

# Оглавление

<b>Введение</b>	<b>5</b>
<b>1 Аналитическая часть</b>	<b>7</b>
1.1 Формализация задачи . . . . .	7
1.2 Цветовая модель . . . . .	8
1.2.1 Цветовая модель RGB . . . . .	8
1.2.2 Цветовая модель HSL . . . . .	8
1.2.3 Цветовая модель HSLuv . . . . .	9
1.2.4 Цветовая модель CIELAB . . . . .	9
1.2.5 Цветовая модель LCH . . . . .	10
1.3 Выбор цветовой модели для решения задачи . . . . .	10
1.4 Конвертация цветов между цветовыми моделями . . . . .	11
1.5 Модели контраста . . . . .	13
1.5.1 Модель Майкельсона . . . . .	13
1.5.2 Модель Вебера . . . . .	14
1.5.3 Модифицированная модель Вебера . . . . .	14
1.6 Выбор модели контраста для решения задачи . . . . .	14
1.7 Анализ возможных решений . . . . .	15
1.7.1 Наивная инверсия . . . . .	15
1.7.2 Изменение цвета компонентов на основе анализа цветовой карты . . . . .	17
1.8 Анализ существующих решений . . . . .	20
<b>2 Конструкторская часть</b>	<b>21</b>
2.1 Требования к программному обеспечению . . . . .	21
2.2 Разработка алгоритмов . . . . .	21
<b>3 Технологическая часть</b>	<b>25</b>
3.1 Средства реализации . . . . .	25
3.2 Детали реализации . . . . .	26
<b>4 Исследовательская часть</b>	<b>30</b>
4.1 Пример работы программного обеспечения . . . . .	30

4.2 Постановка эксперимента . . . . .	35
4.2.1 Цель эксперимента . . . . .	35
4.2.2 Описание эксперимента . . . . .	35
4.2.3 Результат эксперимента . . . . .	35
<b>Заключение</b>	<b>37</b>
<b>Литература</b>	<b>38</b>

# Введение

В мире половина людей владеет смартфонами [1] и персональными компьютерами [2]. Третью часть дня [3] люди пользуются цифровыми устройствами – смартфонами и персональными компьютерами. В этом нет ничего странного, ведь жизнь постепенно переходит в цифровое пространство: уже никого не удивить покупками [4], просмотрами фильмов и сериалов, работой через Интернет.

В таких условиях человеческое здоровье подвергается нагрузке, в том числе при этом страдают глаза [5]. В течение работы за экраном компьютера или телефона, человеческий глаз испытывает перенапряжение. Для решения этой проблемы производители программного обеспечения и разработчики веб-страниц стали добавлять в продукты функцию переключения интерфейса в темный (ночной) режим [6]. Синий свет, излучаемый экранами цифровых устройств, приводит не только к перенапряжению глаз, но и к нарушению секреции мелатонина в организме человека [7], что негативно влияет на сон. Использование темного режима при работе с дисплеями снимает напряжение с глаз и не нарушает секрецию мелатонина.

Помимо пользы для здоровья, темный режим положительно влияет [8] на время работы от батареи устройств, оснащенных OLED [9] дисплеями. Например, при яркости экрана в 50%, мобильное приложение YouTube [10] с включенным темным режимом потребляет на 15% меньше заряда аккумулятора, чем с включенным дневным режимом [8].

Несмотря на плюсы темного режима, остаются веб-страницы, которые не поддерживают такую опцию.

Цель работы – реализовать программное обеспечение для изменения цветовой палитры веб-страниц со светлой на темную.

Чтобы достигнуть поставленной цели, требуется решить следующие задачи:

- проанализировать цветовую карту веб-страницы, чтобы оценить какую часть цветов стоит изменить;
- изменить цветовую палитру веб-страницы в соответствии с полученной цветовой карты;

- обработать изображения, расположенные на веб-странице;
- обработать динамический контент (видео, анимации), расположенные на веб-странице.

Решение поставленной проблемы не очевидно, потому что помимо факторов, влияющих на здоровье пользователя, стоит учесть удобство использования преобразованными веб-страницами. Человеческий глаз лучше воспринимает черный текст на белом фоне, нежели белый текст на черном фоне, как предполагает ночной режим. Так происходит потому, что белый цвет отражает цвет каждой волны в цветовом спектре [11].

Исходя из приведенных выше фактов, стоит задача разработать такую программу, которая будет изменять цветовую палитру веб-страницы в соответствии и с требованиями к снижению нагрузки на глаза, и с требованиями относительно читаемости текста [12].

# 1 Аналитическая часть

В данном разделе описаны цветовые модели, пригодные для решения задачи, выбор цветовой модели, модели контраста, выбор модели контраста, а также анализ возможных и существующих решений.

## 1.1 Формализация задачи

Ресурсы в сети Интернет представлены в виде веб-сайтов [13], веб-сайты в свою очередь состоят из веб-страниц. Веб-страница – это документ, который может быть отображён веб-браузерами, такими как: Firefox, Google Chrome, Microsoft Internet Explorer / Edge или Safari от Apple [13]. Веб-страница состоит из множества компонентов: текст, изображения, фон и т.п. Каждый компонент имеет свои цветовые параметры, количество которых зависит от используемой цветовой модели.

Для преобразования цветовой схемы веб-страниц используется таблица стилей CSS [14]. С помощью CSS можно настраивать параметры отображения компонентов веб-страницы (цвет, яркость и т.п.). Разработчики веб-страниц вручную добавляют расширения для таблицы стилей для поддержки ночного режима. Цель работы – автоматизировать добавление ночного режима для веб-страниц, основываясь на анализе цветовой палитры [15].

В таблице стилей CSS при работе с цветами используется три цветовых модели: RGB, RGBA и HSL, поэтому общее решение задачи можно представить в виде формулы 1.1:

$$CM_{converted} = F(CM_{initial}), \quad (1.1)$$

где  $F$  – функция преобразования цветовой модели,  $CM_{initial}$  и  $CM_{converted}$  – начальная и преобразованная цветовые модели соответственно.

## 1.2 Цветовая модель

Цветовая модель – термин, обозначающий абстрактную модель описания представления цветов в виде кортежей чисел, называемых цветовыми компонентами или цветовыми координатами. Вместе с методом интерпретации этих данных (например, определение условий воспроизведения или просмотра – то есть задание способа реализации), цвета цветовой модели определяет цветовое пространство.

### 1.2.1 Цветовая модель RGB

Цветовая модель RGB описывает излучаемые цвета. Она основана на трёх базовых цветах: красный (Red), зелёный (Green) и синий (Blue). Остальные цвета получаются сочетанием базовых. Цвета такого типа называются аддитивными. В компьютерах для представления каждой из координат традиционно используется один октет, значения которого обозначаются для удобства целыми числами от 0 до 255 включительно. Она применяется в приборах, излучающих свет, таких, например, как мониторы, прожекторы, фильтры.

Инвертировать цвет, заданный в цветовой модели RGB, можно при помощи формулы 1.2 [16]:

$$\begin{aligned} R_{inverted} &= 255 - R_{initial} \\ G_{inverted} &= 255 - G_{initial} \\ B_{inverted} &= 255 - B_{initial} \end{aligned} \tag{1.2}$$

Цветовая модель RGB в рамках задачи позволяет использовать метод наивной инверсии, который описан в 1.7.1.

### 1.2.2 Цветовая модель HSL

Цветовая модель HSL, HLS или HSI (от англ. Hue, Saturation, Lightness (Intensity)) – цветовая модель, в которой цветовыми координатами являются

ются тон, насыщенность и светлота.

- Hue – цветовой тон (например, красный, зелёный или сине-голубой). Варьируется в пределах  $0\text{--}360^\circ$ , однако иногда приводится к диапазону  $0\text{--}100$  или  $0\text{--}1$ .
- Saturation – насыщенность. Варьируется в пределах  $0\text{--}100$  или  $0\text{--}1$ . Чем больше этот параметр, тем «чище» цвет, поэтому этот параметр иногда называют чистотой цвета. А чем ближе этот параметр к нулю, тем ближе цвет к нейтральному серому.
- Lightness (Intensity) – светлота (яркость). Постоянный оттенок ( $d, h$ ) приводит к вертикальному поперечному сечению. Также задаётся в пределах  $0\text{--}100$  и  $0\text{--}1$ .

Цветовая модель HSL в рамках задачи позволяет использовать метод изменения цвета компонентов на основе анализа цветовой карты [17], который описан в 1.7.2, что дает более удобочитаемое [18] представление.

### 1.2.3 Цветовая модель HSLuv

Цветовая модель HSLuv является альтернативной версией модели HSL, которая позволяет с большей точностью (градацией) задать контраст цвета.

Данная цветовая модель непригодна в рамках задачи, так как более плавная градация цвета ведет к получению сильно близких друг к другу цветов, что ухудшает читаемость.

### 1.2.4 Цветовая модель CIELAB

Цветовая модель CIELAB – цветовая модель, в которой цветовыми координатами являются светлота от черного до белого, от зеленого до красного и от синего до желтого цветов.

- L – светлота от черного (0) до белого (100).

- A – светлота от зеленого (-) до красного (+).
- B – светлота от синего (-) до желтого (+).

Данная цветовая модель спроектирована так, что любое изменение числовых параметров компонент соответствующие влияет на визуальное восприятие.

Цветовая модель CIELAB непригодна в рамках задачи, так как не позволяет расчитывать близость цветов в цветовой палитре, как это позволяют сделать модели HSL и HSLuv.

### 1.2.5 Цветовая модель LCH

Цветовая модель LCH – "цилиндрическая" версия цветовой модели CIELAB, что означает, что вместо 6 цветовых параметров, доступных в CIELAB, используется только 4.

- L – светлота от черного (0) до белого (100).
- C – расстояние от серого (0 – 100).
- H – направление цвета (0 – 360) (0 – красный, 90 – желтый, 180 – зеленый, 270 – синий).

Цветовая модель LCH пригодна в рамках задачи, так как позволяет, как и модель HSL, рассчитывать близость цветов в цветовой палитре.

