МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ

ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»



Направление подготовки/специальность   
09.03.01 Информатика и вычислительная техника

направленность (профиль)/специализация

«Технологии разработки программного обеспечения»

**Выпускная квалификационная работа**

Разработка клиентского интерфейса онлайн-микросервиса по редактированию изображений на основе импортозамещенных веб-технологий

| Обучающегося 4 курса  очной формы обучения  Царулковой Анастасии Витальевны |
| --- |
|  |
| Руководитель выпускной квалификационной работы:  кандидат педагогических наук, доцент Государев Илья Борисович |
|  |
|  |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_heading=h.gjdgxs)

[ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И СУЩЕСТВУЮЩИХ РЕШЕНИЙ. ПОДГОТОВКА К РАЗРАБОТКЕ 6](#_heading=h.30j0zll)

[1.1. Анализ клиентских веб-интерфейсов и современных трендов 6](#_heading=h.1fob9te)

[1.2. Анализ существующих зарубежных аналогов 13](#_heading=h.3znysh7)

[1.3. Анализ программных средств, использованных при разработке 16](#_heading=h.2et92p0)

[ГЛАВА 2. РАЗРАБОТКА КЛИЕНТСКОГО ИНТЕРФЕЙСА ОНЛАЙН ГРАФИЧЕСКОГО РЕДАКТОРА 23](#_heading=h.tyjcwt)

[2.1 Создание макета интерфейса 23](#_heading=h.3dy6vkm)

[2.2. Кодирование клиентского интерфейса 27](#_heading=h.1t3h5sf)

[ЗАКЛЮЧЕНИЕ 39](#_heading=h.4d34og8)

[СПИСОК ЛИТЕРАТУРЫ 41](#_heading=h.2s8eyo1)

# ВВЕДЕНИЕ

Пользовательские интерфейсы являются неотъемлемой частью жизни современного человека. Они рассматриваются как инструменты управления устройствами, приложениями, программами. Профессионалы и обычные пользователи ежедневно взаимодействуют с ними, общаясь в социальных сетях, заказывая товары через интернет, смотря фильмы или сериалы онлайн, а также занимаясь своей профессиональной деятельностью, например, разработкой программного обеспечения, ретушью фотографий и другим. Учитывая факты, можно сделать вывод, что при разработке пользовательских интерфейсов необходим серьезный и глубокий исследовательский подход. Успех приложения или программы во многом зависит от качества проработки пользовательского интерфейса. В профессиональных редакторах, таких, как например Adobe Photoshop, большое количество функций и инструментов, они распределены и сгруппированы по всему экрану, и обычному человеку достаточно сложно разобраться, какой инструмент за что отвечает и как реализовать свою идею с помощью многообразия всего предоставляемого функционала. С данной точки зрения, чтобы создать продукт, который сможет использовать любой желающий без глубоких знаний о конкретном программном обеспечении и получать хороший результат, требуется найти баланс в количестве функций и инструментов и расположить их наиболее удобным для пользователя образом. Также стоит отметить, что на данный момент существует не так много доступных для быстрого освоения онлайн графических редакторов российского производства, которые функционируют на российских серверах.

Таким образом, **актуальность** дипломной работы связана с небольшим выбором среди веб-сервисов для обработки изображений с простым интерфейсом и находящихся на серверах Российской Федерации.

**Цель** заключается в разработке клиентского интерфейса онлайн-микросервиса по редактированию изображений на основе импортозамещающих веб-технологий.

**Объект исследования** – пользовательские интерфейсы.

**Предмет исследования** – клиентский интерфейс онлайн-микросервиса по редактированию изображений.

В соответствии с поставленной целью необходимо решить следующие задачи:

* Провести анализ трендов современных клиентских веб-интерфейсов;
* Провести анализ существующих зарубежных аналогов;
* Определить необходимый функционал клиентской части онлайн-микросервиса по редактированию изображений;
* Создать макет интерфейса;
* Разработать клиентскую часть сервиса в соответствии с созданным макетом.

Практическая значимость заключается в получении удобного интерфейса для онлайн-микросервиса по редактированию изображений, который позволит быстро и легко ориентироваться в функционале микросервиса.

В первой главе представлен анализ предметной области, исследование последних тенденций в разработке интерфейсов и выбор основных функций сервиса.

Во второй главе представлены прикладные аспекты разработки интерфейса – техническое задание, макет, прототип, используемые библиотеки и собственно разработка клиентской части.

Выпускная квалификационная работа имеет следующую структуру: введение, две главы, заключение, список литературы, включающий 21 источник.

Общий объем работы включает 44 страницы. В работе представлено 24 рисунка.

# ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И СУЩЕСТВУЮЩИХ РЕШЕНИЙ. ПОДГОТОВКА К РАЗРАБОТКЕ

## 1.1. Анализ клиентских веб-интерфейсов и современных трендов

Качество пользовательских интерфейсов совершенствуется с каждым годом. При разработке очень важно не отставать от современных идей и следить за ситуацией на рынке. Создание «сильного» интерфейса для любой компании требует много времени и средств, но в будущем он станет одним из ключевых критериев выбора именно этого продукта пользователем, а не другого. Но часто разработчики забывают и «сосредотачиваются на технологии и порождают решения, слабоуправляемые и неудобные в применении» [1].

Если пользователю будет не удобно взаимодействовать с интерфейсом какого-то продукта, его будет отталкивать внешний вид, то он быстро закроет и удалит этот продукт, а цель эффективного интерфейса, удержать пользователя в сервисе как можно дольше. Кроме того, неправильно спроектированный интерфейс может стать источником дискомфорта, стресса человека. С целью понять, как этого лучше всего этого избежать, в данном параграфе будут проанализированы существующие клиентские веб-интерфейсы и тенденции последних лет.

Рассмотрим основные использующиеся понятия.

Клиент – это часть программного обеспечения, которая работает на компьютере пользователя и взаимодействует с сервером.

В клиентскую сторону приложения входит интерфейс и часть функций. Данные функции, обычно, могут быть расположены на клиенте по нескольким причинам. Они, в соответствии с поставленными задачами, должны обрабатываться быстро, или же для их работы нет необходимости затрачивать ресурсы сервера.

Клиентский (пользовательский) интерфейс – элементы и компоненты программы, способные оказывать влияние на взаимодействие пользователя с программным обеспечением. К ним относятся:

* Средства отображения информации, отображаемая информация;
* Язык «пользователь-интерфейс»;
* Устройства и технологии ввода данных;
* Диалоги взаимодействия и транзакции между пользователем и компьютером;
* Обратная связь с пользователем и др. [2]

Фронтенд – это непосредственно сам клиентский код. В нем происходит синтез трёх технологий:

* HTML (HyperTextMarkupLanguage) – код, который позволяет структурировать и отобразить веб-страницу и её контент. Также это язык гипертекстовой разметки, взаимодействующий с браузером на стороне клиента [3].
* CSS (Cascading Style Sheets) – каскадные таблицы стилей – код, благодаря которому можно придать странице форму, оригинальный вид. Один из инструментов для создания комфортной среды пользователя.
* JavaScript – язык программирования, который позволяет взаимодействовать с данными в HTML и CSS, обеспечить логику взаимодействия. С его помощью можно добавить на страницу или приложение интерактивные элементы.

Познакомимся с принципами разработки интерфейса – концепциями высокого уровня, которые могут использоваться при проектировании программного обеспечения. После чего определим, какой из них наиболее важен в данной разработке. Кратко они звучат так:

1. Контроль пользователем интерфейса;
2. Уменьшение загрузки памяти пользователя;
3. Последовательность пользовательского интерфейса [4].

В соответствии с первым принципом, интерфейс должен дать пользователю возможность контроля, удовлетворять потребности и желания пользователя. Следовать данному принципу помогают определенные правила. Например, возможность пользователя свободно ориентироваться в интерфейсе: при движении вглубь интерфейса должна быть возможность легко вернуться обратно. Если пути движения для пользователя неочевидны, то нужно вводить контекстные подсказки. Он должен не бояться нажимать на интерактивные элементы, переходить на другие экраны. Также нужно уметь сфокусировать внимание пользователя и не заставлять его заканчивать цепочку действий раньше времени и теряться. Важно дать человеку возможность сохранить произведенные действия.

