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 № 7**

з дисципліни “ Основи web-програмування ”

тема “ Реєстрація та авторизація корстувачів ”

|  |  |  |
| --- | --- | --- |
| Виконав(ла)  студент(ка) II курсу  групи КП-82  Гашинський Василь Юрійович  (*прізвище, ім’я, по батькові*)  варіант № 3 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

Київ 2018

**Мета роботи**

Реалізувати персоналізацію вмісту сайту за допомогою користувачів і авторизації із використанням сесій.

**Постановка завдання**

### Підготовка

Знати:

1. HTML форми, їх призначення, принципи роботи та обмеження.
2. Типи елементів вводу у формах для різних типів даних.
3. Елемент форми для завантаження файлів.
4. Асинхронне виконання JS коду за допомогою callback-функцій.
5. Повернення даних та помилок із callback-функцій.

Потрібно:

1. Створити у навчальному репозиторії директорію проекту: labs/lab4 і перейти до неї у редакторі.
2. Скопіювати у корінь проект лабораторної роботи [ЛР№3](http://progbase.herokuapp.com/modules/webprogbase/labs/lab3) для продовження роботи над ним.
3. Ознайомитись із [асинхронним програмуванням за допомогою callback-функцій](http://progbase.herokuapp.com/posts/2/asinkhronnist-u-javascript).

### Попереднє завдання

1. Додати до скопійованого проекту веб-сайту нову веб-сторінку /{entities}/new із базовим HTML5 шаблоном.
2. Додати на сторінку /{entities} пагінацію списку основних сутностей (див. Додатки).
3. Переписати статичні синхронні функції моделей на [асинхронні аналоги з callback-функціями](https://docs.google.com/document/d/17d93c-kxsFSJgKXFP1tK210ijlQ_vg9dZ60VNlfqB18/edit" \l "heading=h.a1uny8dlwyp), які використовують стандартні асинхронні функції для роботи з файловою системою.

### Завдання

1. Пошук сутностей по назві
   1. Модифікувати веб-сторінку /{entities} з сутностями так, щоби можна було шукати сутності по частковому співпадіння строки назви за допомогою форми із GET запитом.
   2. Сторінка має містити спеціальні компоненти для інформування користувача про те, для якого пошукового запиту відображаються результати.
   3. Якщо в результаті пошуку не було знайдено жодних об'єктів, додавати на сторінку спеціальне повідомлення про це.
2. Створення нових сутностей
   1. Додати на сторінку /{entities}/new 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/fs/. Додати у веб-сервер обробник GET запитів на отримання файлів із data/fs/. 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.

