    4. **About**. Створити сторінку (about.html) із додатковим описом сайту і інформацією про автора сайту.

3. Додаткові зміни:
    1. Меню сайту. Створити список посилань на основні розділи сайту (Home, Users, {Entities} - **замінити на назву ваших сутностей**, About) і додати його до всіх створених сторінок.
    2. Зображення:
        1. Створити у public/ директорію images/ і додати у неї мінімум 3 зображення довільних форматів (наприклад, images/logo.png, images/user.png, images/{entity}.png). Також можна додати декілька зображень для сутностей.
        2. Додати зображення за допомогою тегів img на відповідні сторінки (index, user, {entity}).
    3. Стилізація сайту:
        1. Створити у public/ директорію stylesheets/, додати у неї файл з CSS-стилем (stylesheets/style.css) та підключити цей стиль до всіх створених веб-сторінок.
        2. Оформити стиль веб-сторінок за вимогами.
4. Відкрити файл views/index.html у будь-якому веб-браузері та перевірити показ веб-сторінок та навігації по статичному сайту.

Частина 2. Шаблонізація веб-сторінок

1. Використати ExpressJS веб-сервер із попереднього завдання.
    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 директорії public.
    2. Оновити у HTML-сторінках URL-шляхи до CSS файлів та статичних зображень.
    3. Реалізувати на сервері обробники запитів (роутери і контроллери) для отримання відповідних веб-документів із директорії views. Наприклад:
    4. GET / -> views/index.html
    5. GET /users -> views/users.html
    6. GET /users/:id -> views/user.html
    7. GET /{entities} -> views/{entities}.html
    8. GET /{entities}/:id -> views/{entity}.html
    9. GET /about -> views/about.html

Для включення цієї можливості використайте consolidate і swig та код налаштування із Додатків (Використання HTML view).

2. Шаблонізація
    1. Виділити із HTML-сторінок шаблони для вставки у них об’єктів моделей за допомогою обраного шаблонізатора (наприклад, Mustache). Повинні з'явитися наступні шаблони:

```
views/
    index.mst
    users.mst
    user.mst
    {entities}.mst
    {entity}.mst
    about.mst
```

    9. Видалити HTML файли.
    10. 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
    11. Включити часткові шаблони у всі інші шаблони.

