Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

ИНСТИТУТ ЭЛЕКТРОННЫХ И ИНФОРМАЦИОННЫХ СИСТЕМ

**ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ**

РАзработка веб-сайта для прослушивания музыкальных композиций

Курсовой проект по учебной дисциплине

«Технология разработки программного обеспечения»

по специальности 09.02.03 Программирование в компьютерных системах

ПТК.КП 2993 02.000ПЗ

|  |
| --- |
| Руководители  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. Р. Ильин  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 года |
| Студент группы 2993  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ф. П. Бурлаков  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 года |

Оглавление

[Аннотация 3](#_Toc165573609)

[Введение 5](#_Toc165573610)

[Раздел 1. Общая (теоретическая) часть 7](#_Toc165573611)

[1.1 Постановка задачи (ТЗ) 7](#_Toc165573612)

[1.1.1 Характеристики бизнес-процессов 7](#_Toc165573613)

[1.1.2 Технико-математические описание задачи 7](#_Toc165573614)

[1.1.3 Требования к программе 8](#_Toc165573615)

[1.2 Обоснование проектных решений 9](#_Toc165573616)

[1.2.1 Обоснование выбора языков программирования 9](#_Toc165573617)

[1.2.2 Инструментальные средства 10](#_Toc165573618)

[1.2.3 Обоснование выбора среды программирования 10](#_Toc165573619)

[1.2.4 Информационное обеспечение 10](#_Toc165573620)

[1.3 Обзор существующих программных систем 11](#_Toc165573621)

[Раздел 2. Практическая составляющая 12](#_Toc165573622)

[2.1 Реализация 12](#_Toc165573623)

[2.1.1 Обоснование и описание выбора состава технических и программных средств 12](#_Toc165573624)

[2.1.2 Практическая реализация алгоритмов 14](#_Toc165573625)

[2.1.3 Реализация интерфейса 20](#_Toc165573626)

[2.2 Тестирование 21](#_Toc165573627)

[Заключение 30](#_Toc165573628)

[Список литературы 31](#_Toc165573629)

[Приложение А 32](#_Toc165573630)

[Приложение Б 36](#_Toc165573631)

[Приложение В 40](#_Toc165573632)

# Аннотация

Данный проект представляет собой разработку веб-приложения, которое предлагает пользователям возможность прослушивать песни, а также расширенное использование сервиса для композиторов. Пользователи могут проигрывать аудио файлы композиций, следить за выходом треков разных исполнителей. Поддерживать определенных исполнителей через подписку. Для создания сайта используются HTML для структурирования контента, CSS для стилизации и JavaScript для реализации функциональной части сайта, взаимодействия с локальным хранилищем и работы с аудио файлами.

# Введение

В современном мире, где музыка играет не последнюю роль в жизни человека, в независимости от любимого жанра отдельно взятого человека. Композиторы вкладывают в свои произведения много эмоций, которые отражаются в слушателях. Но, к сожалению, не все могут сочинять новое творчество, так как это требует не только навыков, а ещё многих вложений и хороших специалистов в других областях музыкального мастерства.

Наличие удобной платформы, как для пользователя, так и для композитора становится с каждым годом актуальнее. Настроение, новые идеи, новые желания и свершения в жизни часто зависит от музыки. С точки зрения этих фактов разработка удовлетворяющего приложения для прослушивания и опубликования музыки, является приоритетом развития данного направления.

Целевой аудиторией данного приложения являются люди, которым необходима возможность из любой точки пространства запустить приложение для прослушивания песни. Это могут быть путешественники, телекоммуникационные службы, спортсмены, а также обычные люди, для которых погодные данные важны в повседневной жизни. В основную часть аудитории так же входят музыкальные артисты, которым в свою очередь, необходимо контролировать отчетность прослушиваний, создание качественного материала формата “.mp3” и правильного продвижения. Разработанное веб-приложение предоставляет им удобный и эффективный способ прослушивания или создания музыки.

В рамках данного проекта будет разработано веб-приложение, позволяющее пользователям не только прослушивать музыку, но и приобретать ее, полностью следить за деятельностью музыканта в границах эстрады и другое. В связи с одной из главных проблем – безопасностью в интернете, сайт предоставляет как композитору, так и слушателю возможность контроля монетизации песен. Слушатель может выбирать кому из музыкантов, которых пользователь слушает, будет отправлены деньги, полученные с подписки на платформу. Композиторы в свою очередь, смогут полностью отслеживать поток монетизации в любое время, без третьего юридического лица.

Это сделает мир музыкального творчества ближе к широкой аудитории, что не только привлечёт много новых пользователей на платформу, но и увеличит количество хороших музыкальных произведений. В целом создание сайта с такими возможностями для стороны слушателей и стороны исполнителей является необходимым переходом на новую ступень удобства в данной сфере развлечений и улучшения качества отечественной музыки.

Для достижения поставленной цели будут решены следующие задачи:

1. Анализ предметной области и требований к приложению;
2. Выбор технологий и инструментов разработки;
3. Выбор архитектуры приложения;
4. Проектирование архитектуры и функциональности приложения;
5. Создание диаграмм, описывающих алгоритмы и функции;
6. Разработка макета для выявления самых удобных комбинаций интерфейса;
7. Разработка пользовательского интерфейса;
8. Тестирование и отладка приложения;
9. Развертывание и публикация приложения.

# Раздел 1. Общая (теоретическая) часть

## 1.1 Постановка задачи (ТЗ)

### 1.1.1 Характеристики бизнес-процессов

Бизнес-процесс - совокупность взаимосвязанных мероприятий или работ, направленных на создание определенного продукта или услуги для потребителей.

Основными бизнес-процессами, связанными с разработкой приложения для прослушивания музыки, являются работа с хранилищем аудио файлов из надежных источников, а также их воспроизведение в формате удобного интерфейса. Эти процессы должны выполняться в режиме реального времени.

Для приложения музыкальной платформы были выявлены следующие бизнес-процессы:

1. Получение данных об композиции из объекта, являющимся хранилищем, с возможностью изменения;
2. Обработка и анализ композиции: процесс запроса к собранным ранее данным в удобный для работы и вывода формат;
3. Воспроизведение музыки на основе функции, на основе полученных данных при обращении к объекту с данными;
4. Предоставление изменения воспроизведения пользователям: процесс визуализации кнопок изменения и повторного запроса к алгоритму воспроизведения.

