

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-91  Степаненко Андрій Сергійович  Варіант №21 |  | Перевірив  “” “листопада” 2020р.  викладач  Гадиняк Руслан Анатолійович |

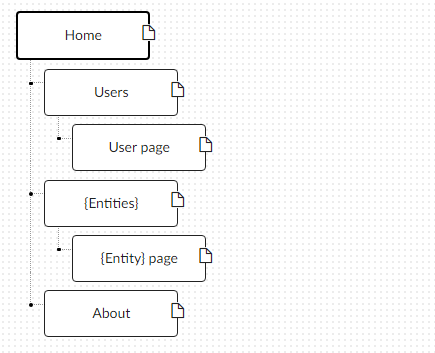
Київ 2020

**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.

**Постановка завдання**

### **Завдання:**

Реалізувати багатосторінковий веб-сайт із заданою структурою розділів:

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

**Вказівки до виконання:**

#### 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](https://docs.google.com/presentation/d/1tXOT3GL0yhRbriRLlsUIn-zcRFQSkWsb2pYesO9Yr2M/edit?usp=sharing).
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:

GET / -> views/index.html

GET /users -> views/users.html

GET /users/:id -> views/user.html

GET /{entities} -> views/{entities}.html

GET /{entities}/:id -> views/{entity}.html

GET /about -> views/about.html

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

1. Шаблонізація
   1. Виділити із HTML-сторінок 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

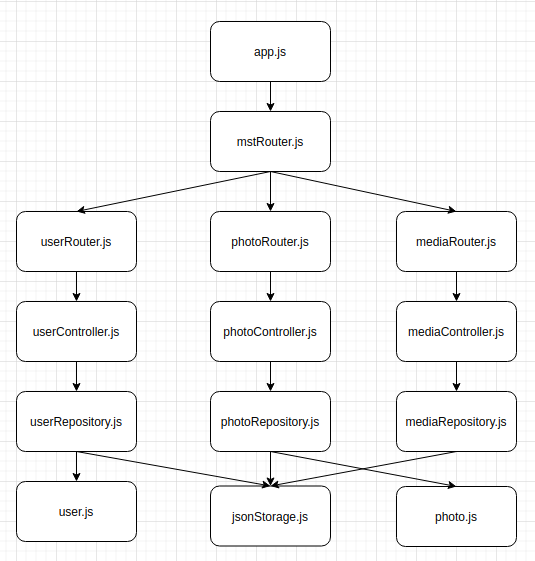
about.mst

* 1. Видалити HTML файли.
  2. 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
  3. 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:**

****

****

**Тексти коду програм**

|  |
| --- |
| app.js |
| const express = require('express');  const app = express();  const consolidate = require('consolidate');  const path = require('path');  const mustache = require('mustache-express');  const body\_parser = require('body-parser')  const busboyBodyParser = require('busboy-body-parser');  const morgan = require('morgan');  const viewsDir = path.join(\_\_dirname, 'views');  app.engine("mst", mustache(path.join(viewsDir, "partials")));  app.set('views', viewsDir);  app.set('view engine', 'mst');  app.set('views', path.join(\_\_dirname, 'views'));  app.set('view engine', 'mst');  app.use(express.static('./public'))  app.use(express.static('./data'))  const mstRouter = require('./routes/mstRouter');  app.use(body\_parser.urlencoded({ extended: true }))  app.use(body\_parser.json());  app.use(busboyBodyParser());  app.use(morgan('dev'));  const expressSwaggerGenerator = require('express-swagger-generator');  const expressSwagger = expressSwaggerGenerator(app);  const options = {  swaggerDefinition: {  info: {  description: 'Trying to understand swagger',  title: 'Pseudo swagger',  version: '1.0.0',  },  host: 'localhost:3000',  produces: [ "application/json" ],  },  basedir: \_\_dirname,  files: ['./routes/\*\*/\*.js', './models/\*\*/\*.js'],  };  expressSwagger(options);  app.get('/', function(req, res) {  res.render('index', {disabled: "disabled"});  });  app.get('/about', function(req, res) {  res.render('about', {aboutDisabled: "disabled"});  });  app.use('', mstRouter);  app.use((req, res) => {  res.status(400).send({ message: "Error in route."});  });  app.listen(3000, function() {  console.log('Server is ready');  }); |

|  |
| --- |
| mstRouter.js |
| const router = require('express').Router();  const userRouter = require('./userRouter');  router.use('/users', userRouter);  const photoRouter = require('./photoRouter');  router.use('/photos', photoRouter);  const mediaRouter = require('./mediaRouter');  router.use('/media', mediaRouter);  module.exports = router; |