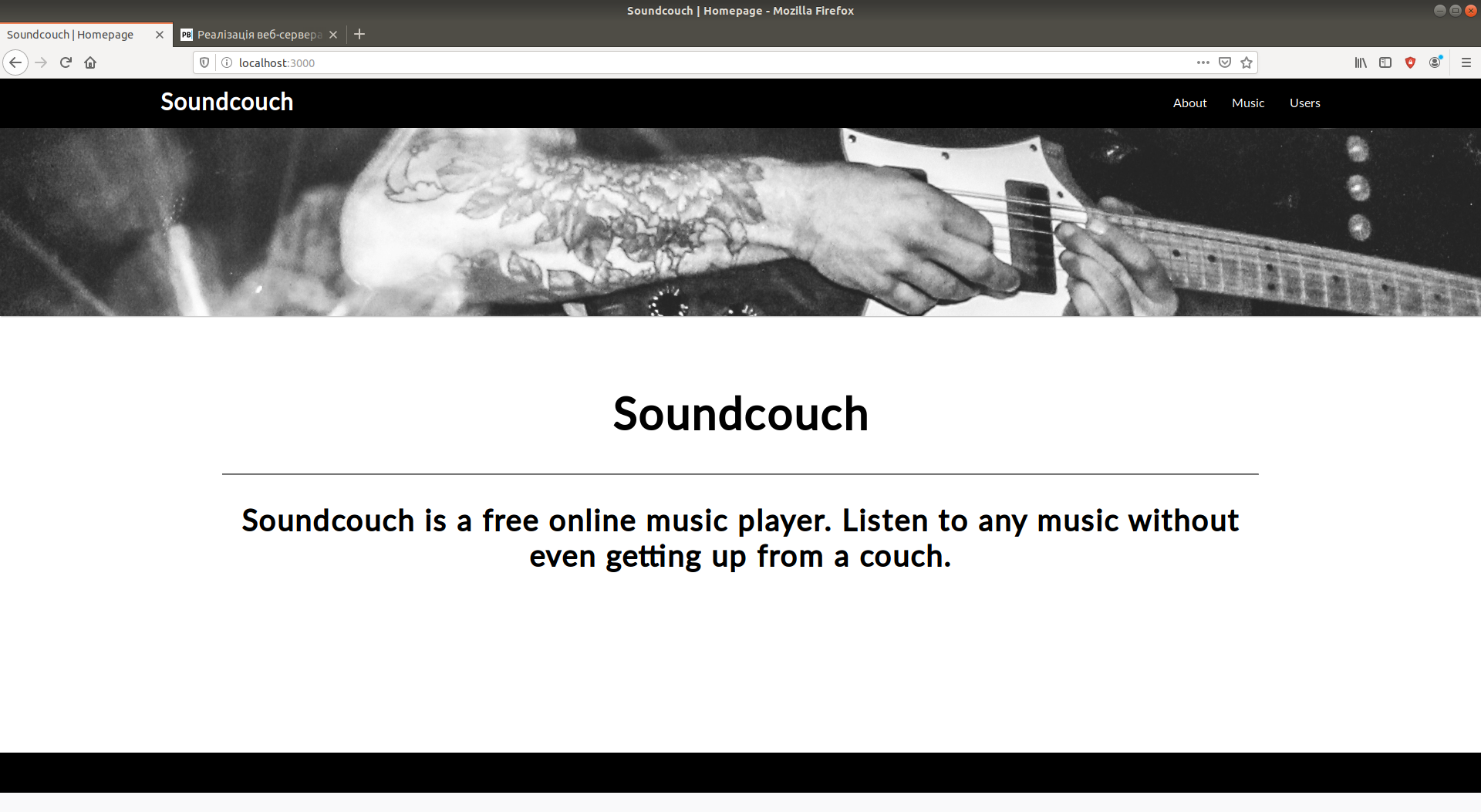
**Тексти коду програм**

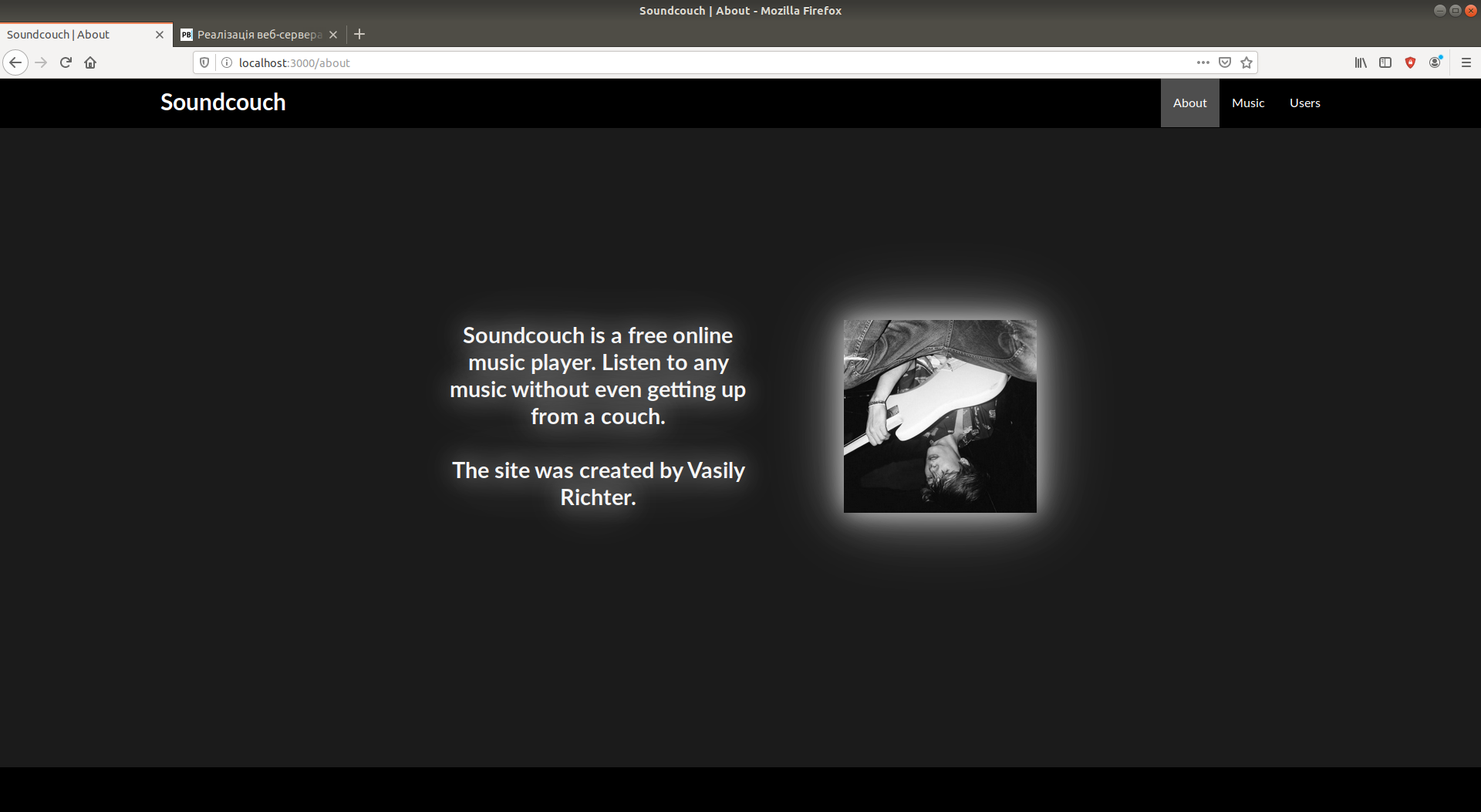
|  |
| --- |
| **app.js** |
| var createError = require('http-errors');  var express = require('express');  var path = require('path');  const consolidate = require('consolidate');  var cookieParser = require('cookie-parser');  var logger = require('morgan');  const bodyParser = require('busboy-body-parser');  var mustacheExpress = require('mustache-express');  var app = express();  app.engine('mst', mustacheExpress());  app.set('view engine', 'mst');  app.engine('mst', mustacheExpress(path.join(\_\_dirname, 'views/partials')));  app.set('views', path.join(\_\_dirname, 'views'));  app.use(bodyParser());  var indexRouter = require('./routes/index');  var usersRouter = require('./routes/users');  var aboutRouter = require('./routes/about');  var musicRouter = require('./routes/music');  var errorRouter = require('./routes/error');  var api = require('./routes/api');  var music = require('./models/music');  var user = require('./models/user');    *// view engine setup*  app.use(express.static('public'));  app.use(express.static('models'));  app.use(express.static('data'));  app.use(logger('dev'));  app.use(express.json());  app.use(express.urlencoded({ extended: false }));  app.use(cookieParser());  app.use('/', indexRouter);  app.use('/about', aboutRouter);  app.use('/users', usersRouter);  app.use('/music', musicRouter);  app.use('/api', api);  app.use('/error', errorRouter);  *// catch 404 and forward to error handler*  app.use(function(*req*, *res*, *next*) {  next(createError(404));  });  app.get('/data/fs/:path', function(*req*, *res*) {  res.sendFile(path.join(\_\_dirname, './data/fs/' + req.params.path))  });  *// error handler*  app.use(function(*err*, *req*, *res*, *next*) {  *// set locals, only providing error in development*  res.locals.message = err.message;  res.locals.error = req.app.get('env') === 'development' ? err : {};  *// render the error page*  res.status(err.status || 500);  console.log(err.message)  });  module.exports = app; |