### 1.1.2 Технико-математические описание задачи

Для реализации были использованы следующие средства:

HTML - стандартизированный язык разметки документов в Интернете. Большинство веб-страниц содержат описание разметки на языке HTML. Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства. Язык HTML до 5-й версии определялся как приложение SGML (стандартного обобщенного языка разметки по стандарту ISO 8879). Спецификации HTML5 формулируются в терминах DOM (объектной модели документа). В интернете HTML-страницы, как правило, передаются браузерам от сервера по протоколам HTTP или HTTPS, в виде простого текста или с использованием шифрования.

CSS - формальный язык описания внешнего вида документа, написанного с использованием языка разметки. Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL.

Протокол передачи данных HTTP (англ. HyperText Transfer Protocol - «протокол передачи гипертекста») - протокол прикладного уровня передачи данных. Основой HTTP является технология «клиент-сервер». Основным назначением HTTP является передача веб-страниц (текстовых файлов с разметкой HTML), хотя с помощью него с успехом передаются и другие файлы, как связанные с веб-страницами (изображения и приложения), так и несвязанные с ними. HTTP предполагает, что клиентская программа - веб-обозреватель - способна отображать гипертекстовые веб-страницы и файлы других типов в удобной для пользователя форме.

### 1.1.3 Требования к программе

Функциональные требования:

* Возможность регистрации пользователя.
* Воспроизведение аудио файлов.
* Переход между страницами сайта.
* Функционирующие и стилизованные модальные окна.

Нефункциональные требования:

* Простой и интуитивно понятный интерфейс.
* Быстрая загрузка и отзывчивость приложения.
* Кроссбраузерная совместимость.

## 1.2 Обоснование проектных решений

### 1.2.1 Обоснование выбора языков программирования

Выбор языков программирования зависит от многих факторов, например задачи и функции, которые нужно решить и реализовать. Для разработки музыкального веб-сайта обусловлен следующими факторами:

HTML был выбран для создания структуры и разметки веб-страницы, так как он является стандартом для веб-разработки и обеспечивает семантическую структуру контента.

CSS, используется для стилизации и оформления интерфейса приложения. CSS, является основным, а главное понятным средством стилизации проекта, позволяющий быстро, удобно и понятно создать красивый дизайн.

JavaScript применяется для реализации интерактивного функционала, обработки данных и взаимодействия с локальным хранилищем. Он имеет простой синтаксис и большое сообщество разработчиков, что делает его удобным для изучения и использования. Кроме того, JavaScript имеет множество библиотек и фреймворков, которые упрощают разработку веб-приложений. JavaScript является мощным и гибким языком программирования, широко используемым в веб-разработке.

Таким образом, выбор языков программирования обоснован тем, что они являются популярными и широко используемыми языками программирования в веб-разработке.

### 1.2.2 Инструментальные средства

Для разработки приложения использовались следующие инструментальные средства:

* Visual Studio Code - кроссплатформенная интегрированная среда разработки (IDE), обеспечивающая удобный редактор кода, отладчик и множество полезных расширений для веб-разработки. Она предоставляет широкий набор инструментов для отладки, рефакторинга кода и автоматизации задач.
* Git - распространенная система контроля версий, позволяющая отслеживать изменения в коде, производить откаты и совместную работу над проектом.
* Figma - это онлайн-редактор дизайна, который позволяет создавать макеты и прототипы веб-сайтов, приложений и других пользовательских интерфейсов.

### 1.2.3 Обоснование выбора среды программирования

Браузер был выбран в качестве среды выполнения для данного веб-приложения, так как он предоставляет все необходимые возможности для отображения и выполнения HTML, CSS и JavaScript-кода. Современные веб-браузеры поддерживают новейшие стандарты и технологии, что делает их идеальной средой для разработки и запуска веб-приложений.

### 1.2.4 Информационное обеспечение

Для сайта для прогноза погоды были выявлены следующие элементы информационного обеспечения:

Для разработки логики работы приложения для прослушивания музыкальных файлов используется язык программирования JavaScript.

Для получения первичных данных аудио файлов используется множество интернет-платформ, включая и официальные ресурсы музыкантов. Что в свою очередь обеспечивает надежность и качество данных.

## 1.3 Обзор существующих программных систем

На рынке существует ряд приложений и веб-сайтов, предназначенных для прогноза погоды. Некоторые из наиболее популярных и широко используемых:

* Яндекс Музыка

Сайт: https://music.yandex.ru/home

* Веб-сайт и мобильное приложение Яндекс Музыка предлагают популярные платформы для прослушивания, создание плейлистов и оповещения. Они известны удобным интерфейсом.
* Spotify

Сайт: https://www.spotify.com/int/why-not-available/

Этот веб-сайт предоставляет очень удобный и интуитивно понятный интерфейс, понятная система монетизации. Главная проблема заключается в том, что Spotify в данный момент не доступен для русской аудитории.

* Vk Music(BOOM)

Сайт: <https://vk.com/boom>

Как такового сайта нету. Данная платформа встроена в сервисы мессенджера Вконтакте и это является ее главным плюсом.

* Apple Music

Сайт: https://music.apple.com/us/browse

Этот веб-сервис и мобильное приложение имеют приятный интерфейс, проработанную систему и качественные функции.

Преимущества разрабатываемого приложения заключаются в его легкости, быстрой загрузке и предлагаемых услуг, что делает его удобным для использования различных пользователей и в условиях медленного интернет-соединения. Кроме того, приложение будет иметь преимущество гибкости и возможности настройки в соответствии с потребностями пользователей.

# Раздел 2. Практическая составляющая

## 2.1 Реализация

### 2.1.1 Обоснование и описание выбора состава технических и программных средств

Для разработки музыкального приложения использовались следующие технические и программные средства:

* Операционная система: Windows 10;
* Браузер: Mozilla Firefox (для тестирования и отладки);
* IDE: Visual Studio Code;
* Система контроля версий: Git;
* Инструменты разработки: npm;
* Онлайн редактор дизайна: Figma;

Windows 10 – это операционная система , выпущенная компанией Microsoft в 2015 году. Она объединяет лучшие элементы Windows 7 и Windows 8, обеспечивая пользователю удобство использования и инновационные возможности. Windows 10 обладает улучшенным интерфейсом, который оптимизирован под работу как на ПК, так и на планшетах и смартфонах. OC Windows 10 имеет много полезных функций, таких как виртуальные рабочие столы, центр уведомлений, голосовой помощник Cortana, браузер Edge, защитные механизмы Windows Defender и Windows Hello для аутентификации с помощью лиц и отпечатков пальцев. Также, эта операционная система предлагает облачные сервисы через OneDrive, а также поддерживает универсальные приложения, которые могут работать на всех устройствах под управлением этой OC. Обновления выпускаются регулярно, чтобы обеспечить безопасность и улучшить производительность Windows 10. Также, используется в большом количестве проектов из-за своей простоты, гибкости, удобности и быстроты.

