МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____ 2021 г.

Создание редактора элементов страниц для веб-разработки с доступом к исходному коду элементов

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.п.н., доцент каф. ЭВМ
_____Ю.Г. Плаксина
«___»_____2021 г.

Автор работы,
студент группы КЭ-405
_____А.С. Герасимов
«___»_____2021 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___»_____2021 г.

Челябинск-2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____2021 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-405
Герасимову Александру Сергеевичу
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы: «**Создание редактора элементов страниц для веб-разработки с доступом к исходному коду элементов» утверждена приказом по университету от 26 апреля 2021 года №714-13/12 (приложение №73)

2. **Срок сдачи студентом законченной работы:** 1 июня 2021 г.

3. **Исходные данные к работе:**

– Документация фреймворка Vue (https://v3.vuejs.org/guide/introduction.html);
– Хавербеке, М. Выразительный JavaScript. Современное веб-программирование: учебник / М. Хавербеке. – Санкт-Петербург: Изд-во Питер, 2019. – 480 с.;

– Фрэйн, Б. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств: учебник / Б. Фрэйн. – Санкт-Петербург: Изд-во Питер, 2017. – 272 с.

4. **Перечень подлежащих разработке вопросов:**

– рассмотрение существующих программных решений для осуществления генерации веб-компонентов;
– анализ актуальности разработки;
– разработка собственного программного решения для создания генератора веб-компонентов;
– оценка работоспособности разработанного программного решения в различных средах и устройствах.

5. **Дата выдачи задания:** 1 декабря 2020 г.

Руководитель работы _____/*Ю.Г. Плаксина*/

Студент _____/*А.С. Герасимов*/

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	01.03.2021	
Разработка модели, проектирование	01.04.2021	
Реализация системы	01.05.2021	
Тестирование, отладка, эксперименты	15.05.2021	
Компоновка текста работы и сдача на нормоконтроль	24.05.2021	
Подготовка презентации и доклада	30.05.2021	

Руководитель работы _____/*Ю.Г. Плаксина*/

Студент _____/*А.С. Герасимов*/

Аннотация

А.С. Герасимов. Создание редактора элементов страниц для веб-разработки с доступом к исходному коду элементов – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2021, 55 с., 31 ил., библиогр. список – 14 наим.

В рамках выпускной квалификационной работы был проведён анализ существующих аналогов, а также схожих по функционалу систем. В результате проведённого анализа были выявлены недостатки и достоинства этих систем. Рассмотрены основные технологии, применяющиеся в веб-разработке, и были выбраны наиболее подходящие для данного проекта.

После анализа были произведены проектирование, разработка и тестирование веб-сайта для создания веб-компонентов.

Результатом работы является веб-сайт, работающий на разных версиях веб-браузеров, с разными разрешениями экрана мониторов, реализующий заявленный функционал.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Разработка веб-сайтов развивается с каждым днём, в мире веб-разработки появляются новые задачи и вызовы, для их решения появляются новые программы и технологии, которые позволяют ускорить процесс разработки, а также строить более сложные системы.

В современном мире веб-разработки сложилось большое количество различных направлений деятельности. Разработка современного сложного веб-сайта состоит из следующих этапов: создание дизайна, верстка, frontend-разработка, backend-разработка, создание и администрирование баз данных, тестирование. В зависимости от специфики функционала веб-сайта к данным этапам могут добавляться и другие.

Конечно, это идеально, когда каждым этапом веб-разработки занимается отдельный специалист или команда специалистов, но реальность такова, что из-за нехватки людей, приходится работать с разными задачами. Довольно частым явлением бывает ситуация, когда frontend-разработчик занимается в том числе и вёрсткой.

Актуальность разработки редактора веб-компонентов заключается в стремлении создать удобный инструмент, позволяющий легко создавать компоненты веб-страниц, что позволит frontend-разработчикам уделять меньше времени вёрстке и больше времени программированию.

Также редактор веб-компонентов позволит наглядно демонстрировать изменения в свойствах веб-компонентов, что будет полезно для начинающих верстальщиков.

Целью представленной выпускной квалификационной работы является разработка веб-сайта, обеспечивающего возможность создания и редактирования веб-компонентов для дальнейшего их использования в собственных проектах.

Для достижения поставленной цели, необходимо решить следующие задачи:

1. Найти существующие на данный момент аналоги и схожие по функционалу системы.
2. Провести анализ с целью выявления преимуществ и недостатков этих систем, сделать выводы.
3. Провести анализ существующих технологий веб-разработки и выбрать те, которые будут использоваться для выполнения выпускной квалификационной работы. Обосновать их выбор.
4. Составить техническое задание и эскизный проект системы.
5. Разработать архитектуру проекта.
6. Выполнить программную реализацию проекта.
7. Определить методы тестирования и провести тестирование разработанного приложения на корректность выполнения задач, удобство интерфейса.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. ОБЗОР АНАЛОГОВ

Сейчас существует множество веб-сайтов для создания и редактирования веб-компонентов, однако каждый из них имеет какие-либо недостатки, например, неудобный интерфейс или ограниченный функционал и т.п. Разберём более подробно различные примеры, выявим их достоинства, которые можно будет реализовать в проекте и недостатки, которые нужно будет не допустить в разрабатываемом веб-сайте.

1. Веб-сайт - https://www.cssfilters.co/ [1] позволяет загружать и редактировать фотографии с помощью различных фильтров, например, фильтров контрастности, яркости, насыщенности и т.п. Пример интерфейса веб-сайта представлен на рисунке 1, на рисунке 2 представлен интерфейс веб-сайта с измененным изображением.

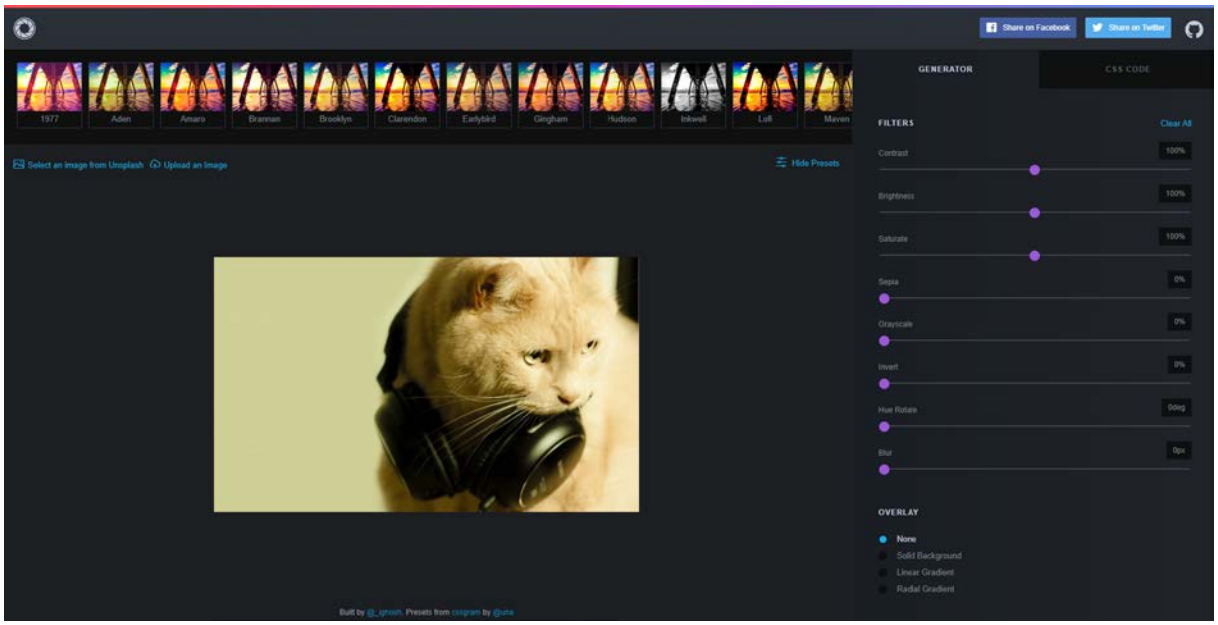


Рисунок 1.1 – Интерфейс веб-сайта https://www.cssfilters.co/

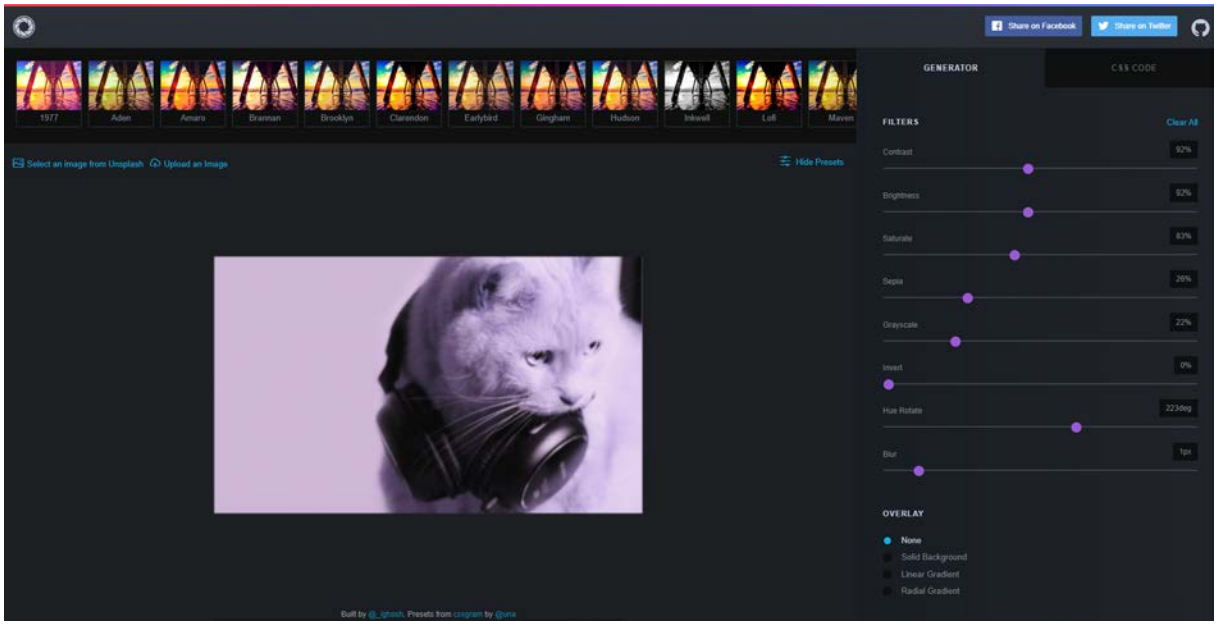


Рисунок 1.2 – Изменённое фильтрами изображение

Достоинства данного проекта:

– хороший UX веб-сайта: у пользователя не возникают вопросы, какой функционал есть у веб-сайта и какие задачи он выполняет, т.к. интерфейс удобен и понятен и позволяет легко накладывать различные фильтры для фотографий;
– содержимое веб-страницы меняется одновременно с изменениями значений фильтров, то есть для применения изменений не нужно перезагружать веб-страницу, нажимать кнопку сохранения и т.п.
– можно получить HTML & CSS код, для того чтобы вставить его в свой проект и получить фотографию с аналогичными фильтрами.

Недостатки данного проекта:

– малый функционал. Кроме наложения фильтров для фотографий веб-сайт больше не выполняет никаких функций по их редактированию;
– при закрытии веб-браузера, закрытии веб-страницы или её обновлении, содержимое страницы возвращается к первоначальному виду: все

настройки фильтров сбрасываются, а изображение изменяется на изображение по умолчанию;

– неудобный экспорт кода: нет специальной кнопки для выполнения этого действия, нужно вручную пролистывать код, выделять и копировать его.

2. Веб-сайт - https://webcode.tools/ [2] предоставляет генераторы HTML & CSS, а также JSON кода. Интерфейс веб-сайта представлен на рисунке 1.3. Генераторы кода позволяют с помощью элементов взаимодействия с пользователями настраивать внешний вид веб-компонентов и получать программный код, генерирующий такие же элементы. Так, например, на рисунке 1.4 продемонстрировано создание email-формы, обязательного элемента любого лендинга или новостного ресурса.

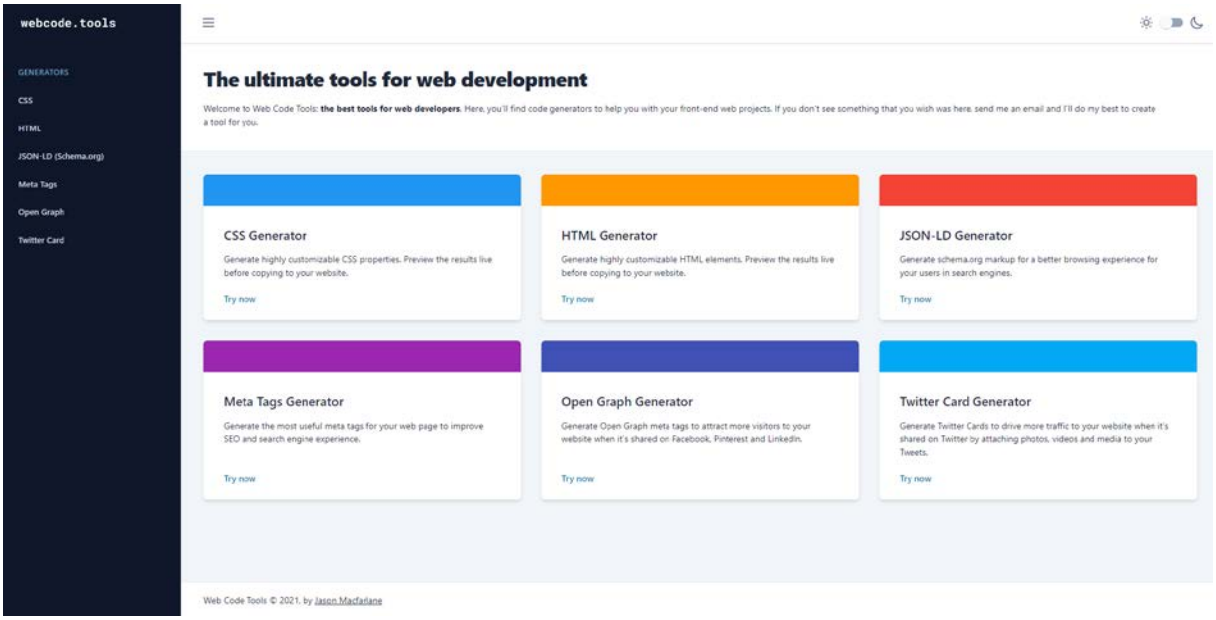


Рисунок 1.3 – Интерфейс веб-сайта https://webcode.tools/

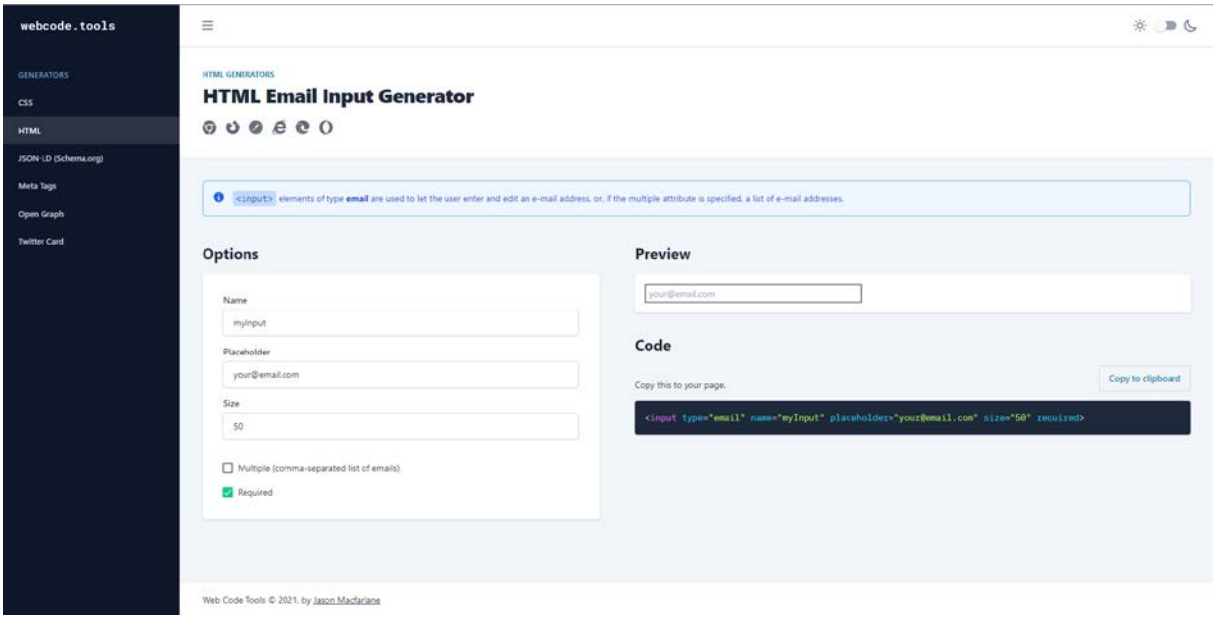


Рисунок 1.4 – Создание email-формы с помощью графического интерфейса

Достоинства данного проекта:

– приятный и современный дизайн;
– сайт не выполняет какую-то одну задачу, например, наложить фильтры на изображение, а имеет генераторы для большого спектра задач;
– динамическое изменение содержимого веб-страницы, не нужно перезагружать веб-страницу или как-то сохранять изменения, это происходит автоматически;
– удобный экспорт программного кода: нажатие одной кнопки копирует в буфер обмена весь код и позволяет тут же вставить его в свой проект.