## 1.3 Выбор цветовой модели для решения задачи

Для решения задачи подходит 3 цветовых модели: RGB, HSL и LCH. В качестве основной модели выбрана модель HSL, так как она:

- позволяет оценить близость цветов в цветовой палитре, что позволит при инверсии цветов подбирать наиболее близкие цвета, отталкиваясь от базового, для сохранения визуальной целостности веб-страницы;

- позволяет провести прямую конвертацию из RGB, в отличие от LCH. RGB – самая распространенная цветовая модель, используемая в таблице стилей CSS [19], поэтому данную конвертацию придется производить с большой частотой;
- изначально поддерживается таблицей стилей CSS, что позволит не производить обратную конвертацию в RGB.

## 1.4 Конвертация цветов между цветовыми моделями

В формулах 1.3 – 1.6 представлены шаги для получения компонент цвета цветовой модели HSL из RGB [20].

$$H = \begin{cases} \text{undefined} & \text{if } MAX = MIN \\ 60^\circ \times \frac{G-B}{MAX-MIN} + 0^\circ, & \text{if } MAX = R \\ & \text{and } G \geq B \\ 60^\circ \times \frac{G-B}{MAX-MIN} + 360^\circ, & \text{if } MAX = R \\ & \text{and } G < B \\ 60^\circ \times \frac{B-R}{MAX-MIN} + 120^\circ, & \text{if } MAX = G \\ 60^\circ \times \frac{R-G}{MAX-MIN} + 240^\circ, & \text{if } MAX = B \end{cases}, \quad (1.3)$$

$$S = \begin{cases} 0 & \text{if } L = 0 \text{ or } MAX = MIN \\ \frac{MAX-MIN}{MAX+MIN} = \frac{MAX-MIN}{2L}, & \text{if } 0 < L \leq \frac{1}{2} \\ \frac{MAX-MIN}{2-(MAX+MIN)} = \frac{MAX-MIN}{2-2L}, & \text{if } \frac{1}{2} < L < 1 \end{cases}, \quad (1.4)$$

или, в общем случае

$$S = \frac{MAX - MIN}{1 - |1 - (MAX + MIN)|}, \quad (1.5)$$

$$L = \frac{1}{2}(MAX + MIN), \quad (1.6)$$

где:

- R, G, B – значения цвета в цветовой модели RGB, значения в диапазоне  $[0; 1]$  (R - красный, G - зелёный, B - синий);
- MAX – максимум из трёх значений (R, G, B);
- MIN – минимум из трёх значений (R, G, B);
- H – тон  $[0; 360]$ ;
- S – насыщенность  $[0; 1]$ ;
- L – светлота  $[0; 1]$ .

В формулах 1.7 – 1.15 представлены шаги для получения компонент цвета цветовой модели RGB из HSL [20].

$$Q = \begin{cases} L \times (1.0 + S), & \text{if } L < 0.5 \\ L + S - (L \times S), & \text{if } L \geq 0.5 \end{cases} \quad (1.7)$$

$$P = 2.0 \times L - Q \quad (1.8)$$

$$H_k = \frac{H}{360} \quad (1.9)$$

$$T_R = H_k + \frac{1}{3} \quad (1.10)$$

$$T_G = H_k \quad (1.11)$$

$$T_B = H_k - \frac{1}{3} \quad (1.12)$$

$$\text{if } T_c < 0 \rightarrow T_c = T_c + 1.0 \quad \text{for each } c = R, G, B \quad (1.13)$$

$$\text{if } T_c > 1 \rightarrow T_c = T_c - 1.0 \quad \text{for each } c = R, G, B \quad (1.14)$$

Для каждого цвета  $c = R, G, B$ :

$$\text{color}_c = \begin{cases} P + ((Q - P) \times 6.0 \times T_c), & \text{if } T_c < \frac{1}{6} \\ Q, & \text{if } \frac{1}{6} \leq T_c < \frac{1}{2} \\ P + ((Q - P) \times (\frac{2}{3} - T_c) \times 6.0), & \text{if } \frac{1}{2} \leq T_c < \frac{2}{3} \\ P, & \text{otherwise} \end{cases} \quad (1.15)$$

## 1.5 Модели контраста

Контраст (фр. *contraste*) – в оптике (сенситометрии и фотометрии) разница в характеристиках различных участков изображения, способность фотографического материала или оптической системы воспроизводить эту разницу, а также характеристика чувствительности глаза (зрительной системы) относительно яркости и цвета [21].

Контрастность (также в различных контекстах употребляется и само слово контраст и коэффициент контраста) – степень контраста, чаще всего выражается безразмерной величиной, отношением или логарифмом отношений.

Вопрос выбора модели контраста важен, потому что контраст является определяющим свойством при определении удобочитаемости текста [22].

В общем случае контрастность можно вычислить по формуле 1.16 [23]:

$$CR = \frac{L_H}{L_L}, \quad 1 \leq CR \leq \infty, \quad (1.16)$$

где

- $L_H$  – относительная яркость [24] самого яркого цвета, представленного на изображении;
- $L_L$  – относительная яркость самого темного цвета, представленного на изображении;

Относительная яркость – величина в диапазоне от 0 до 1, где в случае 0 – черный, 1 – белый.

При  $CR = 1$  считается, что изображение не имеет контраста (полностью однотонное).

### 1.5.1 Модель Майкельсона

Модель контраста Майкельсона определяется по формуле 1.17 [23]:

$$C_M = \frac{L_H - L_L}{L_H + L_L}, \quad 0 \leq C_M \leq 1, \quad (1.17)$$

При  $C_M = 0$  считается, что изображение не имеет контраста (полностью однотонное).

Данная модель не учитывает влияние окружающего освещения, которое может искажить контраст изображения на дисплее.

### 1.5.2 Модель Вебера

Модель контраста Вебера определяется по формуле 1.17 [23]:

$$C_W = \frac{L_H - L_L}{L_H}, \quad 0 \leq C_W \leq 1, \quad (1.18)$$

При  $C_W = 0$  считается, что изображение не имеет контраста (полностью однотонное).

Данная модель также не учитывает влияние окружающего освещения, которое может искажить контраст изображения на дисплее.

### 1.5.3 Модифицированная модель Вебера

Модифицированная модель контраста Вебера определяется по формуле 1.19 [23]:

$$C_m W = \frac{L_H - L_L}{L_H + 0.05}, \quad (1.19)$$

В данном случае в знаменателе добавлен коэффициент, позволяющий более точно смоделировать контраст изображения с учетом окружающего освещения.

## 1.6 Выбор модели контраста для решения задачи

Для решения задачи выбрана модифицированная модель контраста Вебера, так как она:

- учитывает влияние окружающего освещения при подсчете контрастности;
- используется в качестве основной при подсчете контрастности по стандарту WCAG [12].

## 1.7 Анализ возможных решений

В данном подразделе представлен сравнительный анализ возможных решений поставленной задачи.

### 1.7.1 Наивная инверсия

Метод наивной инверсии предполагает собой получение инвертированного значения компонента цвета путем вычитания из максимально возможного значения параметра компонента текущего значения. В формуле 1.2 показан пример инвертирования цвета для цветовой модели RGB.

Наивная инверсия цвета даст некорректный в контексте работы результата: при инверсии цветов некоторых элементов (например, фон и текст), их конечное отображение может оказаться нечитаемым, потому что не будет удовлетворено соотношение 7:1 или 4.5:1 [22] для самого светлого и самого темного цветов страницы, а также соотношение 3:1 [25] для . На рисунках 1.1 и 1.2 показаны исходное состояние веб-страницы и состояние, на котором инвертированы цвета, соответственно.

Данный результат обусловлен тем, что при инвертировании темного цвета в результате получается светлый цвет. Такое изменение корректно в случае, если меняется цвет текста, но некорректно, если меняется цвет фона. Помимо этого, читаемость текста на измененных веб-страницах может не всегда быть приемлемой. Полученные цвета текста и фона могут не давать соотношение контрастности 4.5:1 или, в лучшем случае, 7:1. Также цвета смежных (имеющих общую границу) элементов могут не иметь соотношение контрастности 3:1. Данные соотношения являются рекомендованными согласно стандарту WCAG 2.1 [12], описывающему требования

к отображению элементов веб-страницы. Несоответствие результатов стандарту противоречит поставленной цели.