Mozilla Firefox — свободный браузер на движке Quantum, разработкой и распространением которого занимается Mozilla Corporation. Четвёртый по популярности браузер в мире (после Google Chrome, Safari и Microsoft Edge) и первый среди свободного ПО. Удобен для тестирования веб-проектов консолью отладчика, очень важным инструментом в деятельности веб-разработчика, так же один из лучших браузеров для проверки и настройки адаптивности сайта.

Microsoft Edge - Microsoft Edge - это браузер, разработанный компанией Microsoft, который предлагает быструю и безопасную работу в сети. Он построен на технологии Chromium, что обеспечивает высокую производительность и совместимость с веб-стандартами. Особенности браузера Microsoft Edge включают в себя защиту от онлайн-угроз с помощью встроенных инструментов SmartScreen и защиты непосредственно на уровне браузера. Также в Edge доступен режим "Инкогнито" для незаметного просмотра в сети без сохранения истории и данных. Браузер поддерживает удобный режим чтения, блокировку рекламы и расширения для расширения функционала. Microsoft Edge также обладает возможностью синхронизации данных и настроек между устройствами под управлением Windows 10.Своей удобной и интуитивно понятным интерфейсом Microsoft Edge предлагает удобный и безопасный интернет-опыт для пользователей. Использовался для быстрой проверки стилей наложенных на страницы. Во время работы использовался изредка и только как второстепенный.

Visual Studio Code – текстовый редактор для программирования, разработанный Microsoft для Windows, Linux и macOS, позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом, но готовые сборки распространяются под проприетарной лицензией. Также, используется в большом количестве проектов из-за своей простоты, гибкости, удобности и быстроты. Он поддерживает работу с огромным количество языков программирования, включая JavaScript. Имеет большое количество инструментов для работы с кодом, его фреймворками и библиотеками. Приложение имеет интуитивно понятный дизайн, что даже люди не связанные с IT смогут в нём разобраться довольно быстро. Именно поэтому для разработки конечного продукта была выбрана именно эта среда программирования, а не другие.

Figma - представляет собой онлайн-платформу (имеется веб версия) для дизайнеров и программистов, которая обеспечивает гибкое и удобную работу над проектами. Основными особенностями Figma являются возможность создания проектов в реальном времени, коллаборативный дизайн и интеграция с различными инструментами программирования. Платформа позволяет дизайнерам и программистам работать над проектами одновременно, видеть изменения мгновенно и легко комментировать их друг у друга. Это значительно упрощает процесс совместной работы и ускоряет разработку проектов.

Также Figma предоставляет широкий набор инструментов для создания дизайна и разработки интерфейсов, включая возможность создания прототипов, анимации и визуальных эффектов. Интеграция с популярными инструментами программирования, такими как HTML, CSS и JavaScript, делает Figma удобным и эффективным инструментом для разработчиков.

Кроме того, Figma позволяет создавать много платформенные приложения, которые могут быть просмотрены и отредактированы на любом устройстве с помощью веб-браузера. Это делает работу над проектами еще более удобной и доступной для всех участников команды. К тому же, Figma способна создавать предпросмотр макета сайта в зависимости от настроек, так же он способен создавать анимации, для продумывания функционала ещё на этапе проектирования макета. Именно поэтому я выбрал Figma для создания и настройки дизайна конечного продукта, а не другие приложения .

### 2.1.2 Практическая реализация алгоритмов

Алгоритм — это точное предписание, описывающее порядок действий (план действий), определяющих процесс решения задачи или достижения цели.

Открытие диалоговых окон – диалоговые окна создаются тэгом html <dialog></dialog> и работают следующим образом. Через JS-код, допустим, при нажатии кнопки или при выполнении определенного условия, диалоговому окну присваивается метод .showModal() и диалоговое окно открывается. Для начала было создано множество кнопок с одинаковыми классами, но каждый с уникальным идентификатором для того, чтобы через нажатие на новость открывалась конкретное модальное окно. После обращалось к массиву объектов, хранящему информацию о каждом модальном окне. Данная система работает таким образом, что при прогрузке DOM-элементов присутствует функция, которая получает в Java Script получает HTML элементы, в последующем каждой кнопке из этого файла языка гипертекстовой разметки присваивается обработчик событий addEventListener. У addEventListener присутствует атрибут event, благодаря которому происходит открытие модального окна. Изначально контейнер модального окна имеет стиль в CSS display: none, то есть элемент изначально не отображается на странице. При нажатии на кнопку, через event мы обращаемся к id контейнера модального окна и при помощи Java Script мы изменяем видимость контейнера модального окна через .showModal(). При использовании данного метода нужное нам модальное окно проявляется на странице. После того, как нужное нам модальное окно отображено, в коде присутствует скрипт, который обращается к массиву объектов и достает оттуда информацию, после заполняет ее в специально отведенные поля в теге dialog. В конце всех этих функций реализована еще одна - при нажатии на кнопку, заданную в теге dialog сначала удаляет class и id у которых стоял display не равный значению none. Сразу же после удаления class’ов и id закрывает модальное окно методом .close(). Также присутствует код, который создает эффект блюра на заднем фоне страницы, после закрытия данный эффект пропадает.

Код можно просмотреть в Приложении В.2.