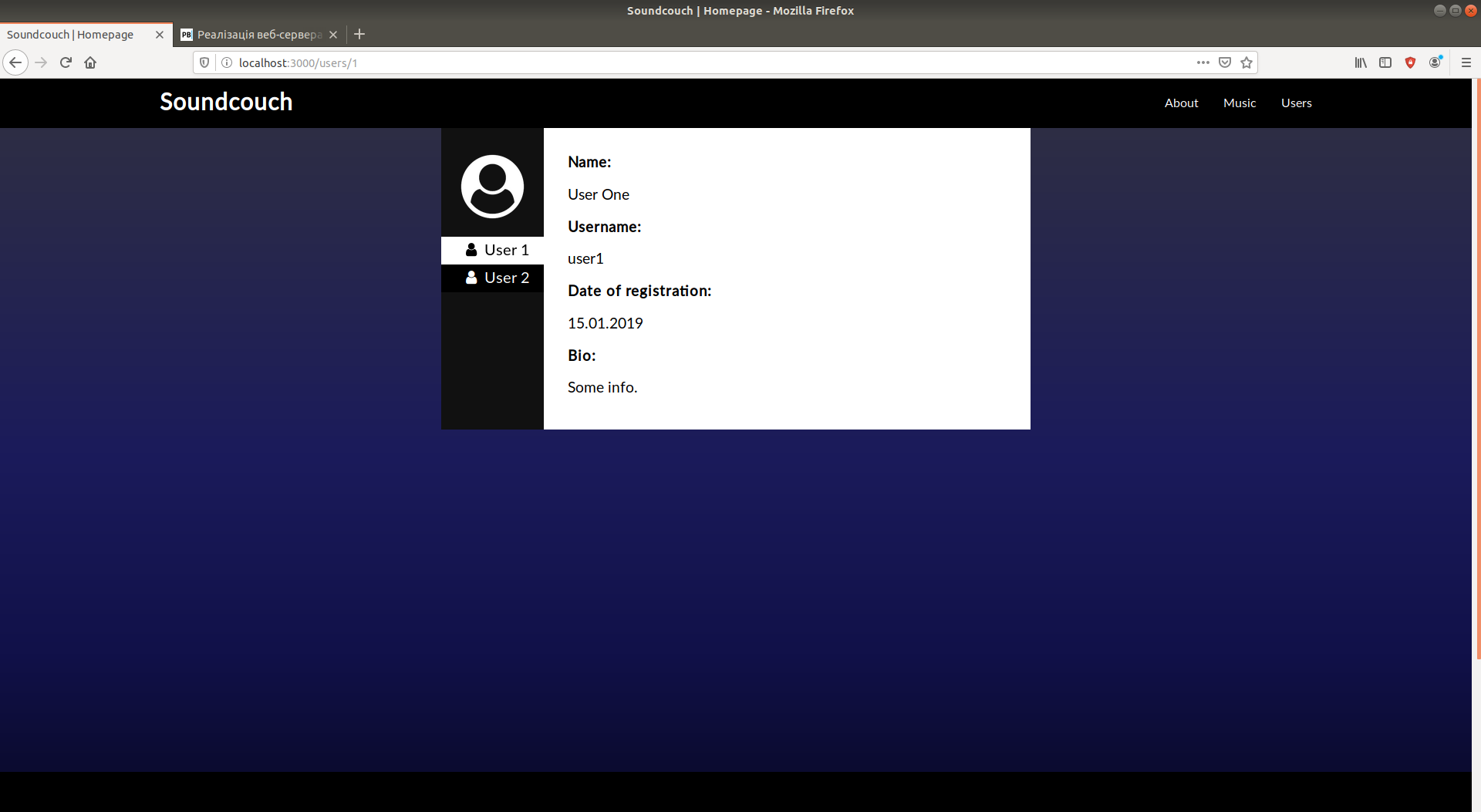
|  |
| --- |
| **music.js** |
| module.exports = {};  const fs = require('fs');  let file = './data/music.json';  let raw = fs.readFileSync('./data/music.json');  var music = JSON.parse(raw);  class Music {  constructor(*id*, *name*, *timeAdded*, *songUrl*, *yearPublished*, *code*) {  this.id = id; *// number*  this.name = name; *// string*  this.timeAdded = timeAdded; *// ISO 8601*  this.songUrl = songUrl; *// string*  this.yearPublished = yearPublished; *// number*  this.code = code; *// some kind of a universal song code like*  }  static insert(*obj*) {  // console.log(music.nextId);  obj.id = music.nextId;  // console.log("run1");  // console.log("run2");  music.nextId++;  // console.log("run3");  // music.items.push(obj);  // music.nextId++;  // fs.writeFileSync(file, JSON.stringify(music, null, 4));  // return obj.id;  // console.log("run4");  // fs.writeFileSync(file, json);  music.items.push(obj);  let json = JSON.stringify(music, null, 4);  fs.writeFile(file, json, (*err*) => {  if (err) throw err;  console.log('The file has been saved!');  });  // console.log("run5");  // return obj.id;  // console.log(obj);  // console.log("\n");  // console.log(music);  }  // static functions to access storage  static deleteById(*id*, *callback*) {  this.getNextId(function(*nextId*) {  if (nextId === undefined)  callback("Internal error, can't get next id.");  else if (id < nextId) {  fs.readFile(file, 'utf-8', function(*err*, *data*) {  if (!err) {  let musicData = JSON.parse(data);  for (let i = 0; i < musicData.items.length; i++)  if (musicData.items[i]["id"] === id) {  let link = musicData.items[i]["themeUrl"];  musicData.items.splice(i, 1);  fs.writeFile(file, JSON.stringify(musicData, null, 4), function(*err*) {  if (!err) {  if (link != "")  fs.unlink(link, function(*err*) {  if (err)  callback(err);  else  callback(undefined);  });  else  callback(undefined);  } else  callback(err);  });  }  } else {  console.log("Delete music by id, Can`t read file");  callback(err);  }  });  } else {  callback("Wrong id!");  }  });  }  static getNextId(*callback*) {  fs.readFile(file, 'utf-8', function(*err*, *data*) {  if (!err) {  callback(JSON.parse(data).nextId);  } else {  console.log("Error:" + err);  callback(undefined);  }  });  }  static getName(*step*, *stepNumber*, *requestedSubString*, *callback*) { *////// rewrite async*  console.log("~~~~~~~~~~~~~NEW SESSION~~~~~~~~~~~~~~~");  fs.readFile(file, "utf-8", function(*err*, *data*) {  if (!err) {  let mItems = JSON.parse(data).items;  let reqResult = [];  let itemcount = mItems.length;  console.log(itemcount);  if (requestedSubString == "") {  let count = 0;  for (let i = 0; i < itemcount; i++) {  if (mItems[i].isDisabled == false) {  reqResult[count] = mItems[i];  count++;  }  }  console.log("\n=======REQRESEMPTY=====\n");  console.log(reqResult);  } else {  for (let item of mItems)  if (item.name.toLowerCase().includes(requestedSubString.toLowerCase()))  reqResult.push(item);  console.log("\n======REQRESFULL======\n");  console.log(reqResult);  }  let resData = {  items: [],  itemsCount: reqResult.length  }  // if (requestedSubString == "") {  // for (let i = stepNumber \* step; i < step && i < step; i++)  for (let i = stepNumber \* step; i < step \* (stepNumber + 1) && i < reqResult.length; i++)  resData.items.push(reqResult[i]);  // console.log("\n======RESDATAEMPTY======\n");  // console.log(stepNumber);  // console.log(resData);  // if (requestedSubString != "" && step != 0) {  // for (let i = stepNumber \* step; i < step \* (stepNumber + 1) && i < reqResult.length; i++)  // resData.items.push(reqResult[i]);  // console.log("\n======RESDATAEMPTY======\n");  // console.log(resData);    callback(undefined, resData);  } else  callback(err, undefined);  });  }  static update(*x*) {  for (let i = 0; i < music.items.length; i++) {  if (music.items[i].id === x.id) {  music.items[i] = x;  break;  }  }  fs.writeFileSync(file, JSON.stringify(music, null, 4));  }  // returns music with id or undefined  static getById(*id*) {  for (let Music of music.items) {  if (Music.id === id) {  return Music;  }  }  }  static getLast() {  for (let Music of music.items) {  if (Music.id === (music.nextId - 1)) {  return Music;  }  }  }  // returns an array of all users in storage  static getAll() {  return music;  }  static deleteById(*id*) {  let object;  for (let i = 0; i < music.items.length; i++) {  if (music.items[i].id === id) {  object = music.items[i];  music.items.splice(i, 1);  break;  }  }  fs.writeFileSync(file, JSON.stringify(music, null, 4));  return object;  }  };  module.exports = Music; |