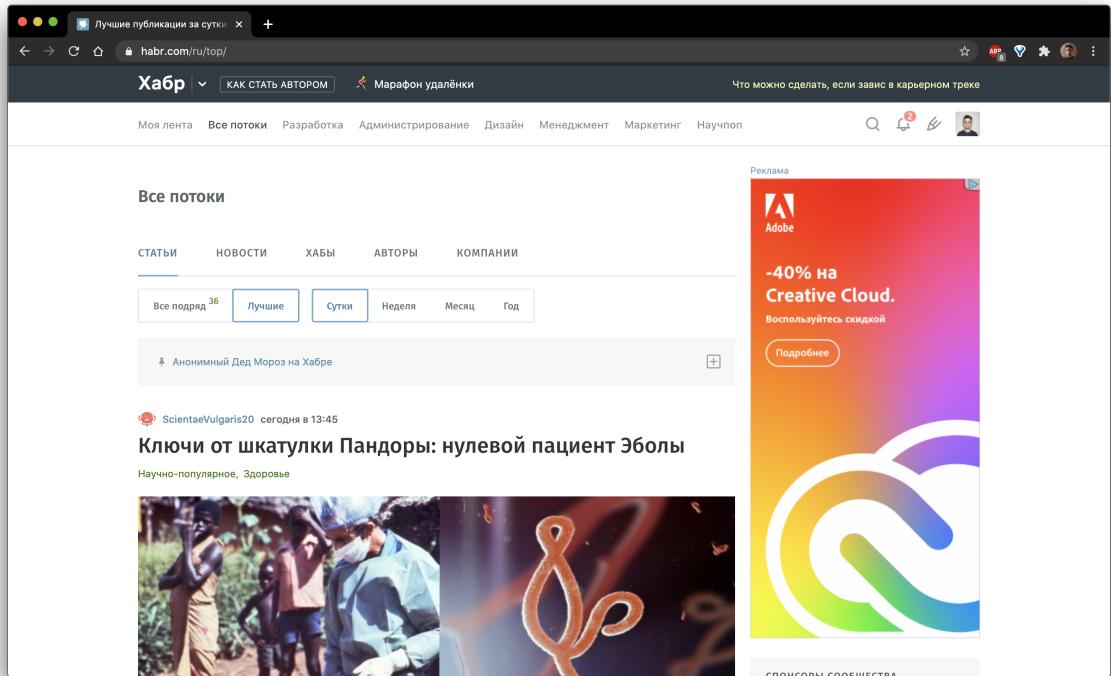


Рисунок 1.1 – Исходное состояние веб-страницы

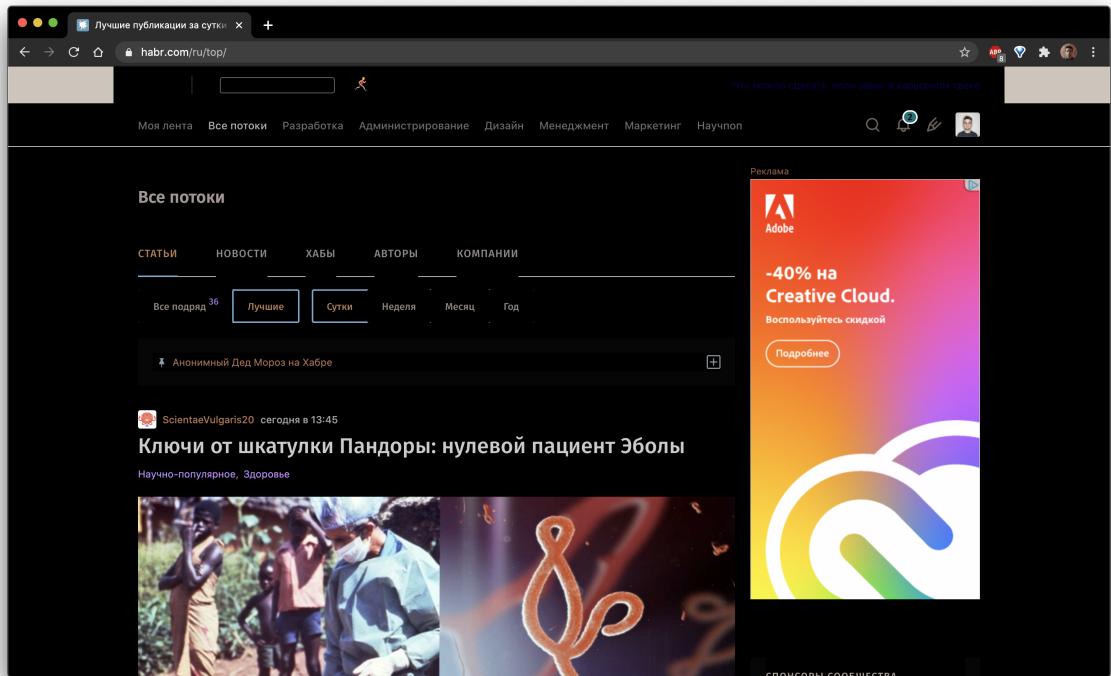


Рисунок 1.2 – Состояние веб-страницы с инвертированными цветами

### 1.7.2 Изменение цвета компонентов на основе анализа цветовой карты

Метод изменения цвета компонентов на основе анализа цветовой карты предполагает собой изменение только базовых цветов страницы (наиболее часто встречающихся на странице). Для определения доминантного цвета можно воспользоваться методом  $k$ -средних [26], который позволяет произвести кластеризацию участков изображения. Причем под изменением предполагается не инверсия, а выбор близкого по модели к инвертированному цвета с учетом условий сохранения читаемости [12] (соотношения контрастности самого светлого и самого темного цветов страницы). После преобразования базовых цветов остальные цвета на странице подбираются в соответствии с преобразованными цветами в том же спектре, что и базовые, но с другими параметрами модели (например, насыщенность в модели HSL). Такое преобразование позволяет сохранить изначальную визуальную структуру страницы.

Выбор цветов стоит производить также опираясь на характеристику контраста [21], чтобы обеспечить приемлемый уровень читаемости текста.

Исходя из понятия контраста и рекомендаций WCAG, для подбора цветов элементов страницы можно воспользоваться формулой 1.20 [?]:

$$\frac{L_1 + 0.05}{L_2 + 0.05}, \quad (1.20)$$

где

- $L_1$  – относительная яркость самого яркого из цветов, представленных на странице.
- $L_2$  – относительная яркость самого темного из цветов, представленных на странице.

Под самым ярким и самым темным цветами подразумеваются цвета, имеющие наибольший и наименьший параметр L (Lightness) в цветовой модели HSL, соответственно.

Данная формула является модифицированной формулой Вебера [23] нахождения контраста. Модификация более точно моделирует потерю кон-

трастности, возникающую при более низкой яркости дисплея из-за условий окружающего освещения. Увеличенную точность обеспечивают дополнительные коэффициенты в числителе и знаменателе.

На рисунках 1.3 и 1.4 представлены исходное состояние веб-страницы и её цветовая карта соответственно.

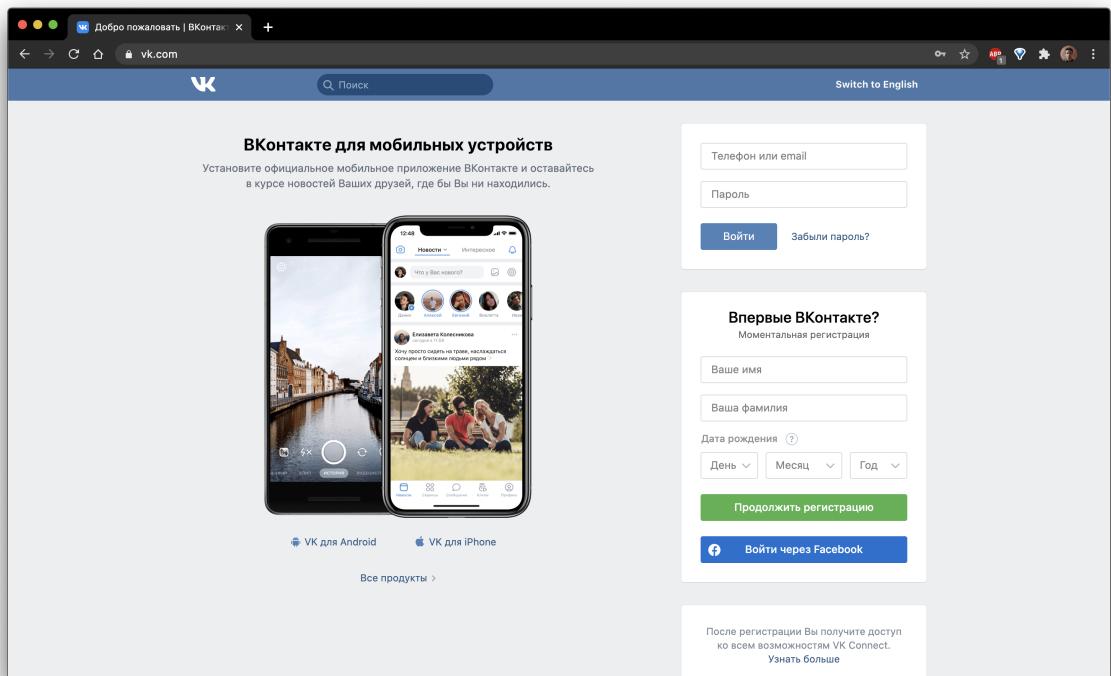


Рисунок 1.3 – Исходное состояние веб-страницы

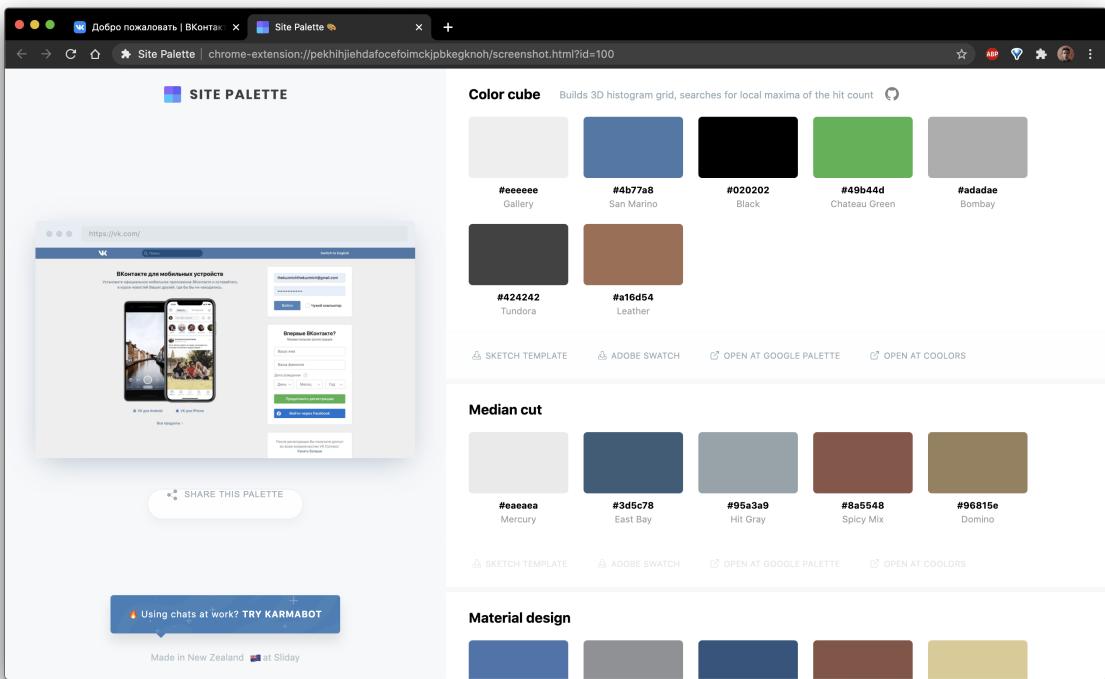


Рисунок 1.4 – Цветовая карта веб-страницы

На основе анализа цветовой карты можно произвести преобразование цветов. Полученный результат показан на рисунке 1.5.