Практическая реализация функционала системы регистрации – система регистрации работает через форму с полями для ввода данных. Присутствует поля: логин, пароль, email, номер телефона, также есть кнопка зарегистрироваться и “Уже есть аккаунт?”. Также присутствует обработчик событий, реализованный через конструкцию else if. Обработчик событий посылает уведомление (тэг <dialog></dialog> в HTML) через диалоговое окно о том, что либо вы не ввели данные в указанное поле, либо ввели недостаточно данных. При успешной регистрации также будет выведено уведомление, затем добавлены данные в массив пользователей, затем отправлены в localStorage путем методов JSON.stringify (преобразование данных в JSON формат) и методов локального хранилища setItem(key, value (поле value обязательно должно быть в JSON формате)) . После регистрации нужно будет авторизоваться. В авторизации всего 2 поля: логин, пароль и кнопка войти. При вводе данных и нажатия на кнопку войти, данные из полей сравниваются с данными в массиве (путем forEach JS – метода переборки массива и условием if (условие) {если true}) с зарегистрированными пользователями. После успешной авторизации, в локальное хранилище через метод setItem() создается элемент с информацией о конкретном аккаунте пользователя, в который вы вошли и элемент auth = true. При auth = true вам доступен весь функционал сайта. Если данные совпали, вам остается закрыть модальное окно(закрывается только по нажатию на клавишу ESC на клавиатуре). База данных представляет собой массив с объектами.

Код можно просмотреть в Приложении В.1.

Реализация функционала автослайдера – для начала создается переменная с значением 0. Это нужно для реализации перехода между последней картинкой и первой, а так же на данной переменной завязано основное расположение картинок. Далее создается функции, которая отвечает за само автоматическое перелистывание. Объявляются переменные самих изменяемых элементов. Также добавлено условие для реализации передвижения картинок. Реализация самого передвижения происходит через свойство translate(). Также вместе с картинками по такому же способу меняется отображающий блок. Сам блок реализован через стили class, который удаляется и добавляется в зависимости от того какая сейчас показывается картинка. При вызове функции присваивается метод .setInterval для установки времени через которое будет выполнятся функция.

Код в Приложении В.3.

Реализация функционала слайдера с текстом, работающим по нажатию – слайдер с текстом работает через addEventListener. Для начала заранее создали четыре объекта. В первом текст об авторе и песне или альбоме, во втором информация об этой песне или альбоме, в третьем объекте данных хранятся цвета в формате hex, которые после будут использоваться, как задний фон, а в четвертом путь ссылающий на картинку. После объявляются, все переменные с которыми мы будет работать(обращаться, изменять). Создаем переменную i, как отображение index значения элементов в объектах. Благодаря этой переменной i при загрузке страницы выводим первые элементы из объектов и присваиваем их значения. Присваиваем кнопке, отвечающую за переход на следующий блок информации, функцию которая прибавляет к переменной i единицу для обращения к следующему элементу объекта. После выводим эти элементы и присваиваем класс для всего блока, который будет отвечать за анимацию. Для появления картинки устанавливаем короткое время для создания интересного эффекта. И удаляем класс анимации. После создаем функцию для перехода назад. Для начала устанавливаем условие проверки на значение переменной i если она строго равна нулю. Это нужно нам для проверки, в случае удачного ответа выводит последние элементы и устанавливает значения равные им же. В данной функции i уже убавляется на единицу. За исключением этих изменений код остается таким же. Так как слайдера с текстом на данной странице два, справедливо отметить, что второй ничем не отличается от первого, за исключением названий переменных, которые обращаются к тегам в языке гипертекстовой разметке, данных в объекте

Код в Приложении В.4.

Реализация функции проигрывания музыкального плеера – музыкальный плеер одна из основных функций данного сайта. В самом начале объявляются переменные обращаются к кнопкам, к тегу audio(нужен для проигрывания аудио звука), к элементам содержащими текст , к элементам прогресс бара(показывает прогресс проигрывания произведения). Задаем переменную index, для отображения значения индекса объектов в массиве данных объектов. Создаем сам объекты в массиве с информацией, а именно название, артист, картинка, ссылка на аудио. Для отображения времени пройденного во время прослушивания нужно форматирование, с этой целью была создана функция, которая сначала обрабатывает минуты методом Math.floor(), после этого функция через использование тернарного оператора форматирует значение минут, если значение минуты меньше 10, то добавить ноль спереди (например, из числа 7 сделать строку "07"), таким же образом принимает и форматирует секунды и в самом конце возвращает “Минуты(00):Секунды(00)”. В следующей функции происходит обращение к объекту в массиве для работы с элементами данного объекта и присваивает их значения через свойства innerhtml и src. И сразу после создаем продолжительность отображения при загрузке метаданных, через событие onloadedmetadata в функции, которого присваиваем значение функции форматирования. Настройка кнопки проигрывания песни реализована, через событие по нажатию которое запускает композицию через метод .play(), удаляет class(“hide”)используемый для скрытия, кнопке паузы и присваивает данный класс себе. Создается переменная “loop” для работы с некоторыми кнопками. Кнопка повтора реализована через событие клика, и условие. Если кнопка имеет класс “active”, то удаляется этот класс, значение переменной ставит на false(ложь), и выводит в консоль – кнопка повтора отключена. Иначе создается класс активности, значение переменной устанавливается true(правда) и в консоль выводит – кнопка повтора работает. Кнопка следующей песни реализована, через условие проверки на работу кнопки случайной песни, если включена, то проверяется на последняя ли песня в списке, если да, то значение переменной индекса сбрасывается на 0, иначе значение индекса прибавляется на единицу, обращается к функции с работой с файлами хранилищем и запускает произведение композиции. Если же не включена, то поиск случайного индекса, так же обращение и получение файлов из хранилища и их воспроизведение. Кнопка паузы работает, через событие нажатия и останавливает воспроизведение, после чего удаляет класс “hide” у кнопки воспроизведения и присваивает его себе. Кнопка предыдущей песни, не предусматривает возможность работы со случайно включенными песнями. Она сначала проверяет условие, если значение индекс больше нуля, то остановка воспроизведения песни, после чего из значения индекса вычитается единица, иначе возвращает значение массива вычитая единицу и присваивает индексу. Запуск функции для получения данных из хранилища, и воспроизводит их. Переход музыкального произведения выполнен через простую функцию, которая ссылается на функцию следующей песни. Функционал кнопки перемешивания песен, создан с помощью события и условия. При нажатии запускается проверка на активность кнопки, если проверка пройдена успешна, то происходит удаление класса активности, переменной “loop” присваивается значение true(Правда), в консоль выводится оповещение об выключении данного функционала, в ином случае добавляется класс активности, переменной присваивается значение false(ложь), а в консоль выводится сообщение об успешной активации данной функции. Для обновления текущего времени проигрывания аудио и полосы прогресса используется setInterval внутри которого значение текущего времени воспроизведения аудиопотока обновляется и отображается в указанном элементе, после рассчитывается процент завершенности воспроизведения аудиопотока и устанавливается в качестве ширины элемента представляющего прогресс воспроизведения. Так же при загрузке страницы запускается функция устанавливающая значение индекса, и вызывающая функции обращения к данным в хранилище.