|  |
| --- |
| style.css |
| \* {  margin: 0;  padding: 0;  }  body {  background-image: url("/images/back3.png");  background-repeat: no-repeat;  background-size: cover;  }  body::after {  content: "";  background-color: rgba(22, 2, 43, 0.44);  background-size: cover;  top: 0;  left: 0;  bottom: 0;  right: 0;  position: absolute;  z-index: -100;  }  header {  position: fixed;  width: 100%;  margin: auto;  top: 0;  height: 40%;  }  .center {  position: relative;  margin: auto;  margin-top: -8%;  width: 40%;  text-align: center;  display: block;  }  .center img {  padding: auto;  }  p, table, input, button {  font-family: 'Kalam';  text-align: center;  vertical-align: center;  }  td, th {  border: 2px inset #f9ebb5;  padding: .3em 1em;  }  .main table, .main .search, .mainPhotos table {  margin: auto;  }  table {  color: #f9ebb5;  font-size: 36px;  }  .center p {  color: #f9ebb5;  margin-top: 2%;  font-size: 60px;  }  .menu p {  color: #f9ebb5;  font-size: 42px;  }  .menu {  position: relative;  margin: auto;  margin-top: 6%;  width: 150vh;  height: 10vh;  text-align: center;  display: block;  z-index: 10;  }  .left {  float: left;  position: relative;  width: 33%;  }  .ul {  width: 50%;  float: left;  position: relative;  }  .right {  float: right;  position: relative;  width: 33%;  }  a {  position: relative;  color: #f9ebb5;  text-decoration: none;  }  span {  position: relative;  color: #888888;  text-decoration: none;  }  a:hover {  animation: spring 1000ms ease-out;  text-shadow: 0 -1px 0 #f9ebb5;  font-weight: bold;  }  a::before {  content: "";  position: absolute;  width: 100%;  height: 2px;  bottom: 0;  left: 0;  background-color: #f9ebb5;  visibility: hidden;  transform: scaleX(0);  transition: all 0.3s ease-in-out 0s;  }  a:hover::before {  visibility: visible;  transform: scaleX(1);  }  .mainPhotos {  margin-left: 10%;  margin-top: 16%;  width: 50%;  text-align: center;  display: block;  }  .aboutInfo, .main, .newPhotoInfo {  margin: 0 auto;  margin-top: 16%;  width: 50%;  text-align: center;  display: block;  }  p, input, button {  color: #f9ebb5;  }  .entityP {  font-size: 48px;  }  footer {  position: fixed;  width: 100%;  margin: auto;  bottom: 0.3%;  }  footer p {  color: #f9ebb5;  font-size: 14px;  }  .underLogoP, .fullnameP{  font-size: 60px;  }  .mainText {  font-size: 48px;  padding-top: 8%;  }  .backTo {  position: fixed;  top: 2%;  left: 2%;  }  .backTo a, .extraInfoP {  font-size: 32px;  }  .userInfo {  width: 60%;  height: auto;  margin: 5% auto;  }  .userAva {  width: 200px;  height: 200px;  margin-left: auto;  margin-right: auto;  overflow: hidden;  border-radius: 50%;  text-align: center;  border: 2px dashed #f9ebb5;  }  .userAva img, .photoImg img {  margin-left: auto;  margin-right: auto;  height: 100%;  width: auto;  display: block;  }  .fullnameP {  padding-top: 3%;  }  .extraInfoP {  padding-top: 1%;  }  .bio {  width: 100%;  text-align: center;  vertical-align: center;  font-size: 28px;  padding-top: 7%;  }  .photoInfo {  width: 100%;  height: auto;  margin: 8% auto;  display: flex;  }  .photoImg {  width: 960px;  height: 540px;  margin-left: 