

**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**

« Анализ изображения при использовании инструментов»

**по дисциплине**

«Web программирование»

**Вариант 14**

**Выполнил:**

Студенты группы ТКИ-541

Моисеенко М.А.

Земсков Н.А.

**Проверил:**

доцент кафедры УиЗИ, к.т.н.

Сафронов А.И.

**Москва 2024 г.**

Оглавление

[1. Цель работы 3](#_Toc182658162)

[2. Формулировка задачи 3](#_Toc182658163)

[3. Таблица соответствия переменных и методов 4](#_Toc182658164)

[4. Содержательная часть 5](#_Toc182658166)

[4.1. Код web-приложения 5](#_Toc182658168)

[4.2. Отображение в браузере 11](#_Toc182658169)

[4.3. Сеть Петри 13](#_Toc182658170)

[5. Вывод 13](#_Toc182658171)

# Цель работы

Средствами SVG или Canvas под управлением фреймворка Vue.js построить программное, браузерное web-обеспечение, реализующее анализ подгруженного изображения (\*.jpeg) по варианту.

# Формулировка задачи

Средством Canvas, с использованием цветовой модели CMYK, под управлением фреймворка Vue.js построить программное, браузерное web‑обеспечение, реализующее анализ подгруженного изображения в формате .jpeg и вывод гистограммы. Режим вывода загруженного изображения сделать обратным (инверсия), режим построения гистограммы сделать прямым (слева-направо). Реализовать управление гистограммой с помощью выбора соответствующей опции (см. рисунок 1).

|  |
| --- |
| Рисунок 1 – Вариант 14 |

# Таблица соответствия переменных и методов



|  |  |  |
| --- | --- | --- |
| Описание | Метод/Переменная | Тип |
| Текущий режим отображения гистограммы | histogramMode: | Переменная |
| Данные изображения | imageData: | Переменная |
| Данные гистограммы для каналов | histogramData: | Переменная |
| Ссылка на холст для изображения | $refs.imageCanvas: | Переменная (ссылка) |
| Ссылка на холст для гистограммы | $refs.histogramCanvas | Переменная (ссылка) |
| Обработка выбора файла | handleFileChange(event) | Метод |
| Обработка изображения после загрузки | processImage(img) | Метод |
| Применение инверсии цветов на изображении | applyInversion(imageData) | Метод |
| Расчет гистограммы | calculateHistogram() | Метод |
| Отрисовка гистограммы | drawHistogram() | Метод |
| Отрисовка одного канала гистограммы | drawChannel(data, color, maxValue, ctx, canvas) | Метод |

# Содержательная часть



## Код web-приложения

<template>

<div class="image-analyzer">

<div class="card">

<p>Выберите изображение из файла</p>

<input type="file" @change="handleFileChange" accept="image/jpeg" />

<br />

<canvas ref="imageCanvas" width="500" height="500"></canvas>

</div>

<div class="card">

<p>Гистограмма</p>

<div>

<input type="radio" value="value" v-model="histogramMode" @change="drawHistogram" />

<label>Значение</label>

<input type="radio" value="color" v-model="histogramMode" @change="drawHistogram" />

<label>Цвет</label>

</div>

<canvas ref="histogramCanvas" width="256" height="150" style="border: solid 1px #999;"></canvas>

</div>

</div>

</template>

<script>

export default {

data() {

return {

histogramMode: 'value',

imageData: null,

histogramData: {

cyan: new Array(256).fill(0),

magenta: new Array(256).fill(0),

yellow: new Array(256).fill(0),

black: new Array(256).fill(0),

},

};

},

methods: {

handleFileChange(event) {

const file = event.target.files[0];

if (file) {

const img = new Image();

img.src = URL.createObjectURL(file);

img.onload = () => {

this.processImage(img);

};

}

},

processImage(img) {

const canvas = this.$refs.imageCanvas;

const ctx = canvas.getContext("2d");

canvas.width = 500;

canvas.height = 500;

ctx.clearRect(0, 0, canvas.width, canvas.height);

ctx.drawImage(img, 0, 0, canvas.width, canvas.height);

const imageData = ctx.getImageData(0, 0, canvas.width, canvas.height);

this.imageData = imageData.data;

this.applyInversion(imageData);

this.calculateHistogram();

this.drawHistogram();