Код в Приложении В.5.

### 2.1.3 Реализация интерфейса

Интерфейс был реализован с использование стандартов UI/UX дизайна через веб-версию утилиты Figma. На сайте была выбрана следующая цветовая гамма.

Цвета заднего фона:

1. #383737;

Цвета заднего фона были выбраны специально таким образом, чтобы пользователю было комфортно находится на сайте продолжительное время. Темные цвета меньше давят на глаза. При этом на сайте в некоторых блоках используются картинки, как задний фон и определение основных и второстепенных цветов сайта.

Основные и второстепенные цвета сайта:

1. # 026873;
2. # 070C73;
3. # 730202;
4. # 012326;
5. #D56BF2;
6. #5CE1D4;

Цвета под номерами один и два является основными, цвет под третьим номером используется для некоторых блоков сайта для привлечения внимания. Изначально, основным цветом сайта был цвет под номером четыре (темно-зелёный), но мною было принято решение, что этот цвет слишком агрессивный для данной тематики сайта как основной. Поэтому как основные цвета были выбраны более мягкие – цвета под номером один (темно-салатовый) и (темно-синий), он является более приятным глазу и создает ощущение уюта. А цвет под номером четыре стал появляться в некоторых блоках для особого выделения и стал использоваться для некоторых кнопок. Цвета под номерами пять и шесть используются для отображения иконок. Под номером пять, иконки на основной части страниц, а под номером шесть только в footer.

Все блоки страниц являются строго структурированными, отступы с левой и с правой стороны сайта являются фиксированными. Для разных заголовков используется разные величины(px) в зависимости от их важности. По аналогии так же отредактирован простой текст.

## 2.2 Тестирование

Тест-кейс №1

|  |  |
| --- | --- |
| Название проекта | space music(космическая музыка) |
| Рабочая версия | 0.8 |
| Тестировщик | Фёдор Бурлаков |
| Дата теста | 03.04.2024 |
| ID теста | TC\_UI\_1 |
| Приоритет | Высокий |
| Название теста | Проверка работоспособности функционала регистрации. |
| Краткое изложение теста | Проверка работоспособности функционала регистрации. |
| Этапы теста | 1. На главной странице сайта нажать на кнопку “Профиль”. 2. В открывшемся модальном окне ввести данные во все поля и выбрать значение из списка. 3. Нажать на кнопку “Регистрация”. 4. Высвечивание сообщения об успешной регистрации. |
| Тестовые данные | Логин – burlak87  Пароль – burlakov87  Email – [burlakovfedor87@gmail.com](mailto:burlakovfedor87@gmail.com)  Телефон – 89517259202 |
| Ожидаемый результат | Аккаунт будет зарегистрирован успешно |
| Фактический результат | Аккаунт зарегистрирован в системе |
| Статус | Зачет |

Тест-кейс №2

|  |  |
| --- | --- |
| Название проекта | space music(космическая музыка) |
| Рабочая версия | 0.8 |
| Тестировщик | Фёдор Бурлаков |
| Дата теста | 05.04.2024 |
| ID теста | TC\_UI\_2 |
| Приоритет | Высокий |
| Название теста | Проверка работоспособности функционала регистрации. |
| Краткое изложение теста | Проверка работоспособности функционала регистрации. |
| Этапы теста | 1. На главной странице сайта нажать на кнопку “Профиль”. 2. Нажать кнопку “Уже есть аккаунт”. 3. В открывшемся модальном окне ввести данные во все поля и выбрать значение из списка. 4. Нажать на кнопку “Авторизация”. 5. Высвечивание сообщения об успешной Авторизации. 6. Закрыть окно по нажатию клавиши “ESC”. |
| Тестовые данные | Логин – burlak87  Пароль – burlakov87  Email – [burlakovfedor87@gmail.com](mailto:burlakovfedor87@gmail.com)  Телефон – 89517259202 |
| Ожидаемый результат | Аккаунт будет авторизован успешно |
| Фактический результат | Аккаунт авторизован в системе |
| Статус | Зачет |

Тест-кейс №3

|  |  |
| --- | --- |
| Название проекта | space music(космическая музыка) |
| Рабочая версия | 0.8 |
| Тестировщик | Фёдор Бурлаков |
| Дата теста | 05.04.2024 |
| ID теста | TC\_UI\_3 |
| Приоритет | Средний |
| Название теста | Проверка работоспособности функционала оповещений об ошибках при регистрации. |
| Краткое изложение теста | Проверка функционала оповещений об ошибках вводимых данных при регистрации по нажатию кнопки. |
| Этапы теста | 1. На главной странице сайта нажать на кнопку “Профиль”. 2. Не заполнение полей. 3. Вписывать значения в меньше четырех символов. 4. Вписать значения меньше шести, но больше четырех символов. 5. Вписать значения меньше восьми, но больше шести символов. 6. Вписать значение меньше десяти, но при этом больше восьми символов. 7. Вписать число, которое будет больше всех проверяемых выше значений. 8. В открывшемся модальном окне ввести данные во все поля и выбрать значение из списка. 9. Нажать на кнопку “Регистрация”. 10. Высвечивание сообщения об успешной ошибки и ее расположении. 11. Закрыть окно по нажатию клавиши “ESC”. |
| Тестовые данные | Пустая строка  Строка номер один - bur  Строка номер два - burla  Строка номер три - burlakv  Строка номер четыре - burlakoff  Строка номер пять - 777777777777 |
| Ожидаемый результат | В случае пустого поля – будет выведено окно с текстом данное (ссылается на не заполненное поле) поле не заполненно  В случая с нехваткой символов – будет выведено окно с текстом Поле “Данное (там где не хватает)” меньше n(кол-во проверяемых символов) символов. |
| Фактический результат | В случае пустого поля – будет выведено окно с текстом данное (ссылается на не заполненное поле) поле не заполненно  В случая с нехваткой символов – будет выведено окно с текстом Поле “Данное (там где не хватает)” меньше n (кол-во проверяемых символов) символов. |
| Статус | Зачет |