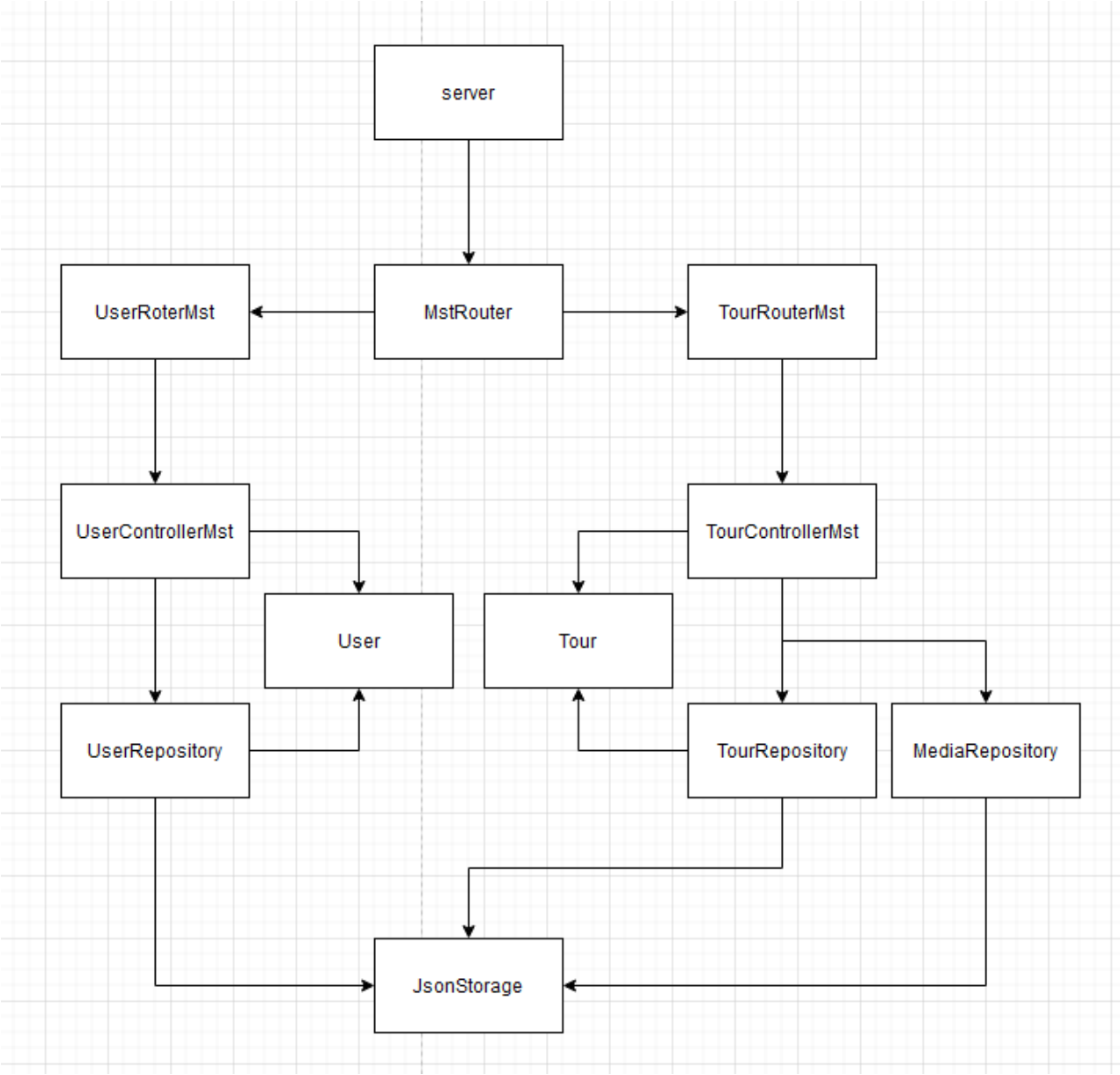
Частина 3. HTML-форми і CRUD операції для сутностей

1. Пошук сутностей по назві
    1. Модифікувати веб-сторінку /{entities} з сутностями так, щоби можна було шукати сутності по частковому співпадінню рядка назви за допомогою HTML форми із GET запитом.
    2. Сторінка має містити спеціальні компоненти для інформування користувача про те, для якого пошукового запиту відображаються результати.
    3. Додати пагінацію результатів. Показувати користувачу номер поточної сторінки та загальну кількість сторінок із посиланнями для переходу на попередню/наступну/довільну сторінку.
    4. Якщо в результаті пошуку не було знайдено жодних об'єктів, додавати на сторінку спеціальне повідомлення про це.
2. Створення нових сутностей

1. Створити сторінку /{entities}/new і додати на неї HTML форму для створення нової сутності (за варіантом) за допомогою POST запиту.
2. Дана форма обов’язково повинна мати одне поле для завантаження на сервер файлу (зображення чи ін.), що пов'язаний із типом об'єкта.
3. Завантажені файли розміщувати у директорії data/media/. Додати у веб-сервер обробник GET запитів на отримання файлів із data/media/. У поля сутностей зберігати URL для отримання цих файлів через GET запити до веб-сервера.
4. Після створення нової сутності і присвоєння їй ідентифікатора, перенаправити на сторінку нової сутності (/{entities}/:id).

3. Видалення сутностей
    1. На сторінці перегляду інформації про окрему сутність (/{entities}/:id) додати форму (яка виглядатиме як кнопка, посилання чи зображення) для видалення поточної сутності за допомогою POST запиту.

