Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ О ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

Кафедра «Информатика»

Место прохождения практики

Разработка веб-приложения для упрощения плетения бисером

тема

Руководитель от университета Д. А. Евдокимов

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Подпись, дата Инициалы, Фамилия

Студент КИ19-17/1Б, №031939174 А. К. Никитин

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Номер группы, зачетной книжки Подпись, дата Инициалы, Фамилия

Красноярск 2022

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc109068620)

[ВВЕДЕНИЕ 3](#_Toc109068621)

[1 Функциональные требования к ПО 4](#_Toc109068622)

[2 Описание процесса разработки ПО 5](#_Toc109068623)

[2.1 Алгоритм обработки изображения 5](#_Toc109068624)

[2.2 Интерфейс приложения 6](#_Toc109068625)

[2.3 Сервер приложения 6](#_Toc109068626)

[2.4 Взаимодействие клиент-сервер 7](#_Toc109068627)

[3 Пример работы ПО 9](#_Toc109068628)

[ЗАКЛЮЧЕНИЕ 14](#_Toc109068629)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 15](#_Toc109068630)

# ВВЕДЕНИЕ

Весьма очевидно, что разрабатываемый программный продукт в первую очередь нацелен на нужды пользователей, отчего можно констатировать, что, чем шире является круг потребителей, тем разнообразнее представляет из себя рынок программного обеспечения.

Из этих также рассуждений следует, что более востребованная сфера человеческой деятельности, требующая использование ПО, рождает более конкурентноспособную среду на рынке IT. Однако справедливо и обратное. Чем более нишевой является эта сфера, тем менее вероятно появление качественного программного продукта, связанного с ней.

В качестве примера подобной нишевой сферы можно привести такое хобби, как плетение из бисера. Естественно, уже существуют программные решения, облегчающие данный процесс рукоделия – BeadTool, MyJane, jbead и пр. Однако вышеперечисленные программы являются desktop-приложениями, что вынуждает пользователя тратить время на установку и конфигурацию. К тому же функционал использования изображения в качестве основы для схемы зачастую является платным, что и без того ограничивает сферу использования этой функции.

Итак, целью производственной практики является разработка веб-приложения, позволяющего создавать и редактировать схемы на основе загруженного изображения.

# Функциональные требования к ПО

Разрабатываемое веб-приложение должно уметь:

1. загружать изображения пользователя на сервер;
2. растрировать изображение на ячейки произвольного размера;
3. выводить схему плетения согласно результату обработки изображения на сайте;
4. поддерживать несколько видов плетения и вариаций расположения бисера;
5. произвольно изменять сгенерированную схему согласно желаниям пользователя;
6. поддерживать произвольное количество цветов;
7. сохранять готовую схему в файл.

# Описание процесса разработки ПО

## Алгоритм обработки изображения

Поставленная задача по своей сути является расширением задачи пикселизации, в которой пиксель представляет из себя не правильный четырехугольник, а прямоугольник. Процесс пикселизации обычно состоит из двух шагов:

1. разбиение изображения по сетке согласно размеру ячейки;
2. присвоение цвета ячейки.

Для вычисления цвета при этом обычно используется среднее значение внутри каждой из ячеек.

Разумеется, стандартный алгоритм пикселизации не применим в данном контексте: средний цвет по ячейке зачастую дает неудовлетворительный результат при обработке, и сетка может иметь форму, отличную от прямоугольной. Поэтому был разработан собственный алгоритм разбиения изображения на ячейки.

Алгоритм был написан на языке Python с использованием библиотек numpy и OpenCV.

Обработка изображения проходит в четыре этапа.

1. *Изменение размера изображения*. Этот шаг необходим для быстродействия алгоритма и унификации пользовательского вывода. Параметр, отвечающий за размер, задается пользователем.
2. *Изменение цветовой палитры*. Так как количество цветов строго ограничено и задается пользователем, изображение должно быть построено по необходимой нам цветовой схеме. Поэтому значение каждого пикселя должно быть перерассчитано согласно цветам палитры.
3. *Растрирование изображения*. Изображение должно быть разбито на сетку из ячеек согласно правилу генерации сетки, уникальной для каждого типа плетения.
4. *Замена цветов*. Данный этап производится с применением сразу двух алгоритмов. Первый алгоритм рассчитывает среднее значение цвета, второй алгоритм – наиболее часто встречающийся цвет. Таким образом, получается два набора ячеек, закрепленных за своим алгоритмом обработки.