Недостатки данного проекта:

– при закрытии веб-браузера, закрытии веб-страницы или её обновлении, содержимое страницы возвращается к первоначальному виду: все настройки сбрасываются и восстанавливаются к первоначальным значениям;

– для каждого свойства открывается отдельная вкладка в веб-браузере, соответственно нельзя в одном окне настроить, например, свойства цвета блока и свойства текста внутри блока. Из-за этого нельзя комбинировать различные свойства компонентов внутри этого веб-сайта. Чтобы настроить несколько параметров компонента, нужно открыть несколько вкладок, настроить эти параметры в отдельных вкладках и несколько раз копировать генерируемый код, что по итогу может занять больше времени, чем если бы разработчик писал этот код вручную;
– представлена лишь малая часть различных CSS-свойств, определённых в спецификации [3], соответственно функционал веб-сайта не может полноценно выполнять функцию генератора CSS-кода.

3. Веб-сайт - https://neumorphism.io/ [4] предоставляет различные параметры для генерации теней у блоков. Например, можно настроить размер, радиус, интенсивность и другие свойства, присущие теням, всё это делается с помощью графического интерфейса, а изменения моментально отображаются на странице. Примеры того, как происходит работа с веб-сайтом приставлены на рисунках 1.5 и 1.6.

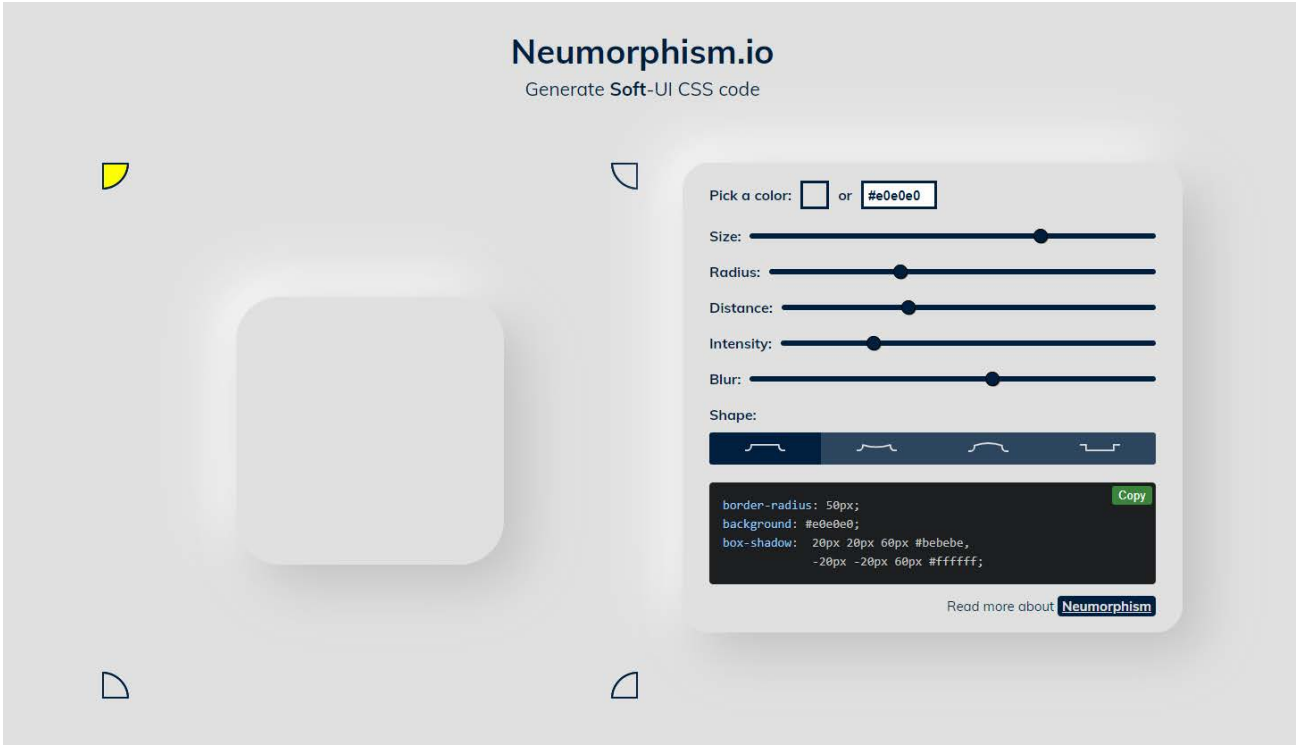


Рисунок 1.5 – Интерфейс веб-сайта https://neumorphism.io/

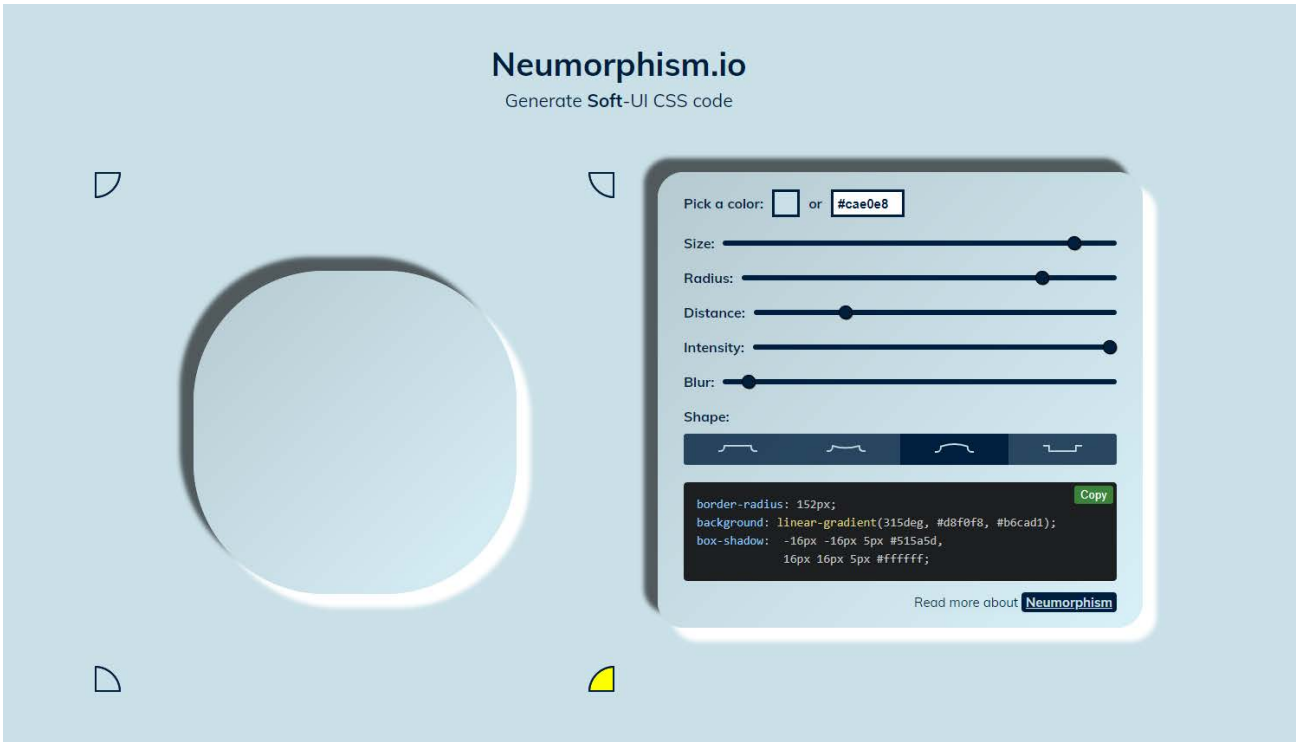


Рисунок 1.6 – Изменение теней у блока с помощью интерфейса

Достоинства данного проекта:

- приятный и современный дизайн;
- динамическое изменение содержимого веб-страницы, не нужно перезагружать веб-страницу или как-то сохранять изменения, это происходит автоматически;
- удобный экспорт программного кода: нажатие одной кнопки копирует в буфер обмена весь код и позволяет тут же вставить его в свой проект.

Недостатки данного проекта:

- при закрытии веб-браузера, закрытии веб-страницы или её обновлении, содержимое страницы возвращается к первоначальному виду: все настройки сбрасываются и восстанавливаются к первоначальным значениям;
- малый функционал. Кроме редактирования теней и размера блоков, этот веб-сайт больше ничего не делает.

1.2 АНАЛИЗ ОСНОВНЫХ ТЕХНОЛОГИЧЕСКИХ РЕШЕНИЙ

1.2.1 ЯЗЫКИ КЛИЕНТСКОЙ ЧАСТИ

Клиентская часть любого веб-сайта строится на трёх основных технологиях: язык разметки – HTML (HyperText Markup Language), язык стилей – CSS (Cascading Style Sheets) и язык программирования JavaScript. В области frontend-разработки на сегодняшний день нет технологий, которые смогли бы их заменить. Такая монополия обусловлена особенностью работы веб-браузеров (программ занимающихся отрисовкой клиентской части любых веб-сайтов), т.к. они работают только с данным стеком технологий.

Для языков HTML и CSS существуют препроцессоры, например Pug для HTML [5] или SASS & SCSS для CSS [6]. Они предоставляют специальный упрощённый синтаксис для этих языков. Используются только во время разработки, т.к. веб-браузер не может с ними работать, поэтому они должны транслироваться в HTML & CSS и только потом запускаться в веб-браузере.

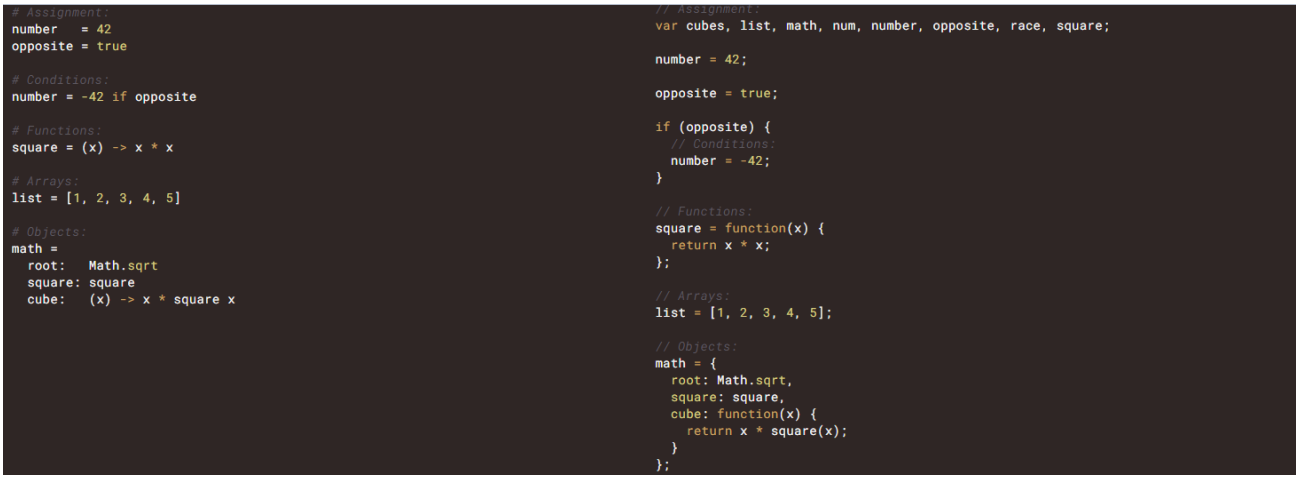
Использование препроцессоров имеет смысл в больших системах с огромной кодовой базой, т.к. они упрощают процесс написания архитектуры интерфейсов и уменьшают количество ошибок. Использование же в малых системах является не таким оправданным, т.к. их достоинства здесь не так ярко выражены, а необходимость настраивать правильную работу препроцессоров и заново проводить компиляцию кода препроцессоров в HTML & CSS код при каждом изменении проекта лишь замедлит разработку системы.

Поэтому для разработки веб-сайта будем пользоваться HTML & CSS, без использования различных препроцессоров.

Для backend-разработки веб-сайтов можно использовать множество различных ЯП: PHP, Python, Ruby, C#, Golang, JS и т.д., для frontend-разработки

может использоваться только ЯП, которые в конечном итоге транслируются в JavaScript, например, языки TypeScript и CoffeeScript.

Язык программирования CoffeeScript является синтаксическим сахаром для языка JavaScript. Целью создания CoffeeScript было сокращение времени разработки на JavaScript, делая синтаксис языка более простым и коротким [7]. Пример синтаксиса на CoffeeScript и его эквивалента на JavaScript представлен на рисунке 1.8.



```
# Assignment:
number   = 42
opposite = true

# Conditions:
number = -42 if opposite

# Functions:
square = (x) -> x * x

# Arrays:
list = [1, 2, 3, 4, 5]

# Objects:
math =
  root:   Math.sqrt
  square: square
  cube:   (x) -> x * square x
```

```
// Assignment:
var cubes, list, math, num, number, opposite, race, square;

number = 42;

opposite = true;

if (opposite) {
  // Conditions:
  number = -42;
}

// Functions:
square = function(x) {
  return x * x;
};

// Arrays:
list = [1, 2, 3, 4, 5];

// Objects:
math = {
  root: Math.sqrt,
  square: square,
  cube: function(x) {
    return x * square(x);
  }
};
```

Рисунок 1.7 – Синтаксис на CoffeeScript (слева) и эквивалентный синтаксис на JavaScript (справа)

И хотя написание кода может благодаря использованию CoffeeScript станет быстрее, разработка может замедлиться из-за следующих проблем: необходимость перекомпиляции в JavaScript при каждом изменении в проекте, проблемы с компиляцией в JavaScript (особенно при использовании новых функций языка), отладка усложняется из-за того, что ошибки возникают при исполнении Javascript, а исправлять их надо в CoffeeScript.

