

**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**(РУТ (МИИТ)**

|  |
| --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Кафедра «Управление и защита информации»

**Отчет по лабораторной работе №6**

**по дисциплине**

«Web программирование»

Вариант 2

**Выполнил:** студенты

группы ТКИ-541

Порхун Д.Д. , Макаров Д.И.

**Проверил:** доцент кафедры УиЗИ, к.т.н. Сафронов А.И.

**Москва 2024 г.**

Оглавление

[Цель работы 3](#_Toc181981334)

[Формулировка задачи 3](#_Toc181981335)

[Таблица соответствия переменных и методов, используемых в web-приложении 3](#_Toc181981336)

[Содержательная часть по наблюдателям 6](#_Toc181981337)

[Вывод 15](#_Toc181981338)

# Цель работы

Создать веб-приложение для загрузки и анализа изображений в формате JPEG, используя HTML5 Canvas и RGB модель. Приложение обеспечит отображение изображения и возможность инвертирования цветов для анализа цветовых компонентов.

# Формулировка задачи

– локально (LSPWA), не прибегая к инструментарию Node.js и npm (Node Package Manager);

* **Задание :**

Создать веб-интерфейс для загрузки изображений в формате JPEG и обработки их с помощью технологии HTML5 SVG. Приложение должно обеспечивать возможность инвертирования цветов в модели RGB, позволяя тем самым анализировать цветовую составляющую изображений.

Модель – RGB

Объект – SVG

Реализация – Суммарная / Каждый канал – своя опция

Режим - Цветовая инверсия

Режим построения гистограммы - Прямой (слева-направо)

# Таблица соответствия переменных и методов, используемых в web-приложении

**Задание :**

|  |  |  |
| --- | --- | --- |
| **Переменная** | **Описание** | **Тип данных** |
| imageSrc | URL исходного изображения, загруженного пользователем. | string |
| invertedImageSrc | URL инвертированного изображения, созданного на основе imageSrc. | string |
| histogramData | Объект, содержащий массивы для хранения данных гистограммы для каждого цветового канала. | object |
| histogramData.red | Массив, содержащий количество пикселей для каждого возможного значения красного канала (0-255). | array |
| histogramData.green | Массив, содержащий количество пикселей для каждого возможного значения зеленого канала (0-255). | array |
| histogramData.blue | Массив, содержащий количество пикселей для каждого возможного значения синего канала (0-255). | array |
| histogramData.value | Массив, содержащий количество пикселей для каждого возможного среднего значения RGB (0-255). | array |
| optState | Строка, определяющая выбранный режим отображения гистограммы (суммарная или по каналам). | string |
| file | Объект файла, выбранный пользователем для загрузки. | File |
| reader | Объект FileReader, используемый для чтения содержимого файла. | FileReader |
| img | Объект изображения, созданный для загрузки и обработки изображения. | HTMLImageElement |
| canvas | Элемент <canvas>, используемый для временного рисования изображения и извлечения данных пикселей. | HTMLCanvasElement |
| ctx | Контекст рисования для элемента <canvas>. | CanvasRenderingContext2D |
| imageData | Объект, содержащий данные пикселей изображения, полученные из контекста canvas. | ImageData |
| maxVal | Максимальное значение в данных гистограммы, используемое для нормализации высоты столбцов. | number |
| barWidth | Ширина каждого столбца гистограммы, вычисленная на основе ширины SVG и количества значений. | number |
| barHeight | Высота текущего столбца гистограммы, вычисленная на основе значений в channelData. | number |
| offsetY | Смещение по оси Y, используемое для корректного отображения столбцов на SVG. | number |
| rect | Элемент <rect> для создания столбца гистограммы в SVG. | SVGRectElement |