Стоит отметить, что шаг 2 алгоритма обработки изображения вычисляет новый цвет для каждого пикселя изображения. Это приводит к проблемам со стороны производительности системы. Чтобы избежать долгой обработки изображения, следует переписать алгоритм на языке C++, обладающим гораздо более высокой производительностью по сравнению с языком Python.

## Интерфейс приложения

В связи с тем, что у разрабатываемого веб-сайта предполагалось непростое взаимодействие пользователя с UI, было решено использовать один из фреймворков языка JavaScript – Vue.js – для облегчения процесса разработки пользовательского интерфейса. Также было использовано браузерное расширение Vue Devtools для более простого процесса отладки приложения при разработке.

Frontend приложения был разработан в стиле Single-page Application (SPA), что исключило неудобные с точки зрения пользователя обновления страницы и переходы между ними.

Данные на странице реактивно связаны, что значит, что изменение данных в одном компоненте UI мгновенно отразится на состоянии другого компонента, ссылающегося на те же данные.

## Сервер приложения

Серверная часть веб-приложения была написана на языке Python с использованием фреймворка FastAPI. Использование FastAPI связано со встроенной асинхронностью, ускоряющей работу сервера при одновременной работе нескольких пользователей.

Сервер тестировался при помощи набора инструментов Postman.

Хостинг сервера произведен на платформе Heroku по URL <https://beads-divider.herokuapp.com/>.

## Взаимодействие клиент-сервер

В связи с тем, что интерфейс представляет из себя SPA, большая часть расчетов и переходов выполнялась на стороне клиента без обращения к серверу. Таким образом, имело место только три диалога между клиентом и сервером: получение главной страницы приложения, получение статических файлов и непосредственно обработка изображения.

На рисунке 1 представлена модель, иллюстрирующая процесс обмена сообщениями клиент-сервер при отправке формы с изображением.

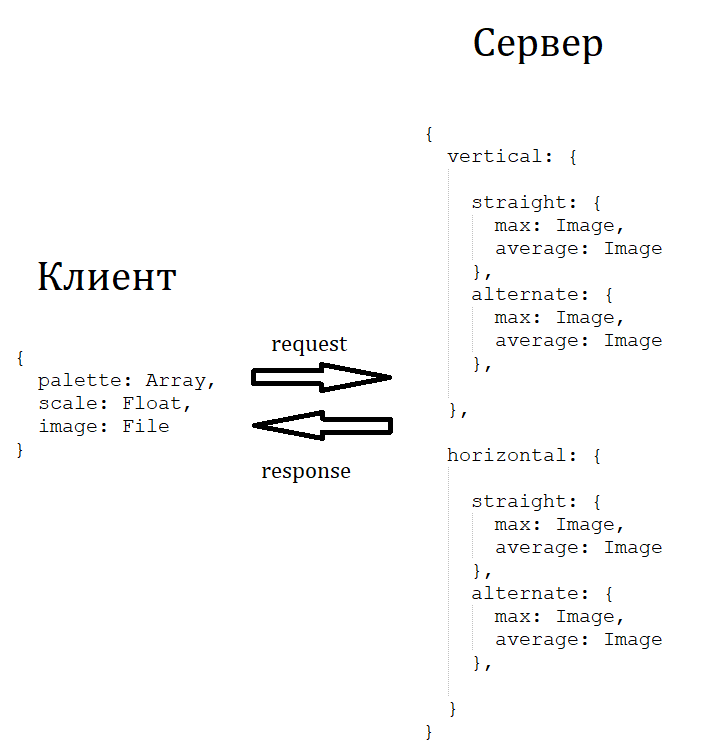


Рисунок 1 – Отправка формы с изображением

Как можно увидеть, на сервере рассчитывается 8 различных вариаций алгоритма и строится схема для каждой вариации. Это необходимый процесс для минимизации обращений клиента к серверу.

# Пример работы ПО

На рисунке 2 изображена приветственная страница сайта, отображаемая, если пользователь впервые заходит на сайт или очищает кэш сайта.