2%;  overflow: hidden;  border-radius: 5%;  text-align: center;  }  .photoNameP {  font-size: 72px;  }  .photoNameP, .locationP , .likesP, .dislikesP, .photoDateP {  margin-left: 40%;  width: 100%;  }  .locationP {  margin-top: 10%;  font-size: 40px;  }  .likesP, .dislikesP, .photoDateP {  margin-top: 4%;  font-size: 40px;  }  .aboutInfo p {  margin-top: 3%;  font-size: 36px;  }  .search {  height: auto;  margin: auto;  width: 82%;  }  .search .nameInput, .findButton, .prev, .next, .currentPageNumber, .pageNumber, .slash {  height: 40px;  line-height: 40px;  }  .search .nameSearch, .pageSearch {  display: flex;  }  .search .nameInput {  width: 74%;  margin-top: 2%;  font-size: 28px;  text-align: center;  vertical-align: center;  background-color: #16092c;  }  .search .findButton {  width: 24%;  font-size: 22px;  margin-top: 2%;  margin-left: 2%;  text-align: center;  vertical-align: center;  background-color: #16092c;  }  .search .prev, .next, .currentPageNumber, .pageNumber, .slash {  margin-top: 2%;  font-size: 22px;  background-color: #16092c;  text-align: center;  }  .search .prev {  width: 18%;  margin-right: 6.25%;  }  .search .next {  margin-left: 6.25%;  width: 18%;  }  .search .currentPageNumber, .perPageNumber {  height: 40px;  line-height: 40px;  width: 10%;  border: 1px solid #f9ebb5;  }  .search .slash {  width: 5%;  }  .search .pageNumber {  width: 10%;  border: 1px solid #f9ebb5;  }  .pageSearch p {  margin-top: 2%;  font-size: 22px;  }  .perPageNumber {  font-size: 22px;  background-color: #16092c;  margin-top: 1.75%;  margin-left: 5.4%;  }  .prev {  margin-top: 2%;  font-size: 22px;  background-color: #16092c;  }  .next {  margin-top: 2%;  font-size: 22px;  background-color: #16092c;  }  .notFound {  font-size: 36px;  }  .bigDiv {  display: flex;  }  .paginationDiv {  margin-top: 20%;  width: 50%;  }  .search .findButtonFixed {  position: fixed;  right: 4%;  width: 8.85%;  font-size: 22px;  margin-top: 0.7%;  margin-left: 0.7%;  height: 44px;  line-height: 44px;  text-align: center;  background-color: #16092c;  }  .goToNewPhoto {  margin-top: 10%;  font-size: 28px;  background-color: #072904;  border: 1px solid #f9ebb5;  padding: 8px 25px;  }  .specialAContainer {  margin-top: 5%;  }  .newForm div div input {  position: fixed;  margin-left: 12%;  height: 50px;  line-height: 50px;  width: 25%;  background-color: #16092c;  font-size: 28px;  }  .newForm div div {  display: flex;  }  .newForm p {  margin-bottom: 1%;  font-size: 32px;  }  .inputSubmit {  height: 50px;  line-height: 50px;  width: 15%;  background-color: #16092c;  font-size: 28px;  }  .newStructure {  margin-left: 13%;  justify-content: center;  }  .fileUpload::-webkit-file-upload-button {  visibility: hidden;  }  .fileUpload::before {  content: 'Select image';  margin-left: 20px;  height: 40px;  line-height: 40px;  padding: 5px 30px;  font-size: 18px;  border: 1px solid #f9ebb5;  }  .deleteInput {  margin-top: 10%;  font-size: 28px;  background-color: #200000;  border: 1px solid #f9ebb5;  padding: 8px 25px;  margin-left: 55%;  }  .disabled {  pointer-events: none;  cursor: default;  } |