По второму принципу не следует заставлять пользователя держать в голове большое количество информации, а переложить данную задачу на клиентский интерфейс. Для этого система должна запоминать и хранить введенную в неё информацию (например форма адреса или пароля). Лучше всего при возможности предоставлять пользователю возможность выбора из списка заранее заготовленных данных, вместо ввода их вручную. Визуально интерфейс должен выглядеть как можно более структурировано: данные и объекты сгруппированы, наиболее важная информация выделена особым образом и общее количество демонстрируемой информации не перегружает человека.

Интерфейсы, спроектированные по третьему принципу, должны предоставить пользователю возможность применить свой прошлый опыт взаимодействия с интерфейсами. Возможно обучение пользователя идеям программного обеспечения на новых ситуациях. Следует предупреждать человека, если результат его действий будет отличаться, от того, что он может ожидать. Важно в таком случае добавить отмену совершенных действий и их воспроизведения, чтобы избавить человека от страха ошибки. Подобный дружественный интерфейс создается с применением различных способов поощрения его изучения.

В данной работе были применены первое и третье правила. Пользователь сможет легко перемещаться по интерфейсу и функционалу, выбирать инструменты, а сам интерфейс будет разработан таким образом, чтобы человек смог применять в нем свои навыки, полученные в прошлом. Инструменты и функции будут всем знакомы, а их поведение ожидаемо и интуитивно.

Теперь перейдем к элементам дизайна интерфейса. Рассмотрим имеющуюся информацию за последние несколько лет.

В 2021 году был актуален глассоморфизм (стекломорфизм) — стиль, в котором имитируется стеклянная поверхность. Пример использования этого стиля на рисунке 1. К его характерным особенностям относятся:

* Эффект матового стекла с использованием размытия фона;
* Многослойность, объекты, плавающие в пространстве;
* Яркие цвета;
* Тонкая светлая граница на стеклянных объектах.

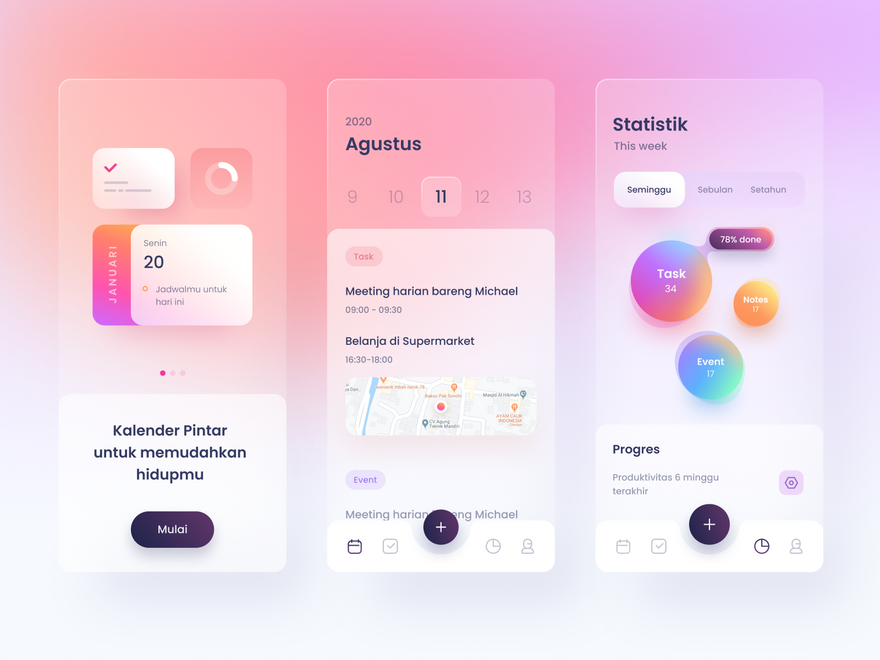


Рисунок 1 — Пример интерфейса в стиле стекломорфизм, автор Ghani Pradita

Этот стиль достаточно сложно правильно использовать в дизайне пользовательских интерфейсов, так как необходимо найти такое сочетание его элементов, чтобы в будущем он не затруднял чтение информации для пользователя. Также он не подойдет для графического редактора, потому что будет сильно мешать пользователю, работающему с изображением, которое может иметь множество цветов. Таким образом общая картинка на экране станет рябящей, что безусловно вызовет дискомфорт у человека.

Следующий актуальный стиль — минимализм, главная идея которого ясна из названия — сведение к минимуму элементов, деталей, цветов. Такой стиль позволяет пользователю легко и быстро ориентироваться в интерфейсе, находить нужные функции, не отвлекаться от поставленных целей. Пример использования минимализма показан на рисунке 2. Характерные черты данного стиля:

1. Крупные, читаемые заголовки
2. Продуманное использование цветов
3. Сосредоточение внимания на контрасте
4. Ограниченное использование эффектов [5]

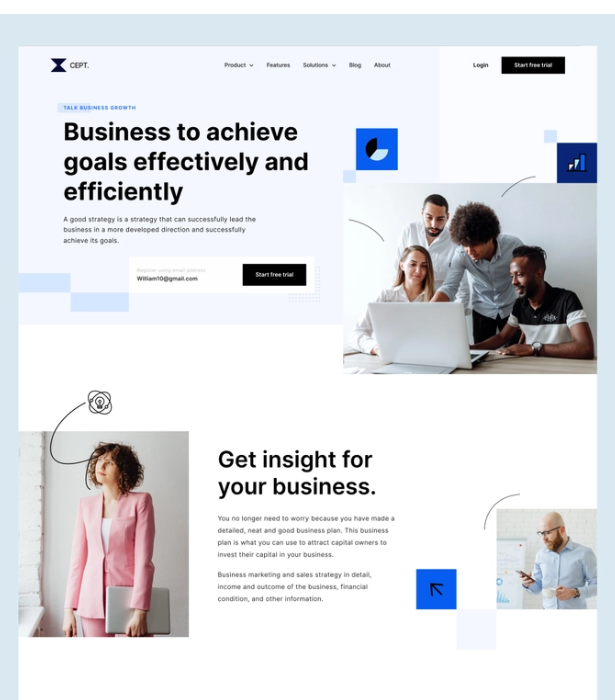


Рисунок 2 — Пример интерфейса в стиле минимализм[[1]](#footnote-0)

Брутализм молодой тренд, отличительными чертами которого являются:

* четкие контуры,
* черный или белый фон,
* мало изображений, они не обработаны,
* ошибки в дизайне,
* один шрифт для всего,
* отсутствие анимации,
* минимум оформления.

Цель брутализма произвести эмоциональный эффект, вывести функциональную часть на первый план, осуществить провокацию, чтобы как можно больше людей узнало о продукте. Данный стиль используют для дизайна всего продукта крайне редко, как эксперимент или высказывание. В реальных проектах могут использоваться отдельные его элементы для придания чувства новизны. На рисунке 3 пример интерфейса в стиле брутализм.

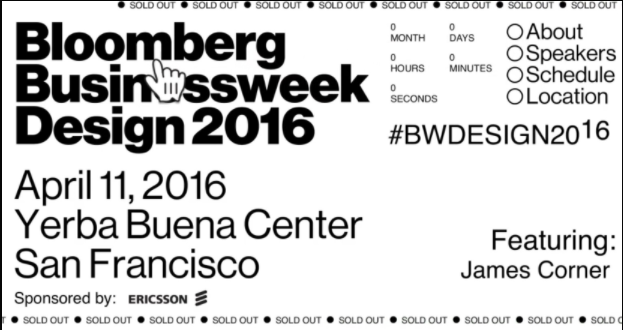
[[2]](#footnote-1)

Рисунок 3 — Пример интерфейса в стиле брутализм

Из всех описанных выше стилей наиболее подходящим для данного проекта будет минималистичный, так как в графическом редакторе на первом месте стоит функционал. Ничто не должно отвлекать пользователя от работы. Этому способствует удобство навигации, однородность иконок. Все нужные инструменты должны быть под рукой и их расположение должно быть легко считываемо. В основной рабочей области центральным элементом безусловно будет изображение, загруженное пользователем, поэтому там будет спокойный однотонный фон.

После выбора стиля оформления интерфейса, необходимо определить функционал, который будет работать на стороне клиента. К нему относятся базовые функции и инструменты графического редактора, такие как:

* отрисовка (рендер) загруженного изображения,
* сохранение изображения
* кисть,
* палитра,
* ластик,
* геометрические фигуры: круга, прямоугольника, треугольника,
* отправка запросов на сервер для осуществления загрузки изображения, поворота.