Язык программирования TypeScript – это статически типизированный язык, основанный на JavaScript [8]. Наличие статической системы типов, интерфейсов, ООП, позволяет писать высоконагруженные крупные проекты с

гораздо меньшим количеством проблем и багов, а также более чистой архитектурой, чем если делать это с помощью JavaScript. Как и на любом другом статически типизированном языке, разработка на TypeScript занимает гораздо больше времени чем разработка на ЯП с динамической типизацией. Также его сложнее поддерживать, а любые изменения в проекте требуют перекомпиляции.

TypeScript и CoffeeScript являются хорошими ЯП для frontend-разработки, но их достоинства дают больший результат в больших и сложных проектах, в малых проектах они предоставляют больше неудобств, чем пользы. Поэтому в рамках ВКР в качестве ЯП для frontend-разработки выберем чистый JavaScript.

1.2.2 ВЫБОР JAVASCRIPT ФРЕЙМВОРКА

Во frontend-разработке существует множество различных фреймворков и библиотек. Согласно опросу более чем 4500 профессиональных разработчиков о фреймворках, которые они используют [9], а также ежегодному опросу среди разработчиков на stackoverflow [10], бесспорным лидером по популярности среди разработчиков является React от компании Facebook. Также стоит обратить внимание на такие фреймворки, как Angular и Vue, которые также пользуются большой популярностью у разработчиков. Рассмотрим их достоинства и недостатки:

1. React – библиотека JavaScript, созданная компанией Facebook в 2013 году и предназначенная для создания интерфейсов веб-приложений, по своему функционалу не уступает различным фреймворкам, а потому часто к ним причисляется.

   Достоинства React [11]:

– быстрый рендеринг интерфейсов, благодаря React Virtual DOM;
– поддержка Progressive Web App (PWA);

– поддержка TypeScript;
– лёгок в изучении, т.к. имеет простой, в сравнении с фреймворками синтаксис и не имеет навязанной архитектуры кода;
– большое сообщество разработчиков, а также большое количество справочной информации.

Недостатки React [11]:

– нет структурности и навязанной архитектуры кода, как у других фреймворков, из-за чего построение архитектуры приложений становится ответственностью разработчиков;
– нет единого стиля написания CSS-кода, что может вызвать путаницу;
– смешивание логики приложений и их представлений, что также плохо сказывается на архитектуре.

2. Angular – один из самых старых JS-фреймворков, созданный в 2009 году, на сегодняшний день полностью переписанный на TypeScript, поддерживаемый компанией Google и представляющий широкий спектр инструментов для создания интерактивных веб-приложений.

   Достоинства Angular [11]:

– является лучшим фреймворком при разработке на TypeScript, т.к. имеет инструменты для работы с этим языком, а его архитектура устроена таким образом, чтобы вести разработку именно на языке TypeScript;
– архитектура, специально созданная для больших масштабируемых приложений.

Недостатки Angular [11]:

– долгое время рендеринга интерфейсов, по сравнению с другими фреймворками;

– односторонняя привязка данных, обеспечивающая минимальный риск возможных ошибок, связанных с изменением данных;
– большое количество потребляемой оперативной памяти, по сравнению с другими фреймворками;
– сложный синтаксис, привязанный к TypeScript, а не к JavaScript, из-за чего порог входа выше, чем у других фреймворков;
– проблемы с миграцией, при переходе от старой версии к новой.

3. Vue – JavaScript фреймворк с открытым исходным кодом и предназначенный для разработки интерфейсов веб-приложений. Vue появился позже, чем Angular и React, а его создатели попытались перенять лучшее из обоих и избавиться от недостатков.

   Достоинства Vue [11]:

– поддерживаемая и постоянно обновляемая документация, содержащая в том числе примеры использования;
– использовании концепции виртуального DOM (как в React), что заметно повышает скорость рендеринга интерфейсов;
– концепция однофайловых компонентов упрощает архитектуру приложений и переиспользование кода;
– двусторонняя привязка данных, позволяющая легко и быстро изменять состояние интерфейсов веб-приложений;
– малое использование памяти, по сравнению с другими фреймворками, Vue может использовать всего 20 КБ и при этом сохранять скорость и гибкость, что заметно улучшает производительность.

   Недостатки Vue [11]:

– проблемы с интеграцией в больших проектах;

– являясь самым молодым фреймворком среди React и Angular, Vue имеет меньшее количество различных библиотек и прочих инструментов для разработки больших проектов.

Для реализации проекта будем использовать фреймворк Vue, т.к. он обладает следующими ключевыми преимуществами:

– в отличие от Angular, во Vue есть реактивность позволяющая легко изменять состояние интерфейсов. Достаточно изменить данные в коде, и они автоматически изменятся в интерфейсе, не нужно постоянно вызывать методы рендеринга при любом изменении данных;
– в отличие от React, во Vue есть навязанная архитектура, упрощающая расширение и переиспользование кода, а также делающая его понятнее.

1.3 ВЫВОД

Суть решаемой задачи: создать веб-сайт с набором графических редакторов для создания веб-компонентов, позволяющих генерировать HTML & CSS код с помощью графического интерфейса, а затем импортировать его в свои проекты. Также такие редакторы позволят наглядно демонстрировать возможности HTML & CSS, что пригодится начинающим разработчикам и верстальщикам.

Для разработки системы будем использовать следующие технологии: HTML – для создания структуры страниц веб-сайта, CSS – для изменения внешнего вида веб-сайта, JavaScript – для программной логики веб-сайта, Vue – для создания архитектуры и простой работы с интерфейсами веб-сайта.

2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

2.1 ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

На этапе проектирования системы были выявлены следующие функциональные требования:

- возможность выбрать веб-компонент из стандартного набора для дальнейшего редактирования. Стандартный набор содержит список веб-компонентов, которые можно создавать;
- наличие рабочей области, в которой отображается редактируемый веб-компонент;
- отображение текущего веб-компонента и его изменений в рабочей области;
- содержимое редактируемого веб-компонента должно обновляться динамически (без полного обновления страницы) при изменении его параметров;
- панель инструментов должна меняться для различных видов веб-компонентов. Панель инструментов представляет собой набор кнопок, форм и прочих средств для получения информации от пользователя. Пользователь с помощью этой панели инструментов сможет легко настраивать длину, ширину, цвет, границы и т.д., а также специфичные для веб-компонентов свойства. Пример набора инструментов для создания и настройки изображения: настройка размеров, относительного положения, растяжения, заполнения, границ, цветов, различных эффектов и т.д.;
- панель инструментов должна работать с системой цветов RGBA;
- экспорт программного кода веб-компонентов на языках HTML & CSS с помощью специальной кнопки;

2.2 НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

На этапе проектирования системы были выявлены следующие нефункциональные требования:

- веб-сайт должен работать без перерывов круглые сутки;
- функционал веб-сайта должен быть интуитивно понятен (user-friendly interface);
- веб-сайт должен поддерживать свой функционал на следующих веб-браузерах, ОС (операционных системах), экранах:

Веб-браузеры: Edge версии не ниже 16, Firefox версии не ниже 52, Chrome (Desktop) версии не ниже 57, Safari (Desktop) версии не ниже 10.1, Opera версии не ниже 44. Другие веб-браузеры могут не поддерживать различный функционал веб-сайта или отображать его неправильно, поэтому их использование не рекомендуется.

ОС: Windows, Linux, Mac OS. Различия между ОС не должны влиять на работу системы, т.к. за её работу отвечают веб-браузеры, таким образом работу системы могут поддерживать и ОС, не указанные ранее, главное, чтобы они поддерживали веб-браузеры, указанные ранее.

Разрешение экрана: система должна быть оптимизирована под различные разрешения и пропорции экранов, поэтому у системы нет ограничений для использования на разных экранах. Рекомендуемое разрешение для комфортной работы с системой: не ниже 1280 x 720. Также система не предусматривает работу на телефонных экранах, т.к. эта система подразумевает использование для разработки, которой не занимаются на телефонах.

3. ПРОЕКТИРОВАНИЕ

3.1 АРХИТЕКТУРА ПРЕДЛАГАЕМОГО РЕШЕНИЯ

Веб-сайт представляет из себя набор редакторов кода, представленных в виде html-страниц, с подключенными к ним файлами стилей (.css), а также подключенным Javascript-кодом.

На каждой странице существуют формы для взаимодействия с пользователем, с помощью которых он может задавать различные параметры редактируемого объекта, пример таких форм представлен на рисунке 3.1.

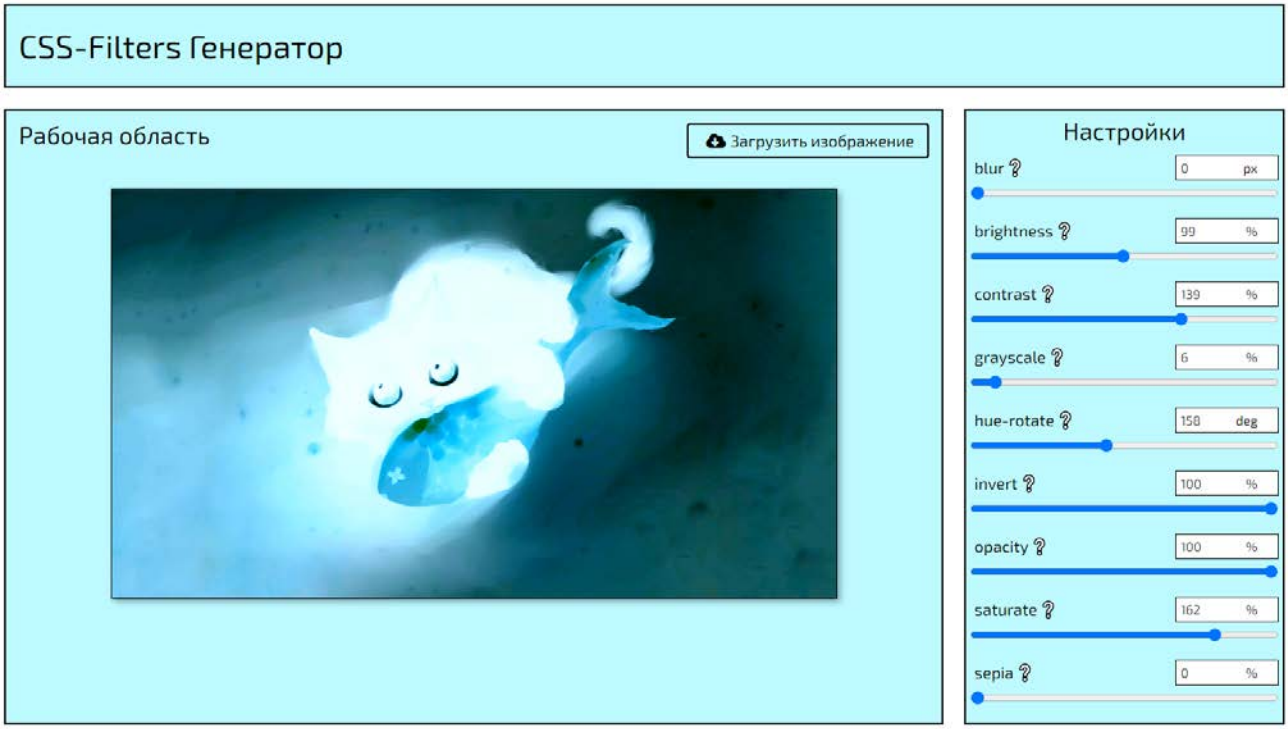


Рисунок 3.1 – Редактор фильтров изображения

Каждая страница, кроме главной, представляет собой подобный редактор со своими специфичными формами взаимодействия с пользователем. Данные этих форм хранятся на стороне пользователя и обрабатываются в режиме

реального времени с помощью фреймворка Vue.js, схема того, как это устроено представлена на рисунке 3.2.

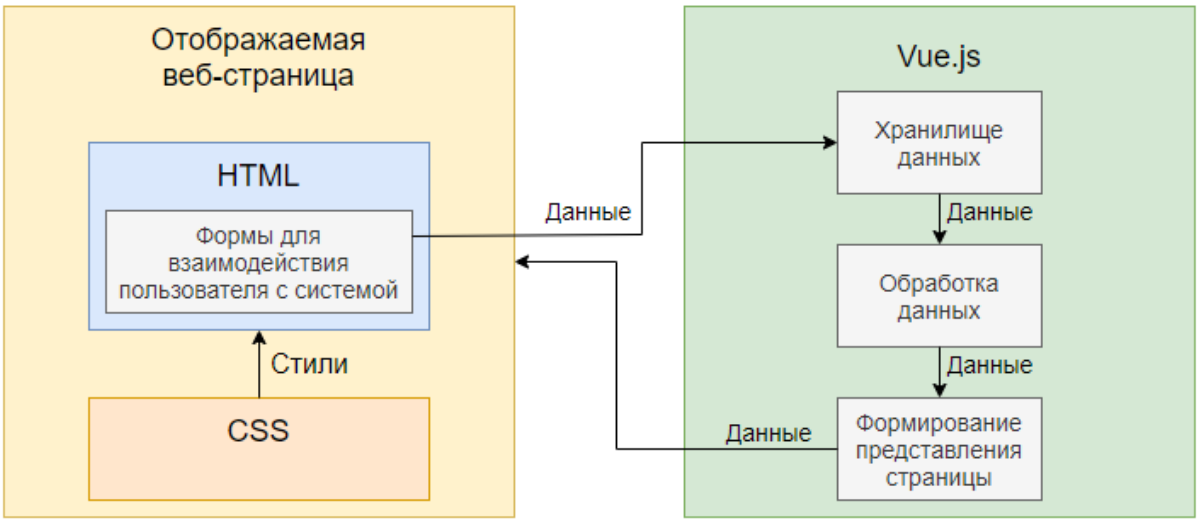


Рисунок 3.2 – Архитектура страницы-редактора

Значения данных форм веб-страницы записываются в хранилище данных объекта Vue. Это могут быть цифровые значения полей, значения цветов, строки и т.д.

Вспомогательные методы обрабатывают эти данные и формируют будущую структуру веб-страницы.

С помощью различных директив и методов фреймворка Vue.js, обработанные данные изменяют содержимое веб-страницы, чтобы значения форм веб-страницы соответствовали его визуальному представлению.

3.2 ОПИСАНИЕ ДАННЫХ

При описании архитектуры разрабатываемого веб-сайта были спроектированы диаграммы потоков данных для отображения взаимодействия данных системы, представленные на рисунке 3.3 и рисунке 3.4.