|  |
| --- |
| photoRepository.js |
| const Photo = require('../models/photo');  const JsonStorage = require('../jsonStorage');  class PhotoRepository {  constructor(filePath) {  this.storage = new JsonStorage(filePath);  }  addPhoto(photoModel) {  const photos = this.storage.readItems();  this.storage.incrementNextId();  const newPhoto = new Photo(this.storage.readItems().nextId, photoModel.photoName, photoModel.location, parseInt(photoModel.likes), parseInt(photoModel.dislikes), photoModel.photoDate, photoModel.photoUrl);  photos.items.push(newPhoto);  console.log("Photo with id = " + this.storage.readItems().nextId + " has been added successfully.");  this.storage.writeItems(photos);  // return newPhoto;  return this.storage.readItems().nextId;  }  getPhotos() {  return this.storage.readItems().items;  }  getPagesNumber(page, per\_page, name) {  const page\_size = 3;  const maxPageSize = 3;  if (per\_page) {  if (per\_page > maxPageSize) {  console.log("Error.");  return undefined;  }  }  else {  per\_page = page\_size;  }  if (!page) {  page = 1;  }  const photos = this.getPhotos();  const photosNumber = Number(photos.length)  const offset = per\_page \* (page - 1);  if (photosNumber <= offset) {  console.log("Error.");  return undefined;  }  let resPhotos = [];  let tempPhotosLen = 0;  if (name) {  for (let i = 0; i < photos.length; i++) {  if (photos[i].photoName.includes(name)) {  resPhotos.push(photos[i]);  }  }  tempPhotosLen = resPhotos.length;  resPhotos = resPhotos.slice(offset, offset + per\_page);  }  const currentPhotos = photos.slice(offset, offset + per\_page);  let pagesNumber = 0;  if ((photosNumber / per\_page) - Math.trunc(photosNumber / per\_page) != 0) {  pagesNumber = Math.trunc(photosNumber / per\_page) + 1;  }  else {  pagesNumber = Math.trunc(photosNumber / per\_page);  }  if (name) {  if ((tempPhotosLen / per\_page) - Math.trunc(tempPhotosLen / per\_page) != 0) {  pagesNumber = Math.trunc(tempPhotosLen / per\_page) + 1;  }  else {  pagesNumber = Math.trunc(tempPhotosLen / per\_page);  }  if (pagesNumber == 0) {  pagesNumber = 1;  }  return pagesNumber;  }  if (pagesNumber == 0) {  pagesNumber = 1;  }  return pagesNumber;  }  getPhotosPaginated(page, per\_page, name) {  const page\_size = 3;  const maxPageSize = 3;  if (per\_page) {  if (per\_page > maxPageSize) {  console.log("Error.");  return undefined;  }  }  else {  per\_page = page\_size;  }  if (!page) {  page = 1;  }  const photos = this.getPhotos();  const photosNumber = Number(photos.length);  const offset = per\_page \* (page - 1);  if (photosNumber <= offset) {  console.log("Error.");  return undefined;  }  let resPhotos = [];  if (name) {  for (let i = 0; i < photos.length; i++) {  if (photos[i].photoName.includes(name)) {  resPhotos.push(photos[i]);  }  }  resPhotos = resPhotos.slice(offset, offset + per\_page);  }  const currentPhotos = photos.slice(offset, offset + per\_page);  if (name) {  return resPhotos;  }  return currentPhotos;  }  getPhotoById(photoId) {  const photos = this.storage.readItems().items;  if (photoId < this.storage.readItems().nextId) {  for (const item of photos) {  if (item.id === photoId) {  return new Photo(item.id, item.photoName, item.location, item.likes, item.dislikes, item.photoDate, item.photoUrl);  }  }  return 0;  }  return 0;  }  updatePhoto(photoModel) {  const photos = this.storage.readItems();  if (photoModel.id < this.storage.readItems().nextId) {  let check = false;  let index = 0;  console.log(photoModel.id);  for (const item of photos.items) {  if (parseInt(item.id) === parseInt(photoModel.id)) {  photoModel.id = parseInt(photoModel.id);  photoModel.likes = parseInt(photoModel.likes);  photoModel.dislikes = parseInt(photoModel.dislikes);  console.log("Photo has been updated successfully.");  check = true;  const updatedPhoto = new Photo(photoModel.id, photoModel.photoName, photoModel.location, photoModel.likes, photoModel.dislikes, photoModel.photoDate);  photos.items[index] = updatedPhoto;  break;  }  index++;  console.log(index);  }  if (check === false) {  console.log("Photo with such id doesn't exist.");  }  }  else {  console.log("Photo with such id doesn't exist.");  }  this.storage.writeItems(photos);  return photoModel;  }  deletePhoto(photoId) {  let result;  const photos = this.storage.readItems();  if (photoId < this.storage.readItems().nextId) {  let check = false;  let index = 0;  for (const item of photos.items) {  if (parseInt(item.id) === parseInt(photoId)) {  check = true;  result = photos.items[index];  photos.items.splice(index, 1);  console.log("Photo has been deleted successfully.");  break;  }  index++;  }  if (check === false) {  console.log("Photo with such id doesn't exist1.");  }  }  else {  console.log("Photo with such id doesn't exist.");  }  this.storage.writeItems(photos);  return result;  }  };  module.exports = PhotoRepository; |

