

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 “Багатосторінковий веб-сайт”

|  |  |  |
| --- | --- | --- |
| Виконав  студент 2 курсу  групи КП-93  Торговських Олександр  варіант № 20 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

Київ 2020

### Мета

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

Ознайомитись із шаблонізаторами веб-сторінок для генерації контенту.

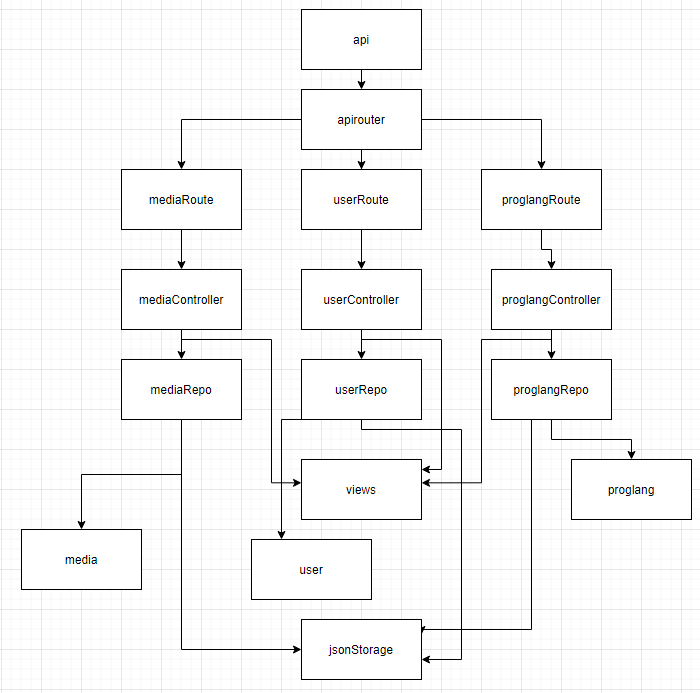
Навчитись створювати HTML-форми для взаємодії користувача із веб-сайтом та обробляти на сервері внесені користувачем дані.

### Завдання

Реалізувати багатосторінковий веб-сайт із заданою структурою розділів: 

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

**діаграма залежностей модулів**



**Код**

|  |
| --- |
| **app.js** |
| **const express = require('express');**  **const morgan = require('morgan');**  **const mustache = require("mustache-express");**  **const app = express();**  **const apiRouter = require('./routes/apirouter');**  **const path = require('path');**  **port = process.env.port || 4444;**    **const expressSwaggerGenerator = require('express-swagger-generator');**  **const expressSwagger = expressSwaggerGenerator(app);**  **const options = {**  **swaggerDefinition: {**  **info: {**  **description: 'http api server',**  **title: 'lab2',**  **version: '1.0.0',**  **},**  **host: 'localhost:4444',**  **produces: [ "application/json" ],**  **},**  **basedir: \_\_dirname,**  **files: ['./routers/\*\*/\*.js', './models/\*\*/\*.js'],**  **};**  **expressSwagger(options);**    **app.use(express.static('public'))**    **const viewsDir = path.join(\_\_dirname, 'views');**  **app.engine("mst", mustache(path.join(viewsDir, "partials")));**  **app.set('views', viewsDir);**  **app.set('view engine', 'mst');**    **app.get('/' , function(req, res) {**  **res.render('index' , {is\_home: true,});**  **});**  **app.get('/about' , function(req, res) {**  **res.render('about' , {title: 'About', is\_about: true ,});**  **});**  **app.use(morgan('dev'));**  **app.use('' , apiRouter);**    **app.listen(port , err => {**  **if(err)**  **{**  **return console.log("ERROR" , err);**  **}**  **console.log('listening on port '+port);**  **});** |

|  |
| --- |
| **partials/head.mst** |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <link type="text/css" href="/stylesheets/style.css" rel="stylesheet">  </head> |

|  |
| --- |
| **partials/header.mst** |
| <body>  <div class="header"> |

|  |
| --- |
| **apiroute.js** |
| **const router = require('express').Router();**  **const userRoute = require('./users.js');**  **const proglanguageRoute = require('./ProgLanguageRoute.js');**  **const bodyparser = require('body-parser');**  **const mediaRouter = require('./mediaRouter');**    **const multer = require('multer');**    **router.use(multer({dest:"./data/media"}).single("image"))**    **router.use(bodyparser.json());**  **router.use(bodyparser.urlencoded({ extended: true }));**    **router.use('/data' , mediaRouter)**  **router.use('/users' , userRoute);**  **router.use('/proglanguages' , proglanguageRoute);**    **module.exports = router;** |

|  |
| --- |
| **usersRoute.js** |
| **const router = require('express').Router();**    **const userController = require('./../controllers/userControllers');**    **router**  **/\*\***  **\* @route GET /api/users/{id}**  **\* @group Users - user operations**  **\* @param {integer} id.path.required - id of the User - eg: 1**  **\* @returns {User.model} 200 - User object**  **\* @returns {Error} 404 - User not found**  **\*/**  **.get("/:id", userController.getUser)**  **/\*\***  **\* @route GET /api/users?page={page}&size={size}**  **\* @group Users - user operations**  **\* @param {integer} page.path.required - page**  **\* @param {integer} size.path.required - size**  **\* @returns {Array.<User>} User - all users**  **\*/**  **.get("/", userController.getUsers)**    **module.exports = router;** |