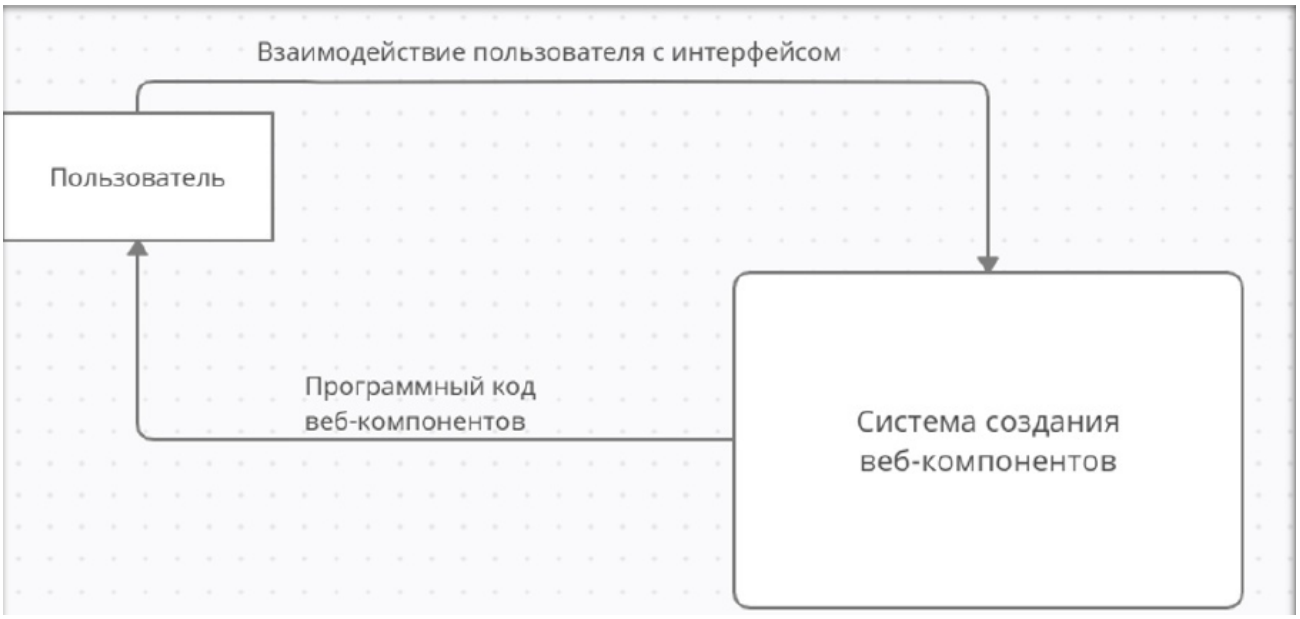


Рисунок 3.3 – Диаграмма потоков данных, 1-ый уровень

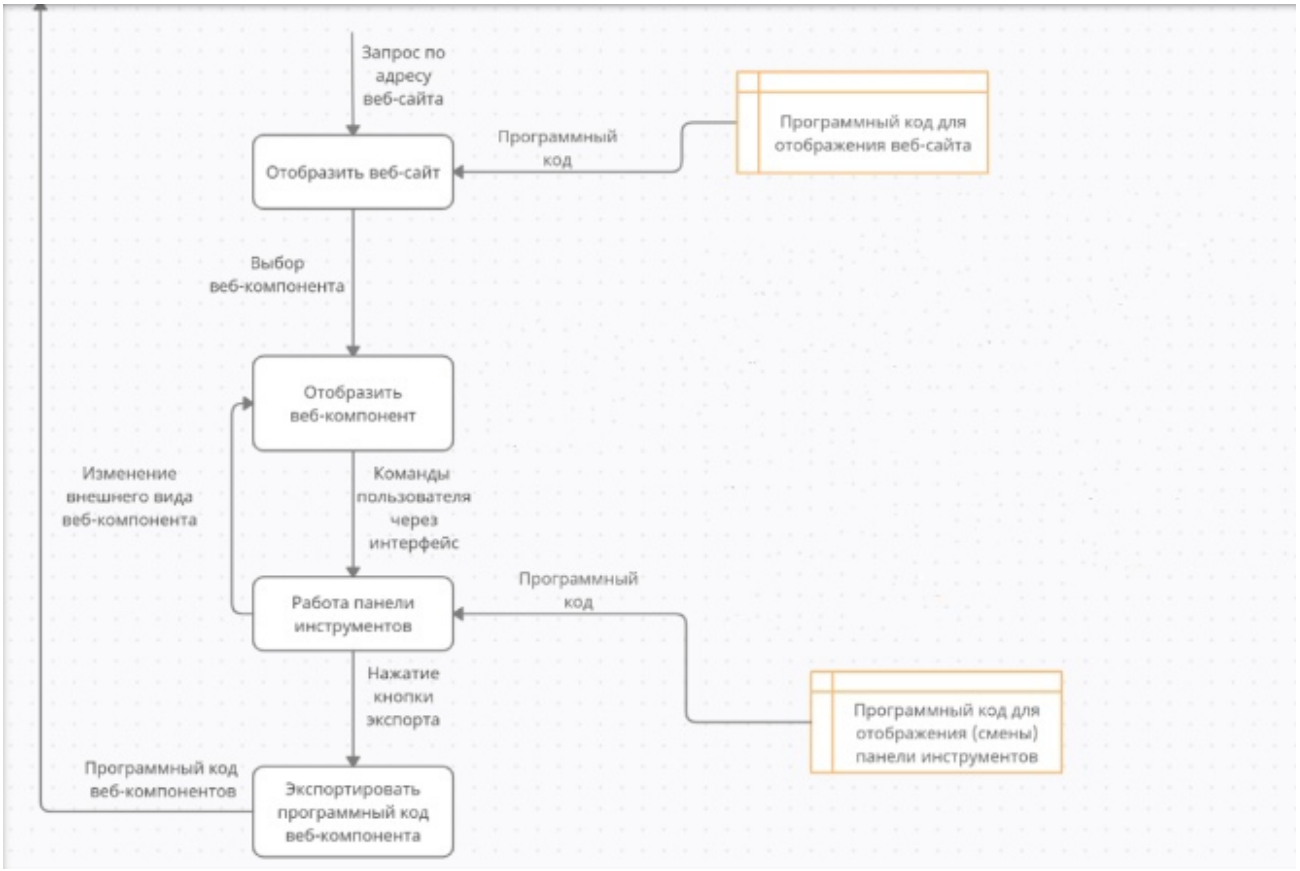


Рисунок 3.4 – Диаграмма потоков данных, 2-ой уровень

4. РЕАЛИЗАЦИЯ

4.1 ВЁРСТКА ВЕБ-САЙТА

Веб-сайт состоит из главной веб-страницы, на которой находится информация о функционале веб-сайта, и веб-страниц редакторов, с помощью которых можно сгенерировать код.

Каждая веб-страница состоит из трёх частей: header, main, footer. Header и footer одинаковые для всех веб-страниц.

В блоке header содержится меню веб-сайта, из которого можно открыть любой редактор, а также название веб-сайта. Header веб-сайта представлен на рисунке 4.1.

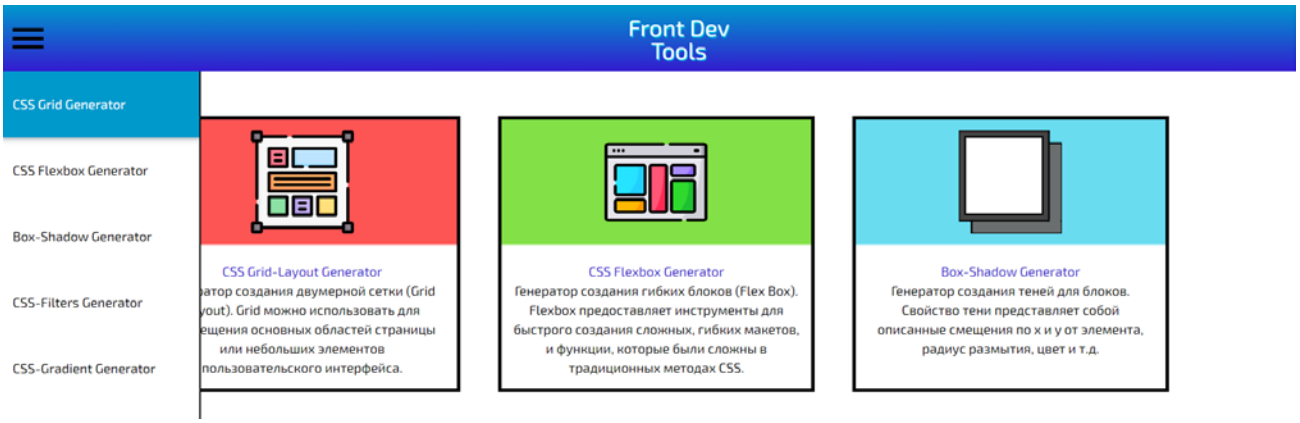


Рисунок 4.1 – Header главной веб-страницы

В блоке footer содержится информация о назначении веб-сайта, ссылки на другие веб-страницы и ссылки на соц. сети автора. Footer веб-сайта представлен на рисунке 4.2.

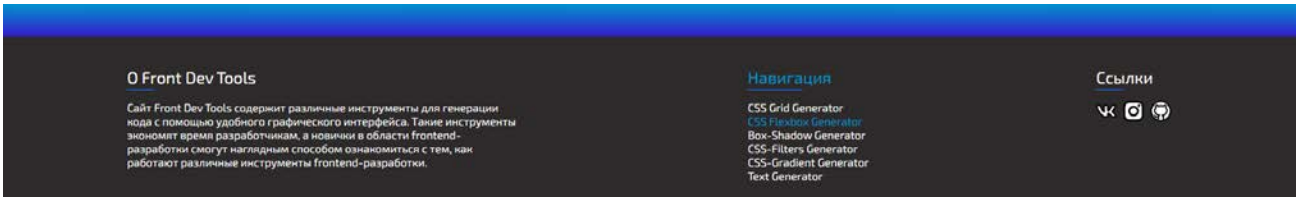


Рисунок 4.2 – Footer главной веб-страницы

В блоке main находится основное содержимое веб-страницы, являющееся уникальным для каждой веб-страницы. Пример того, как выглядит блок main на веб-странице представлен на рисунке 4.3.

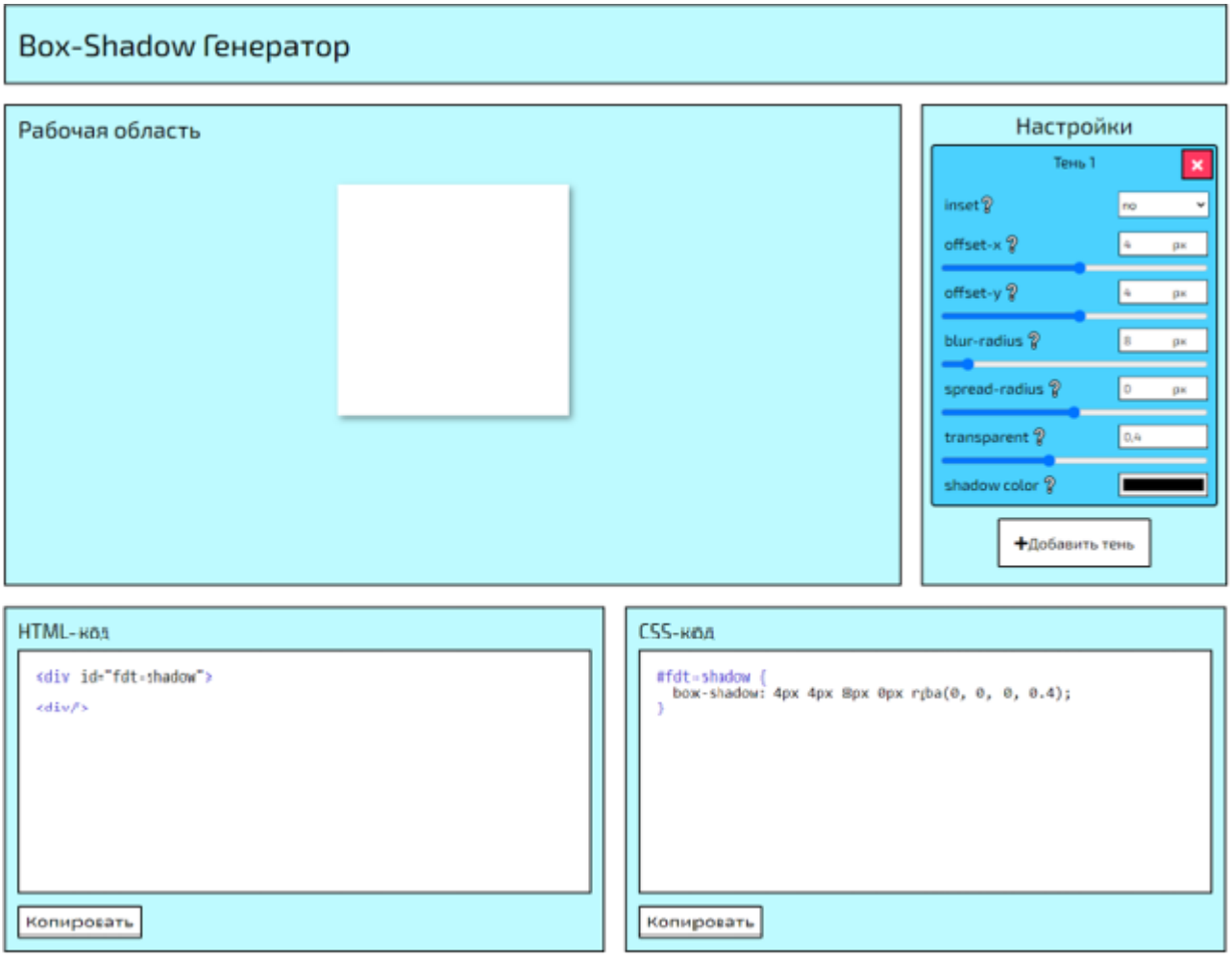


Рисунок 4.3 – Содержимое блока main редактора теней

В рабочей области блока main находится редактируемый объект, вид которого зависит от веб-страницы. Например, в редакторе теней – это блок с тенью, в редакторе grid – это таблица блоков. Справа от рабочей области находится блок настроек редактируемого объекта, именно с этим блоком работает пользователь.

Далее идут блоки, содержащие сгенерированный HTML & CSS код, позволяющий импортировать редактируемый объект в свой проект. Для удобства пользователя там также присутствуют кнопки, обрабатывающие событие клика по ним и копирующие содержимое блоков кода в буфер обмена.

4.2 СВЯЗЫВАНИЕ ДАННЫХ

На каждой веб-странице редакторе есть формы для получения информации от пользователя. Значения этих форм хранятся в объектах, созданных с помощью фреймворка Vue.js и имеют двустороннюю привязку, которую обеспечивает фреймворк. Таким образом, если данные изменяются в форме, они одновременно изменяются в объекте Vue и наоборот. Пример того, как выглядит Vue-объект и его данные, представлен на листинге 4.1.

Листинг 4.1 – Данные Vue-объекта страницы filter.html

```
const vueApp = Vue.createApp({
  data() {
    return {
      filterBlur: "0",
      filterBrightness: "100",
      filterContrast: "100",
      filterGrayscale: "0",
      filterHueRotate: "0",
      filterInvert: "0",
      filterOpacity: "100",
      filterSaturate: "100",
      filterSepia: "0",
      filterStringCSS: "",
    }
  },
}).mount('#app');
```

На рисунке 4.4 продемонстрировано соответствие этих данных с данными форм на веб-странице.