|  |
| --- |
| photoRouter.js |
| const router = require('express').Router();  const photoController = require('./../controllers/photoController');  router.get("/new", (req, res) => { res.status(200).render('new') })  router.get("/:id", photoController.getPhotoById)  router.put("/", photoController.updatePhoto)  router.post("/:id", photoController.deletePhoto)  router.get("/", photoController.getPhotos)  router.post("/", photoController.addPhoto)  module.exports = router; |

|  |
| --- |
| photoController.js |
| const path = require('path');  const PhotoRepository = require('./../repositories/photoRepository');  const photoRepository = new PhotoRepository(path.resolve(\_\_dirname, '../data/photos.json'));  const fs = require('fs');  const MediaRepository = require('./../repositories/mediaRepository');  const Photo = require('../models/photo');  const mediaRepository = new MediaRepository(path.resolve(\_\_dirname, '../data/media'));  module.exports = {  async getPhotoById(req, res) {  try {  console.log(req.params.id);  photo = photoRepository.getPhotoById(parseInt(req.params.id));  if (photo) {  res.status(200).render('photo', {photo: photo});  }  else {  res.status(404).send({photo: null, message: "Not found."});  }  } catch (err) {  console.log(err.message);  res.status(500).send({photo: null, message: 'Server error.'});  }  },  async addPhoto(req, res) {  console.log(req.files);  const fileFormat = req.files['photoUrl'].mimetype.split('/')[1];  fs.writeFileSync(path.resolve(\_\_dirname, '../data/media/' + mediaRepository.getNextId() + '.' + fileFormat), req.files['photoUrl'].data, (err) => {  if (err) {  console.log("Can't load this photo.");  }  })  const photoUrl = '/media/' + mediaRepository.getNextId() + '.' + fileFormat;  const newPhoto = new Photo(mediaRepository.getNextId(), req.body.photoName, req.body.location, Number(req.body.likes), Number(req.body.dislikes), req.body.photoDate, photoUrl)  const newId = photoRepository.addPhoto(newPhoto) - 1;  console.log(newId);  mediaRepository.incrementId();  res.redirect('/photos/' + newId);  },  async updatePhoto(req, res) {  try {  if (!req.body.photoName || !req.body.likes || !req.body.dislikes) {  res.status(400).send({message: 'Bad request.'});  }  photo = photoRepository.updatePhoto(req.body);  if (photo) {  console.log(photo);  res.status(200).send({photo: photo, message: "Success."});  }  else {  res.status(404).send({photo: null, message: "Not found."});  }  } catch (err) {  console.log(err.message);  res.status(500).send({photo: null, message: 'Server error.'});  }  },  async deletePhoto(req, res) {  photoRepository.deletePhoto(Number(req.params.id));  res.redirect('/photos');  },  async getPhotos(req, res) {  try {  photos = photoRepository.getPhotosPaginated(Number(req.query.page), Number(req.query.per\_page), req.query.name);  pagesNumber = photoRepository.getPagesNumber(Number(req.query.page), Number(req.query.per\_page), req.query.name);  let page = req.query.page;  let name = req.query.name;  if (!page) page = 1;  else page = Number(page);  pages = { currentPage: Number(page) }  if (page != 1) pages.prevPage = page - 1;  if (page != pagesNumber) pages.nextPage = page + 1;  if (name) pages.namePage = name;  if (photos) {  res.status(200).render('photos', {photos: photos, pagesNumber: pagesNumber, pages: pages, photoDisabled: "disabled"});  }  else {  res.status(404).send({photos: null, message: "Not found."});  }  } catch (err) {  console.log(err.message);  res.status(500).send({photos: null, message: 'Server error.'});  }  },  }; |

|  |
| --- |
| footer.mst |
| <footer>  <p>Copyright 2020. All rights reserved.</p>  </footer> |

|  |
| --- |
| head.mst |
| <head>  <meta charset="utf-8">  <title>Photos Inc.</title>  <link rel="icon" href="/images/logo2.png" type="image/x-icon">  <link rel="preconnect" href="https://fonts.gstatic.com">  <link href="https://fonts.googleapis.com/css2?family=Kalam&display=swap" rel="stylesheet">  <link href="https://fonts.googleapis.com/css?family=Raleway:400,400i,700" rel="stylesheet">  <link rel="stylesheet" href="/stylesheets/style.css">  </head> |

|  |
| --- |
| header.mst |
| <header>  <div class="menu">  <div class="left">  <div class="ul">  <p><a class="{{disabled}}" class="home" href="/">Home</a></p>  </div>  <div class="ul">  <p><a class="{{userDisabled}}" class="users" href="/users">Users</a></p>  </div>  </div>  <div class="right">  <div class="ul">  <p><a class="{{photoDisabled}}" class="photos" href="/photos">Photos</a></p>  </div>  <div class="ul">  <p><a class="{{aboutDisabled}}" class="about" href="/about">About</a></p>  </div>  </div>  </div>  <div class="center">  <img src="/images/logo7.png" alt="logo" width=25% height=25%>  </div>  </header> |

|  |
| --- |
| about.mst |
| <!DOCTYPE html>  <html>  {{> head}}  <body>  {{> header}}  <div class="aboutInfo">  <p>Creator: Mefta007</p>  <p>Created: 20.11.20</p>  <p>Contact info:</p>  <p>Gmail: qwerty@gmail.com</p>  <p>Address: Leuss str.</p>  <p>Phone: 023-117-89-91</p>  </div>  {{> footer}}  </body>  </html> |

|  |
| --- |
| index.mst |
| <!DOCTYPE html>  <html>  {{> head}}  <body>  {{> header}}  <div class="main">  <p class="underLogoP">  Making your life colourful!  </p>  <p class="mainText">  Feel free to post any photos here and share them among your friends!  </p>  </div>  {{> footer}}  </body>  </html> |