|  |
| --- |
| **ProgLanguageRoute.js** |
| **const router = require('express').Router();**    **const proglanguageController = require('./../controllers/proglanguageControllers');**    **router**  **.get("/new" , proglanguageController.getNew)**  **/\*\***  **\* @route GET /api/proglanguages/{id}**  **\* @group Proglanguages - proglanguage operations**  **\* @param {integer} id.path.required - id of the Proglanguage - eg: 1**  **\* @returns {Proglanguage.model} 200 - Proglanguage object**  **\* @returns {Error} 404 - Proglanguage not found**  **\*/**  **.get("/:id", proglanguageController.getProglanguage)**  **/\*\***  **\* @route GET /api/proglanguages**  **\* @group Proglanguages - proglanguage operations**  **\* @returns {Array.<Proglanguage>} Proglanguage - all proglanguages**  **\*/**  **.get("/", proglanguageController.getProglanguages)**  **/\*\***  **\* @route POST /api/proglanguages**  **\* @group Proglanguages - proglanguage operations**  **\* @param {Proglanguage.model} id.body.required - new Proglanguage object**  **\* @returns {Proglanguage.model} 201 - added Proglanguage object**  **\*/**  **.post("/" , proglanguageController.addProglanguage)**    **/\*\***  **\* @route PUT /api/proglanguages**  **\* @group Proglanguages - proglanguage operations**  **\* @param {Proglanguage.model} id.body.required - new Proglanguage object**  **\* @returns {Proglanguage.model} 200 - changed Proglanguage object**  **\*/**  **.put("/" , proglanguageController.updateProglanguage)**  **/\*\***  **\* @route DELETE /api/proglanguages/{id}**  **\* @group Proglanguages - proglanguage operations**  **\* @param {integer} id.path.required - id of the Proglanguage - eg: 1**  **\* @returns {Proglanguage.model} 200 - deleted Proglanguage object**  **\* @returns {Error} 404 - Proglanguage not found**  **\*/**  **.post("/:id" , proglanguageController.deleteProglanguage)**    **module.exports = router;** |

|  |
| --- |
| **mediaRoute.js** |
| **const router = require('express').Router()**  **//.post('/', (req, res) => res.send(req.files));**  **const mediaController = require('./../controllers/mediaController');**  **router**  **/\*\***  **\* @route POST /api/media**  **\* @group Medias - upload and get images**  **\* @consumes multipart/form-data**  **\* @param {file} image.formData.required - uploaded image**  **\* @returns {integer} 200 - added image id**  **\*/**  **.post("/", mediaController.postMedia)**  **/\*\***  **\* @route GET /api/media/{id}**  **\* @group Medias - media operations**  **\* @param {integer} id.path.required - id of the Proglanguage - eg: 1**  **\* @returns {Proglanguage.model} 200 - Proglanguage object**  **\* @returns {Error} 404 - Proglanguage not found**  **\*/**  **.get("/media/:path" , mediaController.getImage)**  **module.exports = router;** |

|  |
| --- |
| **mediaController.js** |
| **const MediaRepository = require('../repositories/mediaRepository');**  **const Media = require('../models/media');**  **const mediaRepository = new MediaRepository('data/medias.json');**  **const fs = require('fs');**  **const path = require('path');**  **module.exports =**  **{**  **postMedia(req , res)**  **{**  **if(req.file == undefined)**  **{**  **res.status(400).send("400 Bad request");**  **}else{**  **const temp = new Media(0 , req.file.path);**    **temp.id = mediaRepository.addMedia(temp);**  **res.status(201);**  **res.json(temp.id);**  **}**  **},**  **getImage(req , res)**  **{**  **console.log("123")**  **const image\_path = req.params.path;**  **res.contentType('image/jpeg');**  **fs.createReadStream(path.resolve(\_\_dirname + '/../data/media', image\_path)).pipe(res);**  **}**  **};** |

|  |
| --- |
| **userController.js** |
| **const UserRepository = require('../repositories/userRepository');**  **const userRepository = new UserRepository("data/users.json");**    **module.exports =**  **{**  **getUser(req , res)**  **{**  **const userId = parseInt(req.params.id);**  **const user = userRepository.getUserById(userId);**  **if(user == null)**  **{**  **res.status(404).send("404 Not found");**  **}**  **console.log(user.role)**  **console.log(user.fullname)**  **res.render('user' , user);**  **},**    **getUsers(req , res)**  **{**  **//if(req.query.page == null || req.query.size == null)**  **//{**  **// res.status(400).send("400 Bad request");**  **//}else{**  **const page = parseInt(req.query.page);**  **const amount = parseInt(req.query.size);**  **const offset = (page - 1)\*amount;**    **const users = userRepository.getUsers();**  **const new\_users = users;//.slice(offset , offset+amount);**  **res.render('users' , {new\_users , is\_users: true,})**  **//}**  **},**  **};** |