Инструменты, связанные с поворотом изображения, его обрезкой и наложением фильтров будут обрабатываться на стороне сервера. На стороне клиента будут кнопки для отправки запроса на сервер. Также посредством интерфейса будет реализована возможность загрузки изображения на сервер и сохранение его на локальный диск.

## 1.2. Анализ существующих зарубежных аналогов

На зарубежном рынке существует множество продуктов относящихся к онлайн-редакторам изображений. Проведем анализ некоторых из них, чтобы выяснить, какие слабые и сильные стороны в них есть, какие интересные идеи были использованы, и что ещё можно привнести нового в данную работу.

Первым для анализа выбран онлайн-фоторедактор Pixlr E. На официальном сайте сообщается, что это сервис №1 для редактирования фото. Данный сервис имеет бесплатную версию. В ней рядом с интерфейсом редактора на странице находится рекламный блок и есть ограничения некоторых функций, и доступны только базовые инструменты. Для отключения рекламного блока можно приобрести Премиум подписку, которая также дает доступ к большое библиотеке декоративных элементов.

Интерфейс редактора выполнен в нейтральных серо-чёрных тонах с яркими акцентами на активных элементах, как представлено на рисунке 4. Слева находится список инструментов в виде иконок в количестве 26 штук. При наведении на иконку всплывает подсказка по инструменту. Ниже расположены два круга с цветами — основным и дополнительным, которые по нажатию открывают палитру выбора цвета. Наверху помещена навигация по фильтрам и дополнительным функциям редактирования. При нажатии на вкладку открывается выпадающее меню. Чуть ниже расположена полоса параметров выбранного инструмента.

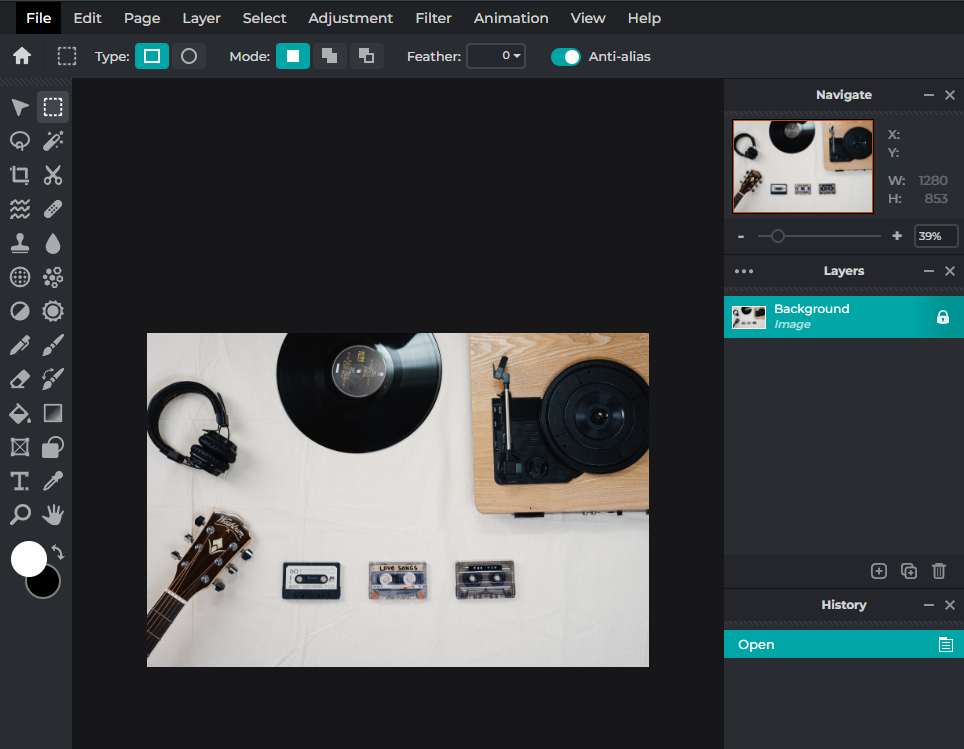


Рисунок 4 — интерфейс Pixlr E

В левой части экрана находится навигация по изображению, список слоёв и блок истории действий. Центральной частью интерфейса является рабочая область. К достоинствам данного интерфейса можно отнести:

* минималистичность;
* привычное расположение инструментов и настроек, для тех, кто уже имел опыт работы с графическими редакторами;
* всплывающие подсказки, для тех, кто только знакомится инструментами подобного рода.

Главной особенностью данного сервиса можно назвать широкий функционал для глубокой обработки изображений. Таким образом, он представляет собой бесплатную альтернативу Adobe Photoshop. Инструментов очень много, что не подходит для решения задачи, поставленной в данной выпускной квалификационной работе, но общую конструкцию можно взять в пример для разработки.

Второй онлайн-редактор под названием Befunky. Позиционируется компанией как сервис для того, чтобы можно было просто и быстро создавать профессиональные фотографии и дизайны без специального образования. Интерфейс представлен на рисунке 5. Он выполнен в тёмных тонах с небольшими яркими цветовыми акцентами. Сервис предлагает несколько разных вариантов для загрузки изображений, как с помощью технологии Drag and Drop, так и стандартным образом, а также можно выбрать что-то из списка предзагруженных изображений.

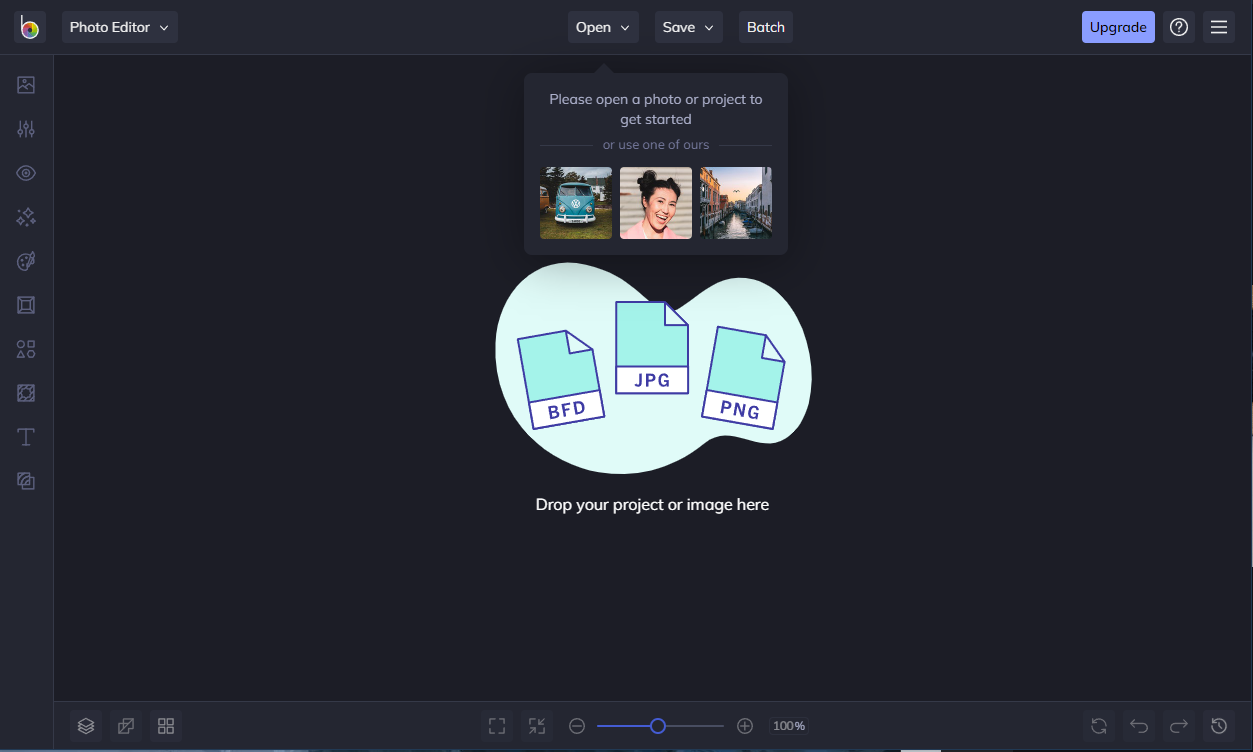


Рисунок 5 — интерфейс Befunky

После загрузки изображения раскрывается обширный список инструментов слева. По нажатию на выбранный инструмент вкладка навигации сменяется на вкладку расширенных настроек. Кроме инструментов присутствуют многочисленные фильтры, маски, стикеры и др. В нижней части экрана находится настройка масштаба, кнопки движения по истории действий и другие полезные функции, но часть из них доступна только после подключения платной подписки. Основную часть экрана занимает рабочая область.