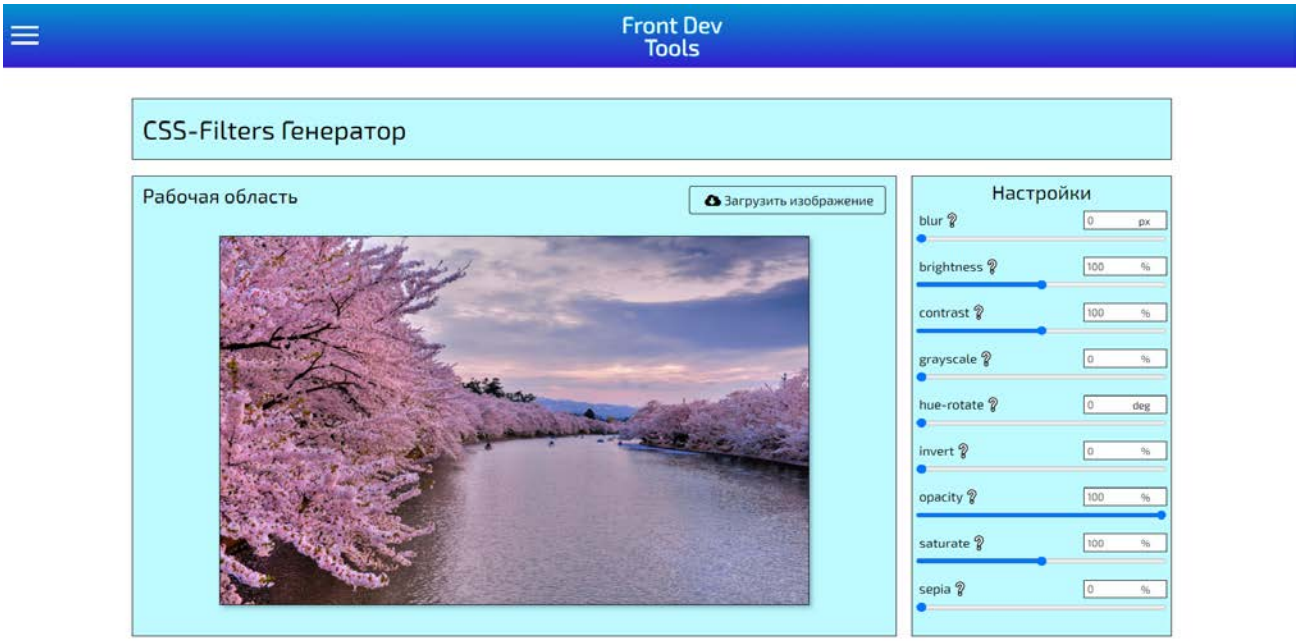


Рисунок 4.4 – Состояние форм на веб-странице filter.html

Двусторонняя связка данных обеспечивается с помощью специальных директив v-model [12]. Они используются в качестве атрибутов html-элементов и обрабатываются фреймворком Vue.js. Так, можно связать значение html-элемента input с данными приложения. Пример того, как выглядит элемент с двусторонней привязкой данных представлен на листинге 4.2.

Листинг 4.2 – Html-элемент, с двусторонней привязкой данных

```
<div class="flexbox-item__div1">
<div class="settings__grid-text">
blur
<div class="tooltip tooltip-settings">&#x2754
<span class="tooltiptext">Размытие. Применяет к изображению размытие по Гауссу. Значение оп
ределяет значение стандартного отклонения функции Гаусса или количество пикселей на экране,
 сливающихся друг с другом, поэтому большее значение приведет к большему размытию. Если пар
аметр не указан, используется значение 0. Параметр указывается как длина CSS, но не принима
ет процентные значения.</span>
</div>
</div>
```

Продолжение листинга 4.2

```
</div>

<div class="flexbox-item__div2">
<p class="control" >
<input type="number" min="0" max="10" v-model="filterBlur" class="input__percent-value">
<span>px</span>
</p>
</div>

<div class="flexbox-item__div3">
<p class="control">
<input type="range" min="0" max="10" v-model="filterBlur" class="control__range-input">
</p>
</div>
</div>
```

Данные форм (html-элементы input) привязаны к данным приложения с помощью директивы v-model.

С помощью директивы Vue: v-bind можно связывать значение атрибутов html-элементов со значением данных Vue-объекта и таким образом задавать inline-стили [13]. Таким образом, директива v-bind:style позволяет изменить значение атрибута style html-элемента, то есть прописать для него css-код.

На листинге 4.3 представлен элемент страницы text.html, для которого стили задаются с помощью данных Vue-объекта, которые задал до этого пользователь веб-сайта. Данные Vue-объекта представлены на листинге 4.4.

Листинг 4.3 – Элемент рабочей области, в котором находится редактируемый текст

```
<div class="working-area__cell">
```

Продолжение листинга 4.3

```
        <textarea name="" id="" cols="75" rows="15" v-
model="textAreaText" placeholder="Введи свой текст" v-bind:style="{ 'font-
size': fontSizeValue + fontSizeUnit, 'font-weight': fontWeight, 'font-
style': fontStyle, 'font-family': fontFamily, 'color': color, 'background-
color': background, 'text-decoration': textDecoration, 'text-
transform': textTransform, 'text-align': textAlign, 'line-
height': lineHeightValue + lineHeightUnit}"></textarea>
            </div>
```

Листинг 4.4 – Vue-объект страницы text.html

```
const vueApp = Vue.createApp({
        data() {
          return {
            fontSizeValue: "1",
            fontSizeUnit: "vw",
            fontWeight: "normal",
            fontStyle: "normal",
            fontFamily: "sans-serif",
            color: "#000000",
            background: "#ffffff",
            textDecoration: "none",
            textTransform: "none",
            textAlign: "left",
            lineHeightValue: "1",
            lineHeightUnit: "vw",
            textAreaText: "Lorem ipsum dolor sit amet consectetur adipisicing elit. Optio p
orro voluptas fuga ducimus hic, animi ut alias dicta repudiandae incidunt quo natus vel. Il
lum modi deserunt porro quisquam provident maxime!",
          }
        },
        methods: {
          copyText(selector) {
            let text = document.querySelector(`${selector}`);
```

Продолжение листинга 4.4

```
      let range = document.createRange();
      range.selectNode(text);
      window.getSelection().addRange(range);
      document.execCommand("copy");
      window.getSelection().removeAllRanges();
    },
  }
}).mount("#app");
```

4.3 ВСПОМОГАТЕЛЬНЫЕ МЕТОДЫ

Все методы Vue-объектов хранятся в свойстве methods. Vue.js автоматически связывает контекст выполнения этих методов с данными Vue-объекта, а значит и с данными веб-страницы [14].

Для веб-страниц проекта существуют, как уникальные методы, обрабатывающие специфичные для веб-страницы данные, так и универсальные методы.

Универсальный метод, позволяющий скопировать содержимое сгенерированного кода в буфер обмена и работающий на каждой веб-странице проекта, является обработчиком события “клик” специальных кнопок и представлен на листинге 4.5.

Листинг 4.5 – Обработчик события нажатия на кнопку “Копировать”

```
copyText(selector) {
            let text = document.querySelector(`${selector}`);
            let range = document.createRange();
            range.selectNode(text);
            window.getSelection().addRange(range);
            document.execCommand("copy");
            window.getSelection().removeAllRanges();
          }
```

На веб-странице grid.html представлен редактор CSS Grid – технологии, позволяющей создавать сетки для разметки страницы. Устройство этих сеток аналогично устройству математических матриц, имеющих в качестве основных параметров – количество строк и количество столбцов. Количество элементов сетки определяется количеством строк и столбцов и должно постоянно обновляться при изменении этих параметров, за это отвечает метод,

представленный на листинге 4.6, это специфичный для это веб-страницы метод, не присутствующий больше нигде в проекте.

Листинг 4.6 – Обработчик, обновляющий количество элементов grid-сетки

```
gridCellsChange() {
            let columnsArrLength = Number(this.gridColumns);
            let rowsArrLength = Number(this.gridRows);
            this.col = columnsArrLength * rowsArrLength;
            while (this.gridTemplateColumnsArr.length != columnsArrLength) {
              if (this.gridTemplateColumnsArr.length < columnsArrLength) {
                this.gridTemplateColumnsArr.push(["100", "px"]);
              } else {
                this.gridTemplateColumnsArr.pop();
              }
            }
            while (this.gridTemplateRowsArr.length != rowsArrLength) {
              if (this.gridTemplateRowsArr.length < rowsArrLength) {
                this.gridTemplateRowsArr.push(["100", "px"]);
              } else {
                this.gridTemplateRowsArr.pop();
              }
            }
          }
```

5. ТЕСТИРОВАНИЕ

5.1 ПРОВЕДЕНИЕ ПРОЦЕДУРЫ ТЕСТИРОВАНИЯ

В рамках дипломного проекта было проведено функциональное тестирование реализованного веб-сайта.

Тестирование работы форм по управлению объектом рабочей области веб-страницы text.html представлены на рисунке 5.1 и рисунке 5.2.

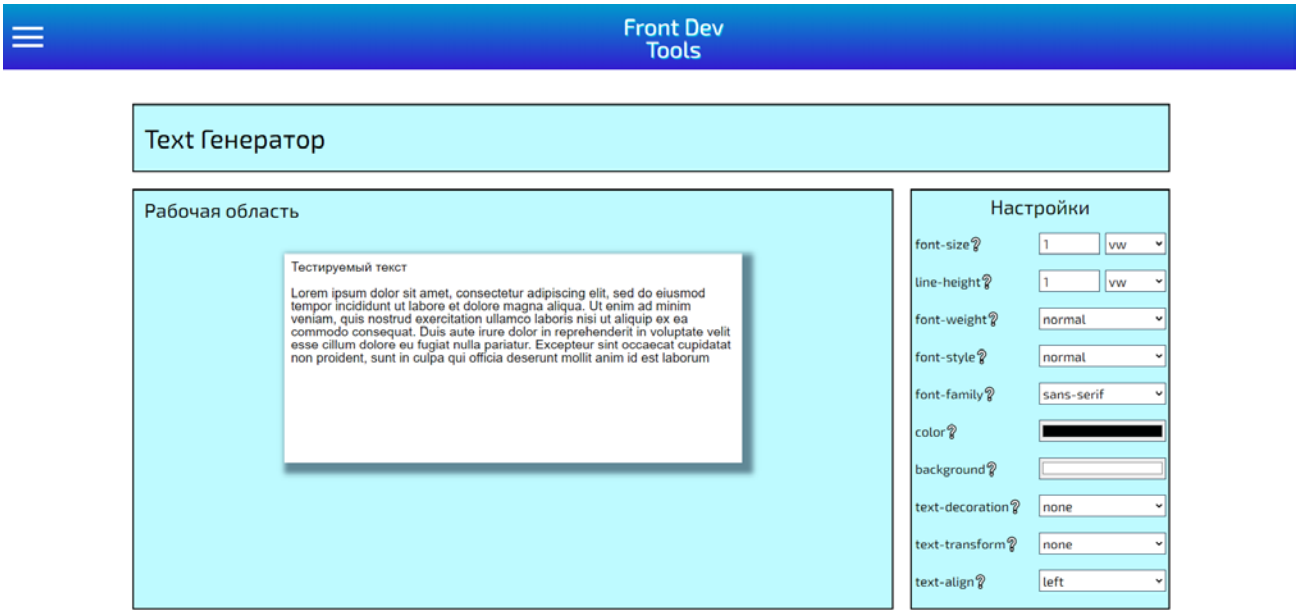


Рисунок 5.1 – Редактор текста и его формы для настройки текста в рабочей области в начальном состоянии

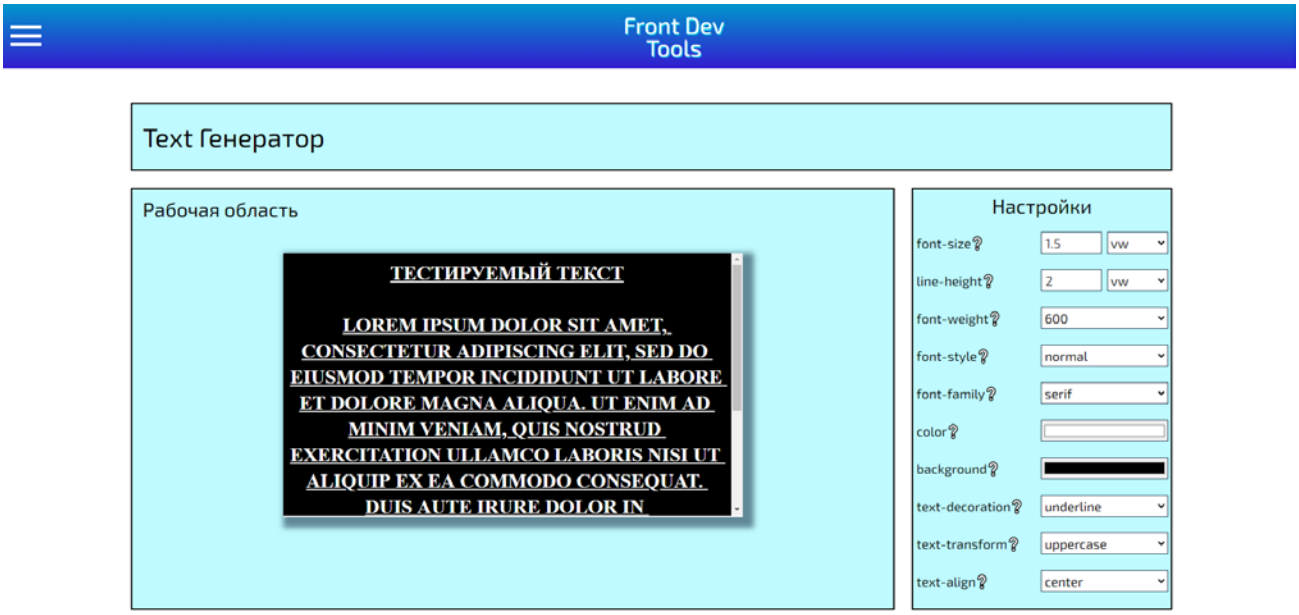


Рисунок 5.2 – Состояние текста после изменения значения форм управления

Как видно из рисунков 5.1 и 5.2, изменение значений форм управления приводит к изменению состояния текста в рабочей области, соответственно формы управления выполняют требуемый функционал.

Помимо форм с фиксированными вариантами выбора, в настройках присутствуют формы для ввода чисел (размер текста и размер высоты линии). При попытке ввести там отрицательные числа или текст, элемент формы подсвечивается красным цветом, сигнализирующим о невалидности вводимого значения. А редактируемый объект не принимает иррациональные значения. Пример ввода таких значений продемонстрирован на рисунке 5.3

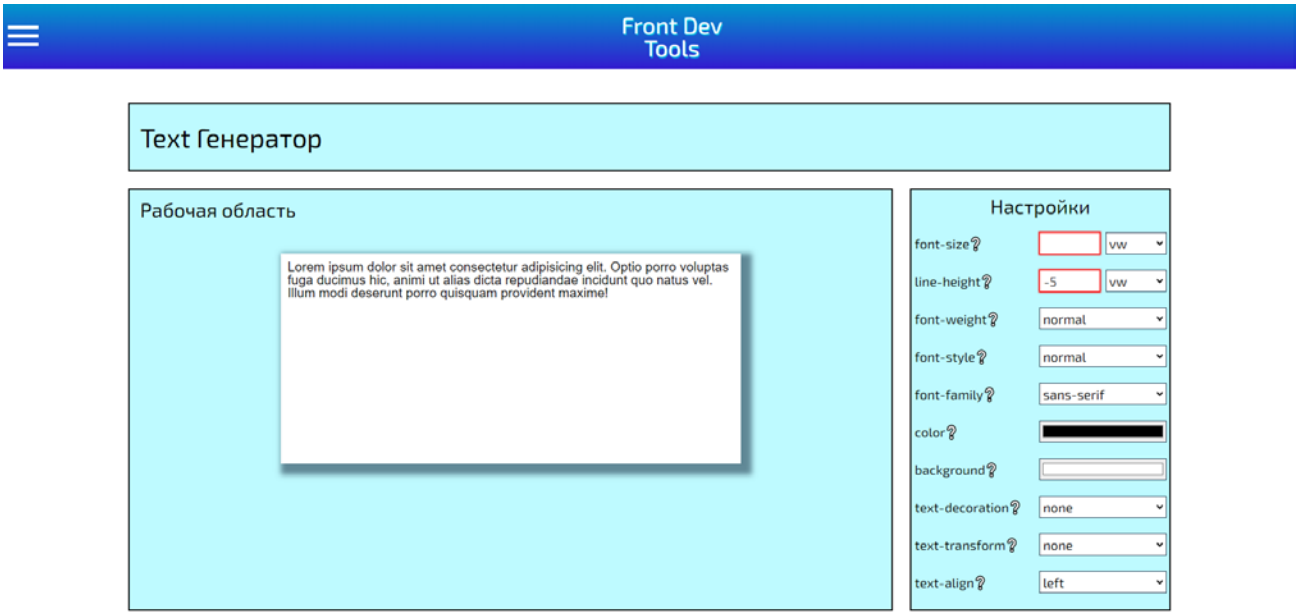


Рисунок 5.3 – Тестирование форм на невалидные значения

Основная задача каждого генератора веб-сайта – это генерация кода, поэтому необходимо её протестировать. Примеры того, как меняется содержимое блоков со сгенерированным кодом продемонстрированы на рисунке 5.4 и рисунке 5.5.