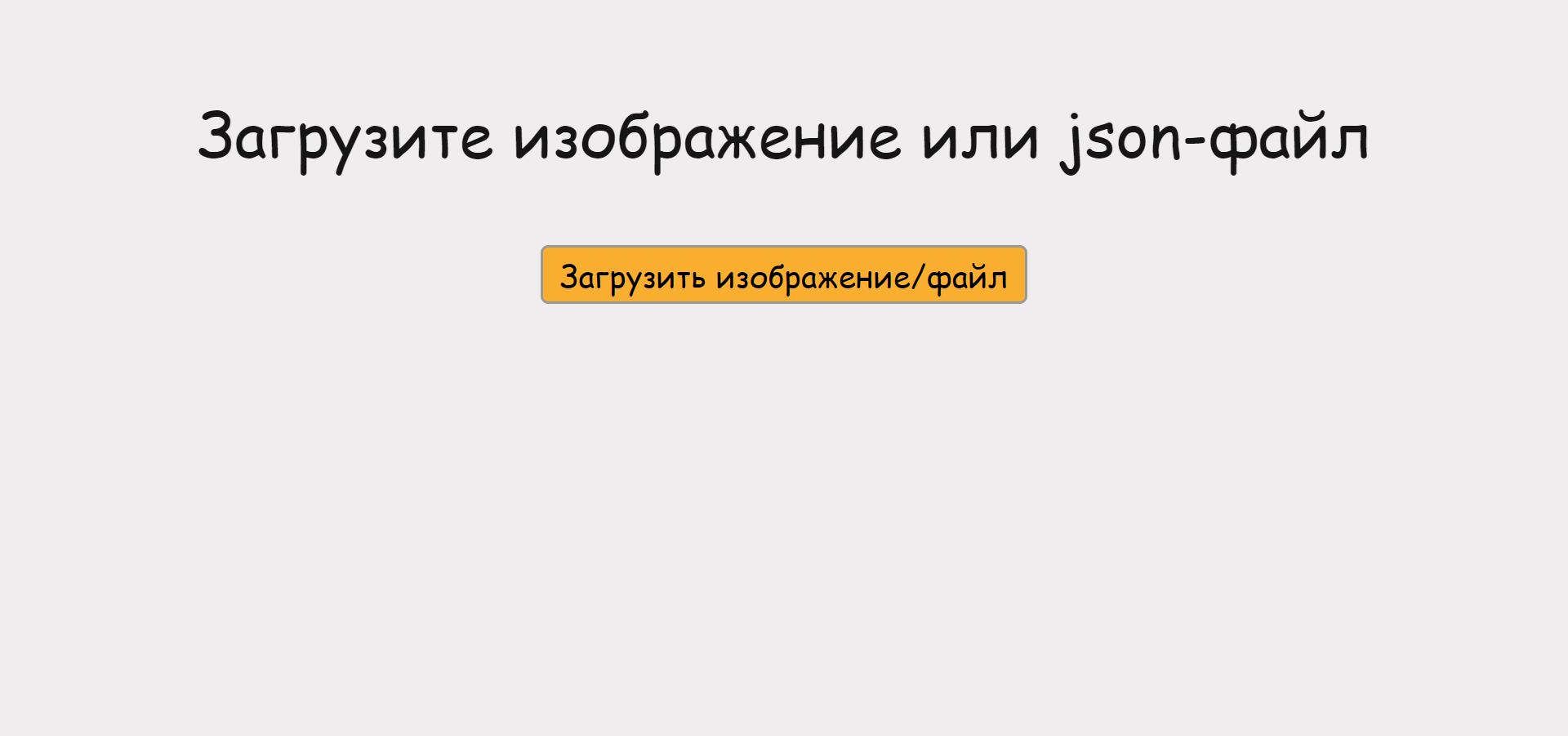


Рисунок 2 – Приветственная страница

На рисунке 3 изображена главная страница приложения.