Тест-кейс №4

|  |  |
| --- | --- |
| Название проекта | space music(космическая музыка) |
| Рабочая версия | 0.8 |
| Тестировщик | Фёдор Бурлаков |
| Дата теста | 09.04.2024 |
| ID теста | TC\_UI\_4 |
| Приоритет | Низкий |
| Название теста | Проверка работоспособности открытия модальных окон. |
| Краткое изложение теста | Проверка функционала открывания модальных окон, также их закрывания эффект блюра и появляющейся информации из хранилища. |
| Этапы теста | 1. Перейти на страницу “Магазин”. 2. Пролистать до секции с блоками товаров 3. Нажать на кнопку с указанным товаром 4. Проверка блюра 5. Проверка закрытия модального окна на клавишу “Escape”. 6. Закрытие на специально созданную кнопку. |
| Тестовые данные |  |
| Ожидаемый результат | При нажатии на кнопку нужную тестировщику, открывается модальное окно. Задний фон размывается, появляются стили компонентов, при закрытии через клавишу “Escape” окно закрывается также как и при нажатии на отдельную кнопку. |
| Фактический результат | При нажатии на кнопку нужную тестировщику, открывается модальное окно. Задний фон размывается, появляются стили компонентов, при закрытии через клавишу “Escape” окно закрывается частично, основная часть стилей уходит на первый блок. При закрытии по кнопке такого не происхрдит |
| Статус | Зачет |

Тест-кейс №5

|  |  |
| --- | --- |
| Название проекта | space music(космическая музыка) |
| Рабочая версия | 0.8 |
| Тестировщик | Фёдор Бурлаков |
| Дата теста | 30.04.2024 |
| ID теста | TC\_UI\_5 |
| Приоритет | Высокий |
| Название теста | Проверка автослайдера на изменения |
| Краткое изложение теста | Проверка автослайдера на изменения при разных величинах перелистывания картинки |
| Этапы теста | 1. В коде js надо изменить величину на большую заявленной изначально 2. В коде js надо изменить величину на меньшую заявленной изначально 3. Проверить на странице |
| Тестовые данные | Величина один – 1200px  Величина два - 300px |
| Ожидаемый результат | Нерабочий автослайдер |
| Фактический результат | Частично-корректно выполненная работа автослайдера, а именно при величине один, одна картинка теряется. А при величине два картинка не доезжает. |
| Статус | Зачет |

Тест-кейс №6

|  |  |
| --- | --- |
| Название проекта | space music(космическая музыка) |
| Рабочая версия | 0.8 |
| Тестировщик | Фёдор Бурлаков |
| Дата теста | 30.04.2024 |
| ID теста | TC\_UI\_6 |
| Приоритет | Высокий |
| Название теста | Проверка слайдера с текстом на изменения |
| Краткое изложение теста | Проверка слайдера с текстом на изменения при разных условиях |
| Этапы теста | 1. В коде js надо изменить данные хранилища на числа 2. Проверка вывода числовых данных 3. Отключение анимаций 4. Проверка работы без анимаций |
| Тестовые данные | Данные числового типа: 12345, 098675433. |
| Ожидаемый результат | Нерабочий слайдер с текстом |
| Фактический результат | Данные числового типа выводятся корректно, а вот код без анимации производит резкую и рванную работу, но не перестает функционировать |
| Статус | Зачет |

# Заключение

В результате выполнения данной курсовой работы были улучшены знания в сфере изучения языков программирования и проектирования информационных систем. Все функции успешно удалось реализовать, готова первая версия новостного сайта space-music.

Проект выполнен без использования фреймворков, сторонних библиотек и баз данных. Все выполнено на чистом Java Script для приобретения новых знаний в рамках этого языка. Получен новый опыт работы в IDE(Visual Studio Code) при помощи разного рода расширений используемых в языке гипертекстовой разметки(html), каскадных таблицах стилях(css) и самом js, что позволило создать гораздо понятливый и красивый интерфейс.

В дальнейшем планируется развитие и расширение функционала сайта, включая добавление личного аккаунта пользователя и композитора, улучшение алгоритма проигрывания произведения музыкального характера, создание алгоритма добавление в избранное некоторые песни, возможность проигрывания видео форматов для простота клипов и записей концертов реализация возможности композитором добавления, изменения и удаления песни или видео в базе данных и другие полезные возможности.