К достоинствам данного интерфейсам можно отнести:

* минимальную загруженность экрана: все лишние меню, навигации можно скрыть;
* минималистичный дизайн;
* интерактивность: все изменения быстро и удобно применяются;
* подсказки к инструментам для новичков.

В данном редакторе также очень много различных инструментов и фильтров, в которых нужно разбираться и ориентироваться. Интерфейс построен не совсем стандартно, к расположению компонентов нужно время, чтобы привыкнуть и запомнить.

Подводя итог, можно сделать вывод, что при создании интерфейсов графических редакторов часто используются определенные паттерны, для того чтобы знакомство пользователя с новым для себя интерфейсом происходило как можно легче и быстрее. Довольно распространено использование широкого списка инструментов и готовых фильтров, но часть функций доступна только за дополнительную плату. Цветовая гамма интерфейсов в основном тёмная.

## 1.3. Анализ программных средств, использованных при разработке

При разработке был использован Visual Studio Code — редактор исходного кода, разработанный Microsoft. Это легкий и бесплатный редактор для разных платформ. Он переведен на множество языков, в том числе и на русский. В нем можно комфортно и эффективно работать с проектами любого размера и направленности благодаря возможности устанавливать нужные расширения, создаваемые сообществом. В данный редактор также входит отладчик, подсветка синтаксиса, автодополнение кода, инструменты работы с Git. Таким образом Visual Studio Code идеально подходит для данной разработки.

Дополнительно при разработке применялись браузеры Mozilla Firefox и Opera для тестирования работоспособности клиентского интерфейса в браузерах на разных движках (Mozilla Firefox на движке Quantum, Opera на Chromium).

Язык программирования, на котором был написан клиентский код — JavaScript. Выбор данного языка программирования обусловлен тем, что он представляет собой инструмент для управления элементами страниц, написания клиентских сценариев и разработку кроссплатформенных программ. Он изначально заточен под работу с браузерами и имеет большое количество библиотек для решения широкого спектра задач.

Из всего многообразия библиотек для JavaScript для разработки пользовательских интерфейсов была выбрана библиотека React. Это одна из самых известных и широко используемых JavaScript-библиотек. В React разрабатываемые интерфейсы состоят из отдельных компонентов, что позволяет сократить количество возможных ошибок в будущем. Данная библиотека основана на декларативном подходе в программировании, т. е. описании конечного результата, она предоставляет мощную ментальную модель. Она использует функциональные и объектно-ориентированные концепции программирования. [6] Также у нее есть хорошо проработанная документация и множество материалов, статей, курсов и т. Д. Важной особенностью React является использование JSX. React использует возможности современного JavaScript для многих своих шаблонов. JSX расширяет синтаксис ECMAScript, так что HTML-подобный код может сосуществовать рядом с ним. Но браузер не распознает JSX, поэтому этот код необходимо преобразовывать с помощью препроцессоров или транспиляторов, например, Babel. Таким образом, используя JSX можно писать краткие структуры, похожие на HTML (древовидные структуры, как DOM), в том же файле, что и основной код JavaScript. После Babel преобразует эти структуры в обычный JavaScript-код.

Преимущества JSX:

* JSX может легко конвертировать HTML-теги в элементы React.
* Быстрее, чем обычный JavaScript.
* Позволяет помещать HTML-элементы в DOM без использования метода appendChild() или createElement().
* Делает проще написание HTML в React.
* Так как JSX — это выражение, мы можем использовать его внутри операторов if и циклов for, присваивать его переменным, принимать в качестве аргументов или возвращать из функций.
* JSX предотвращает атаки XSS (межсайтовый скриптинг), широко известные как инъекционные атаки.
* Он типобезопасен (type-safe), то есть надёжен в применении его системы типов, и большинство ошибок можно обнаружить во время компиляции.

К недостаткам использования библиотеки можно отнести:

* средний порог вхождения,
* увеличение размера приложения,
* приложение может запускаться медленнее, так как нужно выполнить код в браузере,
* затрата ресурсов на хранение виртуальных деревьев DOM. [7]

Изучив статьи и приведенную в них статистику [8, 9, 10], можно сделать вывод о том, что React, на протяжении уже многих лет не только не теряет свою популярность и распространенность, но и наращивает их. В 2020 году был проведен опрос на сайте Stack Overflow, на котором программисты могут задавать друг другу вопросы по своей работе и, следовательно, это сайт, который имеет большое и живое сообщество разработчиков. В нём приняли участие 57 378 человек, 68,9% из которых ответили, что заинтересованы в продолжении работы с React.js. На рисунке 6 представлен график самых используемых в мире фреймворков за 2021 год.

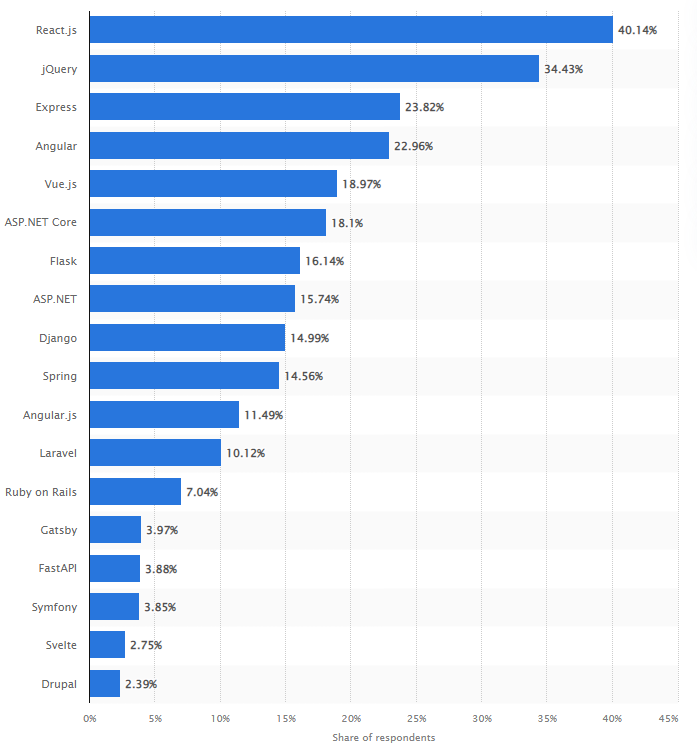


Рисунок 6 — График самых используемых в мире фреймворков за 2021 год[[3]](#footnote-2)

При создании проекта использовалась Node.js — асинхронная исполнительная платформа JavaScript, которая управляется событиями и включает стандартную библиотеку [11]. С её помощью можно писать js-код вне среды браузера. В неё также включено средство для установки и обновления пакетов — Node Package Manager (npm).

**Create React App.** Это инструмент командной строки, который автоматически генерирует проект на React с шаблонами для быстрого начала работы. Благодаря нему не нужно осуществлять ручную настройку Webpack, Babel и Eslint [12]. Также он оптимизирует приложение для финальной стадии разработки, делает сам процесс разработки более комфортным посредством:

* включения всех пакетов JavaScript, необходимых для запуска React-проекта,
* возможности одной командой запустить локальный сервер с приложением, который будет автоматически обновляться при сохранении изменений в проекте
* возможности одной командой осуществить готовую для работы сборку приложения и др.

**Пакет react-dom.** Предназначен для сопряжения с основным пакетом react. В него входят специальные методы для DOM, используемые на верхнем уровне приложения.

**Пакет react-router-dom.** Содержит привязки для использования React Router. В свою очередь React Router — это легкая, полнофункциональная библиотека маршрутизации для React. С её помощью в проекте происходит переключение страницы авторизации на рабочее пространство.

**Пакет react-scripts.** Этот пакет содержит скрипты и конфигурацию, используемые Create React App, именно он устанавливает Babel, Webpack, и Eslint и автоматически настраивает их.