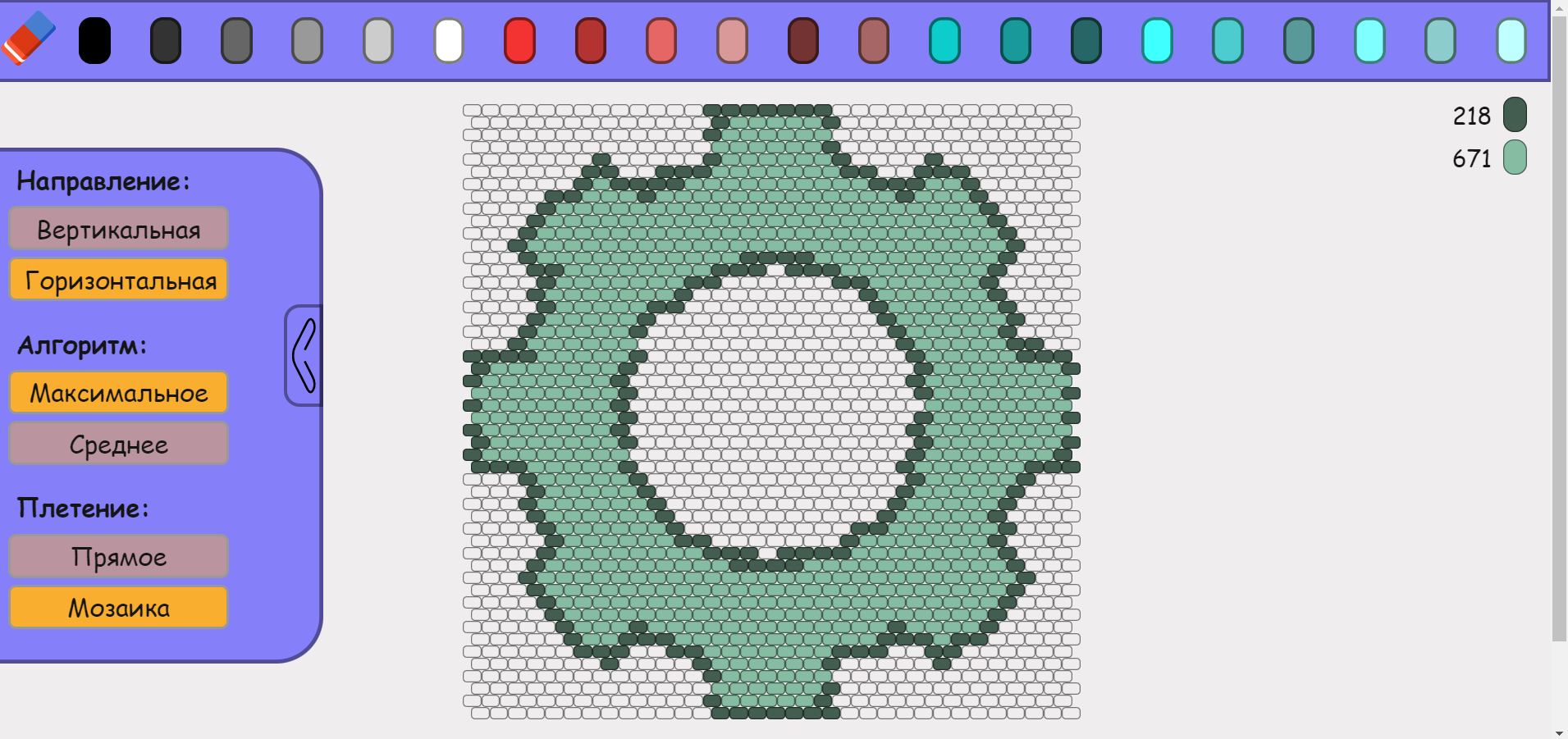


Рисунок 3 – Главная страница

Гиперпараметры алгоритма обработки изображения возможно динамически менять по необходимости, как показано на рисунке 4. «Направление» влияет на форму бусинки – (x, y) или (y, x). «Алгоритм» позволяет выбрать вариант генерации схемы с алгоритмом по максимальному и среднему цвету. «Плетение» изменяет расположение бусин относительно друг друга: при мозаичном плетении каждая вторая строка смещена вниз или вправо на расстояние, равное половине бусины.