Діаграма залежностей модулів



Код всіх модулів

server.js

```
'use strict';
const bodyParser = require('body-parser');
const busboyBodyParser = require('busboy-body-parser');
const morgan = require('morgan');
const consolidate = require('consolidate');
const express = require('express');
const app = express();
const apiRouter = require('./Routes/ApiRouter');
const mstRouter = require('./Routes/MstRouter');
const mustache_ex = require('mustache-express');
const path = require('path');

app.use(bodyParser.urlencoded({ extended: true }))
app.use(bodyParser.json());
app.use(busboyBodyParser());

app.use('/api', apiRouter);
app.use('', mstRouter);
app.use(express.static("./public"));
app.use(express.static("./Data"));
app.use(morgan('dev'));

const expressSwaggerGenerator = require('express-swagger-generator');
const expressSwagger = expressSwaggerGenerator(app);

const options = {
    swaggerDefinition: {
        info: {
            description: 'description',
            title: 'title',
            version: '1.0.0',
        },
        host: 'localhost:3000',
        produces: ["application/json"],
    },
    basedir: __dirname,
    files: ['./routes/**/*.js', './models/**/*.js'],
};
expressSwagger(options);

const viewsDir = path.join(__dirname, 'views');
app.engine("mst", mustache_ex(path.join(viewsDir, "partials")));
app.set('views', viewsDir);
app.set('view engine', 'mst');



app.listen(3000, () => {

});
```

MstRouter.js

```
const userRouter = require('./UserRouterMst');
const tourRouter = require('./TourRouterMst');
const Router = require('express').Router();

Router.use('/users', userRouter);
Router.use('/tours', tourRouter);
Router.get('/', function (req, res) {
    res.render('index', { index_current: 'current', home_link: 'disabled_link' });
});

Router.get('/about', function (req, res) {
    res.render('about', { about_current: 'current', about_link: 'disabled_link' });
});
```

```
module.exports = Router;
```

UserRouterMst.js

```
const Router = require('express').Router();

const userController = require('../Controllers/UserControllerMst');

Router
    .get("/:id", userController.GetUserById)
    .get("/", userController.GetUsers);


module.exports = Router;
```

UserControllerMst.js

```
const path = require('path');
const User = require('../Models/User');
const userRepository = require(path.resolve(__dirname, '../Repositories/UserRepository'));
const userRepo = new userRepository(path.resolve(__dirname, '../Data/Users.json'));


function pagination(items, page, per_page) {
    const startInd = (page - 1) * per_page;
    const endInd = page * per_page;
    return (items.slice(startInd, endInd));

}

function ToUserFriendlyDate(Users) {
    Users.forEach(element => {
        date = new Date(element.registeredAt);
        element.registeredAt = date.toUTCString().replace('GMT', '');
    });
    return Users;

}

module.exports = {
    GetUsers(req, res) {
        let page, per_page;
        if (req.query.per_page && isNaN(req.query.per_page)) {
            res.status(400).send({ mess: 'per_page and page should be a number' });
            return;
        }

        if (req.query.page && isNaN(req.query.page)) {
            res.status(400).send({ mess: 'page and per_page should be a number' });
            return;
        }
        if (!req.query.per_page) {
            per_page = 4;
        } else if (parseInt(req.query.per_page) > 10 || parseInt(req.query.per_page) < 1) {
            res.status(400).send({ mess: 'per_page should`n be more than 10 and less then 1' });
            return;
        } else {
            per_page = parseInt(req.query.per_page);
        }

        if (!req.query.page) {
            page = 1;
        } else if (parseInt(req.query.page) < 0) {
            res.status(400).send({ mess: 'page should`n be less then 0' });
            return;
        } else {
            page = parseInt(req.query.page);
        }

        const Users = userRepo.getUsers();
        const Pag_Users = pagination(Users, page, per_page);
        const User_friendly = ToUserFriendlyDate(Pag_Users);
```

```
        res.status(200).render('users', { users: User_friendly, user_current: 'current',
user_link: 'disabled_link' });
    },

    GetUserById(req, res) {
        if (isNaN(req.params.id)) {
            res.status(400).send({ mess: 'Wrong id' })
            return;
        }
        const User = userRepo.getUserById(parseInt(req.params.id));
        if (User == null) {
            res.status(404).send({ mess: 'No user with such id' });
            return;
        }
        date = new Date(User.registeredAt);
        User.registeredAt = date.toUTCString().replace('GMT', '');

        res.status(200).render('user', { user: User, user_current: 'current' });
    },
};
```