|  |
| --- |
| **proglanguageControllers.js** |
| **const ProgLanguage = require('./../models/ProgLanguage.js');**  **const ProgLanguageRepository = require('./../repositories/ProgLanguageRepository');**  **const { json } = require('body-parser');**  **const proglanguageRepository = new ProgLanguageRepository('data/proglanguages.json')**    **module.exports =**  **{**  **getProglanguage(req , res)**  **{**  **const proglanguageId = parseInt(req.params.id);**  **const proglanguage = proglanguageRepository.getProgLanguagesById(proglanguageId);**  **if(proglanguage == null)**  **{**  **res.status(404).send("404 Not found");**  **}else{**  **res.render('proglanguage',proglanguage);**  **}**  **},**  **getNew(req, res) {**  **res.render('new');**  **},**  **getProglanguages(req , res)**  **{**  **const proglanguages = proglanguageRepository.getProgLanguages();**  **let new\_proglanguages = [];**  **if (req.query.name\_search != null) {**  **search\_str = req.query.name\_search;**  **for (const temp of proglanguages) {**  **str = temp.name.toLowerCase();**  **if (str.indexOf(search\_str.toLowerCase()) == 0) {**  **new\_proglanguages.push(temp);**  **}**  **}**  **} else {**  **new\_proglanguages = proglanguages;**  **}**  **//new\_proglanguages = proglanguages;**  **let page;**  **if (req.query.page == null) {**  **page = 1;**  **} else {**  **page = parseInt(req.query.page);**  **}**  **const amount = 5;**  **const offset = (page - 1) \* amount;**  **max\_page = Math.ceil(new\_proglanguages.length / parseFloat(amount))**  **new\_proglanguages = new\_proglanguages.slice(offset, offset + amount)**  **let pages = [];**  **for (let i = 1; i <= max\_page; i++) {**  **pages.push({ page: i });**  **}**  **if (new\_proglanguages.length == 0) {**  **res.render('proglanguages', { new\_proglanguages, is\_proglanguages: true, pages, search: req.query.name\_search, Empty: true, });**  **} else {**  **res.render('proglanguages', { new\_proglanguages, is\_proglanguages: true, pages, search: req.query.name\_search, });**  **}**  **},**    **addProglanguage(req , res)**  **{**  **if (req.body.name == "" || req.body.founderName == "" || req.body.lastversion == "" || req.body.countofusers == ""|| req.body.lasteditData == "")**  **{**  **res.render('new', { invalid: true, });**  **}else{**  **let path = "";**  **if(req.file != undefined)**  **{**  **path = req.file.path;**  **}**  **const x = new ProgLanguage(0, req.body.name, req.body.founderName, req.body.lastversion, req.body.countofusers, req.body.lasteditData, path);**  **x.id = proglanguageRepository.addProgLanguage(x);**  **res.render('proglanguage' , x);**  **}**  **},**    **updateProglanguage(req , res)**  **{**  **if(req.body.name == null || req.body.founderName == null || req.body.lastversion == null || req.body.countofusers == null|| req.body.lasteditData == null)**  **{**  **res.status(400).send("400 Bad request");**  **}else{**  **const x = new ProgLanguage( req.body.id , req.body.name , req.body.founderName , req.body.lastversion , req.body.countofusers , req.body.lasteditData);**  **if(proglanguageRepository.updateProgLanguages(x) == null)**  **{**  **res.status(404).send("404 Not found");**  **}else{**  **res.json(x);**  **}**  **}**  **},**    **deleteProglanguage(req ,res)**  **{**  **const Id = parseInt(req.params.id);**  **proglanguageRepository.deleteProgLanguages(Id);**  **res.render('proglanguage' , {is\_deleted: true});**  **}**  **};** |

|  |
| --- |
| **style.css** |
| body  {  width: 1200px;  margin-left: auto;  margin-right: auto;  }    #user\_img  {  box-shadow: 4px 4px 3px rgba(0,0,0,0.6);  }    #header\_img  {  float:left;  box-shadow: 4px 4px 3px rgba(0,0,0,0.6);  }    #header\_h1{  margin-left: 110px;  font-size: 30;  font-style: italic;  font-family: Georgia;  }    li.menu\_li{  padding-right:150px;  }    div.header  {  border-style: solid;  border-color: rgba(0, 0, 0, 0.39);  border-width: 1px;  background-color: rgba(255, 255, 255, 0.39);  box-shadow: 7px 7px 5px rgba(0,0,0,0.6);  }    a {  color: black;  text-decoration: none;  }    div.menu\_block , div.table\_block , div.user\_block , div.content\_proglanguage\_block{  display: inline-block;  }    input.new\_data{  height:25px;  font-size:16px;  }    button.new{  float: left;  margin-top: 20px;  height: 40px;  font-size: 20px;  }    table{  border-collapse: collapse;  }    a.table{  color: grey;  text-decoration: underline;  }    #current\_page{  color: grey;  text-decoration: underline;  }    #table{  border-width: 1px;  border-style: solid;  border-color: black;  font-family: 'Times New Roman';  font-size: 24px;  }    div.table\_block{  float:left;  position: relative;  margin-top: 10px;  margin-left:20px;    word-wrap: break-word;  font-family: 'Times New Roman';  font-size: 20px;  }    #user\_img, #user\_info{  display: inline-block;  }    div.menu\_block  {  float:left;  border-style: solid;  border-color: black;  border-width: 1px;  font-family: Georgia;  margin-top:10px;  line-height: 2;  background-color: lightblue;  }    div.content\_block  {  float:left;  position: relative;  padding-top:10px;  margin-left: 20px;  height: 100vh;  word-wrap: break-word;  font-family: 'Times New Roman';  font-size: 20px;  }    div.content\_proglanguage\_block  {  float:left;  position: relative;  padding-top:10px;  margin-left: 20px;  height: 100vh;  word-wrap: break-word;  font-family: 'Times New Roman';  font-size: 20px;  }    div.content\_block  {  position: relative;  padding-top:10px;  margin-left: 320px;  margin-right: 0%;  margin-top: -150px;  height: 300px;  word-wrap: break-word;  font-family: 'Times New Roman';  font-size: 20px;  }    div.bio\_block{  border-style: solid;  border-color: black;  border-width: 1px;  background-color: lightcyan;  box-shadow: 7px 7px 5px rgba(0,0,0,0.6);  }    div.user\_block{  position: relative;  padding-top:10px;  margin-left: 20px;  margin-right: 15%;  height: 100vh;  word-wrap: break-word;  font-family: 'Times New Roman';  font-size: 20px;  }      form.buttons{  float:left;  margin-top:300px;  }    input.buttons{  width: 40px;  font-size: 30px;  }    div.menu\_block , div.content\_block  {  display: inline-block;  } |