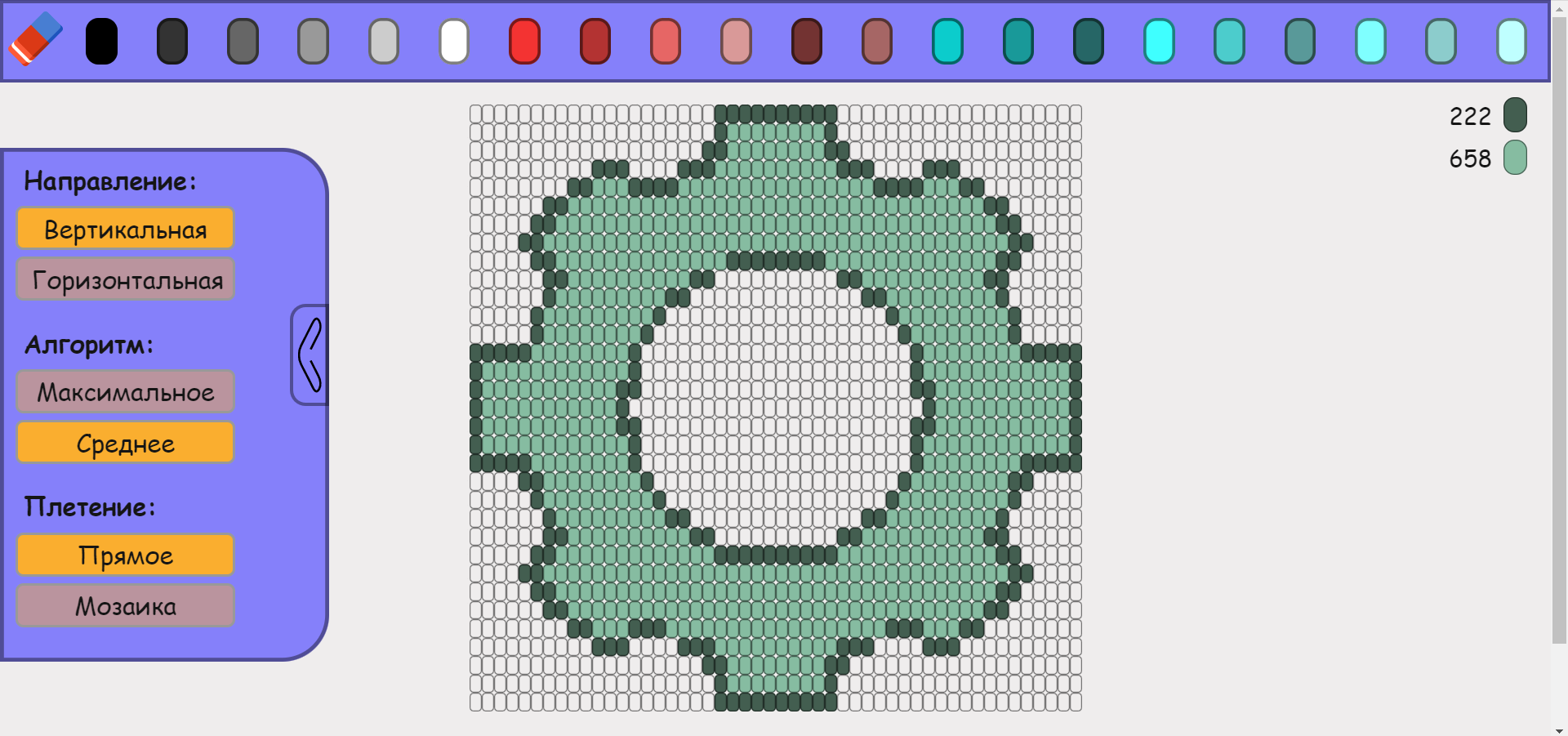


Рисунок 4 – Изменение параметров обработки

Также имеется возможность вручную задавать цвет бусинам, что продемонстрировано на рисунке 5. При этом выбранный цвет выделяется зеленой рамкой. Инструмент ластика позволяет избавиться от любого цвет бисера, сгенерированного при обработке или заданного пользователем.