},

applyInversion(imageData) {

for (let i = 0; i < imageData.data.length; i += 4) {

imageData.data[i] = 255 - imageData.data[i]; // R

imageData.data[i + 1] = 255 - imageData.data[i + 1]; // G

imageData.data[i + 2] = 255 - imageData.data[i + 2]; // B

}

const canvas = this.$refs.imageCanvas;

const ctx = canvas.getContext("2d");

ctx.putImageData(imageData, 0, 0);

},

calculateHistogram() {

this.histogramData.cyan.fill(0);

this.histogramData.magenta.fill(0);

this.histogramData.yellow.fill(0);

this.histogramData.black.fill(0);

for (let i = 0; i < this.imageData.length; i += 4) {

const r = this.imageData[i];

const g = this.imageData[i + 1];

const b = this.imageData[i + 2];

const c = 255 - r;

const m = 255 - g;

const y = 255 - b;

const k = Math.min(c, m, y);

this.histogramData.cyan[c]++;

this.histogramData.magenta[m]++;

this.histogramData.yellow[y]++;

this.histogramData.black[k]++;

}

},

drawHistogram() {

const canvas = this.$refs.histogramCanvas;

const ctx = canvas.getContext("2d");

ctx.clearRect(0, 0, canvas.width, canvas.height);

// Если режим "Цвет"

if (this.histogramMode === 'color') {

const combined = this.histogramData.cyan.map((v, i) =>

v + this.histogramData.magenta[i] + this.histogramData.yellow[i] + this.histogramData.black[i]

);

const maxValue = Math.max(...combined); // Максимум для комбинированного канала

this.drawChannel(combined, "gray", maxValue, ctx, canvas);

} else {

// Если режим "Значение"

['cyan', 'magenta', 'yellow', 'black'].forEach((channel, index) => {

const maxValue = Math.max(...this.histogramData[channel]); // Максимум для каждого канала

const color = ['cyan', 'magenta', 'yellow', 'black'][index];

this.drawChannel(this.histogramData[channel], color, maxValue, ctx, canvas);

});

}

// Добавляем направляющую яркости в верхней части гистограммы

this.drawGradientLine(ctx, canvas);

},

drawChannel(data, color, maxValue, ctx, canvas) {

ctx.fillStyle = color;

data.forEach((value, i) => {

const height = (value / maxValue) \* canvas.height; // Масштабируем значение

ctx.fillRect(i, canvas.height - height, 1, height);

});

},

drawGradientLine(ctx, canvas) {

// Создаем градиент от черного к белому

const gradient = ctx.createLinearGradient(0, 0, canvas.width, 0);

gradient.addColorStop(0, 'black');

gradient.addColorStop(1, 'white');

// Рисуем горизонтальную линию градиента в верхней части холста

ctx.beginPath();

ctx.moveTo(0, 0); // Начало линии в верхнем левом углу

ctx.lineTo(canvas.width, 0); // Конец линии в верхнем правом углу

ctx.strokeStyle = gradient; // Устанавливаем градиент как стиль линии

ctx.lineWidth = 10; // Увеличиваем толщину линии для лучшей видимости

ctx.stroke();

}

},

};

</script>

<style>

.image-analyzer {

display: flex;

flex-direction: column;

align-items: center;

}

.card {

margin: 20px;

padding: 10px;

border: 1px solid #ddd;

border-radius: 5px;