|  |
| --- |
| new.mst |
| <!DOCTYPE html>  <html>  {{> head}}  <body>  {{> header}}  <div class="newPhotoInfo">  <form class="newForm" method="POST" action="/photos" enctype="multipart/form-data">  <p>New photo</p>  <div class="newStructure">  <div><p>Photo name:</p><input type="text" name="photoName" required></div><br>  <div><p>Location:</p><input type="text" name="location" required></div><br>  <div><p>Likes:</p><input type="number" name="likes" required></div><br>  <div><p>Dislikes:</p><input type="number" name="dislikes" required></div><br>  <div><p>Date:</p><input type="date" name="photoDate" required></div><br>  <div><p>Photo (file):</p><input class="fileUpload" type="file" name="photoUrl" required></div><br>  <input class="inputSubmit" type="submit" value="Submit">  </div>  </form>  </div>  {{> footer}}  </body>  </html> |

|  |
| --- |
| photo.mst |
| <!DOCTYPE html>  <html>  {{> head}}  <body>  {{#photo}}  <div class="backTo">  <a href="../photos">Back to Photos</a>  </div>  <div class="photoInfo">  <div class="photoImg">  <img src="{{photoUrl}}">  </div>  <div class="otherPhotoInfo">  <p class="photoNameP"><b>{{photoName}}</b></p>  <p class="locationP">Location: <b>{{location}}</b></p>  <p class="likesP">Likes: {{likes}}</p>  <p class="dislikesP">Dislikes: {{dislikes}}</p>  <p class="photoDateP">Date: {{photoDate}}</p>  <form method="POST" action="/photos/{{id}}">  <input class="deleteInput" type="submit" value="Delete photo">  </form>  </div>  </div>  {{/photo}}  {{> footer}}  </body>  </html> |

|  |
| --- |
| photos.mst |
| <!DOCTYPE html>  <html>  {{> head}}  <body>  {{> header}}  <div class="bigDiv">  <div class="mainPhotos">  <p class="entityP">Photos</p>  <table>  <tr>  <th>Name</th>  <th>Location</th>  <th>Date</th>  </tr>  <tr>  {{#photos}}  <tr>  <td><a href="/photos/{{id}}">{{photoName}}</a></td>  <td>{{location}}</td>  <td>{{photoDate}}</td>  </tr>  {{/photos}}  </tr>  </table>  {{^photos}}  <p class="notFound">No photos found :(<p>  {{/photos}}  </div>  <div class="paginationDiv">  <div class="search">  <form method="get" action="/photos">  <div class="nameSearch">  {{#pages}}  {{#namePage}}  <input class="nameInput" type="text" name="name" placeholder="Enter photo name" value={{namePage}}>  {{/namePage}}  {{^namePage}}  <input class="nameInput" type="text" name="name" placeholder="Enter photo name">  {{/namePage}}  <button class="findButtonFixed">Find</button>  {{/pages}}  </div>  </form>  <form method="get" action="/photos">  <div class="pageSearch">  {{#pages}}  {{#prevPage}}  {{#namePage}}  <a class="prev" href="/photos?page={{prevPage}}&name={{namePage}}">Previous</a>  {{/namePage}}  {{^namePage}}  <a class="prev" href="/photos?page={{prevPage}}">Previous</a>  {{/namePage}}  {{/prevPage}}  {{^prevPage}}  <span class="prev">Previous</span>  {{/prevPage}}  <input type="number" class="currentPageNumber" name="page" min="1" max="{{pagesNumber}}" value={{currentPage}}>  {{#namePage}}  <input type="hidden" name="name" value={{namePage}}>  {{/namePage}}  <p class="slash"> / </p>  <p class="pageNumber">{{pagesNumber}}</p>  {{#nextPage}}  {{#namePage}}  <a class="next" href="/photos?page={{nextPage}}&name={{namePage}}">Next</a>  {{/namePage}}  {{^namePage}}  <a class="next" href="/photos?page={{nextPage}}">Next</a>  {{/namePage}}  {{/nextPage}}  {{^nextPage}}  <span class="next">Next</span>  {{/nextPage}}  <button class="findButtonFixed">Find</button>  {{/pages}}  </div>  </form>  <div class="specialAContainer">  <a class="goToNewPhoto" href="/photos/new">New Photo</a>  </div>  </div>  </div>  </div>  {{> footer}}  </body>  </html> |

|  |
| --- |
| user.mst |
| <!DOCTYPE html>  <html>  {{> head}}  <body>  {{#user}}  <div class="backTo">  <a href="../users">Back to Users</a>  </div>  <div class="userInfo">  <div class="userAva">  <img src="{{avaUrl}}">  </div>  <div class="userBio">  <p class="fullnameP">{{fullname}}</p>  <p class="extraInfoP">({{login}}) {{registeredAt}}</p>  <p class="bio">{{bio}}</p>  </div>  </div>  {{/user}}  {{> footer}}  </body>  </html> |