# Список литературы

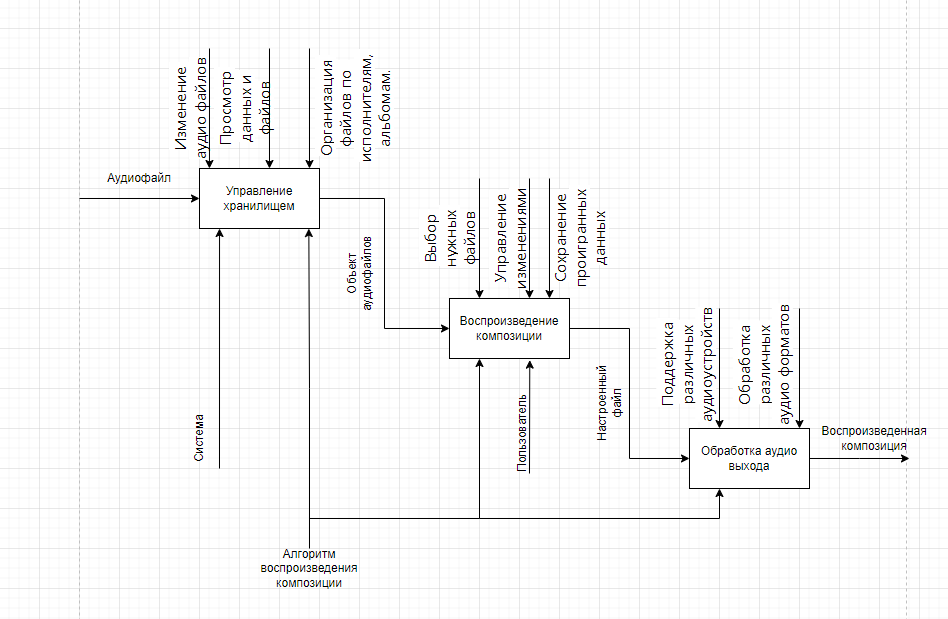
1. <https://lyceum.urfu.ru/fileadmin/user_upload/docs/GOST_7.0.97-2016.pdf> - Система стандартов по информации, библиотечному и издательскому делу ОРГАНИЗАЦИОННО-РАСПОРЯДИТЕЛЬНАЯ ДОКУМЕНТАЦИЯ Требования к оформлению документов
2. <https://www.hse.ru/data/2018/11/28/1144395257/ГОСТы%20и%20ЕСПД%202018-2019.pdf> - Единая Система Программной Документации (ЕСПД) 2018-2019 г.в.
3. <https://learn.javascript.ru/> - современная документация для Java Script
4. <https://developer.mozilla.org/ru/> - профессиональная документация для разработчиков. Использовалось для документации по JS.
5. <https://devdocs.io/> - справочник документаций по разным языкам программирования, фреймворкам и библиотекам.
6. <https://htmlacademy.ru/blog/html-tags/book> - справочник по HTML-тэгам.
7. <https://htmlbook.ru/css> - CSS справочник
8. <https://vc.ru/design/184941-figma-vse-chto-vam-nuzhno-znat> - инструкция по пользованию Figma
9. <https://code.visualstudio.com/docs> - документация по использованию VScode.
10. <https://www.microsoft.com/en-us/Useterms/Retail/Windows/10/UseTerms_Retail_Windows_10_Russian.htm> - Windows 10 лицензионное соглашение.
11. <https://elar.rsvpu.ru/bitstream/123456789/32285/1/Narvatkina_Teknol_postr_buis_proc_2020.pdf> - технология построения бизнес-процессов.
12. “Web на практике CSS, HTML, JavaScript, MySQL, PHP для fullstack-разработчиков” Дубровский Е.В., – книга использовалась для более углубленного понимания написания алгоритов(Год выпуска 2021).
13. “Чистый код” Роберт Мартин – книга читалась для обучения написания чистому коду во время работы над курсовой(Год выпуска 2023).

# Приложение А

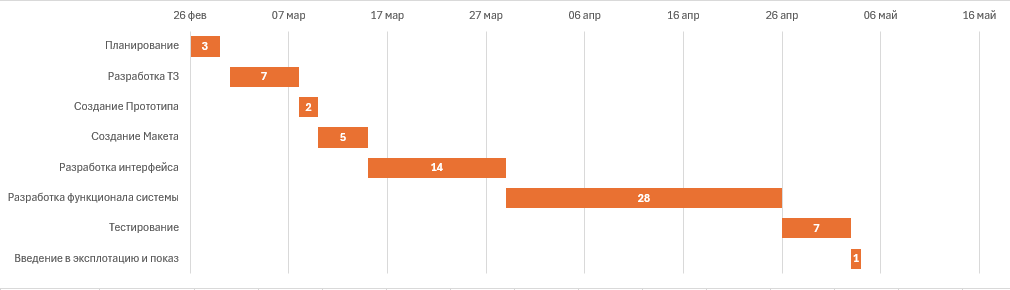
Изображение выглядит как текст, диаграмма, снимок экрана, линия

Автоматически созданное описание

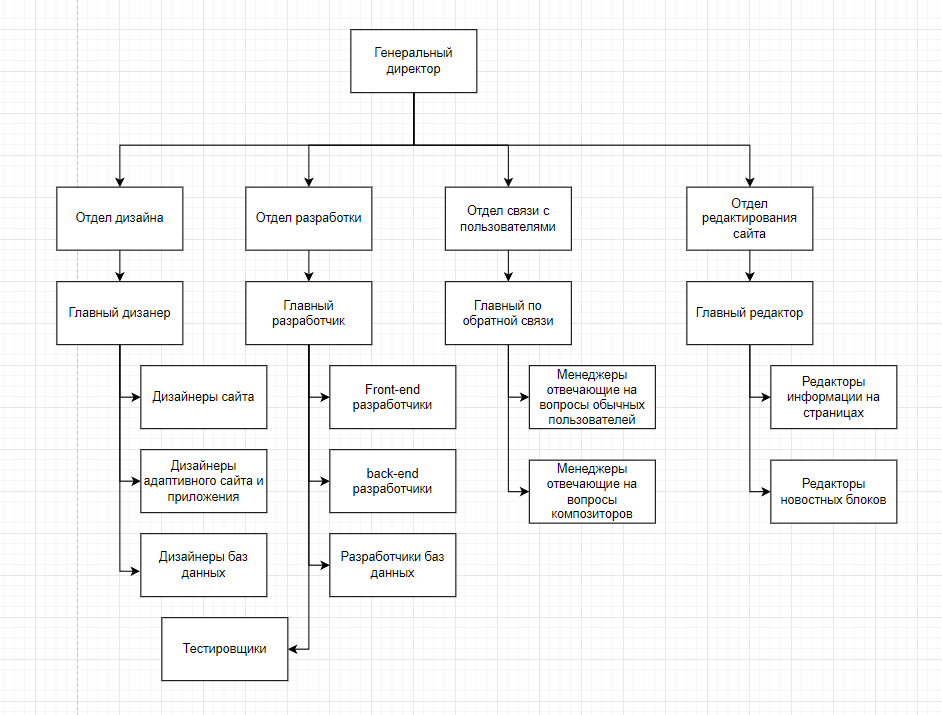
Изображение А.1. idef0-диаграмма



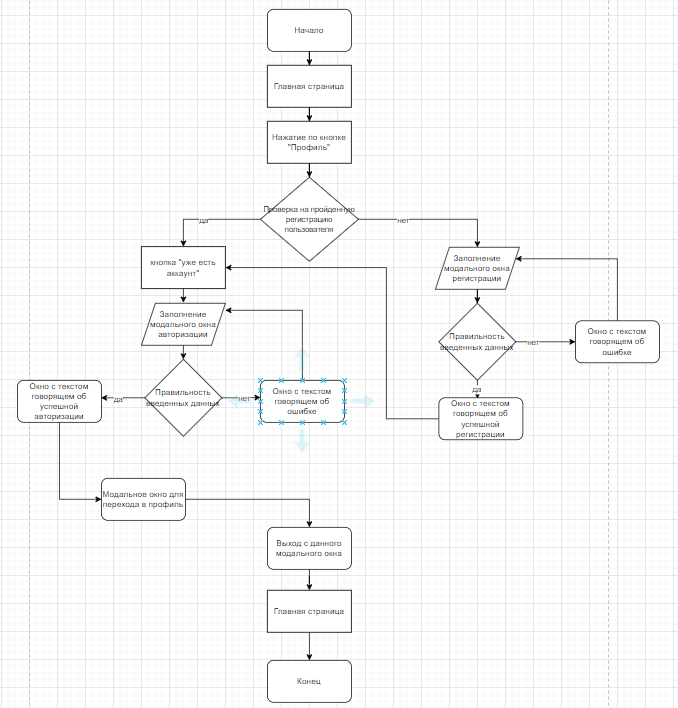
Изображение А.2. декомпозиция диаграммы-idef0