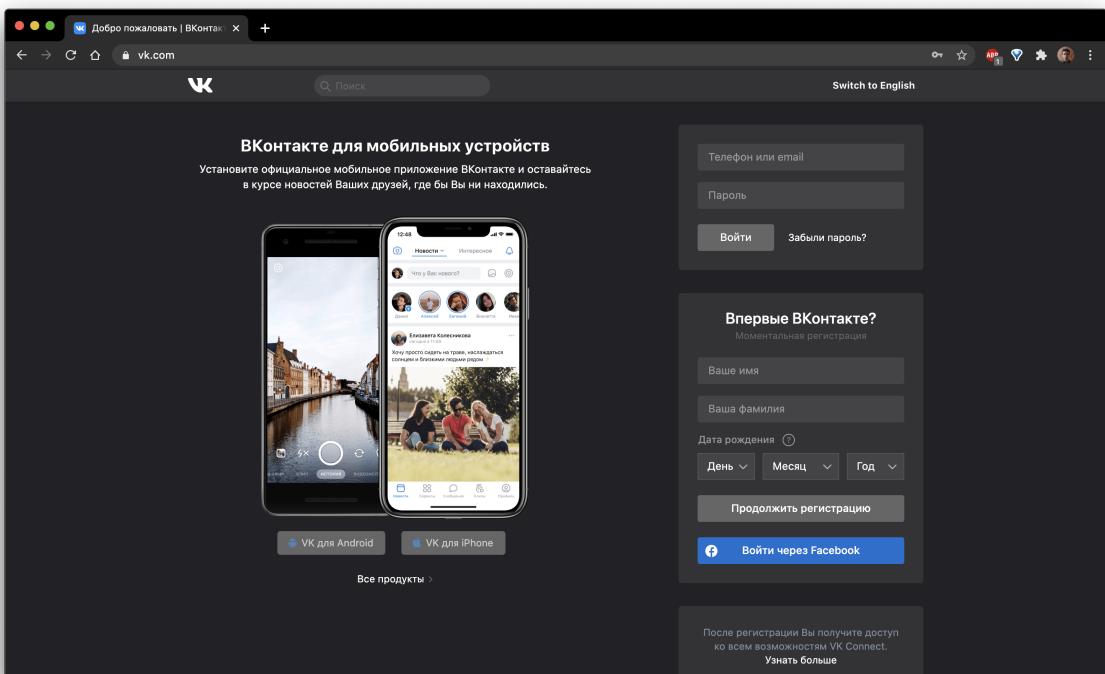


Рисунок 1.5 – Состояние веб-страницы с преобразованными цветами

Полученного результата трудно достичь, используя цветовую модель

RGB, потому что она не позволяет определить близость расположения цветов в цветовой палитре, что необходимо для обеспечения удобочитаемого распределения цветов при преобразовании цветовой схемы. Чтобы достичь такого результата, можно использовать цветовую модель HSL, которая позволяет определять близость цветов.

Данное решение вписывается в цель работы и будет реализовано в дальнейшем.

## 1.8 Анализ существующих решений

В качестве существующих решений для анализа выбраны расширение для браузеров Night Eye [27] и встроенная в браузер Google Chrome [28] опция Chrome Dark Mode [29].

В таблице 1.1 представлен сравнительный анализ существующих решений.

Таблица 1.1 – Анализ существующих решений

Название решения	Используемая цветовая модель	Используемая модель контраста	Метод изменения цветов
Night Eye	RGB	Не указано	Наивная инверсия
Chrome Dark Mode	HSL	Микельсон	Нейронная сеть анализа

## Вывод

В данном разделе был проведен анализ цветовых моделей, возможных и существующих решений поставленной задачи. В качестве основной цветовой модели, при помощи которой будет решаться задача, была выбрана модель HSL, а метода изменения цветовой палитры – метод изменения цвета компонентов на основе анализа цветовой карты.

## **2 Конструкторская часть**

В данном разделе представлены требования к программному обеспечению, схема выбранного для решения поставленной задачи алгоритма.

### **2.1 Требования к программному обеспечению**

Программа должна предоставлять доступ к возможностям:

- 1) преобразования цветовой палитры веб-страницы из светлой в темную;
- 2) сохранения визуальной структуры веб-страницы

К программе предъявляются требования:

- размер программы не должен превышать 700 килобайт в исходном виде и 170 килобайт в сжатом [30];
- программа не должна задействовать такое количество ресурсов устройства, при котором загрузка веб-страницы занимает больше 2 секунд [31].

### **2.2 Разработка алгоритмов**

На рисунках 2.1 – 2.4 представлена схема алгоритма, реализующего преобразование цветовой палитры веб-страницы.