}

</style>

## Отображение в браузере

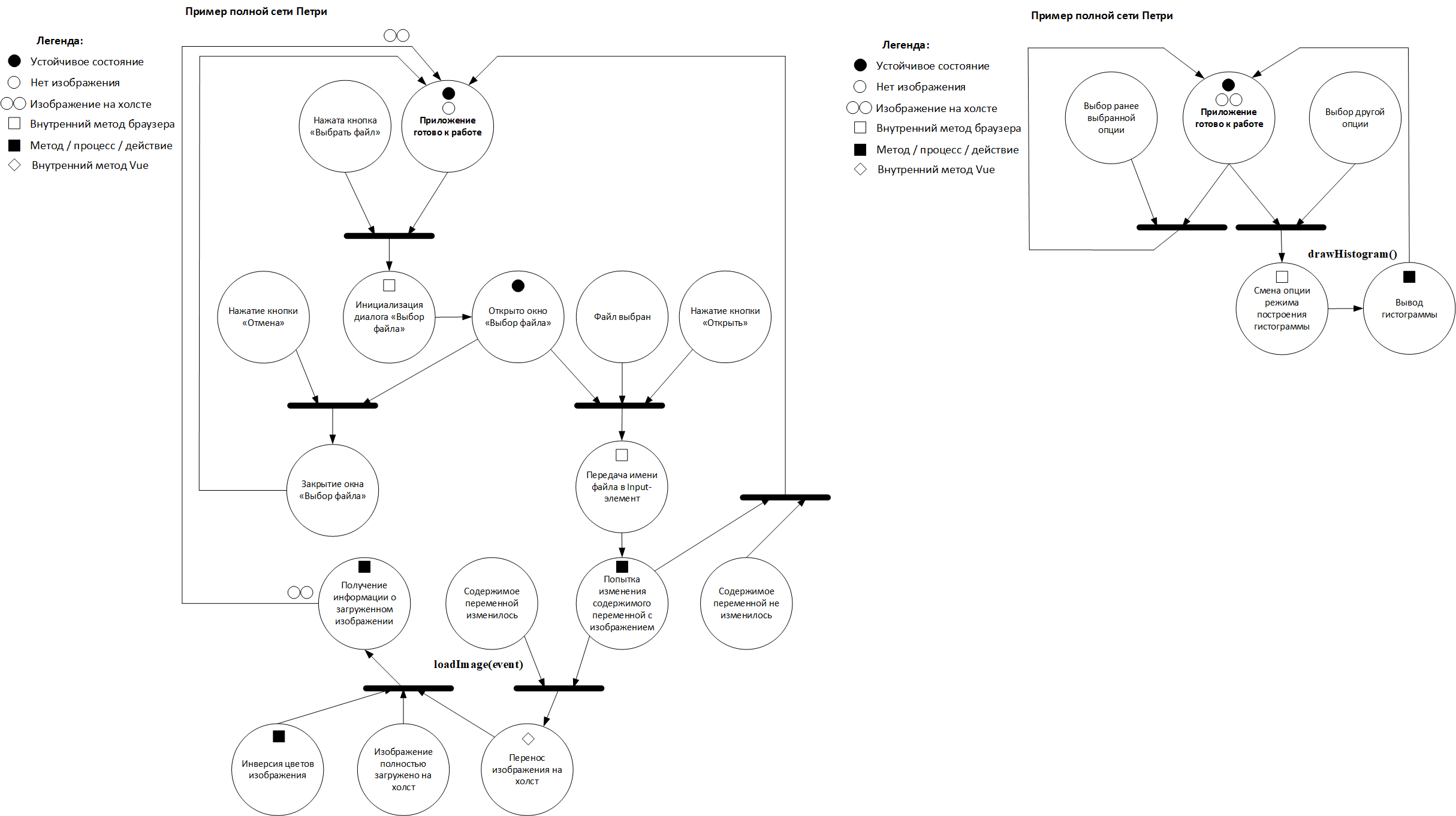
На рисунке 2-8 представлено отображение в браузере нашего web-приложения.

|  |
| --- |
| Рисунок 2 – Отображение в браузере |

|  |
| --- |
| Рисунок 3 – Загрузили изображение в формате .jpg |
| Рисунок 4 – Гистограмма |
| Рисунок 5 – Гистограмма |

## Сеть Петри

Ниже представлена сеть Петри процесса с момента запуска файла страницы до окончания процесса рассмотрения гистограмм.



# 5. Вывод

В данной работе были приобретены навыки по созданию программного, браузерного web-обеспечение под управлением фреймворка Vue.js, изучены основные понятия, принципы и методы работы с гистограммами, цветовой моделью CMYK и средством Canvas. Поставленные задачи выполнены в полном объёме.