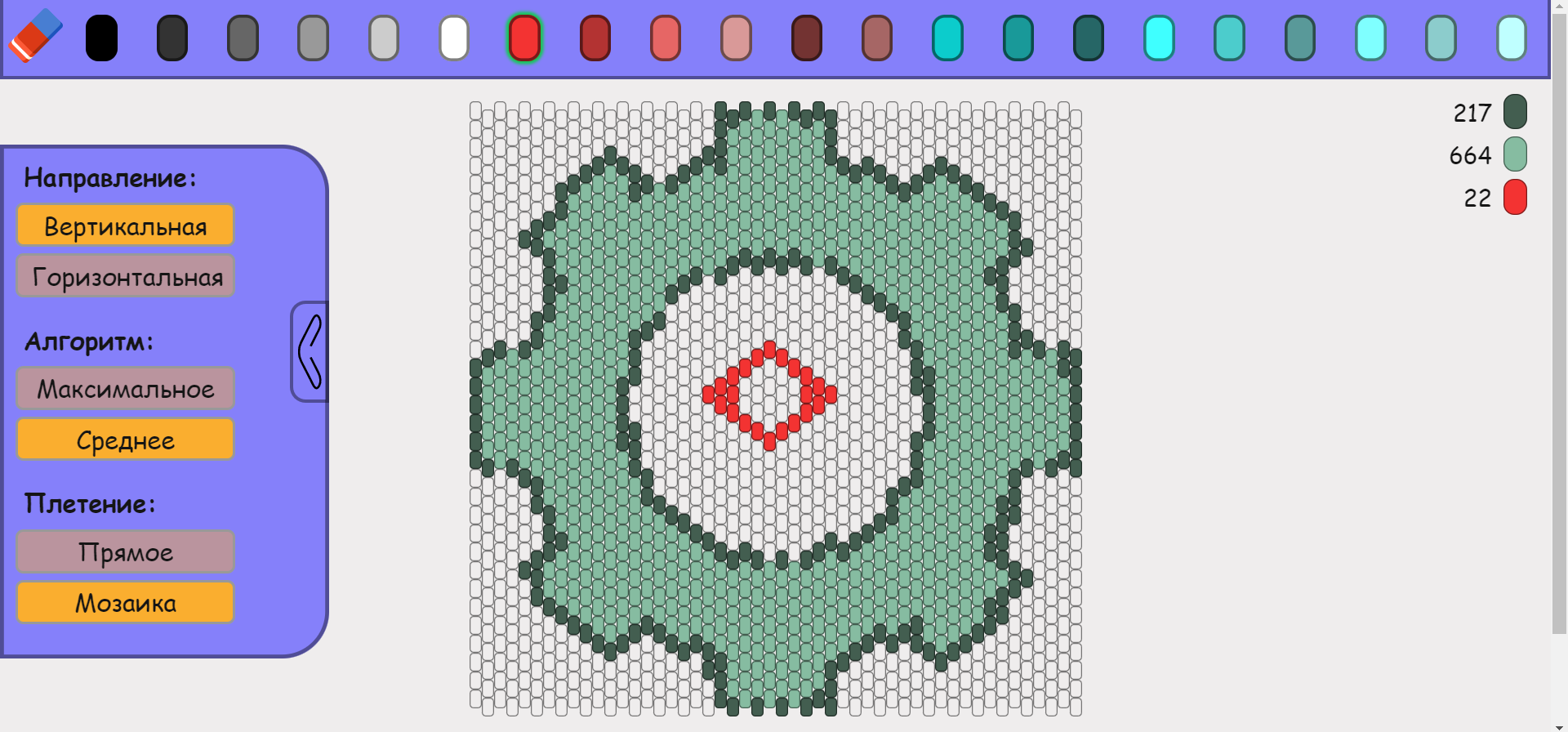


Рисунок 5 – Изменение цвета

Форма загрузки изображения, позволяющая определить гиперпараметры алгоритма и отправить запрос по обработке изображения на сервер, представлена на рисунке 6. При неверном вводе параметров или загрузке файла расширения не формата png, jpg, jpeg или tiff выводится окно предупреждения с просьбой изменить введенные параметры.

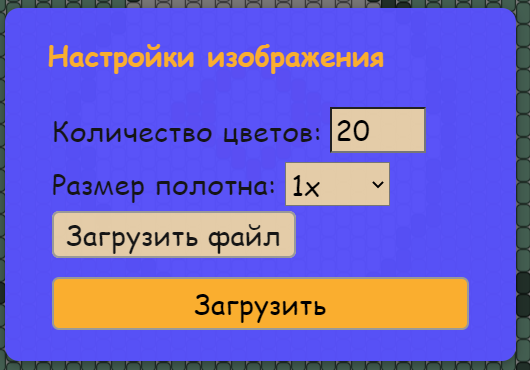


Рисунок 6 – Форма загрузки изображения

Количество цветов и размер полотна можно изменять, что влияет на результат обработки изображения (рисунок 7). Список цветов при этом динамически отображается в контейнере в верхней части сайте