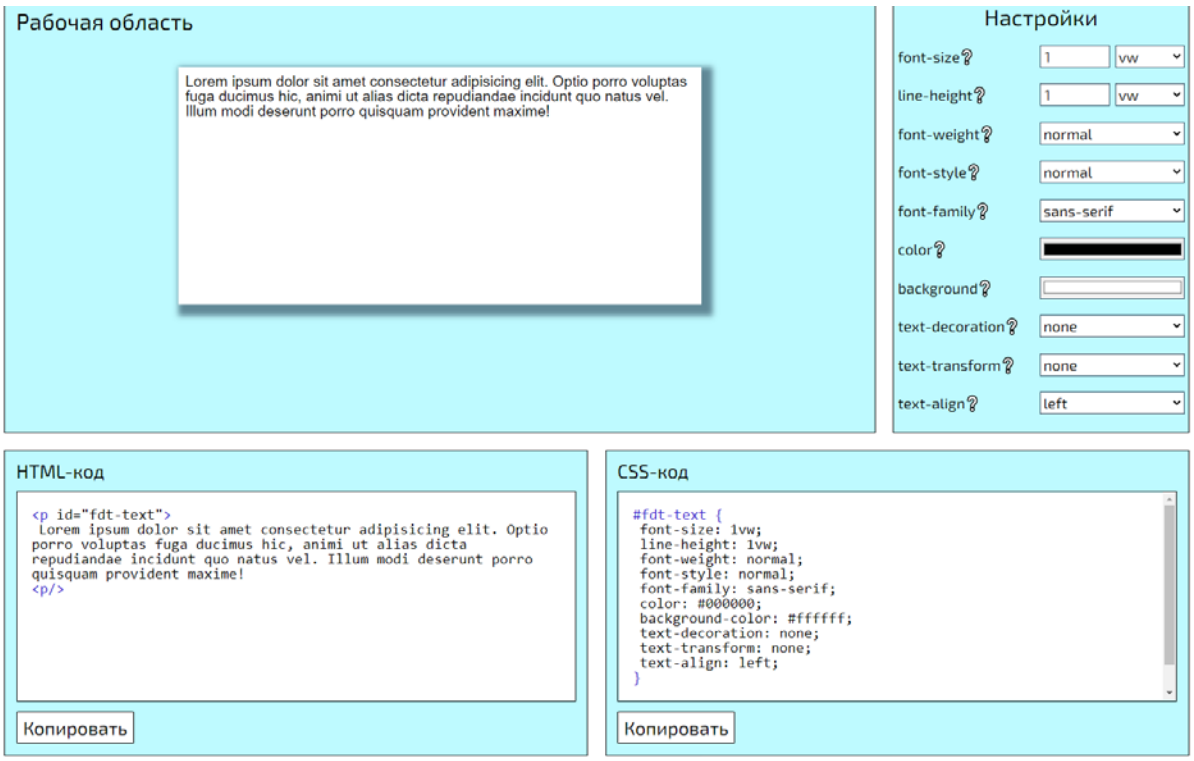


Рисунок 5.4 – Состояние сгенерированного кода до изменений пользователя

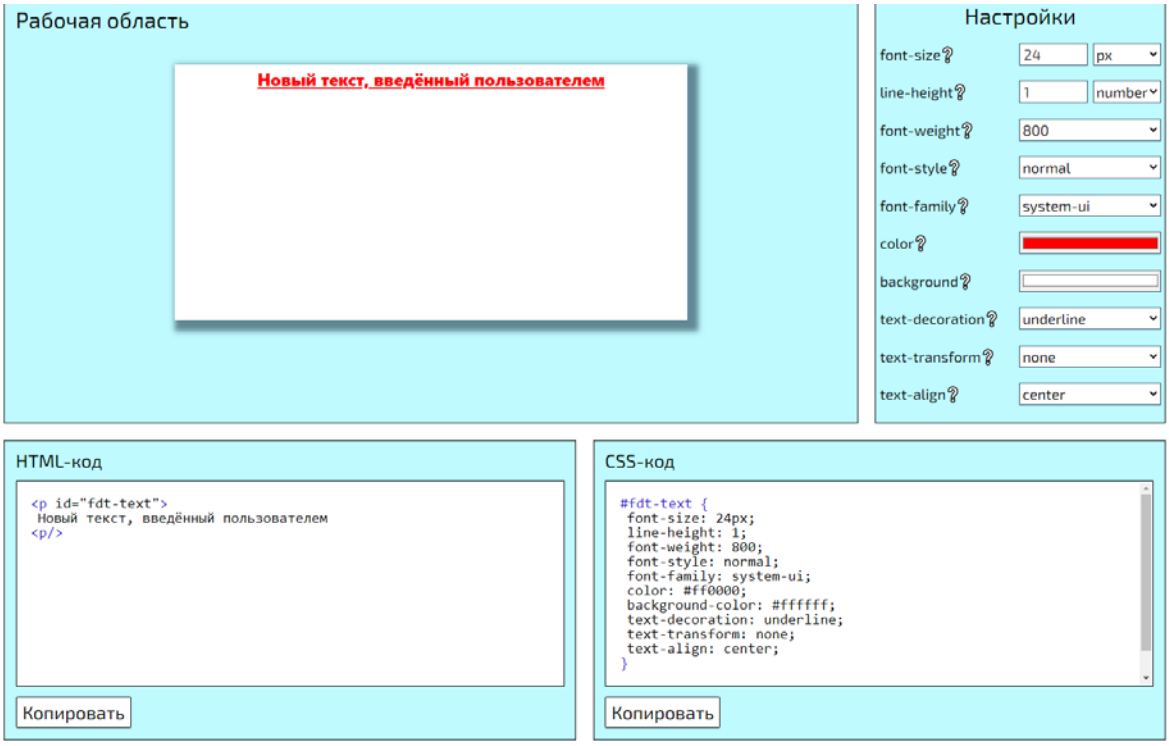


Рисунок 5.5 – Состояние сгенерированного кода после изменений пользователя

На странице shadow.html в блоке настроек расположены формы для настроек теней редактируемого объекта. Особенностью редактора теней, в отличие от, например, редактора текста является то, что здесь настройки привязаны к конкретной тени, которых может быть больше одной, соответственно должен работать функционал добавления и удаления теней. На рисунке 5.6 продемонстрировано состояние редактируемого объекта в начальном состоянии.

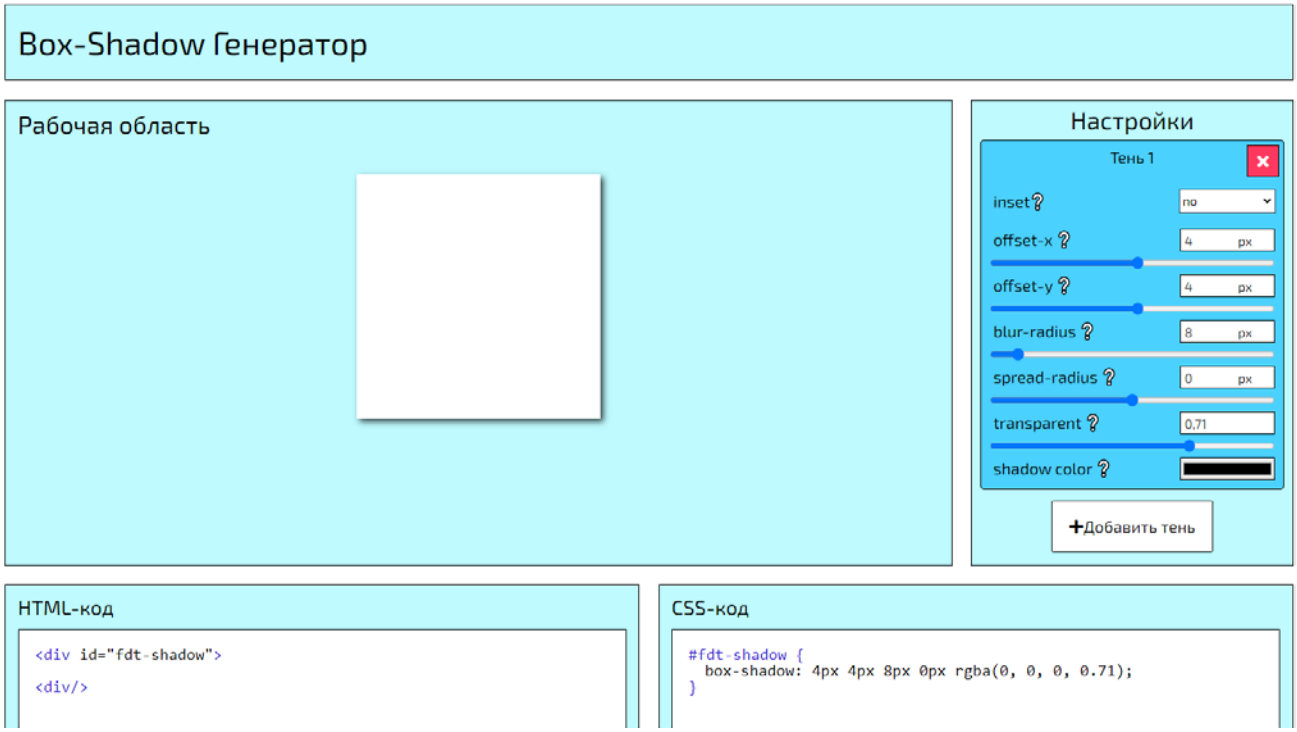


Рисунок 5.6 – Состояние веб-страницы shadow.html до изменений

На специальной кнопке для добавления теней находится обработчик события клика, который позволяет добавить новую тень и её блок настроек.

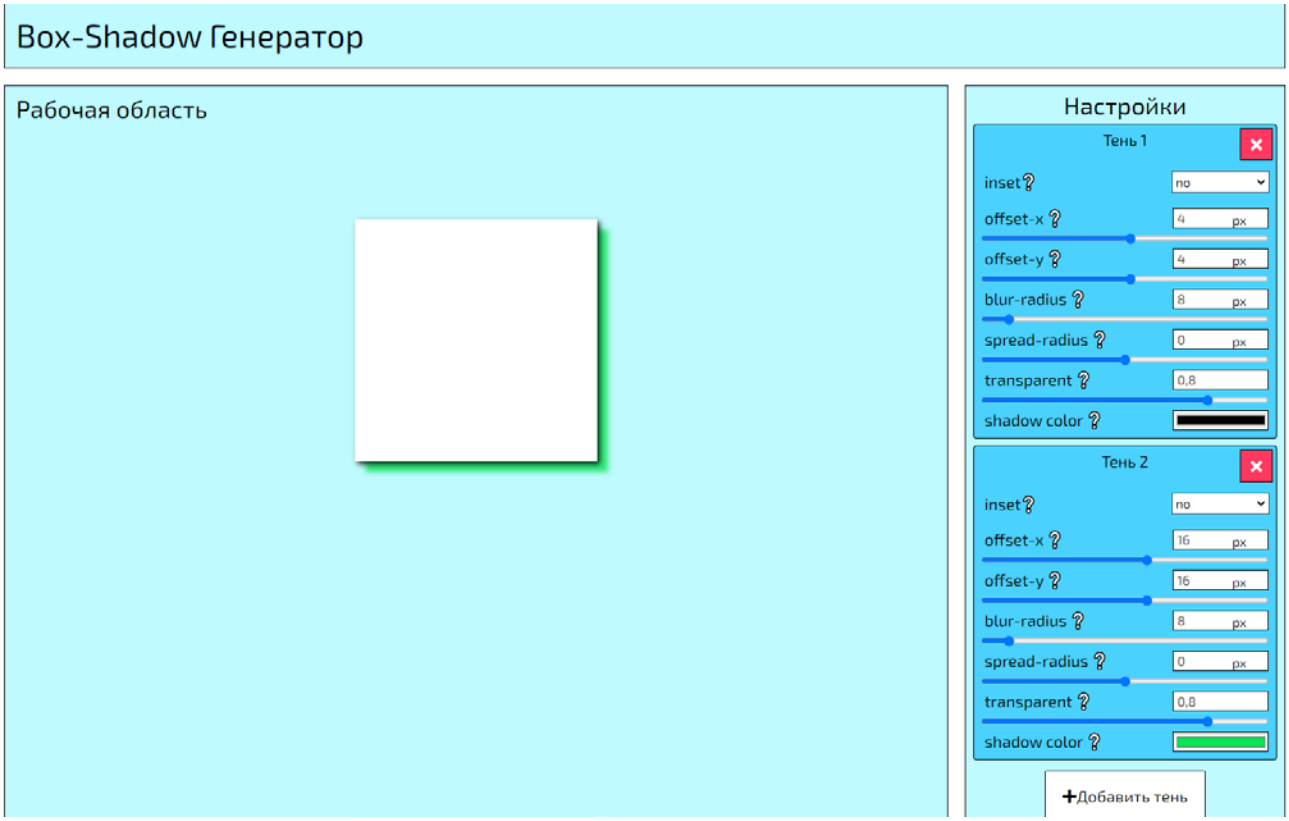


Рисунок 5.7 – Состояние редактируемого объекта после добавления новой тени

Каждый блок настроек теней содержит специальную кнопку удаления тени, после нажатия на неё, срабатывает обработчик, фиксирующий блок, на котором нажата кнопка, удаляет его, автоматически меняя: индексы блоков настроек, содержимое рабочей области и сгенерированный код. На рисунке 5.8 продемонстрирован результат удаления тени №1.

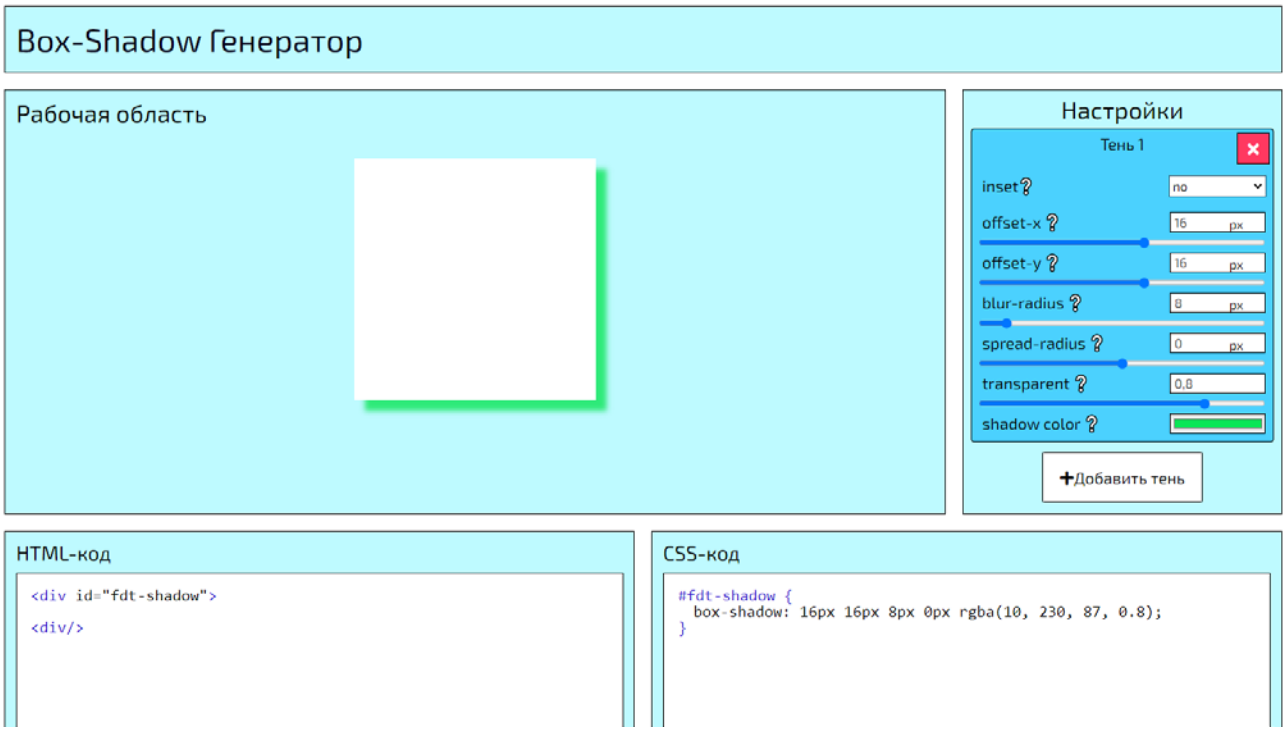


Рисунок 5.8 – Состояние редактируемого объекта после удаления тени

Далее на рисунках 5.9, 5.10, 5.11, 5.12 продемонстрированы остальные генераторы кода и их возможности.