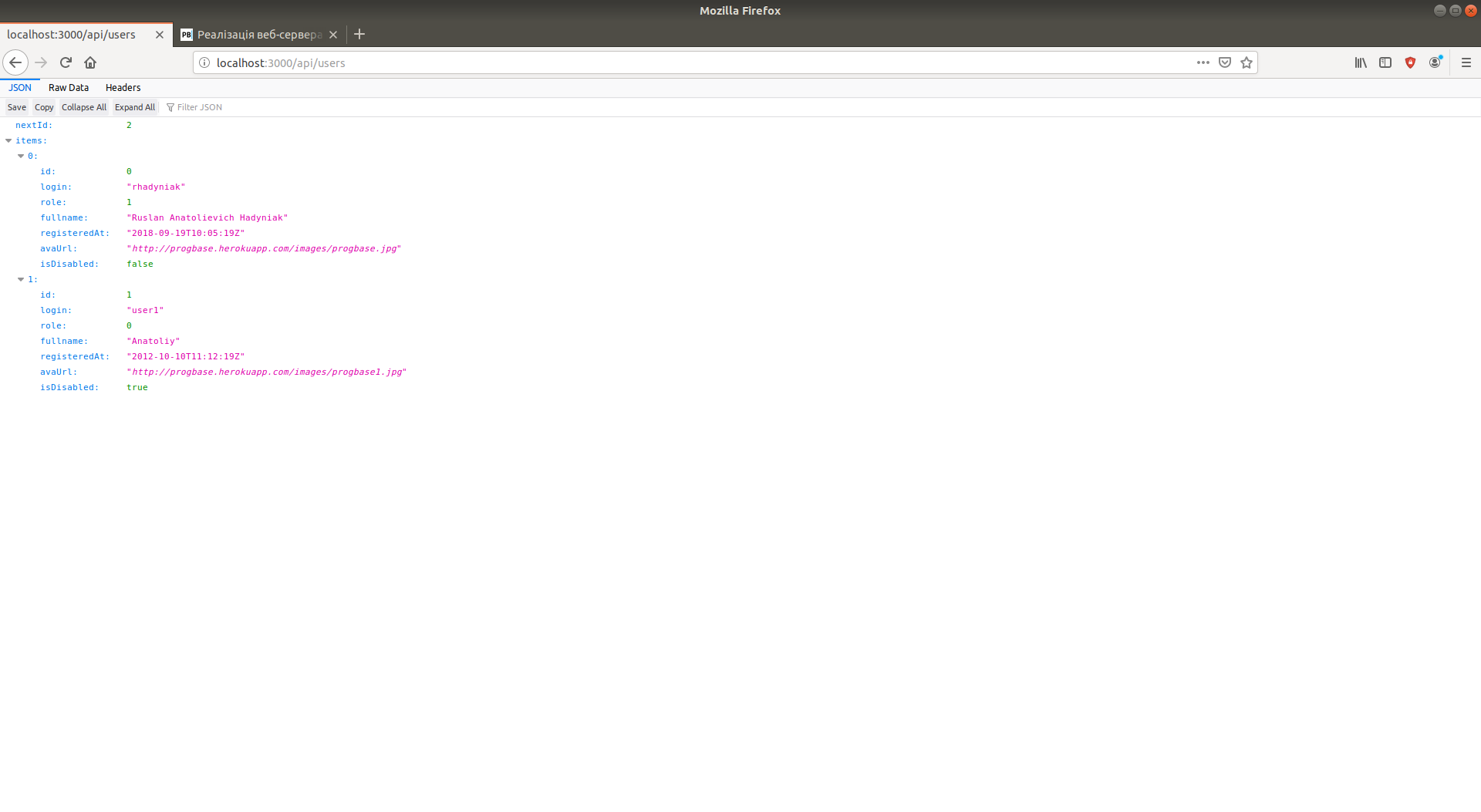
|  |
| --- |
| **user.js** |
| module.exports = {};  const fs = require('fs');  const file = 'data/user.json';  const user = require('../data/users.json');  class User {  constructor(*id*, *login*, *role*, *fullname*, *registredAt*, *avaUrl*, *isDisabled*) {  this.id = id; *// number*  this.login = login; *// string*  this.role = role; *// number 0 - regular, 1 - admin*  this.fullname = fullname; *// string*  this.registredAt = registredAt; *// string ISO 8601*  this.avaUrl = avaUrl;  this.isDisabled = isDisabled;  }  // static functions to access storage  // returns user with id or undefined  static getById(*id*) {  for (let User of user.items) {  if (User.id === id) {  return User;  }  }  }  // returns an array of all users in storage  static getAll() {  return user;  }  };  module.exports = User; |