Рисунок 2.1 – Схема общей постановки задачи

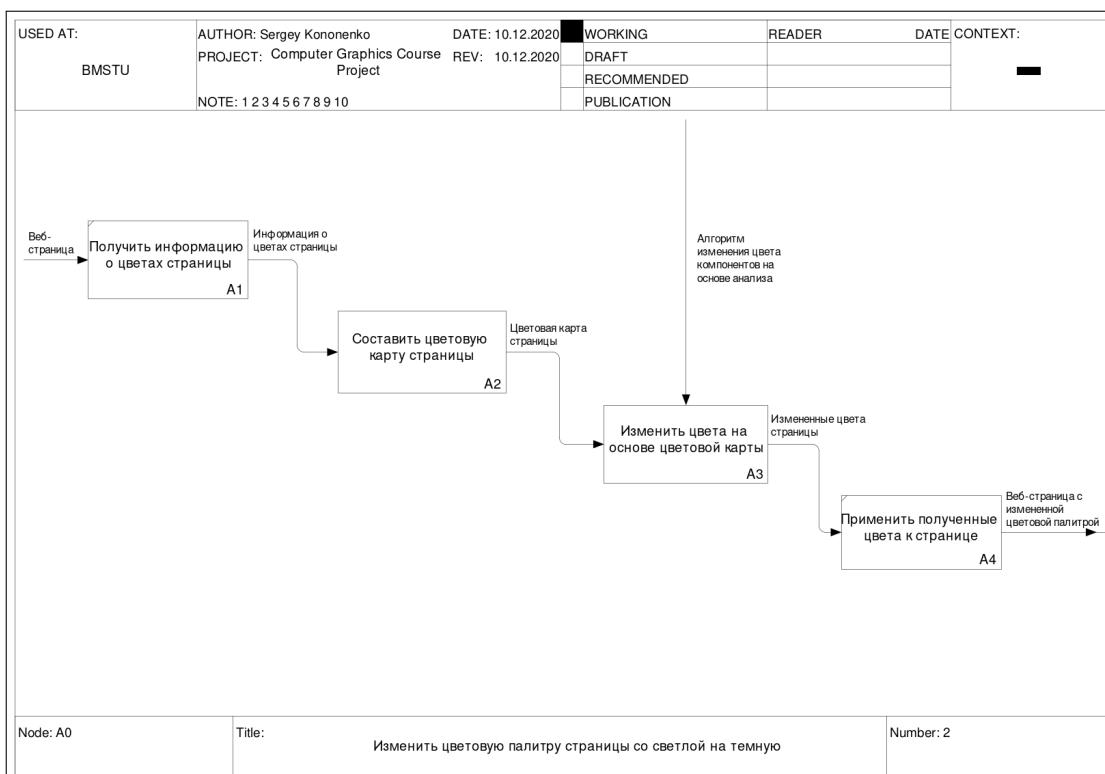


Рисунок 2.2 – Схема декомпозиции общей постановки задачи

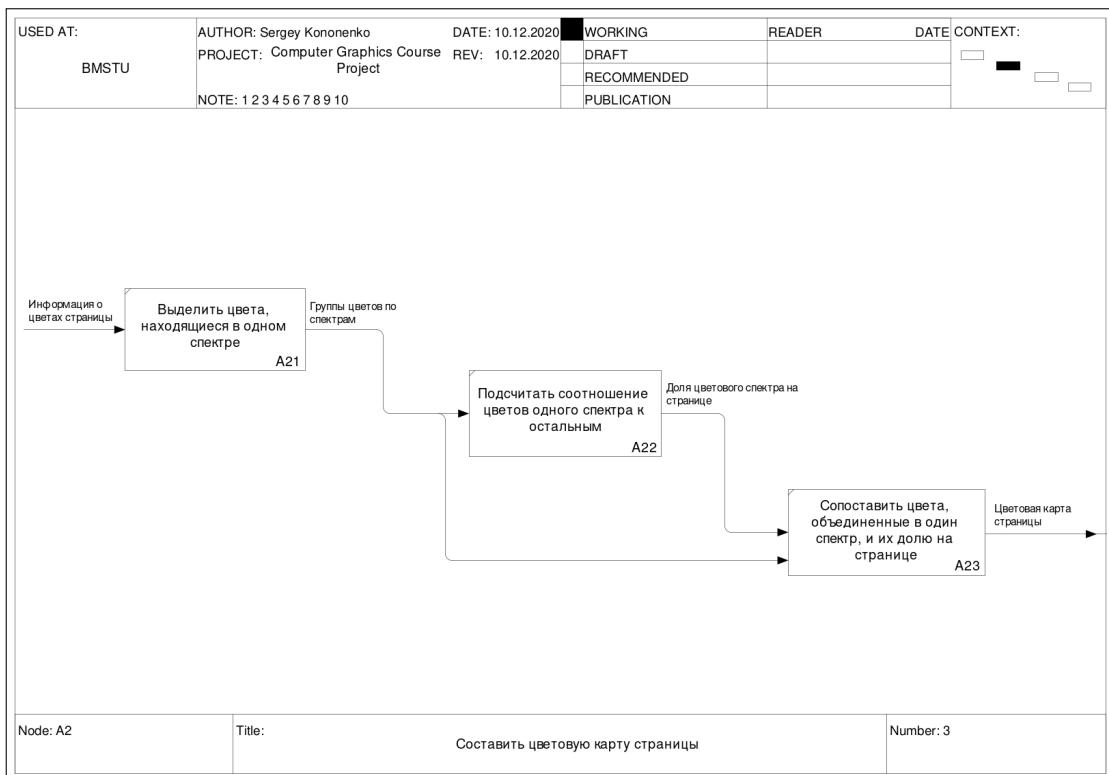


Рисунок 2.3 – Схема декомпозиции задачи составления цветовой карты

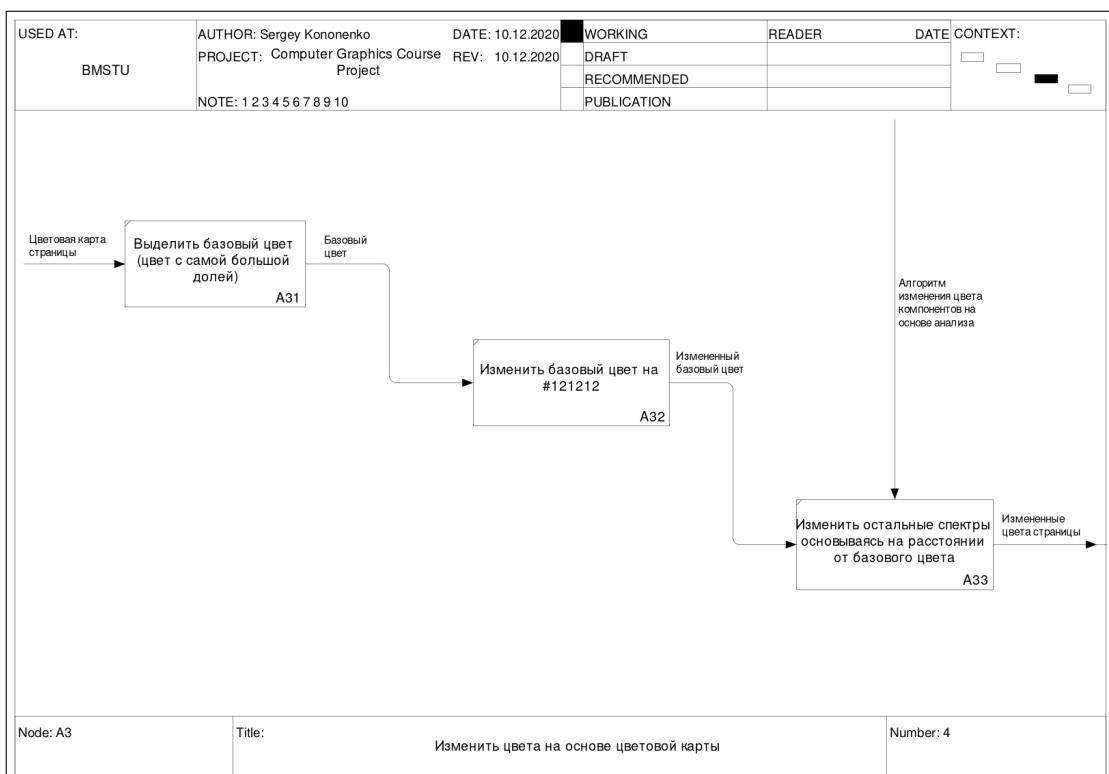


Рисунок 2.4 – Схема декомпозиции задачи изменения цветов на основе цветовой карты

## **Вывод**

В данном разделе были представлены требования к программному обеспечению и разработаны схемы реализуемых алгоритмов.

### 3 Технологическая часть

В данном разделе представлены средства разработки программного обеспечения, детали реализации и тестирование функций.

#### 3.1 Средства реализации

В качестве языка программирования, на котором будет реализовано программное обеспечение, выбран язык TypeScript 4 [32]. Выбор языка обусловлен тем, что динамический контент на веб-страницах разрабатывается и управляется на языке JavaScript [33]. TypeScript является типизированной версией JavaScript и позволяет избежать ошибок, связанных с неправильным использованием типов при разработке. На выходе TypeScript транспилируется в JavaScript.

Разработанный скрипт будет подключаться к уже существующей веб-странице. Поэтому стоит задача минимизации размера скрипта для уменьшения времени загрузки страницы.

Для минимизации размера выходного файла будет использоваться упаковщик webpack [34]. Вместе с ts-loader [35] он позволяет автоматически транспилировать файлы TypeScript в JavaScript и минимизировать их размер путем сокращения названий переменных, функций, удаления пробелов в исходном коде.

Для тестирования программного обеспечения будет использоваться фреймворк [36] Jest [37]. Данный инструмент предоставляет широкие возможности для тестирования приложений, такие как разработка заглушек, имитация запросов и т.п.

Для обеспечения качества кода был использован инструмент ESLint [38], позволяющий во время написания кода контролировать наличие синтаксических и логических ошибок.

В качестве среды разработки выбран текстовый редактор Visual Studio Code [39]. Данный выбор обусловлен большим количеством плагинов под разные инструменты. Плагины позволяют облечить и ускорить разработку программного обеспечения.

Для определения доминантного цвета на странице были использованы

в связке библиотеки html2canvas [40] и Color Thief [?]. html2canvas позволяет пробразовать страницу к изображению, а Color Thief выделяет карту цветов и доминантный цвет.

## 3.2 Детали реализации

В листингах 3.1 – 3.5 приведены реализации алгоритмов конвертации цветов из RGB в HSL для последующего подбора цвета, нахождения относительной яркости, нахождение доминантного цвета, преобразования представлении цветов для записи в таблицу стилей CSS и изменения цветов на основе доминантного соответственно.

Листинг 3.1 – Реализация алгоритма конвертации цвета из RGB в HSL

```
1 const convertRGBToHSL = (baseColor: RGB): HSL => {
2   const red = baseColor.red / 255;
3   const green = baseColor.green / 255;
4   const blue = baseColor.blue / 255;
5   const min = Math.min(red, green, blue);
6   const max = Math.max(red, green, blue);
7   const delta = max - min;
8   let hue, saturation;
9
10  if (max === min) {
11    hue = 0;
12  } else if (red === max) {
13    hue = (green - blue) / delta;
14  } else if (green === max) {
15    hue = 2 + (blue - red) / delta;
16  } else {
17    hue = 4 + (red - green) / delta;
18  }
19  hue = Math.min(hue * 60, 360);
20  if (hue < 0) {
21    hue += 360;
22  }
23
24  const lightness = (min + max) / 2;
25
26  if (max === min) {
27    saturation = 0;
28  } else if (lightness <= 0.5) {
29    saturation = delta / (max + min);
30  } else {
```

```

31     saturation = delta / (2 - max - min);
32 }
33
34 const convColor: HSL = {
35   hue: Math.round(hue),
36   saturation: Math.round(saturation * 100),
37   lightness: Math.round(lightness * 100),
38 };
39
40 return convColor;
41};

```

Листинг 3.2 – Реализация алгоритма нахождения относительной яркости

```

1 const getRelativeLuminance = (color: RGB): number => {
2   const rgb = [color.red / 255, color.green / 255, color.blue / 255];
3
4   for (let i = 0; i < 3; ++i) {
5     rgb[i] = rgb[i] <= 0.03928 ? rgb[i] / 12.92 : Math.pow((rgb[i] + 0.055) / 1.055,
6       2.4);
7   }
8
9   const relLum = 0.2126 * rgb[0] + 0.7152 * rgb[1] + 0.0722 * rgb[2];
10
11 return relLum;
12};

```

Листинг 3.3 – Реализация алгоритма нахождения доминантного цвета

```

1 const getDominantColor = async (): Promise<RGB> => {
2   const canvas: HTMLCanvasElement = await html2canvas(document.body);
3   const dataUrl = canvas.toDataURL();
4
5   const pageImg = document.createElement("img");
6   pageImg.src = dataUrl;
7
8   const colorThief = new ColorThief();
9   let rgb;
10  if (pageImg.complete) {
11    rgb = colorThief.getColor(pageImg);
12  } else {
13    pageImg.addEventListener("load", () => {
14      rgb = colorThief.getColor(pageImg);
15    });
16  }
17
18  await new Promise((r) => setTimeout(r, 100));
19
20  const domColor: RGB = {

```

```

21     red: rgb[0],
22     green: rgb[1],
23     blue: rgb[2],
24   };
25
26   return domColor;
27 }

```

Листинг 3.4 – Реализация алгоритмов нахождения преобразования представлений цветов

```

1 const printRGB = (color: RGB) => {
2   return `rgb(${color.red}, ${color.green}, ${color.blue})`;
3 };
4
5 const printHSL = (color: HSL) => {
6   return `hsl(${color.hue}, ${color.saturation}%, ${color.lightness}%)`;
7 };
8
9 const stringToRgb = (color: string) => {
10   const colors = color
11     .slice(4, color.length - 1)
12     .split(",")
13     .map((c) => Number(c.trim()));
14   const convColor: RGB = {
15     red: colors[0],
16     green: colors[1],
17     blue: colors[2],
18   };
19
20   return convColor;
21 }

```

Листинг 3.5 – Реализация алгоритма подбора цветов на основе базового

```

1 const main = async () => {
2   const BASE_BACKGROUND_RGB: RGB = {
3     red: 18,
4     green: 18,
5     blue: 18,
6   };
7   const BASE_BACKGROUND_HSL = convertRGBToHSL(BASE_BACKGROUND_RGB);
8   const BASE_FOREGROUND_COLOR = "rgba(255, 255, 255, 0.87)";
9   const BASE_COLOR = await getDominantColor();
10
11   document.body.style.backgroundColor = printRGB(BASE_BACKGROUND_RGB);
12   for (const styleSheet of document.styleSheets) {
13     for (const rule of styleSheet.cssRules) {
14       if (rule.type == rule.STYLE_RULE) {

```

```

15     const styleRule = (rule as CSSStyleRule).style;
16     for (const key in styleRule) {
17         if (key === "color") {
18             if (styleRule[key]) {
19                 styleRule[key] = BASE_FOREGROUND_COLOR;
20             }
21         } else if (key.toLowerCase().includes("color")) {
22             if (styleRule[key]) {
23                 const initColor = stringToRgb(styleRule[key]);
24
25                 let initLum = getRelativeLuminance(initColor);
26                 let backLum = getRelativeLuminance(BASE_COLOR);
27                 if (initLum > backLum) {
28                     [backLum, initLum] = [initLum, backLum];
29                 }
30
31                 const multiplier = (backLum + 0.05) / (initLum + 0.05) + 1;
32                 const colorHsl: HSL = {
33                     hue: BASE_BACKGROUND_HSL.hue,
34                     saturation: BASE_BACKGROUND_HSL.saturation,
35                     lightness: multiplier
36                     ? BASE_BACKGROUND_HSL.lightness * multiplier
37                     : BASE_BACKGROUND_HSL.lightness,
38                 };
39                 styleRule[key] = printHSL(colorHsl);
40             }
41         }
42     }
43 }
44 }
45 }
46 };

```

## Вывод

В данном разделе были представлены средства реализации программного обеспечения и листинги программного обеспечения, разработанного на основе схем конструкторского раздела.