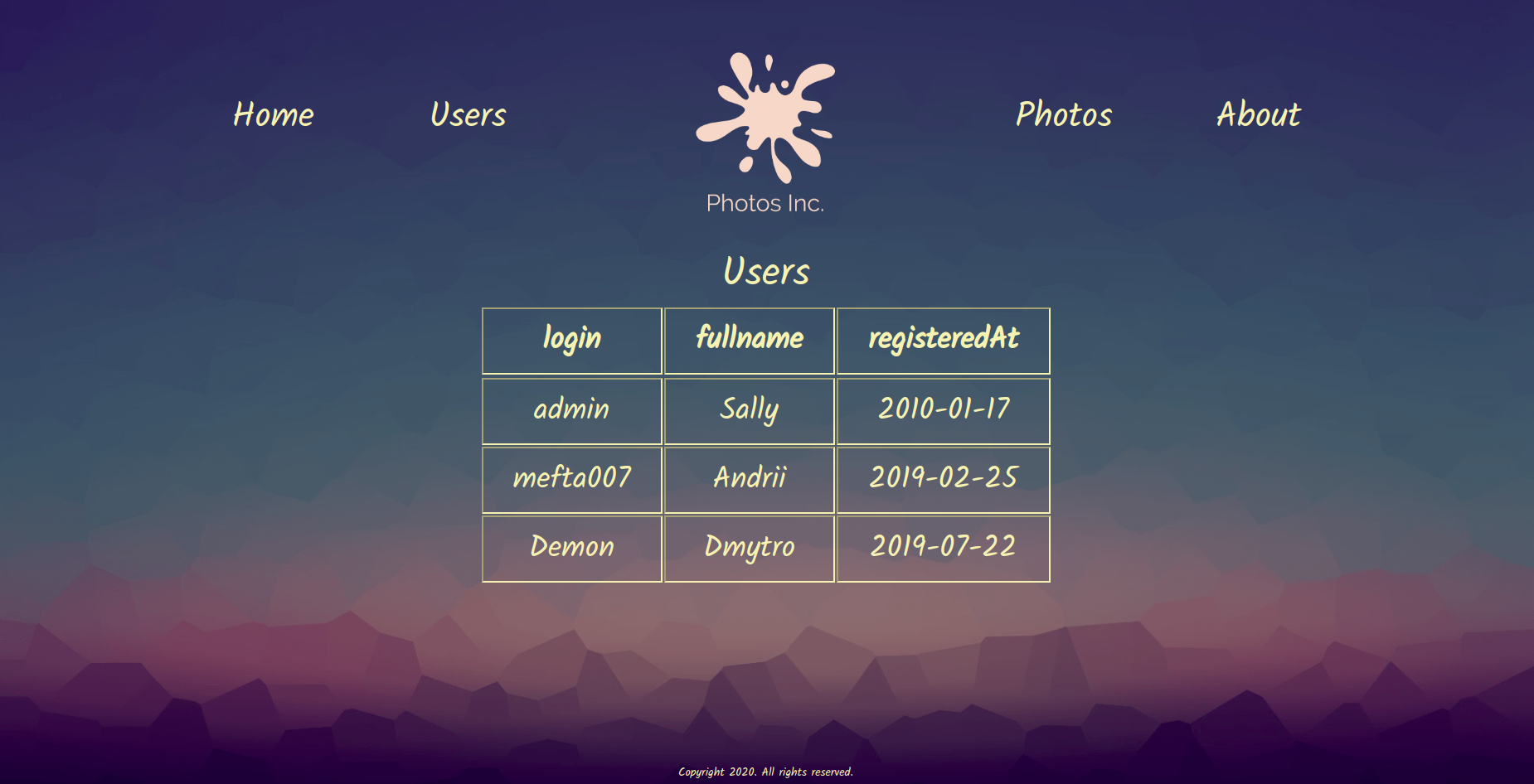
|  |
| --- |
| users.mst |
| <!DOCTYPE html>  <html>  {{> head}}  <body>  {{> header}}  <div class="main">  <p class="entityP">Users</p>  <table>  <tr>  <th>login</th>  <th>fullname</th>  <th>registeredAt</th>  </tr>  <tr>  {{#users}}  <tr>  <td><a href="/users/{{id}}">{{login}}</a></td>  <td>{{fullname}}</td>  <td>{{registeredAt}}</td>  </tr>  {{/users}}  </tr>  </table>  </div>  {{> footer}}  </body>  </html> |