TourRouterMst.js

```
const express = require('express');
const tourController = require('./../Controllers/TourControllerMst');

const Router = express.Router();

Router
    .get('/', tourController.GetTours)
    .get("/new", tourController.NewTour)
    .get('/:id', tourController.GetTourHandler, tourController.GetTourById)
    .post('/', tourController.AddTour)
    .post('/:id', tourController.GetTourHandler, tourController.DeleteTourById);
module.exports = Router;
```

TourControllerMst.js

```
const path = require('path');
const fs = require('fs');
const TourRepository = require('./../Repositories/TourRepository');
const tourRepo = new TourRepository('./Data/Tours.json');
const tourProperty = Symbol('tour');

const MediaRepository = require(path.resolve(__dirname, '../Repositories/MediaRepository'));
const mediaRepo = new MediaRepository(path.resolve(__dirname, '../Data/Media'));
const Media = require('../Models/Media');


module.exports = {
    GetTours(req, res) {
        let page;

        if (req.query.page && isNaN(req.query.page)) {
            res.status(400).send({ mess: 'page and per_page should be a number' });
            return;
        }

        if (!req.query.page) {
            page = 1;
        } else {
            page = parseInt(req.query.page);
        }

        const per_page = 4;
        let class_prev;
        let class_next;

        if (page > 1) {
            class_prev = "";
        } else {
            class_prev = "disabled_link";
        }
```

```
        let Tours;
        let name = "";
        if (req.query.name) {
            Tours = tourRepo.GetToursByName(req.query.name);
            name = req.query.name;
        } else {
            Tours = tourRepo.GetTours();
        }
        if (page * per_page < Tours.length) {
            class_next = "";
        } else {
            class_next = "disabled_link";
        }


        let max_page = Math.ceil(parseInt(Tours.length) / (per_page));
        if (max_page == 0) {
            max_page = 1;
        }
        res.render('tours',
            {
                tours: Tours.slice((page - 1) * per_page, page * per_page),
                search_name: name, tour_current: 'current', next_page: page + 1,
                previous_page: page - 1, page, class_prev, class_next, max_page, tour_link:
'disabled_link'
            });
    },
    GetTourById(req, res, next) {
        res.render('tour', { tour: req[tourProperty], tour_current: 'current' });
    },
    NewTour(req, res) {
        res.render('add_tour');
    },
    DeleteTourById(req, res, next) {
        tourRepo.DeleteTour(req[tourProperty].id);
        res.redirect('/tours');
    },
    AddTour(req, res, next) {
        let media_path = "";
        if (req.files['photo']) {
            const media_id = mediaRepo.AddMedia(req.files['photo']);
            media_path = mediaRepo.GetMediaPathById(media_id);
        }
        req.body.media = media_path;
        const id = tourRepo.AddTour(req.body);
        res.status(201);
        res.redirect('/tours/' + id);
    },
    GetTourHandler(req, res, next) {
        const tour = tourRepo.GetTourById(parseInt(req.params.id));
        if (tour) {
            req[tourProperty] = tour;
            next();
        }
        else {
            res.sendStatus(404);
        }
    }
};
```