# Содержательная часть по наблюдателям

|  |
| --- |
| <!DOCTYPE html>  <html lang="ru">  <head>  <meta charset="UTF-8">  <title>Анализ изображения</title>  <style>  .card {  margin: 20px;  padding: 10px;  border: 1px solid #cccccc;  border-radius: 5px;  box-shadow: 2px 2px 5px rgba(0,0,0,0.2);  }  img {  max-width: 100%;  height: auto;  margin-top: 10px;  }  svg {  width: 100%;  height: 150px;  border: 1px solid black;  margin-top: 10px;  }  </style>  </head>  <body>  <div id="hist" class="card">  <h1>Анализатор изображения</h1>  <p>Выберите изображение из файла</p>    <!-- Кнопка выбора файла перемещена вверх -->  <input type="file" @change="handleFileChange($event)" accept="image/jpeg">  <img :src="invertedImageSrc" v-if="invertedImageSrc">  <h2>Настройка режимов гистограммы</h2>  <div>  <input id="val" type="radio" name="mode" value="val" v-model="optState" @change="drawHistogram">  <label for="val">Суммарная</label>  <input id="col" type="radio" name="mode" value="col" v-model="optState" @change="drawHistogram">  <label for="col">Каждый канал</label>  </div>  <h2>Область гистограммы</h2>  <svg ref="svgHistogram"></svg>  </div>  <script src="https://unpkg.com/vue@2.6.14/dist/vue.js"></script>  <script>  new Vue({  el: '#hist',  data: {  imageSrc: null,  invertedImageSrc: null,  histogramData: {  red: new Array(256).fill(0),  green: new Array(256).fill(0),  blue: new Array(256).fill(0),  value: new Array(256).fill(0),  },  optState: 'col',  },  methods: {  handleFileChange(event) {  const file = event.target.files[0];  if (file && file.type === 'image/jpeg') {  const reader = new FileReader();  reader.onload = (e) => {  this.imageSrc = e.target.result;  this.processImage();  };  reader.readAsDataURL(file);  } else {  alert('Пожалуйста, загрузите изображение в формате JPEG.');  }  },  processImage() {  const img = new Image();  img.onload = () => {  const canvas = document.createElement('canvas');  const ctx = canvas.getContext('2d');  canvas.width = img.width;  canvas.height = img.height;  ctx.drawImage(img, 0, 0);  const imageData = ctx.getImageData(0, 0, canvas.width, canvas.height);    // Инверсия цветов  this.invertColors(imageData);    // Создание нового изображения с инвертированными цветами  ctx.putImageData(imageData, 0, 0);  this.invertedImageSrc = canvas.toDataURL(); // Сохраняем инвертированное изображение  this.calculateHistogram(imageData);  };  img.src = this.imageSrc;  },  invertColors(imageData) {  for (let i = 0; i < imageData.data.length; i += 4) {  imageData.data[i] = 255 - imageData.data[i]; // Red  imageData.data[i + 1] = 255 - imageData.data[i + 1]; // Green  imageData.data[i + 2] = 255 - imageData.data[i + 2]; // Blue  }  },  calculateHistogram(imageData) {  this.resetHistogramData();  for (let i = 0; i < imageData.data.length; i += 4) {  const r = imageData.data[i];  const g = imageData.data[i + 1];  const b = imageData.data[i + 2];  const value = Math.floor((r + g + b) / 3);  this.histogramData.red[r]++;  this.histogramData.green[g]++;  this.histogramData.blue[b]++;  this.histogramData.value[value]++;  }  this.$nextTick(this.drawHistogram);  },  drawHistogram() {  const svg = this.$refs.svgHistogram;  svg.innerHTML = ''; // Очищаем предыдущие данные  const width = svg.clientWidth;  const height = svg.clientHeight;  if (this.optState === 'col') {  this.drawHistogramChannel(svg, width, height / 3, this.histogramData.red, 'red', 0);  this.drawHistogramChannel(svg, width, height / 3, this.histogramData.green, 'green', height / 3);  this.drawHistogramChannel(svg, width, height / 3, this.histogramData.blue, 'blue', 2 \* height / 3);  } else {  this.drawHistogramChannel(svg, width, height, this.histogramData.value, 'gray', 0);  }  },  drawHistogramChannel(svg, width, height, channelData, color, offsetY) {  const maxVal = Math.max(...channelData);  const barWidth = width / channelData.length;  for (let i = 0; i < channelData.length; i++) {  const barHeight = (channelData[i] / maxVal) \* height;  const rect = document.createElementNS("http://www.w3.org/2000/svg", "rect");  rect.setAttribute('x', i \* barWidth);  rect.setAttribute('y', height + offsetY - barHeight);  rect.setAttribute('width', barWidth);  rect.setAttribute('height', barHeight);  rect.setAttribute('fill', color);  svg.appendChild(rect);  }  },  resetHistogramData() {  Object.keys(this.histogramData).forEach(channel => {  this.histogramData[channel].fill(0);  });  },  }  });  </script>  </body>  </html> |
| Код программы Задания |

|  |
| --- |
|  |
| Рисунок 1 Отображение в браузере задание |
|  |
| Рисунок 2 Отображение в браузере задания в изменённом состоянии |
|  |
| Рисунок 3 Сеть Петри к заданию |

# Вывод

Веб-приложение для анализа изображений успешно обеспечивает загрузку, отображение и инверсию цветов в формате JPEG, используя технологии HTML5 SVG и Vue.js. Функционал приложения позволяет детально анализировать цветовые компоненты изображения, делая его полезным инструментом для визуального контента.