|  |
| --- |
| **partials/footer.mst** |
| <footer>  <hr>  <span>Laboratory work #3</span>  /  <a href="https://bitbucket.org/alextorgvoskyh/webprogbase/src/master/labs/lab3/">Repository</a>  </footer> |

|  |
| --- |
| **partials/menu.mst** |
| <div class="menu\_block">  <menu>  {{#is\_home}}  <li class="menu\_li" id="current\_page"><a href='/'>HOME</a></li>  {{/is\_home}}  {{^is\_home}}  <li class="menu\_li"><a href='/'>HOME</a></li>  {{/is\_home}}    {{#is\_users}}  <li class="menu\_li" id="current\_page"><a href= '/users'>USERS</a></li>  {{/is\_users}}  {{^is\_users}}  <li class="menu\_li"><a href= '/users'>USERS</a></li>  {{/is\_users}}    {{#is\_proglanguages}}  <li class="menu\_li" id="current\_page"><a href= '/proglanguages'>PROGLANGS</a></li>  {{/is\_proglanguages}}  {{^is\_proglanguages}}  <li class="menu\_li"><a href= '/proglanguages'>PROGLANGS</a></li>  {{/is\_proglanguages}}    {{#is\_about}}  <li class="menu\_li" id="current\_page"><a href= '/about'>ABOUT</a></li>  {{/is\_about}}  {{^is\_about}}  <li class="menu\_li"><a href= '/about'>ABOUT</a></li>  {{/is\_about}}  </menu>  </div> |

|  |
| --- |
| **about.mst** |
| {{> head}}  <title>About</title>  {{> header}}  <img id="header\_img" src="https://static.onecms.io/wp-content/uploads/sites/34/2018/05/12170411/cat-kitten-138468381.jpg" width = 120 alt="question mark">  <h1 id="header\_h1">Додатково про сайт та автора</h1>  </div>  {{> menu}}  <main>  <div class ="content\_block">Даний сайт був створений студентом КПІ Торговським Олександром з групи КП-93.  <br>Сайт був створений для лабораторної роботи №3 за допомогою вреймворку <strong>Express</strong> <br>Та шаблонізатора <strong> Mustache</strong>  {{> footer}}  </div>  </main>  </body>  </html> |

|  |
| --- |
| **proglanguage.mst** |
| {{> head}}  <title>Progrmaing language</title>  {{> header}}  <img id="header\_img" src="https://www.goodcore.co.uk/blog/wp-content/uploads/2019/08/coding-vs-programming-2.jpg" width = 100 alt="proglanguage">  <h1 id="header\_h1">Інформація про {{name}}</h1>  </div>  {{> menu}}  <div class ="content\_proglanguage\_block">  {{#is\_deleted}}  <p>Видалено</p>  {{/is\_deleted}}  {{^is\_deleted}}    <img id="user\_img" src="/{{path}}" width=80 alt="lang image"><br>    id: {{id}} <br>  Назва: {{name}}<br>  Розробник: {{founderName}}<br>  Остання версія: {{lastversion}}<br>  Кільікість користувачів: {{countofusers}}<br>  Дата останньої зміни: {{lasteditData}}  <form action="/proglanguages/{{id}}" method="POST">  <input class ="new\_data" type="submit" name="delete" value="Видалити">  </form>  {{/is\_deleted}}  {{> footer}}  </div>  </body>  </html> |

|  |
| --- |
| **proglanguages.mst** |
| {{> head}}  <title>Proglanguages</title>  {{> header}}  <img id="header\_img" src="images/c++\_logo.png" width = 70 alt="proglanguage">  <h1 id="header\_h1">Список мов</h1>  </div>  {{> menu}}  <div class="table\_block">  <form action="/proglanguages/" method="GET">  Пошук по назві:<br>  <input type="text" name="name\_search" value={{search}}><br>  </form>  {{#Empty}}  <p>Жодних співпадінь</p>  {{/Empty}}  <table>  {{#new\_proglanguages}}  <tr>  <td id="table"><a class="table" href=/proglanguages/{{id}}>{{name}}</a></td>  </tr>  {{/new\_proglanguages}}  </table>  </div>  <form class="buttons" action="/proglanguages" method="GET">  {{#pages}}  <input class="buttons" type="submit" name="page" value={{page}}>  {{/pages}}  {{> footer}}  <input hidden type="text" name="name\_search" value={{search}}>  </form>  <button class="new"><a href='/proglanguages/new'>Додати нову</button>  </body>  </html> |