Библиотека **reactstrap.** Это библиотека react-компонентов со стилями Bootstrap 5. В отличие от некоторых библиотек компонентов, Reactstrap не встраивает свои собственные стили, а вместо этого зависит от CSS-фреймворка Bootstrap. Это позволяет иметь согласованные стили для собственных компонентов на основе React и статических частей сайта, а также позволяет при необходимости включать собственную тему Bootstrap. В отличие от использования Bootstrap в HTML, Reactstrap автоматически экспортирует все правильные классы Bootstrap, и ему не нужно использовать или включать файлы из данной библиотеки или добавлять атрибуты данных для активации функций. Вместо этого компоненты определяются в дружественных к React компонентах с соответствующими параметрами, которыми можно управлять.

**Пакет react-sidenav.** В него входит компонент боковой навигации с интерактивными элементами: открытием и закрытием по нажатию на кнопку «гамбургер» / «крестик», меню, разворачивающимся при наведении на раздел.

**Пакет react-colorful.** Компонент с простой палитрой цветов. Она легкая и быстрая, потому что:

* не имеет зависимостей (никаких рисков с точки зрения уязвимостей, никаких неожиданных изменений размера пакета);
* построена только с помощью хуков и функциональных компонентов (для них нет классов и полифиллов);
* включает только минимальное количество оптимизированных вручную алгоритмов преобразования цветов (в то время как большинство популярных сборщиков импортируют целые библиотеки цветовых манипуляций, которые увеличивают размер пакета более чем на 10 КБ и замедляют работу приложения).

**Выводы по главе 1:**

1. Проведен анализ требований к клиентским веб-интерфейсам и современные в их оформлении. Сделан вывод о том, что при разработке необходимо применять правило контроля пользователем интерфейса и последовательность в опыте взаимодействия с данным интерфейсом. Пользователь должен иметь возможность применять свой предыдущий опыт и свободно контролировать действия. В отношении дизайна следует применять принципы минимализма.
2. Проведен анализ существующих зарубежных аналогов. Сделан вывод об общих тенденциях в разработке интерфейсов графических редакторов: тёмная цветовая гамма, слева меню инструментов, сверху дополнительные функции, справа или снизу навигация.
3. Определён функционал и инструментарий, который будет работать на основе клиентской части приложения. Это основные инструменты графического редактора.
4. Проанализированы и определены инструменты и библиотеки, которые будут использоваться и при разработке. Среда для написания кода Visual Studio Code, язык программирования JavaScript, библиотеки React.js и Reactstrap, а также дополнительные зависимости.

# ГЛАВА 2. РАЗРАБОТКА КЛИЕНТСКОГО ИНТЕРФЕЙСА ОНЛАЙН-МИКРОСЕРВИСА ПО РЕДАКТИРОВАНИЮ ИЗОБРАЖЕНИЙ

Разработка программного продукта проводилась на основе процесса разработки в соответствии с жизненным циклом ПО, взятого из ГОСТ Р 51904-2002. Он включает в себя такие этапы, как [13]:

* процесс определения требований к ПО;
* процесс проектирования ПО;
* процесс кодирования ПО;
* процесс интеграции.

На первом этапе было составлено техническое задание на основе ГОСТ 19.201-78 [14]. Полный текст технического задания находится в приложении А.

## 2.1 Создание макета интерфейса

Второй этап разработки проводился посредством создания макета в сервисе Pidoco. Pidoco позволяет создавать интерактивные макеты, каркасы и прототипы графического интерфейса без программирования. Он работает в веб-браузере. Также, своей работой можно поделиться с коллегами, рецензентами и пользователями тестирования для редактирования, обсуждения и утверждения в режиме реального времени. Подобный подход улучшает процесс разработки пользовательских интерфейсов для веб-, мобильных и настольных приложений.

По требованиям из технического задания был создан макет интерфейса, представленный на рисунке 7. В состав интерфейса будут входить такие элементы как:

* рабочая область,
* меню инструментов,
* дополнительные модальные окна.

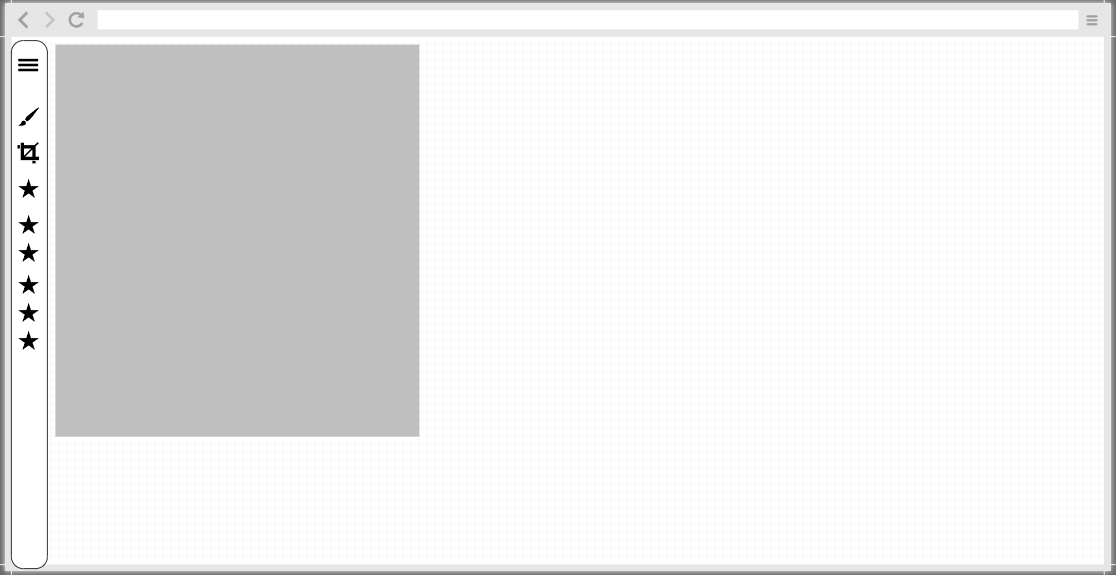


Рисунок 7 — общий вид интерфейса в макете

Пользователь может раскрыть меню, нажав на кнопку «Гамбургер», и увидеть подписи-подсказки к инструментам, что видно на рисунке 8.

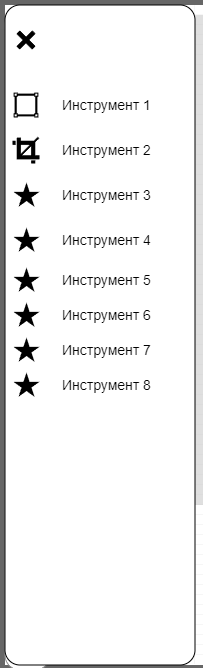


Рисунок 8 — раскрытое меню инструментов

Некоторые инструменты или функции будут иметь возможность настройки. При нажатии на соответствующую кнопку будет открываться модальное окно. Для каждого инструмента будет спроектировано свое окно. Закрытие любого модального окна происходит по нажатию на любую точку области вокруг окна. Рабочая область во время открытия модального окна становится неактивной.

Первое проектируемое окно — это окно загрузки изображения, оно изображено на рисунке 9. В нём будет несколько вариантов загрузки. Пользователь сможет выбрать наиболее удобный для себя в конкретной ситуации. Первый способ стандартный — загрузка через выбор файла в окне Проводника своей операционной системы. Второй способ «Drag and Drop» — загрузка с помощью перетаскивания нужного файла в специально выделенное поле.



Рисунок 9 — модальное окно загрузки изображения

Следующее окно — окно настроек кисти, его вид представлен на рисунке 10. В нем будет расположен бегунок, позволяющий изменить размер кисти, и область палитры, в которой посредством передвижения указателей щелчком мыши или перетаскиванием можно будет выбрать нужный пользователю оттенок. Для подтверждения смены цвета нужно будет нажать кнопку «Поменять цвет».



Рисунок 10 — модальное окно настроек инструмента Кисть

Модальное окно для настройки ластика будет иметь только одну настройку — размер. Оно представлено на рисунке 11.