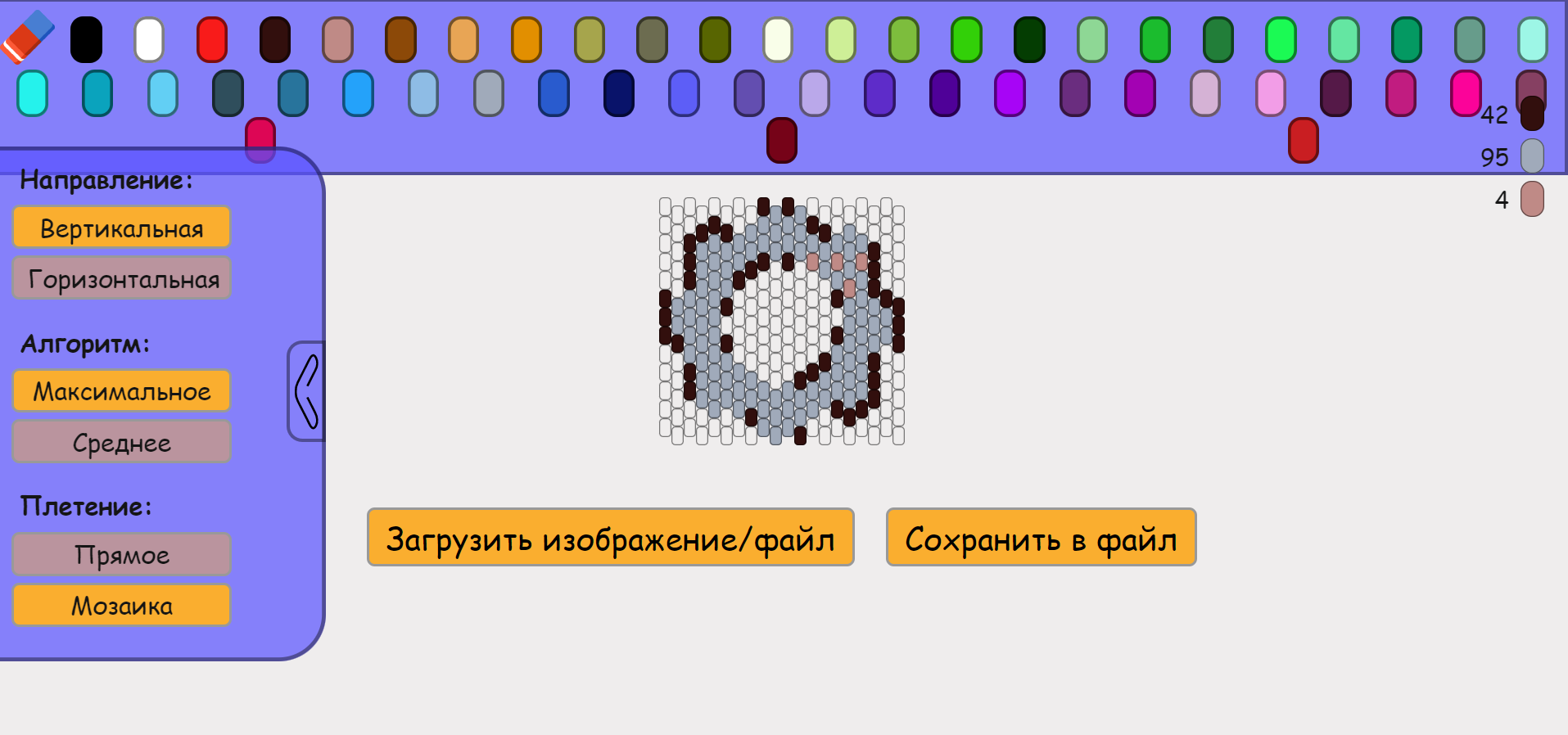


Рисунок 7 – Схема изображения с измененными параметрами

Панель с конфигурацией генерации схемы можно скрыть для удобства просмотра и редактирования схемы (рисунок 8).



Рисунок 8 – Скрытие конфигурационной панели

Процесс обработки изображения длительный и занимает определенное время, напрямую зависящее от заданных пользователем параметров. Для удобства пользователя во время обращения к серверу при получении схемы отображается компонент загрузки (рисунок 9).