# **4 Исследовательская часть**

В данном разделе представлены примеры работы программного обеспечения и постановка эксперимента по сравнению результата изменения цветовой палитры изображения методом наивной инверсии и методом изменения компонентов на основе анализа цветовой карты.

## **4.1 Пример работы программного обеспечения**

На рисунках 4.1 – 4.6 приведены исходные состояния веб-страниц и состояния веб-страниц с измененной цветовой палитрой соответственно.



# WineChecker



Developed by [hackfeed](#)

## Input wine parameters

Fixed acidity, g/dm<sup>3</sup> (from 4.9 to 15.6)

Volatile acidity, g/dm<sup>3</sup> (from 0.12 to 1.58)

Citric acid, g/dm<sup>3</sup> (from 0 to 1)

Residual sugar, g/dm<sup>3</sup> (from 0.9 to 15.5)

Chlorides, g/dm<sup>3</sup> (from 0.01 to 0.61)

Рисунок 4.1 – Исходное состояние веб-страницы 1

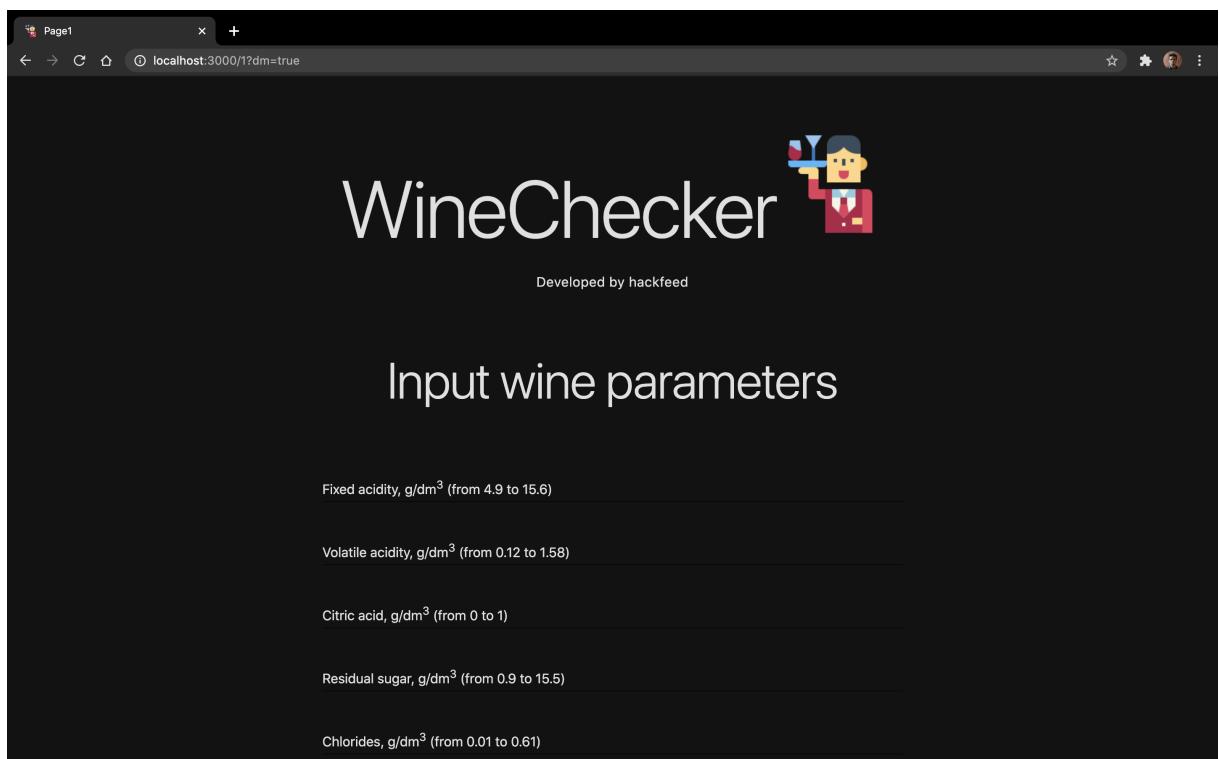


Рисунок 4.2 – Состояние веб-страницы 1 с измененной цветовой палитрой

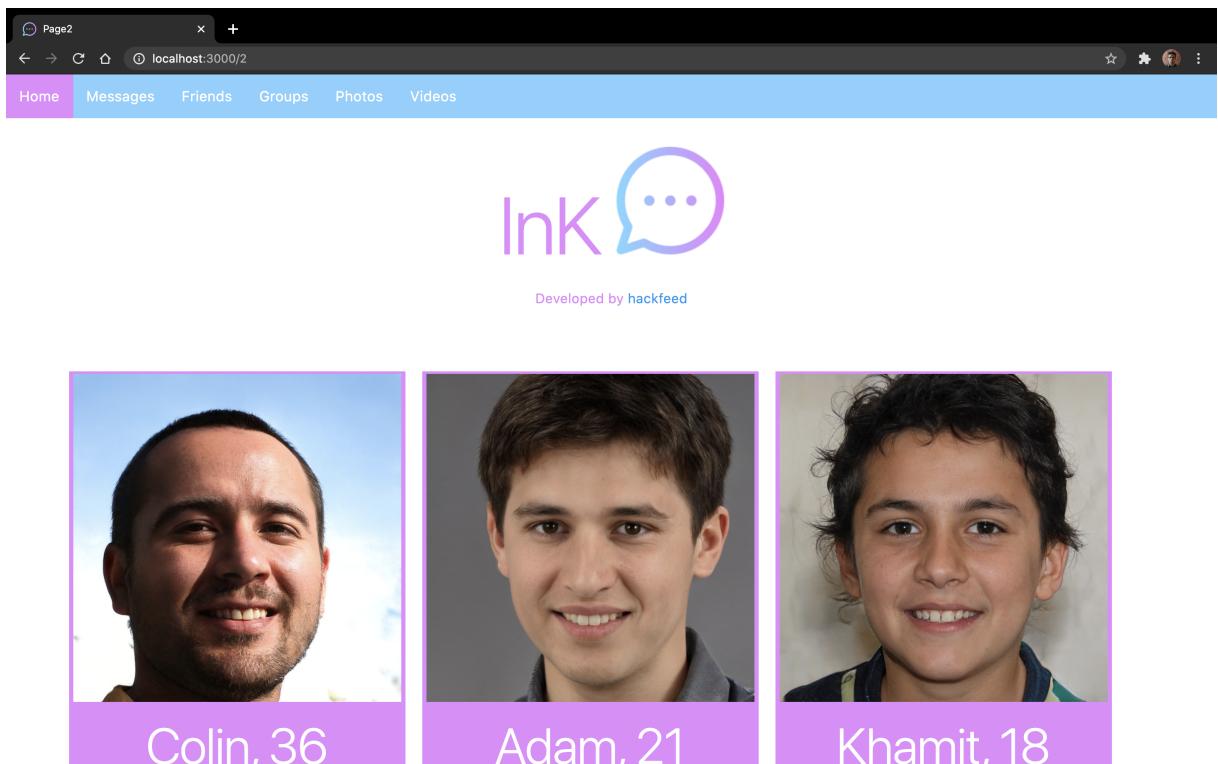


Рисунок 4.3 – Исходное состояние веб-страницы 2

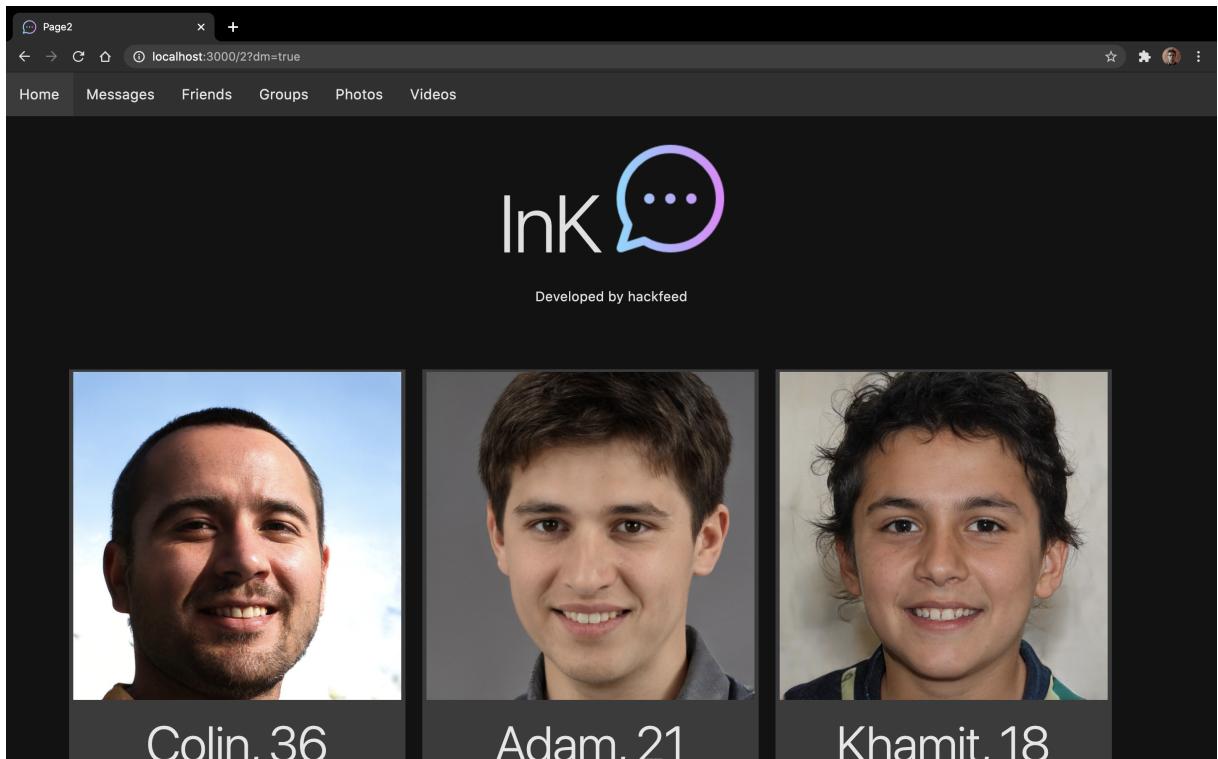


Рисунок 4.4 – Состояние веб-страницы 2 с измененной цветовой палитрой



Рисунок 4.5 – Исходное состояние веб-страницы 3

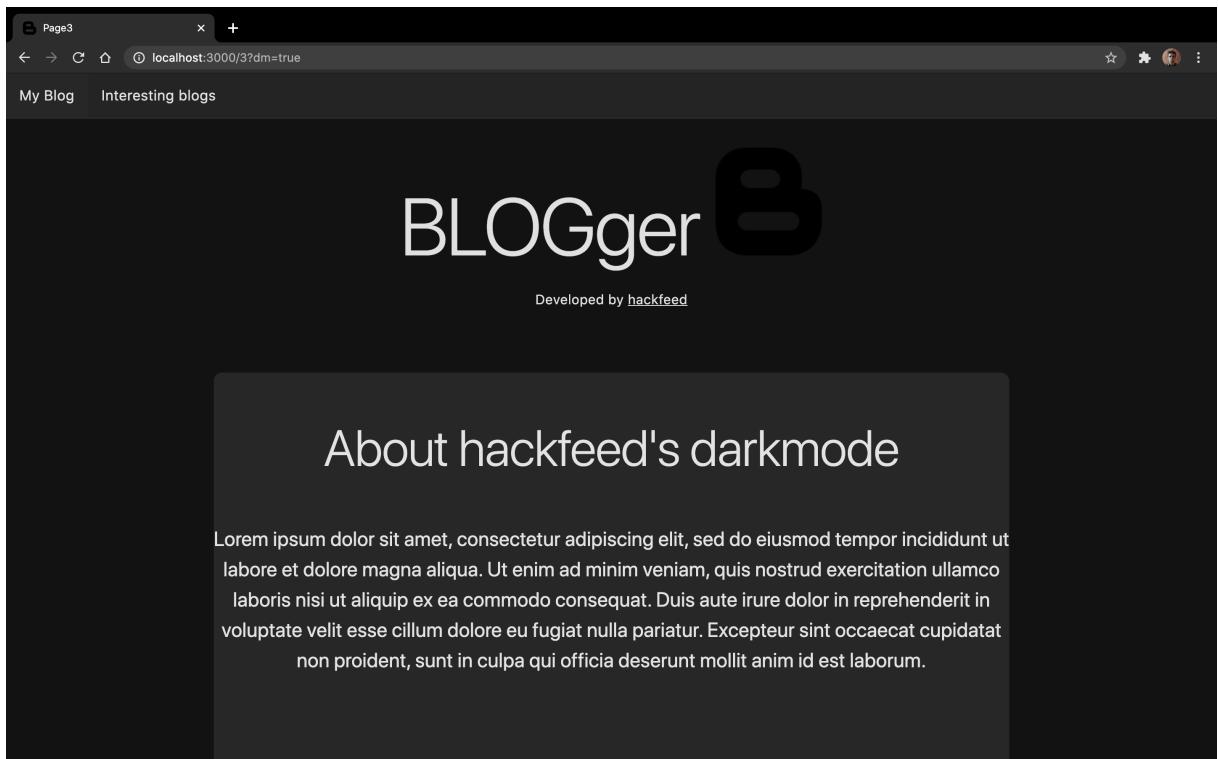


Рисунок 4.6 – Состояние веб-страницы 3 с измененной цветовой палитрой

На рисунках 4.7 – 4.9 приведены данные загрузки веб-страниц. Разработанное программное обеспечение, как и планировалось, производит преобразование в течение времени до двух секунд, что удовлетворяет постав-

ленному условию.

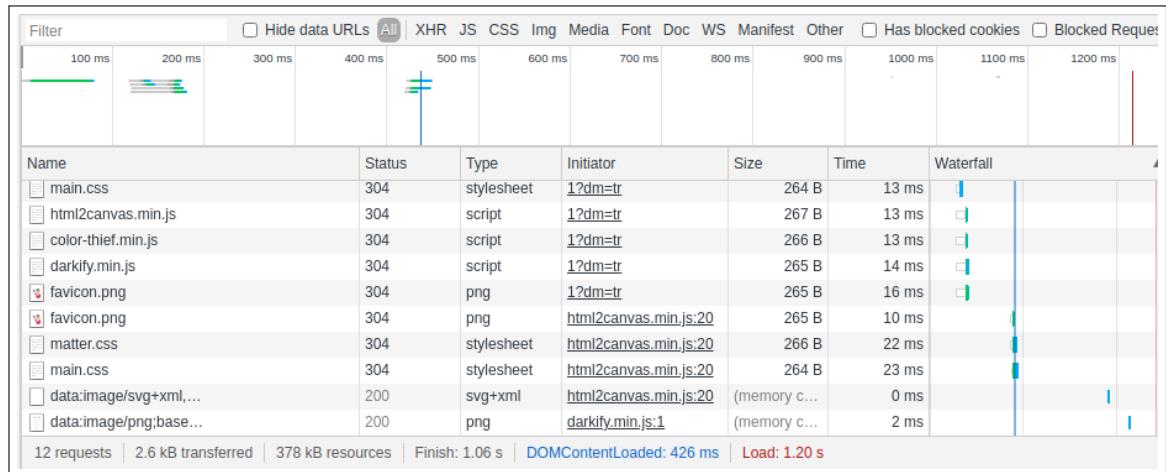