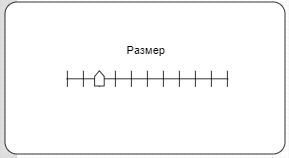


Рисунок 11 — модальное окно настроек инструмента Ластик

В окно с возможностью поворота изображения будет входить поле для ввода целочисленного значения угла поворота, кнопка для подтверждения ввода и две кнопки для быстрого поворота на 90° по часовой стрелке и против часовой, что представлено на рисунке 12.

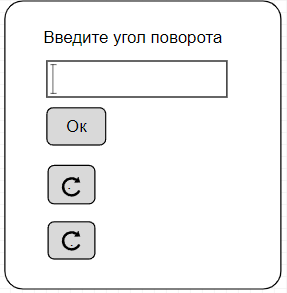


Рисунок 12 — модальное окно настроек инструмента Поворот

На этом завершается этап создания макета данного программного продукта.

Неотъемлемой частью разработки интерфейса является выбор цветовой палитры, в которой он будет исполнен. Основным цветом был выбран #8A2BE2 (оттенок фиолетового). Он не слишком яркий и не слишком тусклый. Цвет фона рабочей области должен быть нейтральным, не перетягивающим внимание на себя. Таким образом, был выбран цвет #EDEEF7 (оттенок серого). Текст должен быть хорошо читаем на фоне основного цвета интерфейса, для этого идеально подходит чисто белый цвет.

## 2.2. Кодирование клиентского интерфейса

Этап непосредственного написания кода клиентской части продукта начинается с создания проекта и установки необходимых зависимостей, описанных в главе 1, с помощью npm. Базовый проект был создан с применением инструмента create-react-app. Структура проекта клиентской части отображена на рисунке 13.

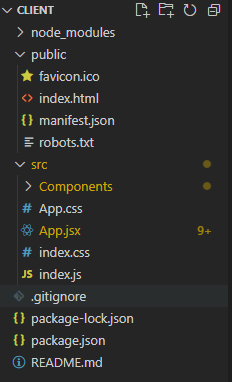


Рисунок 13 — структура проекта

Все установленные пакеты хранятся в папке node\_modules. В папке public хранятся статические файлы, связанные со страницей, на которой будет отрисовываться наш интерфейс и сам файл этой страницы — index.html. Файл favicon.ico является иконой, которая будет отображаться на вкладке браузера при загрузке страницы. Она представлена на рисунке 14. Иконка создана под стиль графического редактора, с использованием фирменных цветов и первой буквы названия.



Рисунок 14 — иконка онлайн-графического редактора «Палитра»

Файл manifest.json — это конфигурационный файл, который содержит метаданные в соответствии с критериями Прогрессивных Веб Приложений (PWA — Progressive Web Apps) [15]. И последний файл в данной папке — это robots.txt, который отвечает за предоставление поисковым системам информации о том, в какие разделы поисковой робот может заходить, а в какие нет [16].

Рассмотрим индексную начальную страницу приложения index.html на рисунке 15.

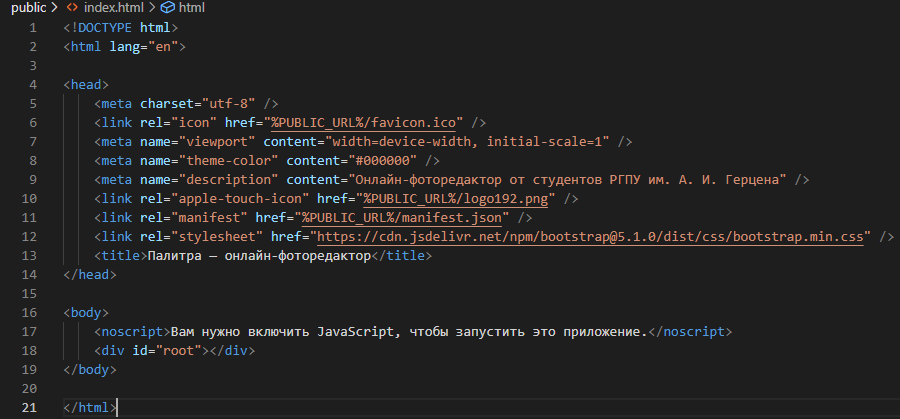


Рисунок 15 — код index.html

Здесь нужно обратить внимание на строки 13, 17, 18.

* На строке 13 находится html-тэг заголовка вкладки.
* На строке 17 находится тэг noscript, который выводит поясняющий текст, если у пользователя не поддерживается или отключена работа JavaScript.
* На строке 18 находится тэг div с id="root". Именно в это место будет помещаться наш интерфейс.

Итоговое содержание зависимостей и скриптов находится в файле package.json, который хранит всю основную информацию о проекте, представлено на рисунке 16.

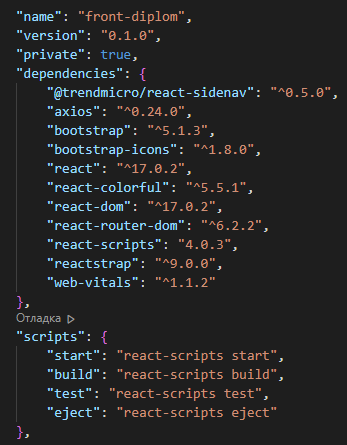


Рисунок 16 — содержание package.json

Самая главная папка проекта — src. В ней находятся такие файлы, как:

* App.jsx — содержит главный компонент приложения.
* App.css — содержит стили компонента App.jsx.
* index.js — содержит стартовую точку приложения.
* index.css — хранит базовые стили приложения.

Также была создана папа Components, в которой хранятся все специально написанные компоненты, которые будут использоваться в интерфейсе. Всего для проекта разработано три компонента:

* MyCanvas.jsx — основное рабочее пространство, на котором будет отрисовываться загружаемое изображение, и на котором будут происходить манипуляции с этим изображением.
* MySideNav.jsx — боковая навигация по инструментам и функциям графического редактора.
* Modal.jsx — модальное окно для отображения расширенных параметров инструментов и функций.

Для каждого компонента существует свой файл с каскадными стилями: MyCanvas.module.css, MySideNav.module.css и modal.css.

Модальное окно представляет собой два блока div, вложенных друг в друга. Каждый из них имеет несколько состояний, которые передаются внутрь компонента. Внешний блок отвечает за перекрытие всей рабочей области и её затемнение. Это необходимо для направления фокуса пользователя на окно с параметрами, а также для возможности закрывать появившееся окно по нажатию на затемненную область.

Открытие и закрытие модального окна реализовано с помощью состояний React и отслеживания событий. В компонент передается:

* переменная active, которая хранит состояние true или false, в зависимости от которого будет применен тот или иной класс css (строки 7 и 11 на рисунке 17);
* функция setActive, с помощью которой изменяется состояние переменной active;
* переменная children. В ней содержатся все элементы, которые находятся между тэгами данного компонента.

В событие onClick на строке 8 помещается функция setActive, которая меняет состояние на false, что означает закрытие модального окна. Но так как внутри находится блок с контентом, необходимо вылавливать событие onClick, когда нажатие происходит именно в этом блоке.



Рисунок 17 — код компонента modal.jsx

Для этого существует метод stopPropagation, который прекращает движение события по фазам [17]. Последний уровень вложенности, это переменная children, с помощью которой помещается нужное содержимое окна.

В стилях modal.css прописаны свойства обоих блоков для каждого из двух состояний. Блок с затемнением должен быть во весь экран, поэтому применяются свойства height: 100v; и width: 100vw;. Сам эффект затемнения создается за счет заливки фона черным цветом с непрозрачностью в 20%. Каждое модальное окно должно появляться по центру экрана, поэтому применяются свойства:

* display:flex
* align-items: center;
* justify-content: center;

Для того чтобы все созданные окна не отображались до момента их активации, добавляется свойство прозрачности opacity со значением 0. Свойство transition: 0.5s управляет скоростью появления элементов, тем самым добавляя плавности. Pointer-events отвечает за управление событиями курсора. В скрытом режиме это свойство имеет значение none, запрещая какие-либо события. При активации элемента происходит переключение на подкласс .modal.active, в котором выставляется значение прозрачности 1 и разрешаются все события курсора, т. е. все кнопки становятся активными.