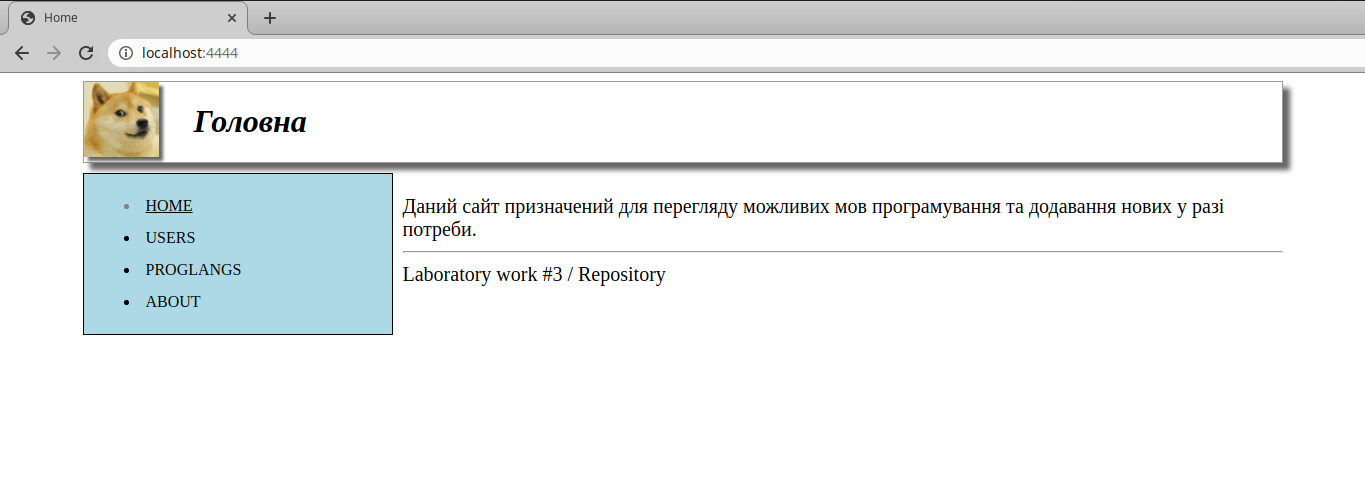
|  |
| --- |
| **index.mst** |
| {{> head}}  <title>Home</title>  {{> header}}  <img id="header\_img" src="./images/doge.png" width = 75 alt="home" >  <h1 id="header\_h1">Головна</h1>  </div>  {{> menu}}  <div class ="content\_block">  Даний сайт призначений для перегляду можливих мов програмування та додавання нових у разі потреби.  {{> footer}}  </div>  </body>  </html> |

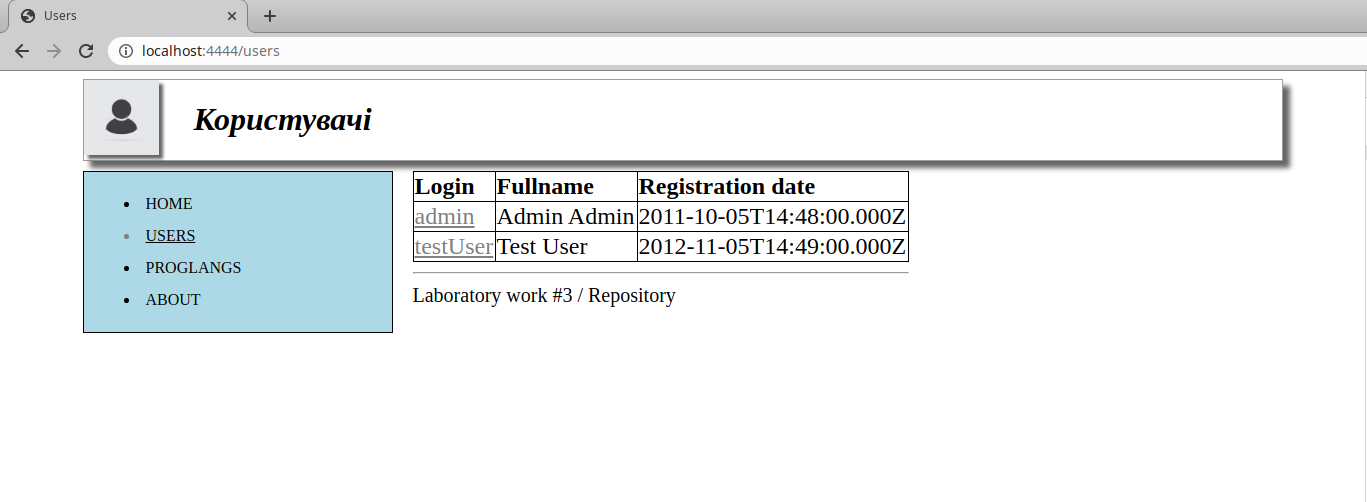
|  |
| --- |
| **new.mst** |
| {{> head}}  <title>New language</title>  {{> header}}  <h1 id="header\_h1">Додання нової мови програмування</h1>  </div>  {{> menu}}  <div class="table\_block">  <form enctype="multipart/form-data" action="/proglanguages" method="POST">  Введіть назву<br>  <input class="new\_data" type="text" name="name"><br>  Введіть ім'я засновника<br>  <input class="new\_data" type="text" name="founderName"><br>  Введіть останню версію<br>  <input class="new\_data" type="text" name="lastversion"><br>  Введіть кількість користувачів<br>  <input class="new\_data" type="number" name="countofusers"><br>  Введіть останню дату змін<br>  <input class="new\_data" type="date" name="lasteditData" pattern="^(0?[1-9]|[12][0-9]|3[01])[-](0?[1-9]|1[012])[-]\d{4}$"><br>  <input class="new\_data" type="file" name="image" accept="image/\*"><br>  <input class="new\_data" type="submit" name="submit" value="Додати">  </form>  {{#invalid}}  <p>Заповніть всі поля</p>  {{/invalid}}  </div>  {{> footer}}  </body>  </html> |