Рисунок 4.7 – Затраченные ресурсы при загрузке страницы 1

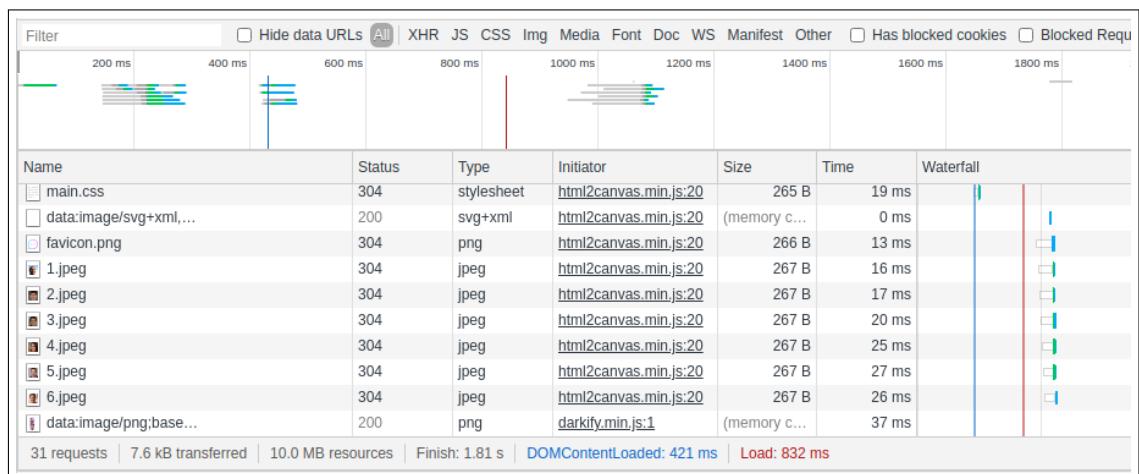


Рисунок 4.8 – Затраченные ресурсы при загрузке страницы 2

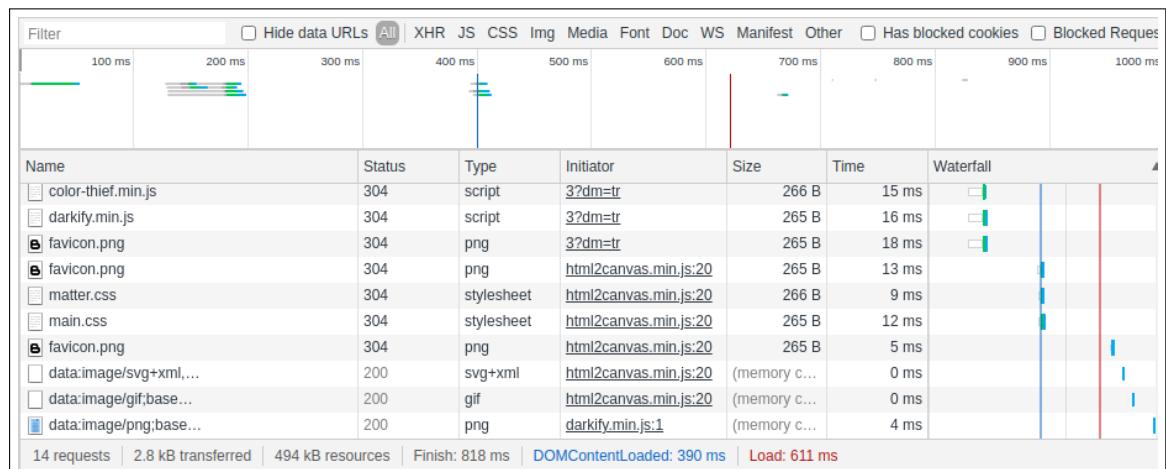


Рисунок 4.9 – Затраченные ресурсы при загрузке страницы 3

## 4.2 Постановка эксперимента

В данном подразделе представлены цель, описание и результаты эксперимента.

### 4.2.1 Цель эксперимента

Целью эксперимента является сравнение методов изменения цветовой палитры изображения. Критерием сравнения будет являться контрастность полученного изображения [22].

### 4.2.2 Описание эксперимента

Сравнить результат преобразования цветовой палитры изображения можно при помощи сравнения полученной контрастности [?].

Хорошим считается отношение 4.5:1 и больше, отличным – 7:1 и больше.

### 4.2.3 Результат эксперимента

В таблице 4.1 представлены результаты поставленного эксперимента.

Таблица 4.1 – Результаты сравнения методов изменения цветовой палитры изображения

Веб-страница	Наивная инверсия	Анализ цветовой карты
WineChecker	21:1	13.92:1
InK	21:1	13.92:1
BLOGger	1.61:1	7:1

## Вывод

В результате сравнения методов изменения цветовой палитры изображения было выявлено, что метод наивной инверсии может давать требу-

мую контрастность, но только в некоторых случаях (например, белый фон страницы и черный текст). Метод изменения цветовой палитры на основе анализа цветовой карты позволяет добиться нужной контрастности в любом случае, потому что метод предполагает собой сравнение и анализ полученных цветов изображения и выбор наиболее подходящих цветов для изменения.

