**Липецкий государственный технический университет**  
Факультет автоматизации и информатики  
  
Кафедра Автоматизированных систем управления

Лабораторная работа №1

по курсу «Организация графических систем»

Система наложения фильтров на загруженное изображение

|  |  |
| --- | --- |
| Студенты | Попов А.Д. |
| Группа М-АС-21 |  |
| Руководитель  доцент | Кургасов В.В. |

Липецк 2022 г.

**Задание**

Применение не менее 3-х фильтров к загруженному изображению и экспорт результата в иной графический формат. Наличие графического меню пользователя. Обработка на CPU.

**Теоретические сведения**

Система реализованы с использованием языка JavaScript.

Фронтенд написан на чистом JS без использования сторонних инструментов. Бекэнд написан с использованием Node.js (серверная платформа для работы с JS), Express (фреймворк для Node.js) и библиотеки sharp, которая отвечает за редактирование изображений.

Веб-приложения развернуто с использованием Docker.

**Основная часть**

**Frontend**

index.html

<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <link rel="stylesheet" href="style.css">  
 <title>Наложение фильтров</title>  
</head>  
<body>  
 <div class="block">  
 <div class="block\_\_image-div"></div>  
 <span>Загрузка...</span>  
 <input class="block\_\_input" type="file" id="input" onchange="*handleFiles*(this.files)" accept=".png,.jpeg,.jpg">  
 <a href="" class="download" download>Скачать</a>  
 </div>  
</body>  
</html>  
  
<script src="https://html2canvas.hertzen.com/dist/html2canvas.js"></script>  
<script src="index.js"></script>

index.js

|  |
| --- |
| async function *handleFiles*(files) {  const file = files[0];  const div = document.querySelector('.block\_\_image-div');  const span = document.querySelector('span');  let img = document.querySelector('.block\_\_image-div-obj');   if (!img) {  img = document.createElement("img");  img.classList.add("block\_\_image-div-obj");  img.file = file;  div.append(img);  }  div.style.display = 'none';  span.style.display = 'block';   img.addEventListener('load', () => {  span.style.display = 'none';  div.style.display = 'block';  });   await *sendFile*(img, file); }  async function *sendFile*(img, file) {  const fd = new FormData();  fd.append('uploadFile', file);  let newFile = undefined  await *fetch*('http://45.130.151.203:3000/', {  method: 'POST',  body: fd  })  .then((resp) => {  return resp.blob();  })  .then((blob) => {  newFile = blob;  });  const a = document.querySelector('.download');  const url = window.URL.*createObjectURL*(newFile);  a.href = url;  a.click();  window.URL.*revokeObjectURL*(url);  *loadImg*(newFile, img); }  function *loadImg*(file, div) {  const reader = new FileReader();  reader.onload = (  function(aImg) {  return function(e) {  aImg.src = e.target.result;  };  }  )(div);   reader.readAsDataURL(file); } |

**Backend**

index.js

|  |
| --- |
| const express = require('express'); const *fileUpload* = require('express-fileupload'); const bodyParser = require('body-parser'); const fs = require('fs'); const *cors* = require('cors'); const *sharp* = require('sharp'); const app = express(); const port = 3000;  app.use(*cors*()); app.use(*fileUpload*(undefined)); app.use(bodyParser.urlencoded({ limit: '10mb', extended: true }));  app.post('/', async (req, res) => {  try {  const { uploadFile = undefined} = req?.files;  if (uploadFile) {  const uploadPath = \_\_dirname + "/uploads/" + uploadFile.name;  await uploadFile.mv(uploadPath);   const sharpRes = await sharpImage(uploadFile.name.split('.')[0], uploadPath);  if (sharpRes.status) {  res.sendFile(sharpRes.fileName);  } else {  res.status(500).send({ message: 'Ошибка сохранения файла' })  }  } else {  res.status(500).send({ message: 'Ошибка чтения файла!' });  }  } catch(err) {  console.log(err);  res.sendStatus(500);  } });  async function sharpImage(name, data) {  try {  const metadata = await *sharp*(data).metadata();  if (metadata) {  const { format } = metadata;  const fileName = `filtered/${name}.${format === 'jpeg' ? 'png' : 'jpeg'}`;   await *sharp*(data)  .grayscale()  .blur(10)  .negate()  .toFile(fileName);   return { status: true, fileName: `${\_\_dirname}/${fileName}` };  }  } catch(err) {  console.log(err);  return { status: false };  } }  app.listen(port, () => {  console.log(`Example app listening on port ${port}`) }); |

**Алгоритм работы программы**

Пользователь загружает в систему изображение, которое по средствам HTTP-запроса передается на сервер, где уже с помощью библиотеки sharp на наго накладывается 3 фильтра (размытие, оттенки-серого и негатив) и меняется расширение изображения (png становится jpeg, а jpeg наоборот png), после чего уже отредактированное изображение возвращается на клиент и автоматически скачивается на устройство пользователя.

**Результат работы**

****

Рисунок - Исходное изображение



Рисунок - Обработанное изображение

**Вывод**

В результате лабораторной работы была разработана система в виде веб-приложения которая редактирует загруженное изображение посредством наложения фильтров и изменения расширения.

Ссылка на приложение: <http://45.130.151.203:8080/>

Ссылка на github: <https://github.com/DrPoseidon/lab1-filters>