Блоку с контентом применяются свойства минимального размера, с целью сохранения формы при уменьшении размера экрана, применяется выбранный в первом параграфе основной фоновый цвет и закругляются углы свойством border-radius. Скругления позволяют сделать дизайн более органичным [18], дружественным. Также, как и во внешнем блоке, здесь используются flex-свойства для центрирования внутренних элементов и свойства, создающие небольшую анимацию при появлении:

* transform: scale(0.5); — масштабирование элемента
* transition: 0.4s all; — скорость появления, которая применяется ко всем дочерним элементам

При активации происходит переключение на подкласс .modal\_\_content.active, в котором свойство transform принимает значение scale(1), приводя элементы к изначально заданным размерам.

Следующий компонент отвечает за меню инструментов — MySideNav.jsx, представлен на рисунке 18. В нём применялся готовый компонент навигации из зависимости @trendmicro/react-sidenav. К нему было добавлено отслеживание состояния (булева переменная isVisible и функция изменения состояния setVisible), для того чтобы меню могло раскрываться, и возможность передавать внутрь дочерние элементы.



Рисунок 18 — код компонента MySideNav.jsx

К примененным стилям относится основной цвет и фиксированное расположение меню. Так, при прокрутке страницы, меню всегда будет находиться удобно в зоне досягаемости.

Самый главный и основной компонент интерфейса — это MyCanvas.jsx — холст, на котором будет происходить работа с изображением, и меню с инструментами. Условно его можно разделить на две части: функциональную и то, что будет отрисовано на странице.

Разработка началась с создания холста, код представлен на рисунке 19. Было решено, что сервис будет работать с двумя слоями: фоновым и верхним, слоем для рисования. Таким образом можно рисовать кистью или с помощью фигур, не портя исходное изображение, так как все рисунки легко стираются ластиком.

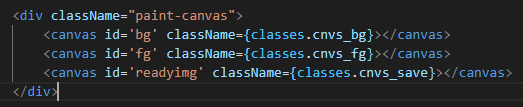


Рисунок 19 — код элемента холста

Каждый слой является отдельным элементом canvas. Третий холст необходим для сохранения итогового изображения. На нем происходит объединение двух созданных слоёв. Следующим шагом было добавлено меню и кнопки для вызова инструментов и функций. Пример того, как выглядит код кнопки на рисунке 20.

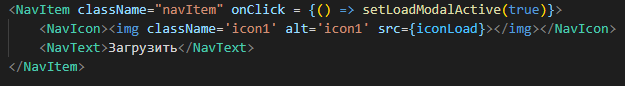


Рисунок 20 — код кнопки «Загрузить»

Кнопка представляет собой элемент навигации, к которому прикреплён слушатель события onClick. Он в свою очередь запускает работу соответствующей функции. У кнопки загрузки изображения это функция активации модального окна. Внутри этого элемента находится блок с иконкой и блок с заголовком, который видно только при раскрытии бокового меню.

Модальное окно загрузки изображения, в соответствии с макетом состоит из формы загрузки файла и поля, в которое загружаемый файл можно перетащить. На форме висит слушатель события onSubmit, которое срабатывает при нажатии пользователем кнопки, относящейся к этой форме, то есть при подтверждении отправки выбранного изображения. Сначала, при выборе файла запускается асинхронная функция, которая конвертирует изображение в понятный для сервера формат. После этого запускается асинхронная функция, которая уже отправляет результат работы предыдущей функции непосредственно на сервер. Далее клиент обращается к серверу для отрисовки полученного изображения на фоновом холсте. На последнем шаге используется такой объект JavaScript как Promise (промис). Промис — это объект, замещающий значение, которое станет доступным в какой-то момент в будущем. Это позволяет другому коду, использующему этот промис, подготовиться к обработке ожидаемого значения [19].

Поле Drag and Drop реализовано с помощью тернарного оператора React и слушателей событий. Тернарный оператор определяет, какой текст будет показываться пользователю при определенных условиях. В данном случае условием является наличие над полем файла или его отсутствие, о котором сигнализирует состояние булевой переменной drag. Когда файла нет, поле подсказывает пользователю, что в него нужно перетащить файл (рисунок 21, а), а когда файл появляется, подсказка сменяется на “Отпустите файл” (рисунок 21, б). После этого происходят аналогичные действие с конвертацией картинки и отправкой её на сервер. По окончании загрузки, для удобства пользователя окно автоматически закрывается.

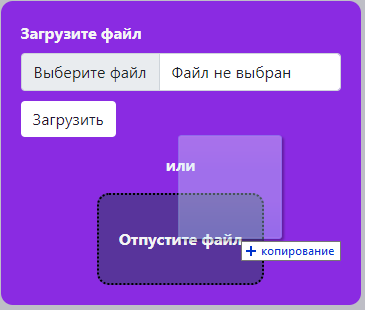
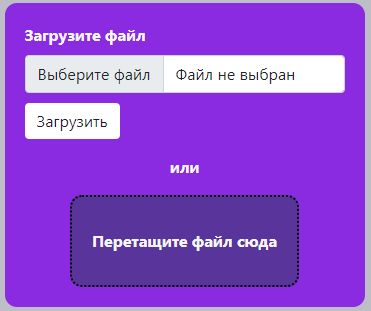


Рисунок 21 — а) начальное состояние поля Drag and Drop, б) состояние во время наведения файла на поле

Сохранение изображение на компьютер происходит по нажатию соответствующей кнопки в меню. Она запускает функцию saveImgBtn, которая совмещает два изображения на третьем невидимом холсте и создает ссылку, которая в свою очередь автоматически активируется, и браузер сохраняет изображение по этой ссылке.

После кнопки сохранения в меню идет кнопка первого инструмента — кисть. Данный инструмент реализован с помощью трёх обработчиков событий: когда кнопка мыши была нажата (событие mouseDown), когда она удерживалась (mouseOver), и когда она была отпущена (mouseUp). С каждым из событий в массив записываются данные с координатами курсора. В момент, когда клавиша мыши нажимается, запускается процесс рисования при обращении к методу контекста холста beginPath(), после чего перо перемещается из своей изначальной точки [0, 0] в точку, в которой произошло нажатие. При этом используется метод moveTo() [20]. При удержании клавиши мыши применяется метод stroke(), который отрисовывает линию по заданным координатам.

Все три обработчика прикрепляются к соответствующим слушателям событий в момент нажатия кнопки в меню инструментов. Также во время нажатия на кнопку в меню свойству контекста холста globalCompositeOperation присваивается значение "source-over". Данное свойство отвечает за режимы наложения графики. Для кисти должно сохранятся значение по умолчанию — "source-over". При этом выбранном свойстве новая фигура накладывается на старые и перекрывает их [21].

Выбор данного инструмента в меню также вызывает открытие модального окна с настройкой его параметров: цвета и толщины, как показано на рисунке 22.

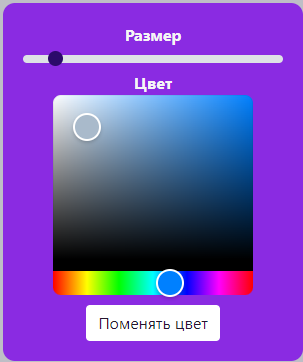


Рисунок 22 — вид модального окна настройки инструмента Кисть со стилями

Настройка размера кисти представлена в виде формы типа «Ползунок» (type="range"). В ней при возникновении события onChange, т. е. при изменении положения ползунка происходит изменение состояния переменной, хранящей размер кисти (brushVal), и присвоение этого значения свойству контекста холста, отвечающей за толщину линии — lineWidth. Начальное значение размера задано как «10». Настройка цвета кисти осуществляется посредством использования компонента HexColorPicker из зависимости react-colorful. Он также изменяет переменную состояния, только заданной для хранения цвета (color). Но для того, чтобы выбранный цвет был применен к контексту холста, необходимо подтвердить выбор нажатием кнопки «Применить цвет». В контексте холста за цвет отвечает свойство strokeStyle. Начальное значение цвета задано как «#aabbcc».

Инструмент Ластик использует те же обработчики событий, что и кисть, но при его активации свойство globalCompositeOperation принимает значение "destination-out", что значит, что отображается часть фигуры, на которую не накладывается новая. Новая фигура не выводится. При выборе данного инструмента в меню активируется модальное окно с настройкой размера ластика, как на рисунке 23. Она реализована аналогично настройке размера кисти, но изменяет другую переменную состояния — eraseVal.