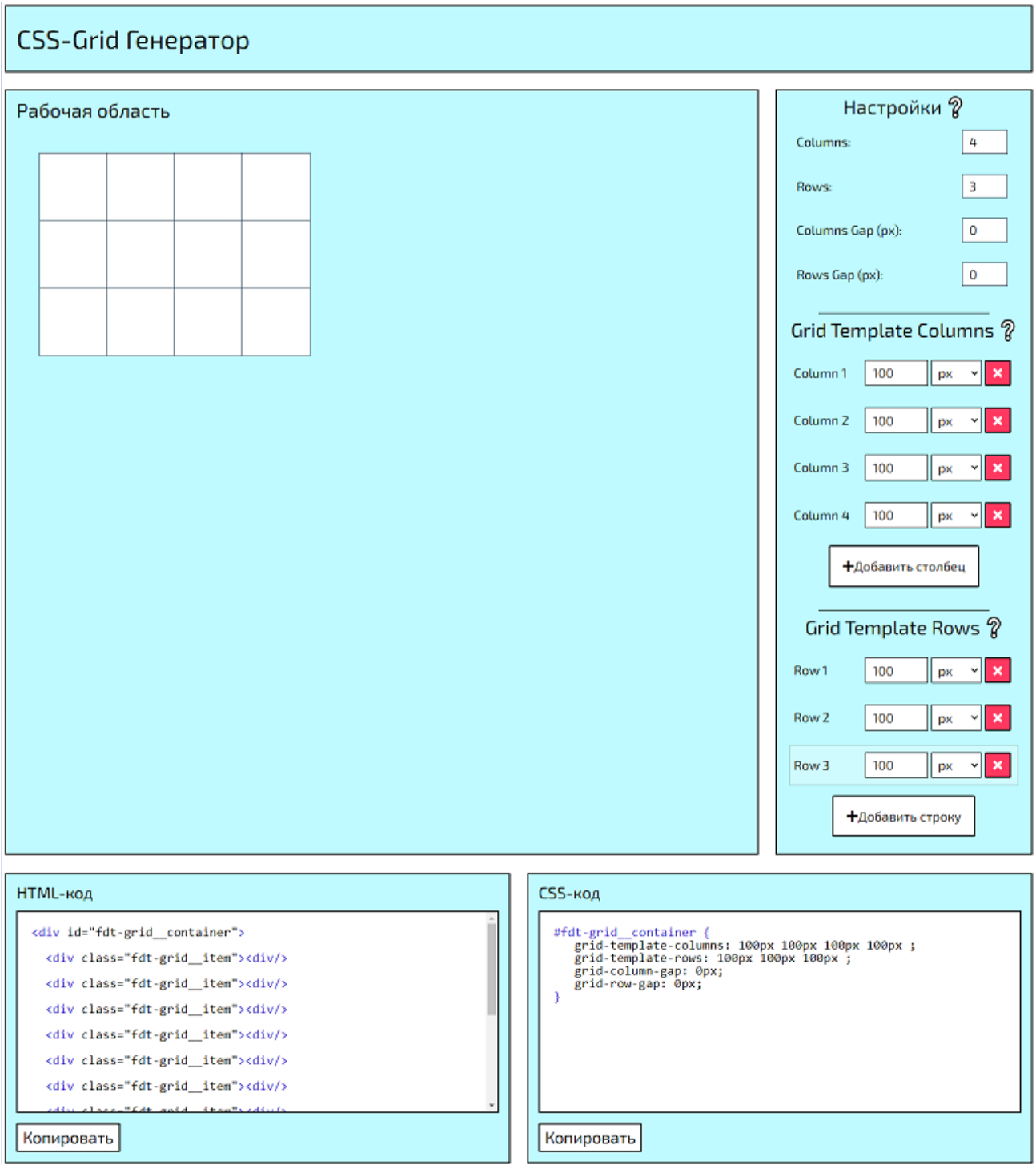


Рисунок 5.9 – Страница grid.html до изменений пользователя

Рисунок 5.10 – Страница grid.html после изменений пользователя

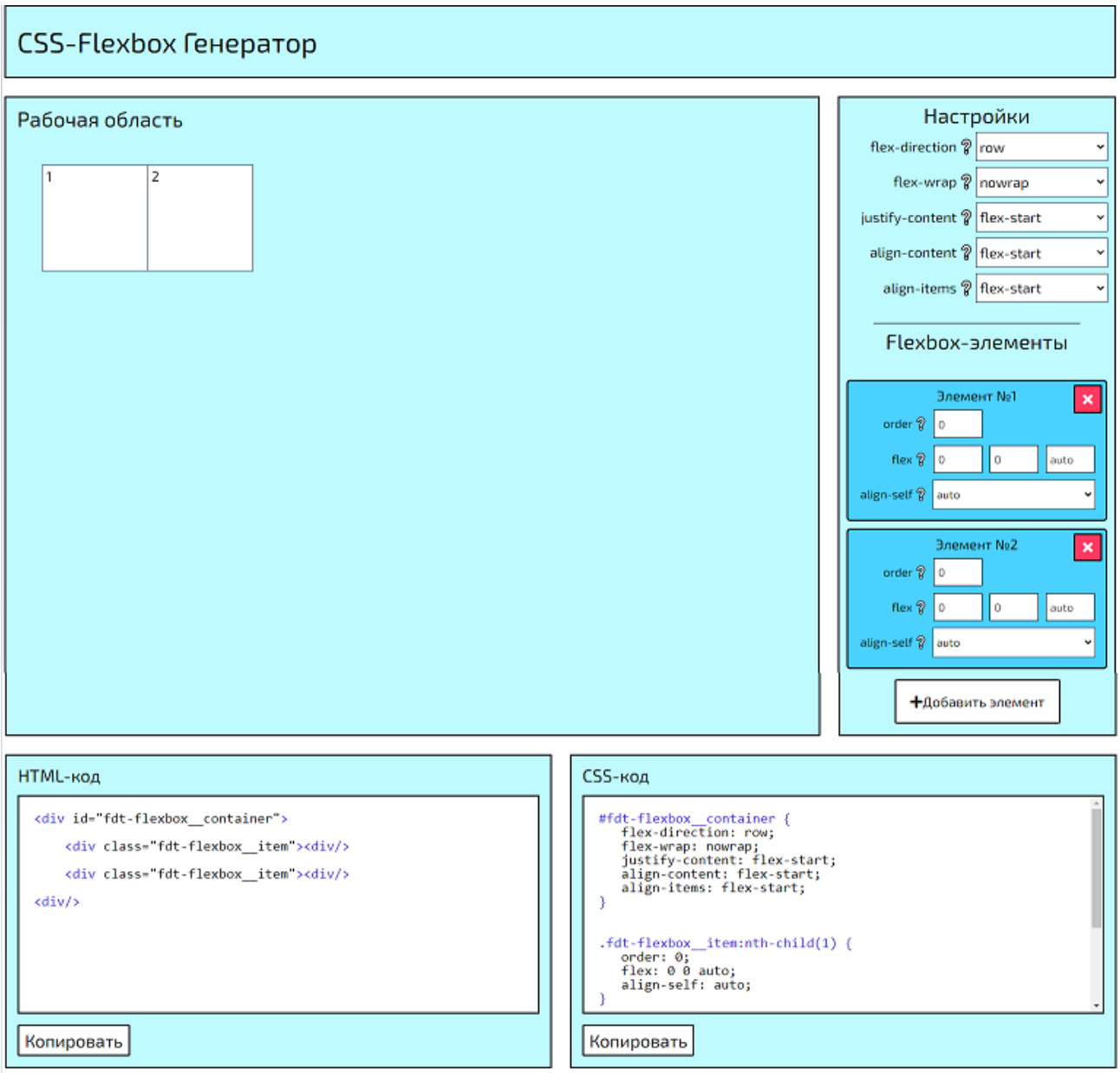


Рисунок 5.11 – Страница flex.html до изменений пользователя

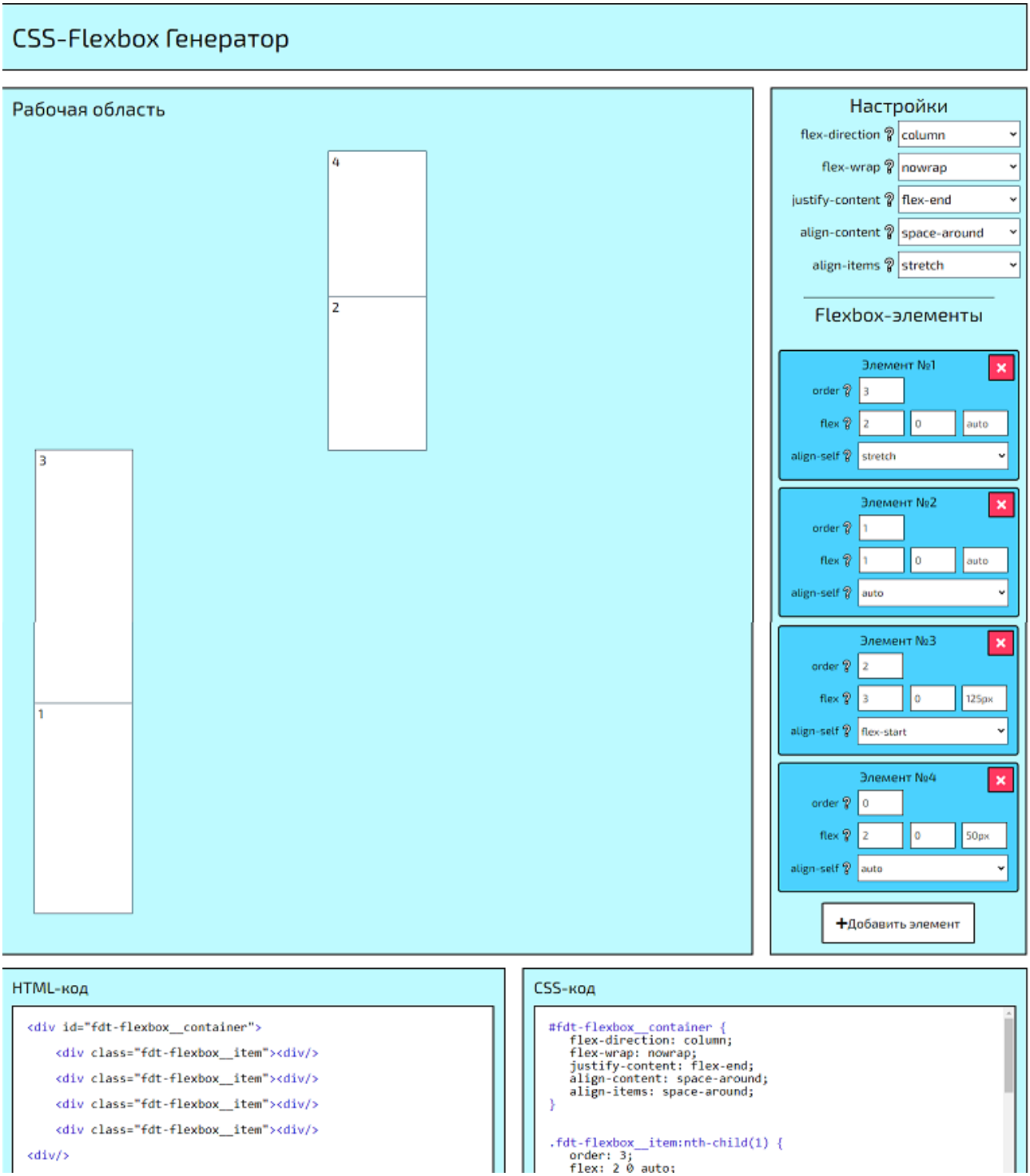


Рисунок 5.12 – Страница flex.html после изменений пользователя

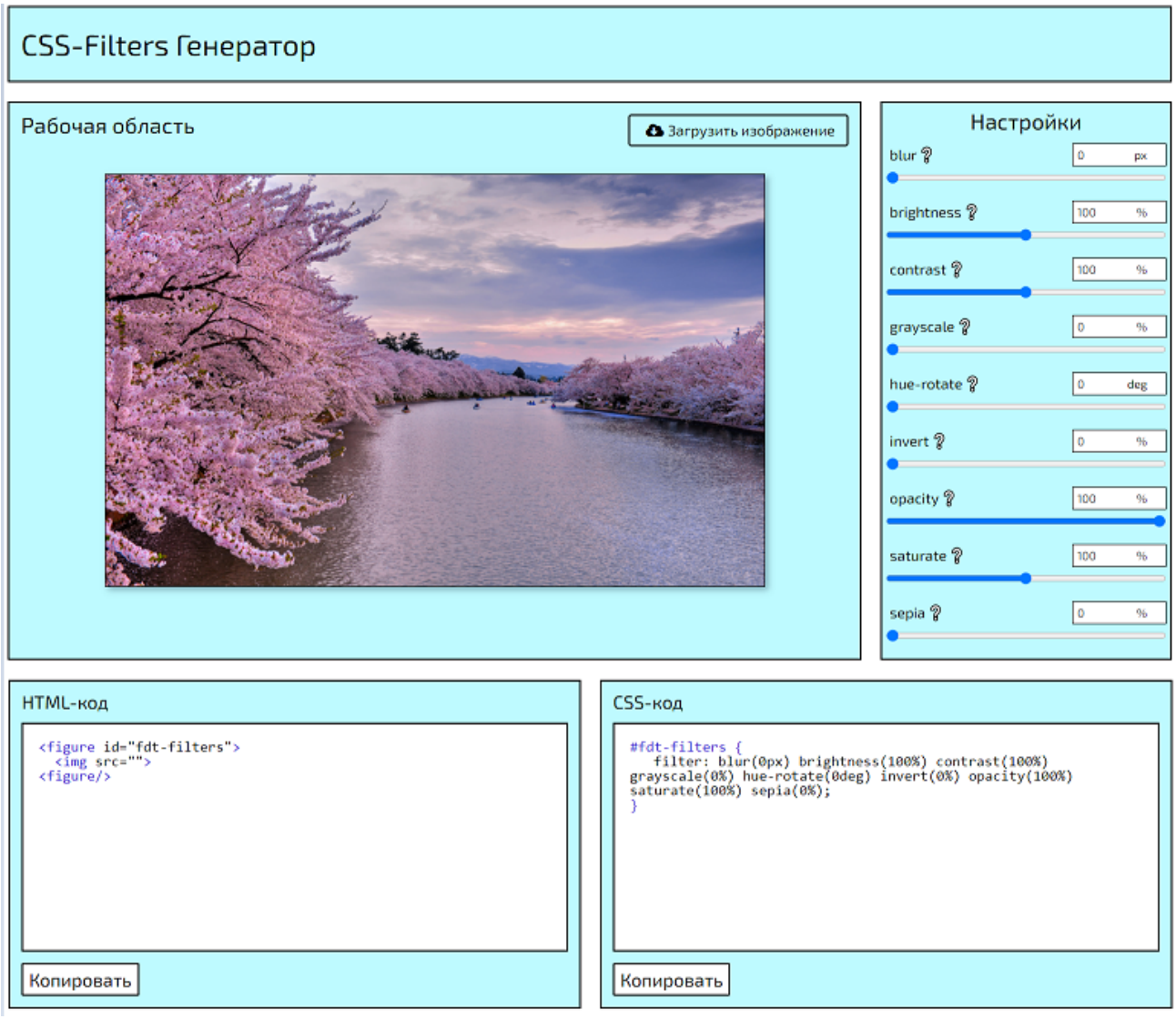


Рисунок 5.13 – Страница filter.html до изменений пользователя

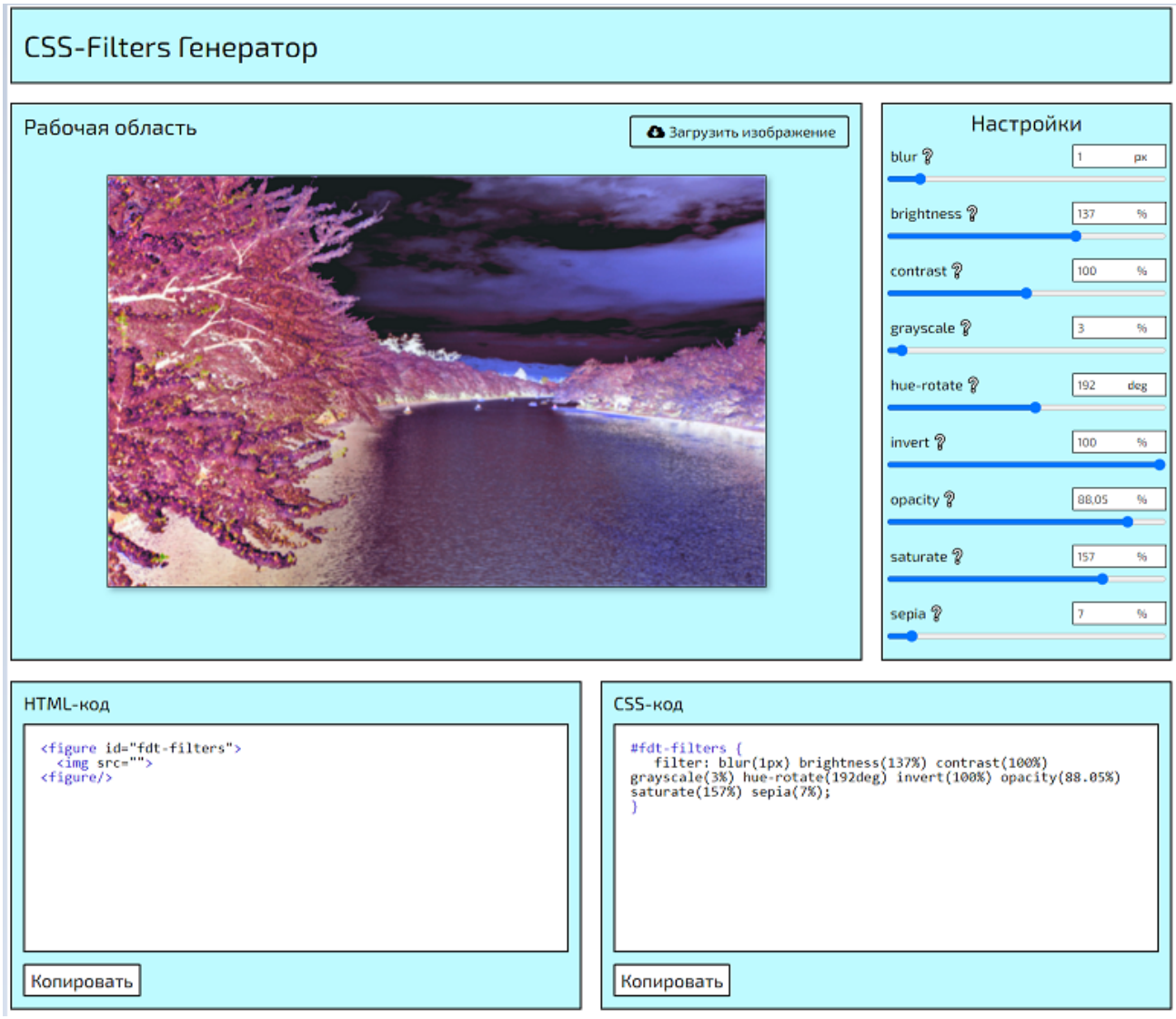


Рисунок 5.14 – Страница filter.html после изменений пользователя

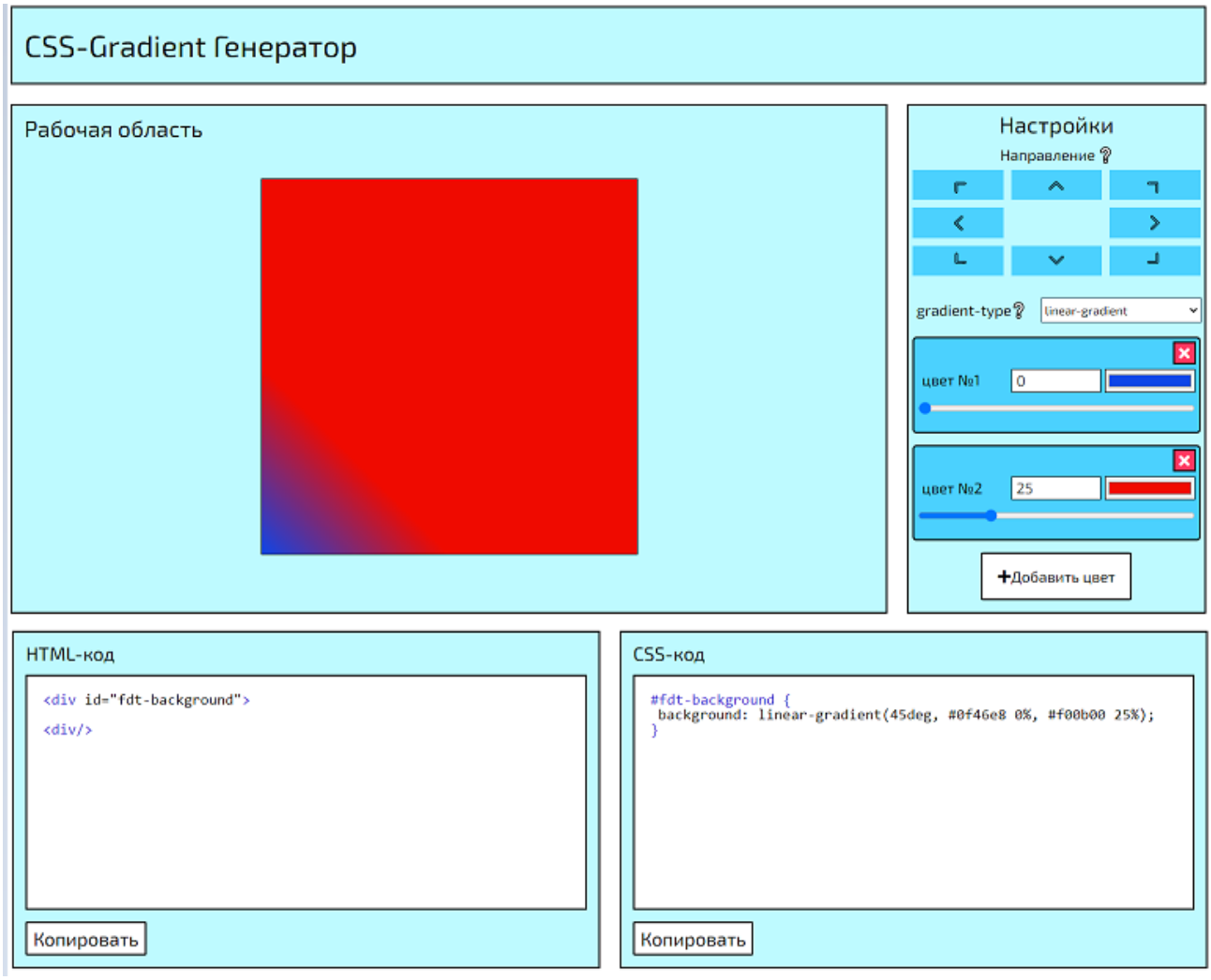


Рисунок 5.15 – Страница gradient.html до изменений пользователя

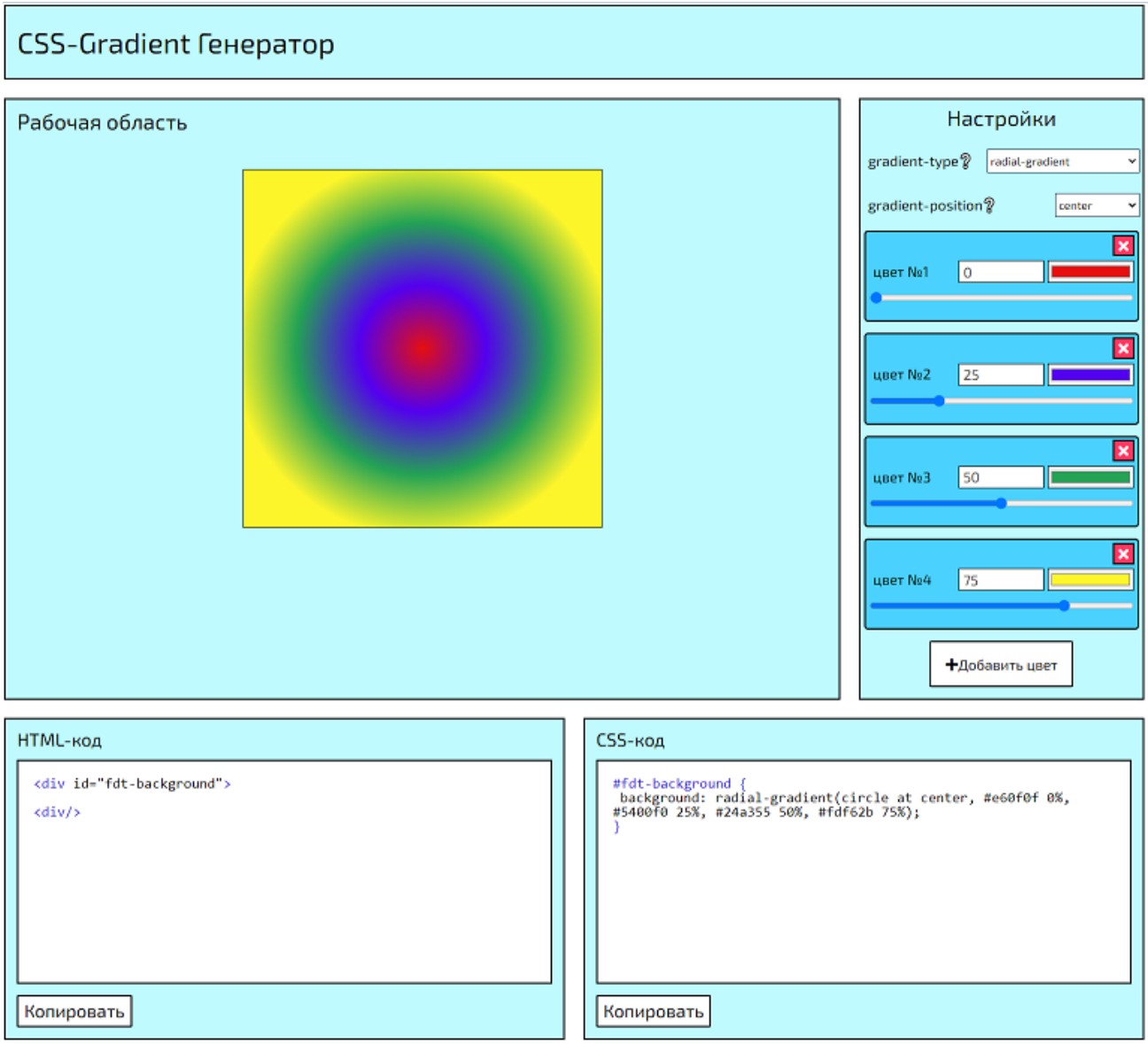


Рисунок 5.16 – Страница gradient.html после изменений пользователя

6. ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы были выполнены следующие пункты:

1. Анализ предметной области разрабатываемого продукта.
2. Анализ похожих систем на достоинства и недостатки.
3. Определение функциональных и нефункциональных требований разрабатываемого решения.
4. Проектирование архитектуры веб-сайта.
5. Создание дизайна веб-сайта.
6. Реализация и тестирование.

Разработанный веб-сайт содержит следующие генераторы веб-компонентов:

1. Генератор CSS Grid-сеток.
2. Генератор CSS Flex-блоков.
3. Генератор теней.
4. Генератор фильтров изображений.
5. Генератор градиентов.
6. Генератор текста.

Проект соответствует изначальным целям его создания, а именно: удобное создание и редактирование веб-компонентов, возможность импорта программного кода этих веб-компонентов с целью дальнейшего использования, наличие справочной информации по редактируемым компонентам и их свойствам, позволяющей лучше разобраться с инструментами начинающим разработчикам и верстальщикам.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. cssfilters // cssfilters.co URL: https://www.cssfilters.co/ (дата обращения: 12.04.2021).