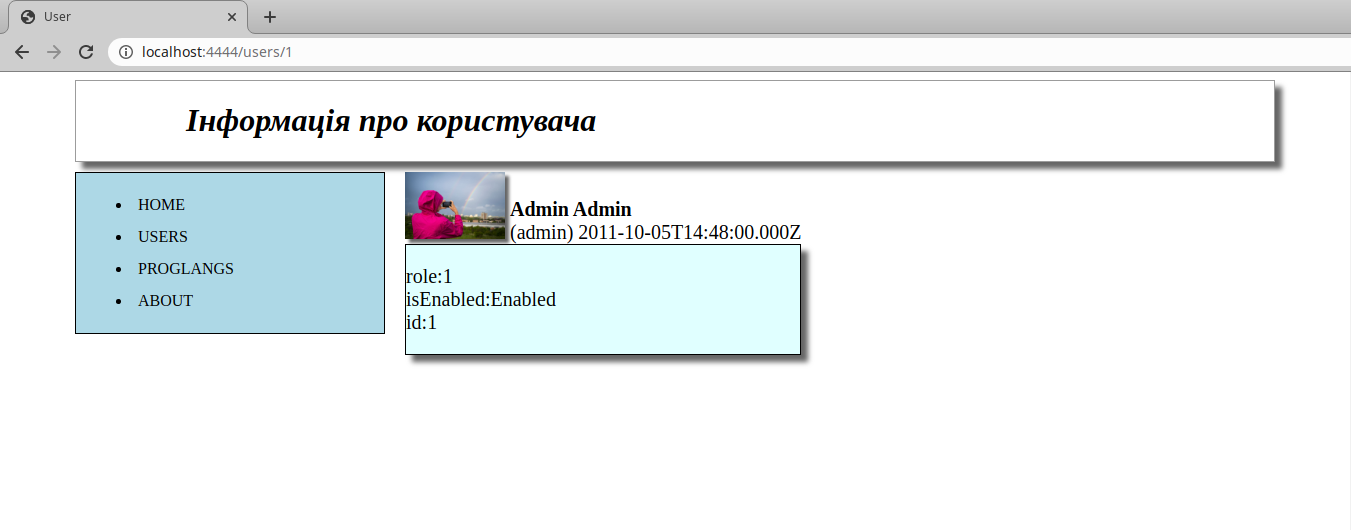
|  |
| --- |
| **user.mst** |
| {{> head}}  <title>User</title>  {{> header}}  <h1 id="header\_h1">Інформація про користувача</h1>  </div>  {{> menu}}  <div class="user\_block">  <img id="user\_img" src="{{avaUrl}}" width = 100 alt="user ava" >  <div id="user\_info">  <strong>{{fullname}}</strong><br>({{login}}) {{registeredAt}}  </div>  <div class="bio\_block">  <p>role:{{role}}<br>isEnabled:{{isEnabled}}<br>id:{{id}}</p>  </div>  </div>  {{> footer}}  </body>  </html> |

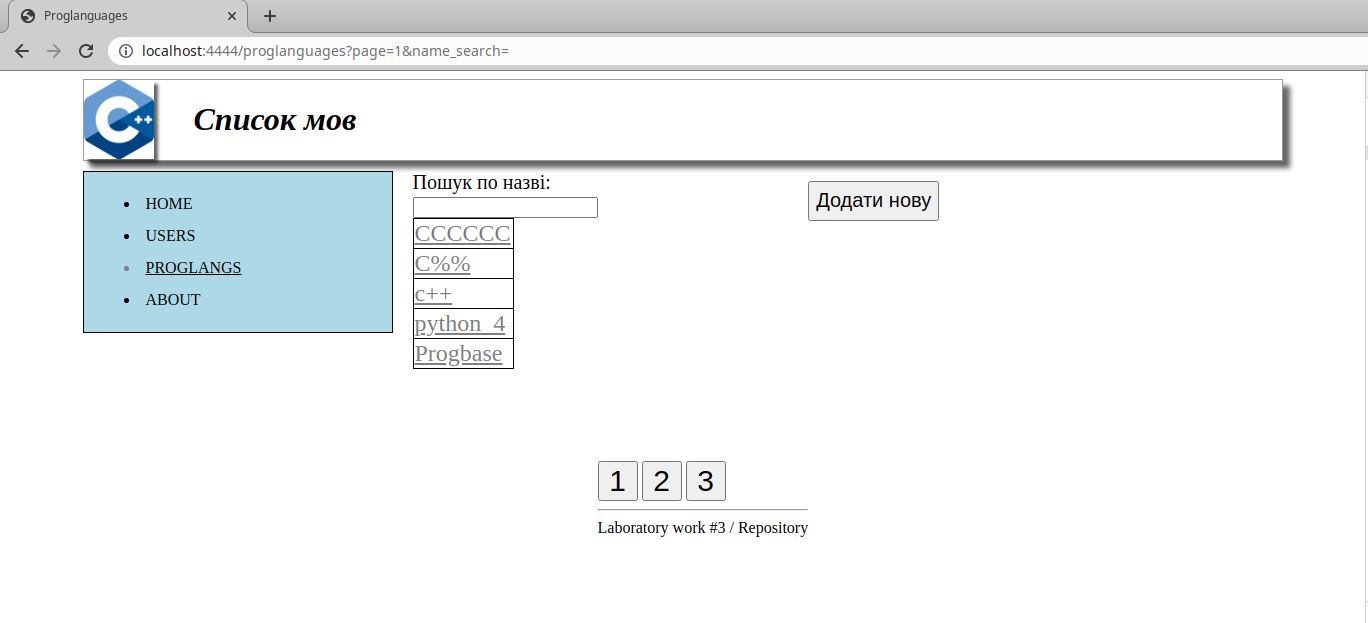
|  |
| --- |
| **users.mst** |
| {{> head}}  <title>Users</title>  {{> header}}  <img id="header\_img" src='./images/user\_logo.jpg' width = 75 alt="user">  <h1 id="header\_h1">Користувачі</h1>  </div>  {{> menu}}  <div class="table\_block">  <table>  <tr>  <td id="table"><strong>Login</strong></td>  <td id="table"><strong>Fullname</strong></td>  <td id="table"><strong>Registration date</strong></td>  </tr>  {{#new\_users}}  <tr>  <td id="table"><a class="table" href=/users/{{id}}>{{login}}</a></td>  <td id= "table">{{fullname}}</td>  <td id="table">{{registeredAt}}</td>  </tr>  {{/new\_users}}  </table>  {{> footer}}  </body>  </html> |