about.mst

```
<!DOCTYPE html>
<html>
  {{>head}}
  <body>
    {{>header}}
    <h1>About</h1>
    <p><b>Name</b>: Klimchuk Nikita</p>
    <p><b>E-mail</b>: klimchuk_nik@gmail.com</p>
    <p><b>About</b>: You can use our site for searching, adding and deleting tours</p>
    {{>footer}}
  </body>
</html>
```

add_tour.mst

```
<!DOCTYPE html>
<html>
  {{>head}}
  <body>
    {{>header}}
    <form action="/tours/" method="POST" enctype="multipart/form-data" class="add_tour">
    <div class="main_inputs">
    Name:<br>
    <input type="text" name="name" required><br>
    Country:<br>
    <input type="text" name="country" required><br>
    Price:<br>
    <input type="number" name="price" min=0 required><br>
    Max tourists:<br>
    <input type="number" name="maxTouristsCount" min=1 required><br>
    Start date:<br>
    <input class="date" type="date" name="startDate" max=2030 min=0 required><br>
    <input class="photo" type="file" name="photo" value="Add file"><br>
    </div>
    <div class="add_tour_button">
    <input class="button create" type="submit" value="Create">
    </div>
    </form>
    {{>footer}}
  </body>
</html>
```

tour.mst

```
<!DOCTYPE html>
<html>
  {{>head}}
  <body>
    {{>header}}
    <div class="tour_inf">
    <div class="tour_img">
    <img class="tour_images" src="{{tour.media.path}}" alt="*Tour picture">
    </div>
    <div class="info">
    <h1>Tour</h1>
    <div class="add">
    <p>Name:<br>
    <em>{{tour.name}}</em></p>
    <p>Country:<br>
    <em>{{tour.country}}</em></p>
    <p>Price:<br>
    <em>{{tour.price}}</em></p>
    <p>Max tourists:<br>
    <em>{{tour.maxTouristsCount}}</em></p>
    <p>Start date:<br>
    <em>{{tour.startDate}}</em></p>
    </div>
    </div>
    </div>
    <form action="/tours/{{tour.id}}" method="POST" >
    <input class="button delete" type="submit" value="Delete">
    </form>
    {{>footer}}
  </body>
</html>
```

tours.mst

```
<!DOCTYPE html>
<html>
  {{>head}}
  <body>
    {{>header}}
    <h1>Tours</h1>
    <form action="/tours?name={{search_name}}&page={{page}}" method="GET">
```

```
    <div>
    Name:<br>
    <input type="text" name="name" value={{search_name}}><br>
    <input class="button find" type="submit" value="Find">

    <div class="search">
    <p>Search for:</p>
    <p class="search_name">{{search_name}}</p>
    </div>
    <div class="items_table">
    <table border="1">
    <tr>
    <th>NAME</th>
    <th>COUNTRY</th>
        </tr>
        {{#tours}}
        <tr>
          <td><a href="/tours/{{id}}">{{name}}</a></td>
          <td>{{country}}</td>
        </tr>
        {{/tours}}
        {{^tours}}
          <tr><td colspan=3 style="text-align: center;">There no tours with such name</td></tr>
        {{/tours}}
    </table>
    </div>
    <input class="button Goto" type="submit" value="Go to">
    <input type="number" name="page" min=1 max={{max_page}}><br>
    </div>

    </form>

    <div class="pagination">
      <a href="/tours?page={{previous_page}}&name={{search_name}}" class={{class_prev}}>Previ-
ous</a>
      <a class="page">{{page}}/{{max_page}}</a>
      <a href="/tours?page={{next_page}}&name={{search_name}}" class={{class_next}}>Next</a>
    </div>

    <form action="/tours/new" method="GET">
    <input class="button create" type="submit" value="Create">
    </form>
    {{>footer}}
  </body>
</html>
```

index.mst

```
<!DOCTYPE html>
<html>
  {{>head}}
  <body>
    {{>header}}
    <div class="home_page">
    <h1>Travel gid</h1>
    <p class="home_text">Travel with us</p>
    <div class="home_images">
    <img src="/images/home_pic.jpg" alt="*home img"/>
    </div>
    </div>
    {{>footer}}
  </body>
</html>
```