# Заключение

Во время выполнения курсового проекта было реализовано программное обеспечение, в котором были реализованы алгоритмы изменения цветовой палитры изображения со светлой на темную.

В ходе выполнения поставленной задачи были получены знания в области компьютерной графики. Были изучены принципы работы цветовых моделей, моделей контраста. Поиск подходящего решения для поставленной задачи позволил повысить навыки поиска и анализа информации.

В результате проведенной работы было разработано программное обеспечение, доказывающее применимость квантовых алгоритмов в области компьютерной графики. Разработанный программный продукт реализует алгоритм изменения цветовой палитры изображения на основе анализа цветовой карты, который в дальнейшем возможно применить для изменения цветовой палитры веб-страниц и изображений.

В ходе выполнения экспериментально-исследовательской части было установлено, что алгоритм изменения цветовой палитры изображения на основе анализа цветовой карты выигрывает у алгоритма наивной инверсии, так как благодаря предварительному анализу дает требуемый результат во всех случаях.

# Литература

- [1] Smartphone users | Statista [Электронный ресурс]. Режим доступа: <https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/> (дата обращения: 8.11.2020).
- [2] How many people have access to a computer | Statista [Электронный ресурс]. Режим доступа: <https://www.statista.com/statistics/748551/worldwide-households-with-computer/> (дата обращения: 8.11.2020).
- [3] Global Online Content Consumption Doubled In 2020 – Forbes [Электронный ресурс]. Режим доступа: <https://www.forbes.com/sites/johnkoetsier/2020/09/26/global-online-content-consumption-doubled-in-2020/?sh=248eeecd2fde> (дата обращения: 8.11.2020).
- [4] Find out How Many People Shop Online in 2020 | Oberlo [Электронный ресурс]. Режим доступа: <https://www.oberlo.com/statistics/how-many-people-shop-online> (дата обращения: 8.11.2020).
- [5] Digital Eye Strain | The Vision Council [Электронный ресурс]. Режим доступа: <http://web-old.archive.org/web/20200612073503/https://www.thevisioncouncil.org/content/digital-eye-strain> (дата обращения: 8.11.2020).
- [6] Dark Mode - What Is It, and Why Do We Need It? | techahead [Электронный ресурс]. Режим доступа: <https://www.techaheadcorp.com/blog/dark-mode/> (дата обращения: 8.11.2020).
- [7] Blue light has a dark side – Harvard Health [Электронный ресурс]. Режим доступа: <https://www.health.harvard.edu/staying-healthy/blue-light-has-a-dark-side> (дата обращения: 8.11.2020).
- [8] Cost of a Pixel Color (Android Dev Summit '18) – YouTube [Электронный ресурс]. Режим доступа: [https://www.youtube.com/watch?v=N\\_6sPd0Jd3g](https://www.youtube.com/watch?v=N_6sPd0Jd3g) (дата обращения: 8.11.2020).

- [9] OLED – Wikipedia [Электронный ресурс]. Режим доступа: <https://en.wikipedia.org/wiki/OLED> (дата обращения: 8.11.2020).
- [10] YouTube – Wikipedia [Электронный ресурс]. Режим доступа: <https://en.wikipedia.org/wiki/YouTube> (дата обращения: 8.11.2020).
- [11] Is Dark Mode Better for Your Eyes? – Rx Optical [Электронный ресурс]. Режим доступа: <https://rxoptical.com/eye-health/is-dark-mode-better-for-your-eyes/> (дата обращения: 16.11.2020).
- [12] Web Content Accessibility Guidelines (WCAG) 2.1 [Электронный ресурс]. Режим доступа: <https://www.w3.org/TR/WCAG21/> (дата обращения: 06.12.2020).
- [13] Веб-страницы, веб-сайты, веб серверы и поисковики – Изучение веб-разработки | MDN [Электронный ресурс]. Режим доступа: [https://developer.mozilla.org/ru/docs/Learn/Pages\\_sites\\_servers\\_and\\_search\\_engines](https://developer.mozilla.org/ru/docs/Learn/Pages_sites_servers_and_search_engines) (дата обращения: 21.11.2020).
- [14] CSS – Википедия [Электронный ресурс]. Режим доступа: <https://ru.wikipedia.org/wiki/CSS> (дата обращения: 21.11.2020).
- [15] Цветовая палитра – Википедия [Электронный ресурс]. Режим доступа: [https://ru.wikipedia.org/wiki/%D0%A6%D0%B2%D0%B5%D1%82%D0%BE%D0%B2%D0%B0%D1%8F\\_%D0%BF%D0%B0%D0%BB%D0%B8%D1%82%D1%80%D0%B0](https://ru.wikipedia.org/wiki/%D0%A6%D0%B2%D0%B5%D1%82%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BB%D0%B8%D1%82%D1%80%D0%B0) (дата обращения: 27.11.2020).
- [16] color – How do you find an inverse colour? – Graphic Design Stack Exchange [Электронный ресурс]. Режим доступа: <https://graphicdesign.stackexchange.com/questions/95084/how-do-you-find-an-inverse-colour> (дата обращения: 13.12.2020).
- [17] Palette (computing) – Wikipedia [Электронный ресурс]. Режим доступа: [https://en.wikipedia.org/wiki/Palette\\_\(computing\)](https://en.wikipedia.org/wiki/Palette_(computing)) (дата обращения: 01.12.2020).
- [18] Критерии 100% читаемости сайтов / Хабр [Электронный ресурс]. Режим доступа: <https://habr.com/ru/post/48862/> (дата обращения: 01.12.2020).

- [19] Defining Colors in CSS [Электронный ресурс]. Режим доступа: <http://web.simmons.edu/~grovesd/comm244/notes/week3/css-colors#:~:text=Hexadecimal%20Color%20Values,way%20to%20represent%20RGB%20values>. (дата обращения: 01.12.2020).
- [20] HSL – Википедия [Электронный ресурс]. Режим доступа: <https://ru.wikipedia.org/wiki/HSL> (дата обращения: 13.12.2020).
- [21] Contrast (vision) – Wikipedia [Электронный ресурс]. Режим доступа: [https://en.wikipedia.org/wiki/Contrast\\_\(vision\)](https://en.wikipedia.org/wiki/Contrast_(vision)) (дата обращения: 09.12.2020).
- [22] Understanding Success Criterion 1.4.6: Contrast (Enhanced) [Электронный ресурс]. Режим доступа: <https://www.w3.org/WAI/WCAG21/Understanding/contrast-enhanced.html> (дата обращения: 08.12.2020).
- [23] Display contrast – Wikipedia [Электронный ресурс]. Режим доступа: [https://en.wikipedia.org/wiki/Display\\_contrast#Luminance\\_contrast](https://en.wikipedia.org/wiki/Display_contrast#Luminance_contrast) (дата обращения: 09.12.2020).
- [24] Web Content Accessibility Guidelines (WCAG) 2.1 [Электронный ресурс]. Режим доступа: <https://www.w3.org/TR/WCAG21/#dfn-contrast-ratio> (дата обращения: 08.12.2020).
- [25] Understanding Success Criterion 1.4.11: Non-text Contrast [Электронный ресурс]. Режим доступа: <https://www.w3.org/WAI/WCAG21/Understanding/non-text-contrast.html> (дата обращения: 18.12.2020).
- [26] Piyush. S. Atram Prof. Pramila M. Chawan. Finding Dominant Color in the Artistic Painting using Data Mining Technique: A Survey. International Research Journal of Engineering and Technology (IRJET), 2019. с. 236.
- [27] Night Eye – Dark mode extension [Электронный ресурс]. Режим доступа: <https://nighteye.app/> (дата обращения: 22.11.2020).

- [28] Веб-браузер Google Chrome [Электронный ресурс]. Режим доступа: [https://www.google.com/intl/ru\\_ru/chrome/](https://www.google.com/intl/ru_ru/chrome/) (дата обращения: 22.11.2020).
- [29] Browse in Dark mode or Dark theme – Android – Google Chrome Help Center [Электронный ресурс]. Режим доступа: <https://support.google.com/chrome/answer/9275525?co=GENIE.Platform%3DAndroid&hl=en> (дата обращения: 22.11.2020).
- [30] The cost of JavaScript in 2019 – V8 [Электронный ресурс]. Режим доступа: <https://v8.dev/blog/cost-of-javascript-2019> (дата обращения: 10.12.2020).
- [31] How Fast Should A Website Load And How To Speed It Up [Электронный ресурс]. Режим доступа: <https://www.hobo-web.co.uk/your-website-design-should-load-in-4-seconds/> (дата обращения: 14.12.2020).
- [32] TypeScript: Typed JavaScript at Any Scale. [Электронный ресурс]. Режим доступа: <https://www.typescriptlang.org/> (дата обращения: 10.12.2020).
- [33] JavaScript | MDN [Электронный ресурс]. Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/JavaScript> (дата обращения: 10.12.2020).
- [34] webpack [Электронный ресурс]. Режим доступа: <https://webpack.js.org/> (дата обращения: 10.12.2020).
- [35] TypeStrong/ts-loader: TypeScript loader for webpack [Электронный ресурс]. Режим доступа: <https://github.com>TypeStrong/ts-loader> (дата обращения: 10.12.2020).
- [36] Фреймворк – Википедия [Электронный ресурс]. Режим доступа: <https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA> (дата обращения: 10.12.2020).
- [37] Jest – Delightful JavaScript Testing [Электронный ресурс]. Режим доступа: <https://jestjs.io/> (дата обращения: 10.12.2020).

- [38] ESLint – Pluggable JavaScript linter [Электронный ресурс]. Режим доступа: <https://eslint.org/> (дата обращения: 10.12.2020).
- [39] Visual Studio Code – Code Editing. Redefined [Электронный ресурс]. Режим доступа: <https://code.visualstudio.com/> (дата обращения: 10.12.2020).
- [40] niklasvh/html2canvas: Screenshots with Javascript [Электронный ресурс]. Режим доступа: <https://github.com/niklasvh/html2canvas> (дата обращения: 18.12.2020).