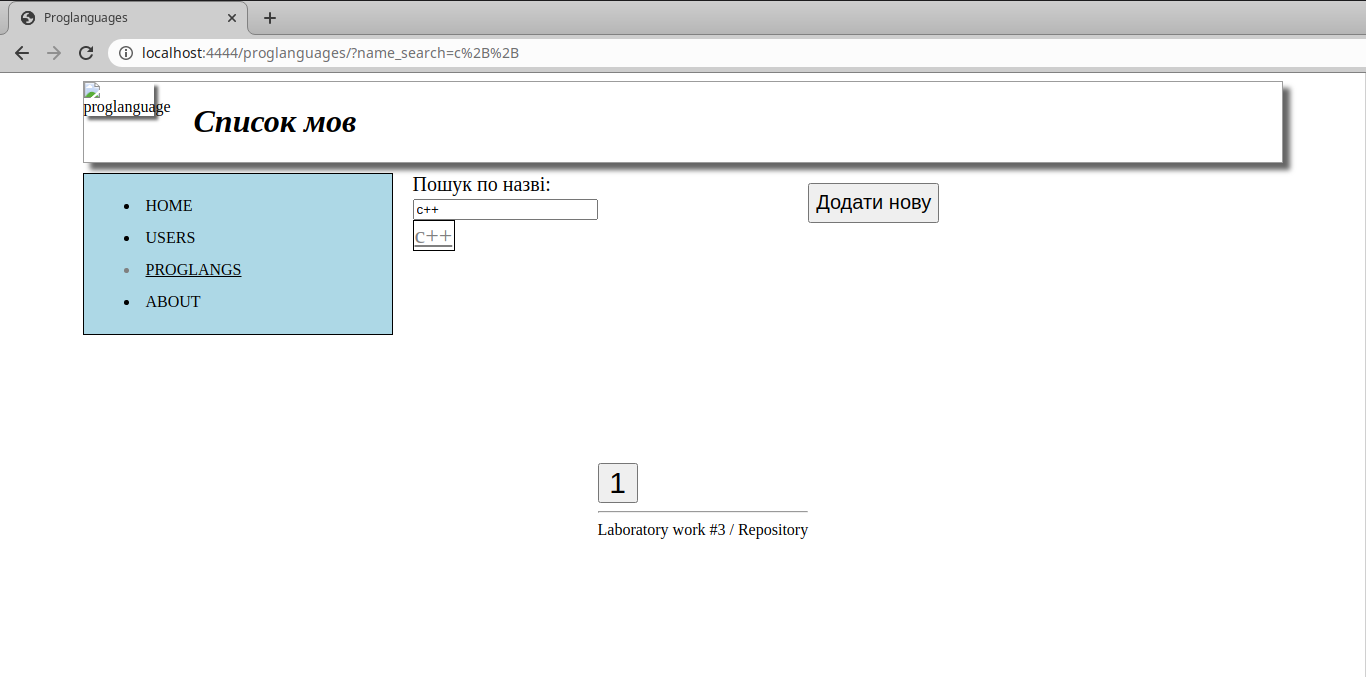
**Приклади результатів**

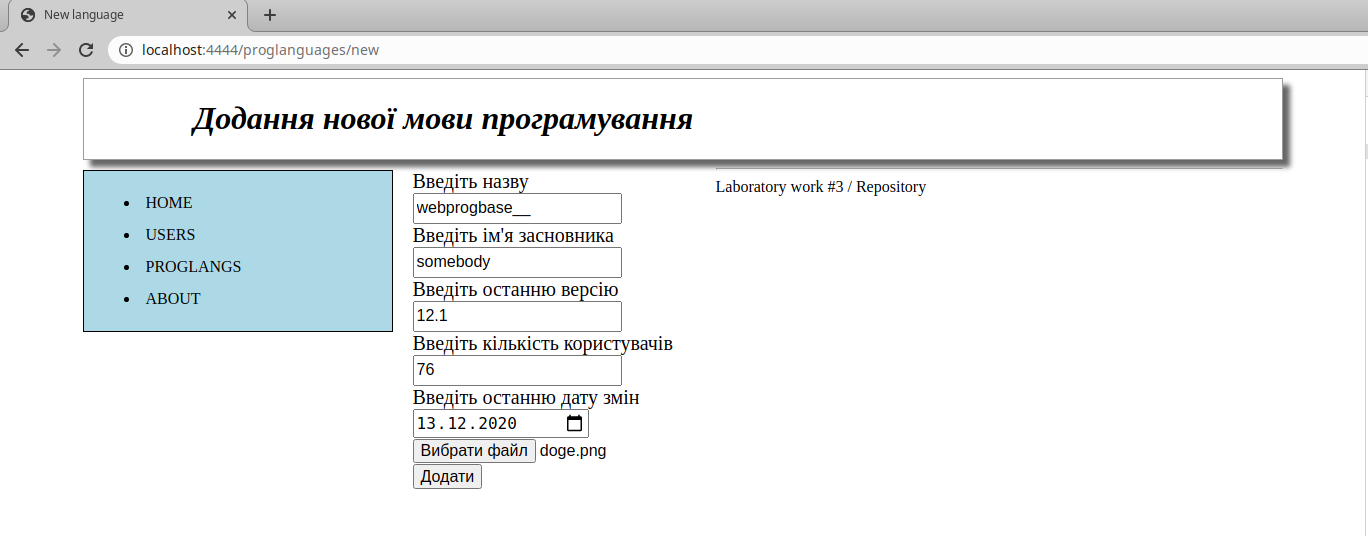
****

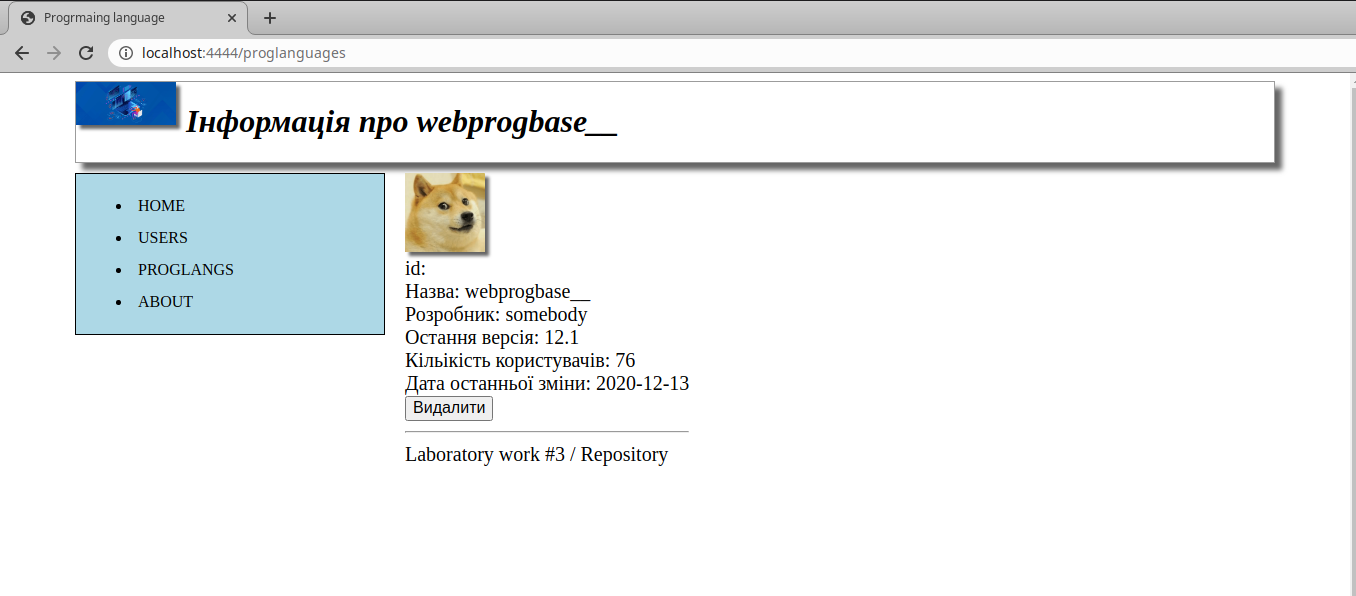
****

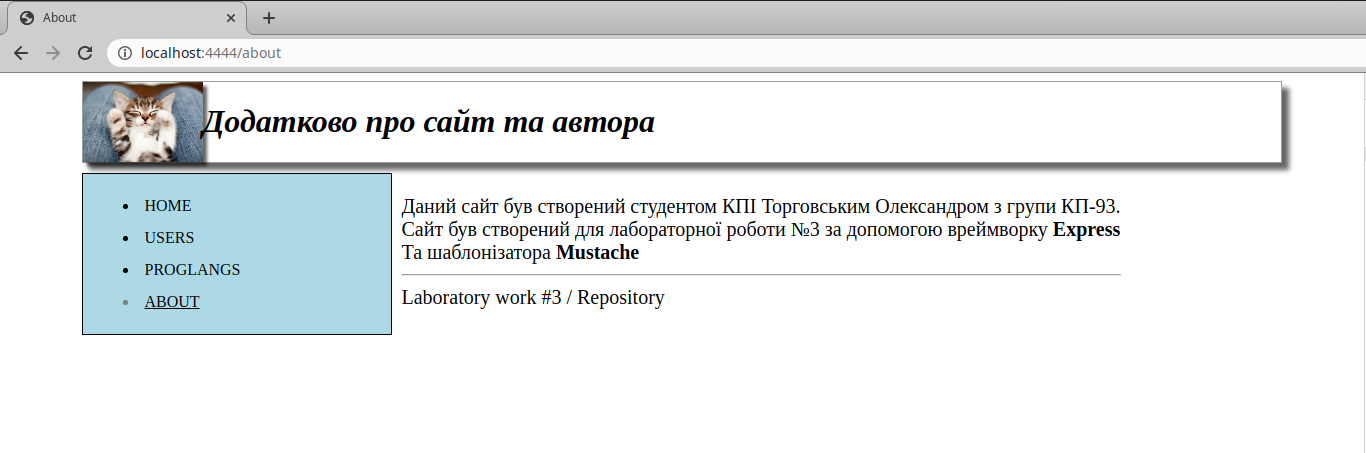
****

****

****

****

****

****

**Висновки**

Виконуючи дану лабораторну роботу я навчився створювати веб-сайт, що надає CRUD доступ до ресурсів веб-сервера. Ознайомився із шаблонізаторами веб-сторінок для генерації контенту. Навчився створювати HTML-форми для взаємодії користувача із веб-сайтом та обробляти на сервері внесені користувачем дані.