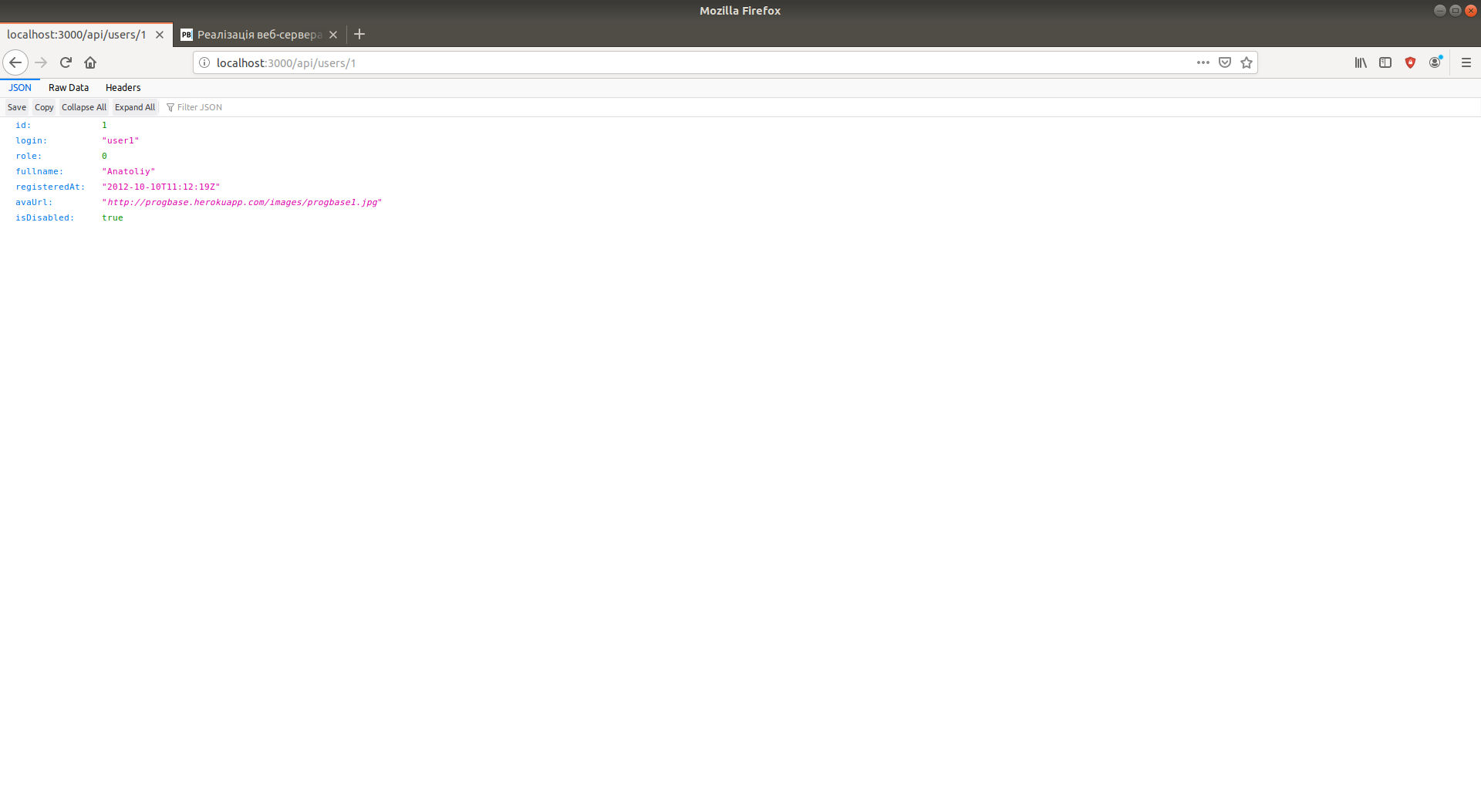
**Приклади результатів**

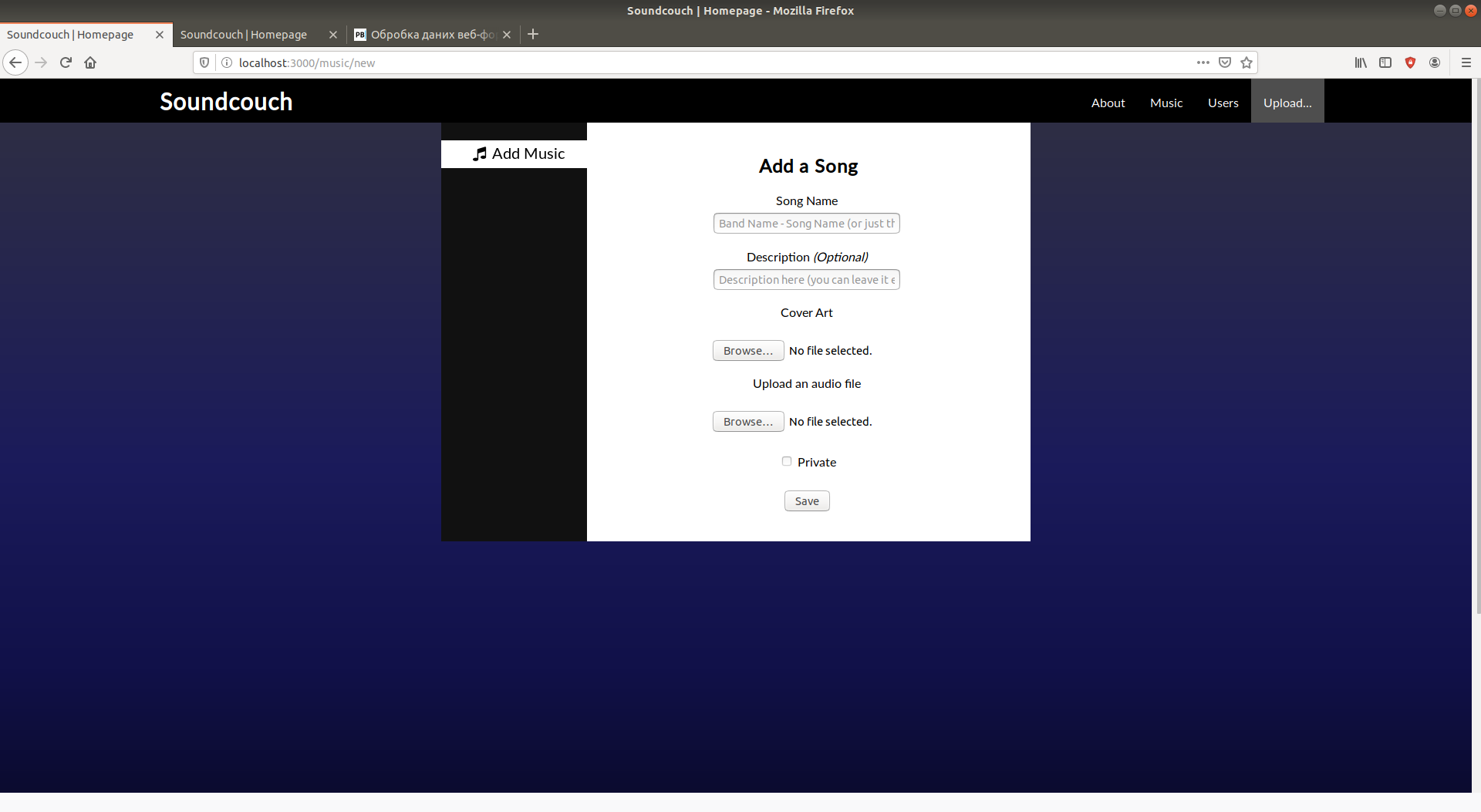
localhost:3000/

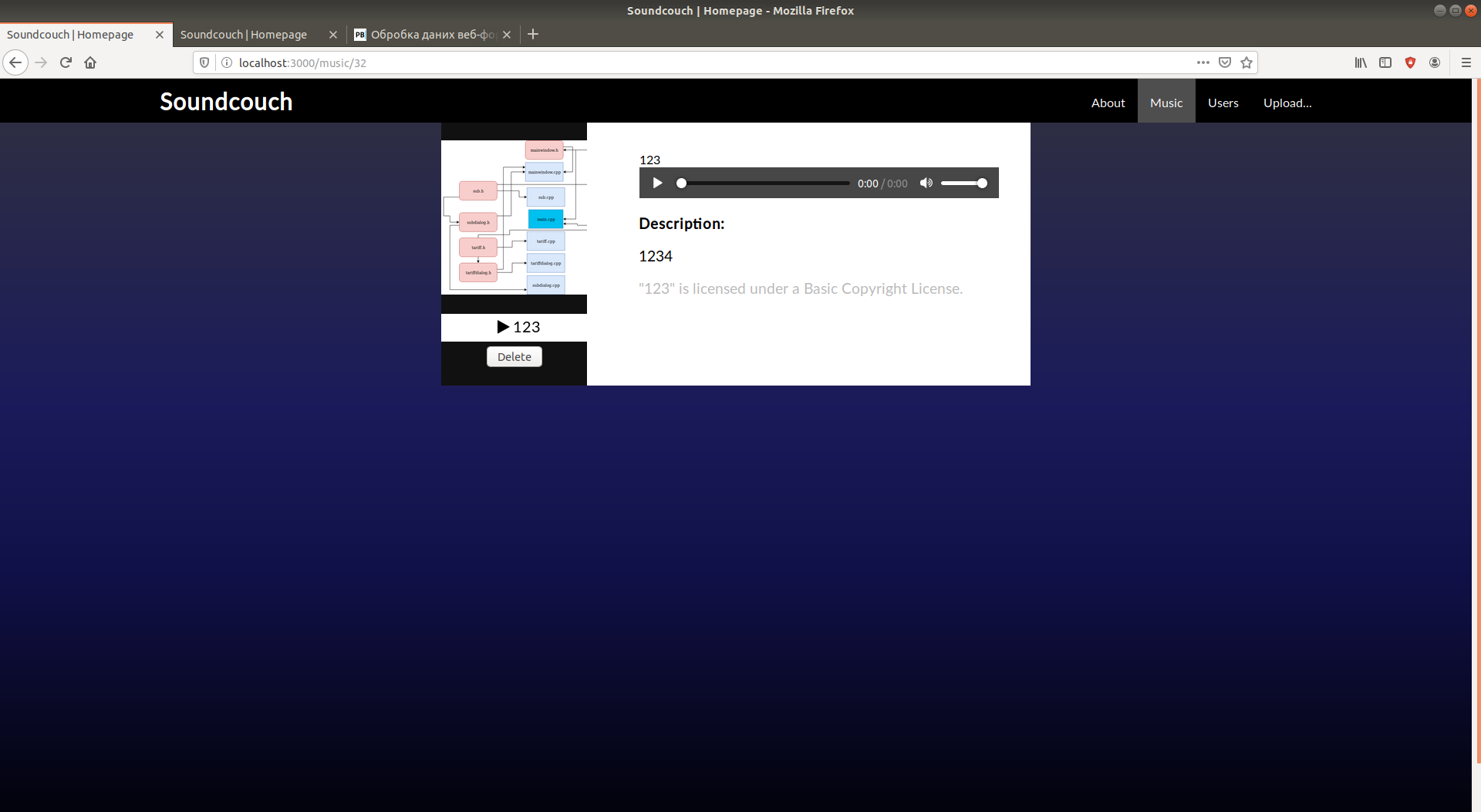
localhost:3000/about