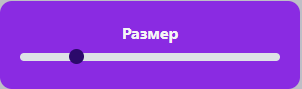


Рисунок 23 — вид модального окна настройки инструмента Ластик со стилями

Инструмент Прямоугольник после его выбора в меню считывает координаты двух нажатых точек, вычисляет ширину и длину прямоугольника. Потом, используя метод контекста холста strokeRect(), происходит отрисовка контура вычисленной фигуры на холсте. В конце массив точек очищается и нужно активировать инструмент заново.

Инструмент Треугольник рисует треугольник по трём выбранным точкам, которые соответствуют расположению трёх углов, используя стандартные методы canvas: beginPath(), moveTo(), lineTo(), closePath(), stroke().

Инструмент Круг рисует фигуру круга по двум точкам: первая точка — это центр круга, вторая точка — это размер радиуса круга. За отрисовку круга на холсте отвечает метод arc().

Все инструменты для рисования фигур за раз позволяют добавить только одну фигуру. Цвет контура фигуры и его толщина зависят от выбранных параметров у кисти или ластика.

Инструмент Поворот при выборе в меню активирует небольшое модальное окно, как на рисунке 24, соответствующее по своему содержанию макету из первого параграфа.

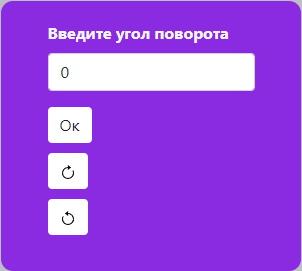


Рисунок 24 — вид модального окна настройки инструмента Поворот со стилями

Форма для ввода угла поворота изменяет переменную состояния. При нажатии кнопки «Ок» происходит подтверждение ввода и отправка запроса о повороте изображения угол, равный введенному значению, на сервер. Две кнопки со стрелками ниже являются дополнительными и нужны для быстрого совершения популярного действия. Они поворачивают изображение на 90 градусов по часовой и против часовой стрелки.

Таким образом, разработанный клиентский интерфейс решает задачи, поставленные в первой главе первом параграфе данной выпускной квалификационной работы.

**Выводы по главе 2:**

1. Создан макет интерфейса в соответствии техническим заданием и обозначенными функциями.
2. Разработан код клиентского интерфейса в соответствии с макетом, который решает поставленные задачи.

# ЗАКЛЮЧЕНИЕ

Цель разработать клиентской интерфейс онлайн-микросервиса по редактированию изображений на основе импортозамещенных веб-технологий, которая была поставлена в данной выпускной квалификационной работе, была выполнена.

В процессе выполнения данной работы были решены следующие задачи:

1. Осуществлен анализ существующих принципов разработки клиентских интерфейсов и современных трендов в дизайне. В результате анализа было определено, что для данного программного продукта наиболее подходящим будет правило контроля пользователем интерфейса и последовательность в опыте взаимодействия с данным интерфейсом. В дизайне интерфейса будут применяться принципы минимализма.
2. Проведен анализ существующих зарубежных аналогов, сделан вывод о применении определенных паттернов построения интерфейсов, использовании преимущественно тёмной цветовой гаммы и большого числа функций и инструментов.
3. Определены инструменты и функции, которые должны быть реализованы на клиентском интерфейсе. К функциям относятся: отрисовка изображения на рабочей области, возможность сохранить изображение на компьютер, отправка запросов на сервер для осуществления загрузки изображения, поворота. К инструментам относятся: кисть, ластик, фигуры (прямоугольник, треугольник, круг).
4. Выбрана для работы среда для написания кода Visual Studio Code, язык программирования JavaScript, библиотеки React.js и Reactstrap, а также дополнительные зависимости.
5. Создан макет будущего клиентского интерфейса на основе требований и заявленного функционала.
6. Разработан код клиентского интерфейса онлайн-микросервиса по редактированию изображений на основе импортозамещенных веб-технологий.

# СПИСОК ЛИТЕРАТУРЫ

1. Купер А. Алан Купер об интерфейсе. Основы проектирования взаимодействия / Купер А., Рейман Р., Кронин Д. – Пер. с англ. – Санкт-Петербург: Символ-Плюс, 2009. – 688 с.
2. Сергеев С. Ф. Введение в проектирование интеллектуальных интерфейсов: Учебное пособие / Сергеев С. Ф., Падерно П. И., Назаренко Н.А. – Санкт-Петербург: СПБГУ ИТМО, 2011. – 108 с.
3. Основы HTML - Изучение веб-разработки | MDN – URL: https://developer.mozilla.org/ru/docs/Learn/Getting\_started\_with\_the\_web/HTML\_basics (дата обращения: 13.03.2022) – Режим доступа: свободный.
4. Принципы и правила проектирования пользовательского интерфейса / В. В. Астафьева. – Текст: непосредственный // Молодой ученый. – 2020. – № 1 (291). – С. 10-14. – URL: https://moluch.ru/archive/291/66058/ (дата обращения: 05.04.2022) – Режим доступа: свободный.
5. Руководство по Modern Minimal стилю UI дизайна – UXPUB – URL: https://ux.pub/editorial/rukovodstvo-po-modern-minimal-stiliu-ui-dizaina-4m2h (дата обращения: 05.04.2022) – Режим доступа: свободный.
6. Тиленс Т. М. React в действии / Тиленс Т. М. – Санкт-Петербург: Питер, 2018. – 368 с.
7. Что такое React и как его освоить? — Академия Яндекса – URL: https://academy.yandex.ru/posts/chto-takoe-react-i-kak-ego-osvoit (дата обращения: 07.04.2022) – Режим доступа: свободный.
8. React Usage Statistics – URL: https://trends.builtwith.com/javascript/React (дата обращения: 09.04.2022) – Режим доступа: свободный.
9. Infographic: React.js stats for 2020 – URL: https://www.thinslices.com/blog/infographic-react.js-statistics (дата обращения: 09.04.2022) – Режим доступа: свободный.
10. Usage Statistics and Market Share of React for Websites, May 2022 – URL: <https://w3techs.com/technologies/details/js-react> (дата обращения: 09.04.2022) – Режим доступа: свободный.
11. Янг А. Node.js в действии. 2-е издание / Янг А., Мек Б., Кантелон М. – Санкт-Петербург: Питер, 2018. – 432 с.
12. Бэнкс А. React: современные шаблоны для разработки приложений 2-е издание / Бэнкс А., Порселло Е. – Санкт-Петербург: Питер, 2021. – 320 с.
13. ГОСТ Р 51904-2002. Программное обеспечение встроенных систем. Общие требования к разработке и документированию // Электронный фонд правовых и нормативно-технических документов – URL: https://docs.cntd.ru/document/1200030195 (дата обращения: 25.04.2022) – Режим доступа: свободный.
14. ГОСТ 19.201-78. Единая система программной документации. Техническое задание. Требования к содержанию и оформлению // Электронный фонд правовых и нормативно-технических документов – URL: https://docs.cntd.ru/document/1200007648 (дата обращения: 25.04.2022) – Режим доступа: свободный.
15. Chiarelli A. Beginning React: Simplify Your Frontend Development Workflow and Enhance the User Experience of Your Applications with React / Chiarelli A. – Великобритания: Packt Publishing, 2018. – 96 с.
16. Новицкий И. Интернет-маркетинг. Продвижение СЕО / Новицкий И. – Россия: ЛитРес, 2019. – 180 с.
17. Чиннатхамби К. JavaScript с нуля / Чиннатхамби К. – Санкт-Петербург: Питер, 2022. – 400 с.
18. Берд Дж. Веб-дизайн. Руководство разработчика / Берд Дж. – Санкт-Петербург: Питер, 2012. – 224 с.
19. Ришкуция В. Программируй & типизируй. / Ришкуция В. – Санкт-Петербург: Питер, 2021. – 352 с.
20. Дронов В. А. JavaScript. 20 уроков для начинающих / Дронов В. А. – Санкт-Петербург: БХВ-Петербург, 2020. – 352 с.
21. Прохоренок Н. А. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера, 5 издание / Прохоренок Н. А., Дронов В. А. – Санкт-Петербург: БХВ-Петербург, 2019. – 912 с.

1. Источник: google.com [↑](#footnote-ref-0)
2. Источник: https://infogra.ru/ui/9-glavnyh-trendov-v-ux-ui-dizajne-v-2021-godu?hcb=1 [↑](#footnote-ref-1)
3. Источник: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/ [↑](#footnote-ref-2)