**Сторінки**

**Index.html**

****

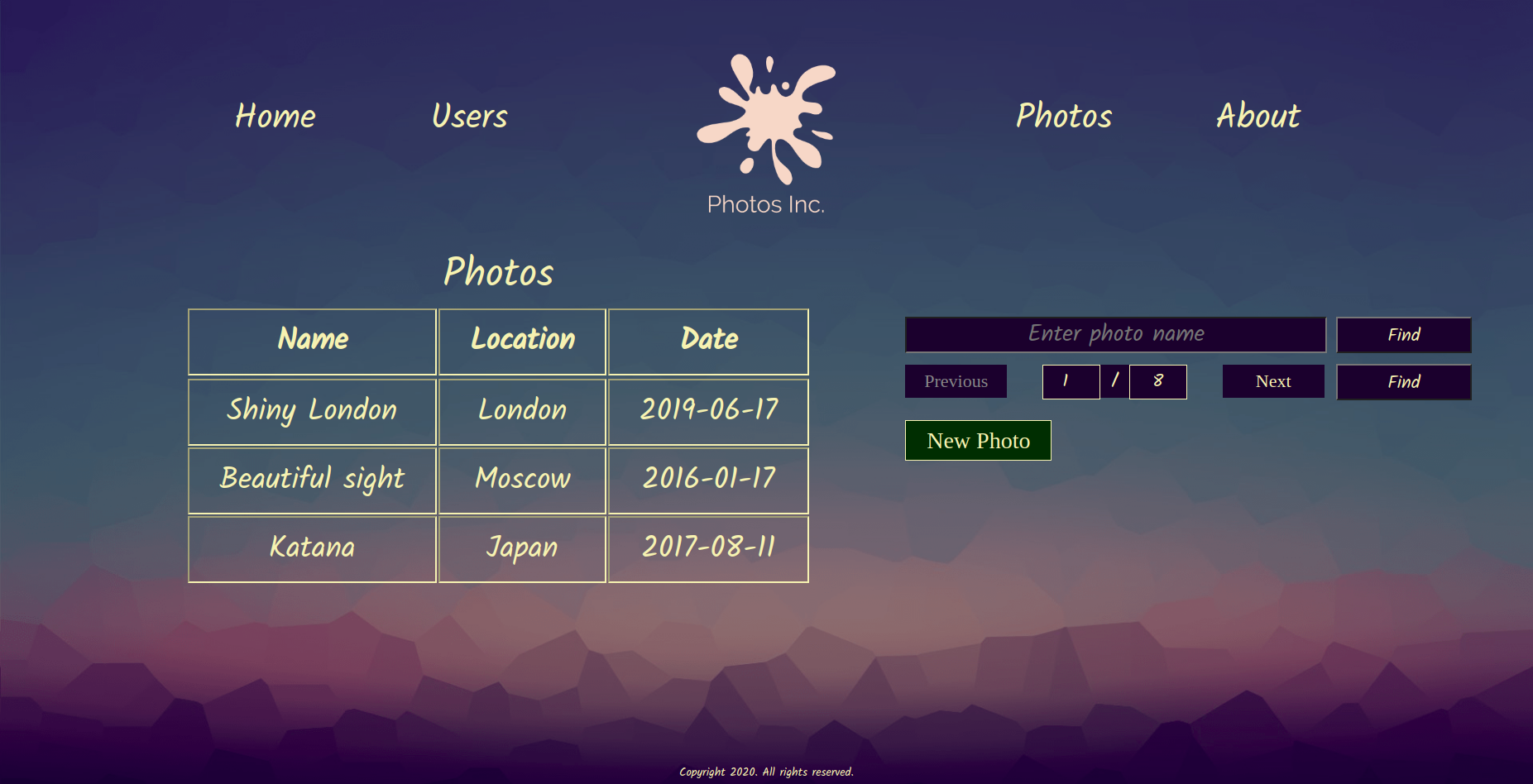
**Users.html**

****

**User.html**

****

**Photos.html**

****

**Photo.html**

****

**New.html (new photo)**

****

**About.html**

****

**Висновки**

Виконавши цю лабораторну роботу ми навчилися створювати веб-сайт, що надає CRUD доступ до ресурсів веб-сервера.  
Також ми ознайомилися з шаблонізаторами веб-сторінок для генерації контенту та навчились створювати HTML-форми для взаємодії користувача із веб-сайтом та обробляти на сервері внесені користувачем дані.