2. The ultimate tools for web development // webcode.tools URL: https://webcode.tools/ (дата обращения: 12.04.2021).
3. CSS Snapshot 2020 // w3.org URL: https://www.w3.org/TR/CSS/#css (дата обращения: 13.04.2021).
4. Neumorphism.io Generate Soft-UI CSS code // neumorphism.io URL: https://neumorphism.io/ (дата обращения: 13.04.2021).
5. Getting Started // pugjs.org URL: https://pugjs.org/api/getting-started.html (дата обращения: 14.04.2021).
6. Sass Basics // sass-lang.com URL: https://sass-lang.com/guide (дата обращения: 17.04.2021).
7. CoffeeScript is a little language that compiles into JavaScript // coffeescript.org URL: https://coffeescript.org/ (дата обращения: 17.04.2021).
8. Typed JavaScript at Any Scale // typescriptlang.org URL: https://www.typescriptlang.org/ (дата обращения: 17.04.2021).
9. State of Frontend 2020 // tsh.io URL: https://tsh.io/state-of-frontend/#developers (дата обращения: 19.04.2021).
10. Web Frameworks // insights.stackoverflow.com URL: https://insights.stackoverflow.com/survey/2020#developer-profile-formal-education-importance-all-respondents3 (дата обращения: 19.04.2021).
11. React vs Angular vs Vue.js — What to choose in 2020? // medium.com URL: https://medium.com/techmagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d (дата обращения: 20.04.2021).

12. Form Input Bindings // v3.vuejs.org URL: https://v3.vuejs.org/guide/forms.html#basic-usage (дата обращения: 02.06.2021).
13. Class and Style Bindings // v3.vuejs.org URL: https://v3.vuejs.org/guide/class-and-style.html#binding-inline-styles (дата обращения: 02.06.2021).
14. Data Properties and Methods // v3.vuejs.org URL: https://v3.vuejs.org/guide/data-methods.html#data-properties (дата обращения: 02.06.2021).