user.mst

```
<!DOCTYPE html>
<html>
  {{>head}}
  <body>
    {{>header}}
    <h1>User</h1>
    <div class="user">
    <div class="user_img">
```

```
    <img class="user_images" src="{{user.media.path}}" alt="User picture">
    </div>
    <div class="user_info">
    <p>{{user.fullname}}</p>
    <p>({{user.login}}) {{user.registeredAt}}</p>
    <p class="Bio"> {{user.Bio}}</p>
    </div>
    </div>
    {{>footer}}
  </body>
</html>
```

users.mst

```
<!DOCTYPE html>
<html>
  {{>head}}
  <body>
    {{>header}}
    <h1>Users</h1>
    <div class="items_table">
    <table border="1">
      <tr>
      <th>login</th>
      <th>fullname</th>
      <th>registered_at</th>
      </tr>
      <tr>
      {{#users}}
      <tr><td><a href="/users/{{id}}">{{login}}</a></td><td><a href="/us-
ers/{{id}}">{{fullname}}</a></td><td>{{registeredAt}}</td></tr>
      {{/users}}
      </tr>
    </table>
    </div>
    {{>footer}}
  </body>
</html>
```

footer.mst

```
<div class="footer"><hr>Tours 2021</div>
```

head.mst

```
<head>
  <title>Tours</title>
  <link rel="stylesheet" href="/stylesheets/style.css">
</head>
```

header.mst

```
<div class="header">
  <img class="logo" src="/images/logo1.png" alt="logo">
  <div class="menu">
    <nav>
    <ul>
    <li> <a href="/" class="{{home_link}}" class="{{index_current}}" > Home </a></li>
    <li> <a href="/users" class="{{user_link}}" class="{{user_current}}"> Users </a></li>
    <li> <a href="/tours" class="{{tour_link}}" class="{{tour_current}}"> Tours </a></li>
    <li> <a href="/about" class="{{about_link}}" class="{{about_current}}"> About </a></li>
    </ul>
    </nav>
  </div>
</div>
<hr>
```

Зображення всіх розроблених сторінок

Home Users Tours About

Travel gid

Travel with us

Tours 2021

Home Users Tours About

Users

Login	Fullname	Registered at
admin	Admin Admin	Thu, 01 Jan 1970 00:00:00
testUser	Test User	Thu, 01 Jan 1970 00:00:00

Tours 2021

User

Admin Admin

(admin) Thu, 01 Jan 1970 00:00:00

Bio1

Tours

Name:

Find

Search for:

Name	Country
Name	Country
Name1	Country
Name	Country
Name7	Country

Go to

Previous 1/2 Next

Create

Home Users Tours About

Tours

Name:
1

Find

Search for: 1

Name	Country
Name1	Country
Name1	Country

Go to

Previous 1/1 Next

Create

Tours 2021

Home Users Tours About

Tours

Name:
11

Find

Search for: 11

Name	Country
There no tours	

Go to

Previous 1/1 Next

Create

Tours 2021

Name:

Country:

Price:

Max tourists:

Start date:

dd / mm / yyyy

Обзор... Файл не выбран.

Create

Tour

Name:
Name

Country:
Country

Price:
1000

Max tourists:
12

Start date:
2021-05-06

Delete

About

Name: Klimchuk Nikita

E-mail: klimchuk_nik@gmail.com

About: You can use our site for searching, adding and deleting tours

Висновки

Ми навчилися створювати веб-сайт, що надає CRUD доступ до ресурсів веб-сервера.

Ознайомились із шаблонізаторами веб-сторінок для генерації контенту. Навчились створювати HTML-форми для взаємодії користувача із веб-сайтом та обробляти на сервері внесені користувачем дані.

Був налаштований express-swagger-generator, body-parser, busboy-body-parser. Також використовувався nodemon.