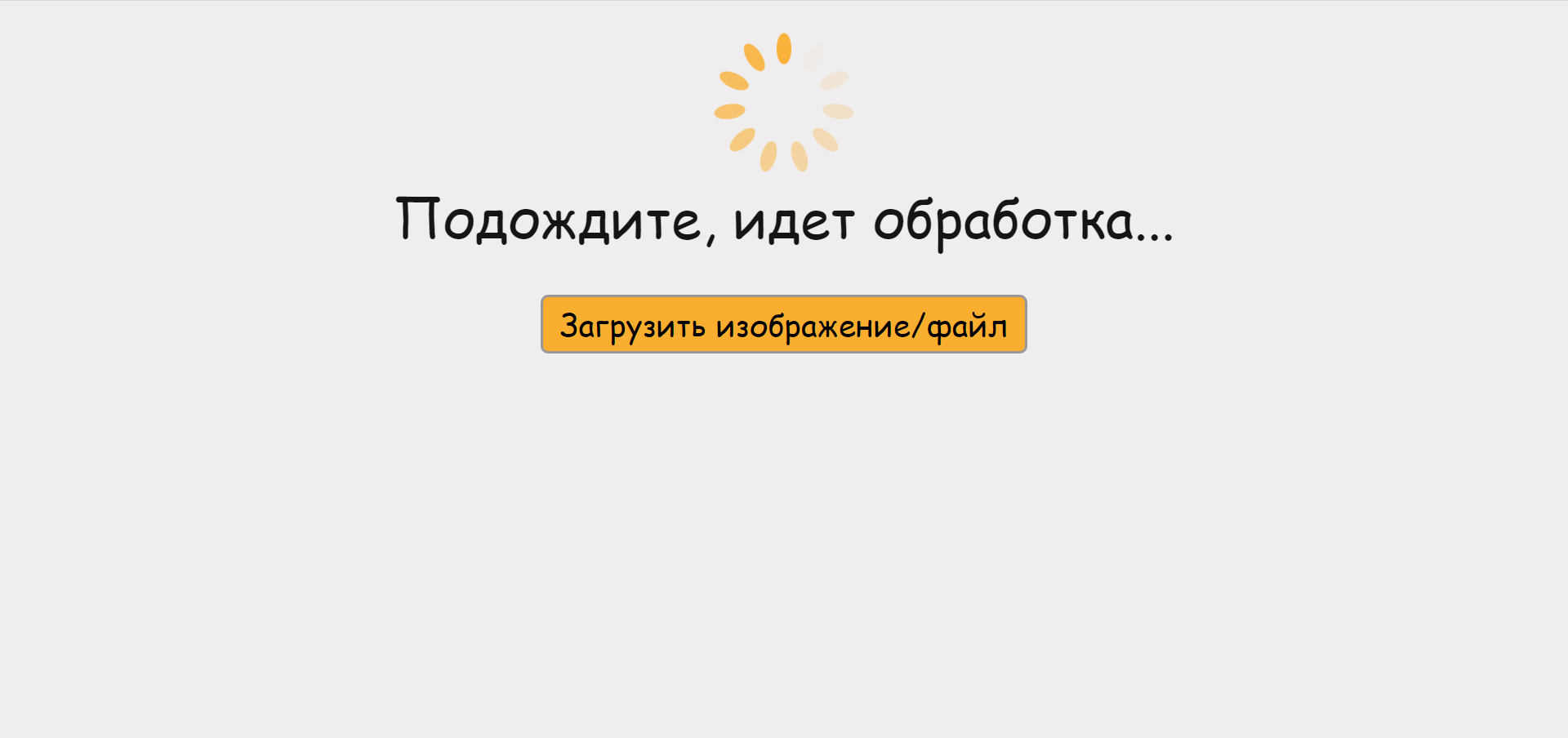


Рисунок 9 – Загрузка изображения

# ЗАКЛЮЧЕНИЕ

В процессе разработки программного продукта были приобретены ценные знания, касающиеся обработки изображений и frontend-разработки. В частности, был изучен Vue.js – фреймворк на язык JavaScript – и освоен принцип реактивности переменных. Были закреплены навыки написания асинхронного сервера с использованием фреймворка FastAPI и обработки изображений при помощи библиотеки алгоритмов OpenCV.

Результатом работы стало готовое веб-приложение для составления схемы плетения бисером на основе изображения, способное существенно упростить труд пользователя.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Хэнчетт Э., Листуон Б. Vue.js в действии. - СПб.: Питер, 2019. - 304 с. – ISBN 978-5-4461-1098-8
2. Vue.js 2.x Документация и руководство [Электронный ресурс] Режим доступа: <https://ru.vuejs.org/v2/guide/> (дата обращения: 28.06.2022).
3. Single-Page Application Development: The Definitive Overview [Электронный ресурс]: Software Development Services – Режим доступа: <https://www.softermii.com/blog/single-page-app-development> (дата обращения: 28.06.2022).