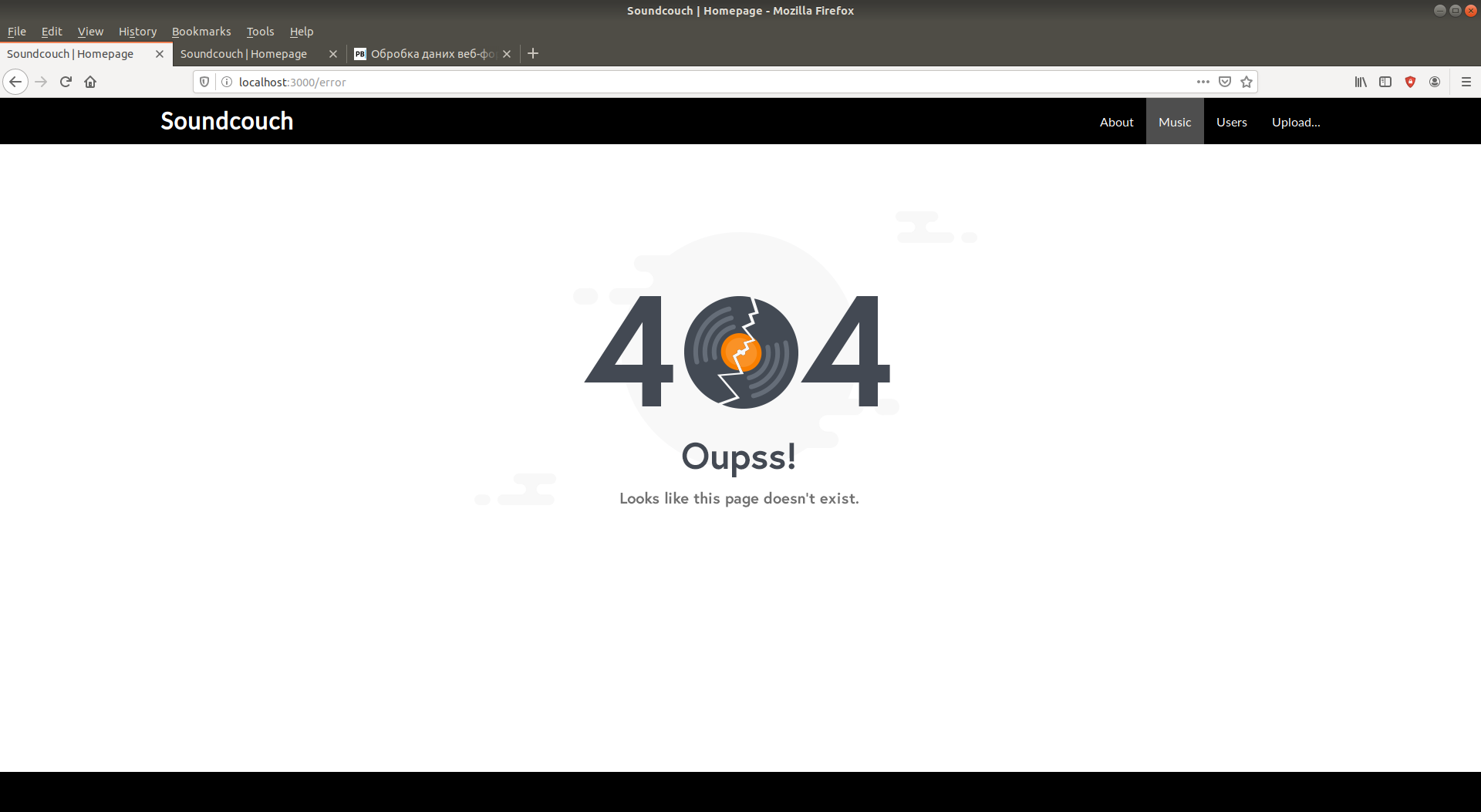
*localhost:3000/users/1*

*localhost:3000/api/users*

*localhost:3000/api/users/1*

Додавання музики

Створений крізь Upload… файл

Цей файл після видалення

**Висновок**

Навчився створювати HTML-форми для взаємодії користувача із веб-сайтом та обробляти на сервері внесені користувачем дані.  
Вивчив основні принципи асинхронного програмування за допомогою callback-функцій.