Изображение А.3. Диаграмма Ганта

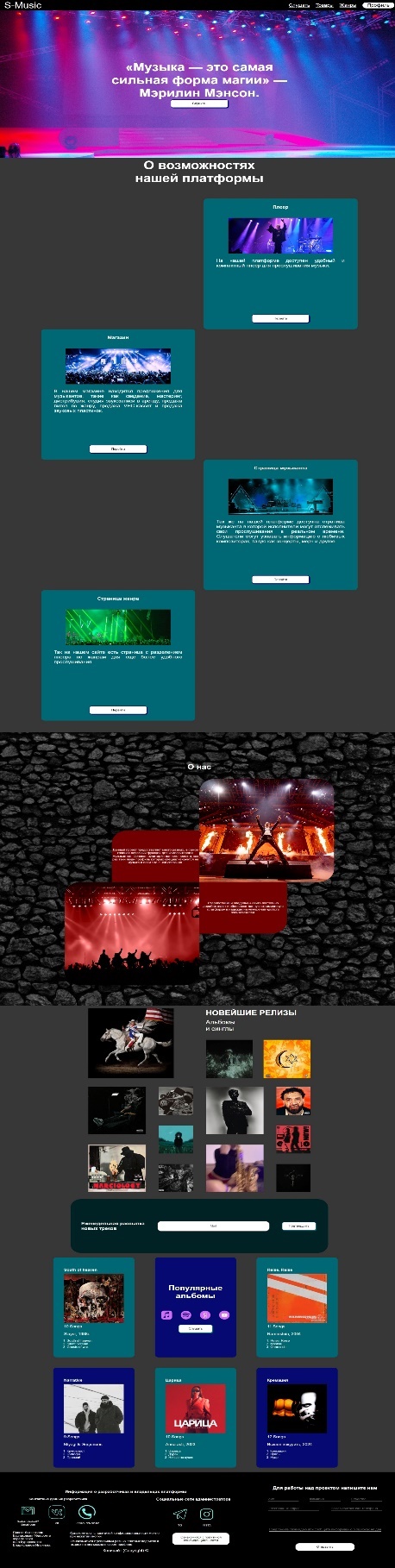


Изображение А.4. Организационная диаграмма



Изображение А.5. Блок-схема Регистрации

# Приложение Б



Изображение Б.1. Главная страницаИзображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Изображение Б.2. Страница музыкального плеера

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Автоматически созданное описание

Изображение Б.3. Страница магазина



Изображение Б.4. Страница с политикой конфидинциальности

Изображение Б.5. Ссылка на figma: https://www.figma.com/file/9WLFJU2l5TrEKzwtjxEAZE/Untitled?type=design&node-id=78-168&mode=design&t=6o1NrtbruKx1cOLT-0

# Приложение В

// новый тег <script> и его расположение

function scriptInclude() {

    const js = document.createElement("script");

    js.type = "text/javascript";

    js.src = "../script.js";

    document.body.appendChild(js);

}

// добавление модальных окон с данными из локального хранилища

document.getElementById('regButtonModal').addEventListener('click', () => {

    let modalAppSection = document.querySelector('.modalAppSection')

    let modalAppContent\_article = document.querySelector('.modalAppContent')

    if (modalAppSection.style.display === "block") {

        modalAppSection.style.display = "none"

        modalAppContent\_article.style.display = "none"

        document.body.style.overflow = "unset"

    } else {

        modalAppSection.style.display = "block"

        modalAppContent\_article.style.display = "flex"

        document.body.style.overflow = "hidden"

    }

})

// скрытие блока

document.getElementById("selectorGroup").addEventListener("click", () => {

    if(document.getElementById("optinBase").style.display = "block") {

        document.getElementById("optinBase").style.display = "none"

    }

})

// пустой массив для дальнейшей работы

let users = []

// запись данных из хранилища в массив

if (localStorage.getItem("users")) {users = JSON.parse(localStorage.getItem("users"))}

document.forms.regForm.submit.addEventListener("click", (event) => {

    // предотвращение стандартного действия браузера

    event.preventDefault()

    // Получение значений из инпутов

    let login = document.forms.regForm.login

    let password = document.forms.regForm.password

    let group = document.forms.regForm.group

    let email = document.forms.regForm.email

    let numberPhone = document.forms.regForm.numberPhone

    // Создание пустых переменных

    let errText = null

    let error = null

    // Функция проверки длины и заполненности полей

    // При несоответсвии выводит ошибку через errText

    // В ином случае error = folse

    function err() {

        if (login.value.length === 0) {

            return errText = `Ошибка: поле "${login.placeholder}" не заполнено`

        } else if (login.value.length < 4) {

            return errText = `Ошибка: поле "${login.placeholder}" меньше 4 символов`

        } else if (password.value.length === 0) {

            return errText = `Ошибка: поле "${password.placeholder}" не заполнено`

        } else if (password.value.length < 6) {

            return errText = `Ошибка: поле "${password.placeholder}" меньше 6 символов`

        } else if (email.value.length === 0) {

            return errText = `Ошибка: поле "${email.placeholder}" не заполнено`

        } else if (email.value.length < 8) {

            return errText = `Ошибка: поле "${email.placeholder}" меньше 8 символов`

        } else if (numberPhone.value.length === 0) {

            return errText = `Ошибка: поле "${numberPhone.placeholder}" не заполнено`

        } else if (numberPhone.value.length < 10) {

            return errText = `Ошибка: поле "${numberPhone.placeholder}" меньше 10 символов`

        } else {error = false}

    }

    err()

    // Проверка длины значений всех полей(инпутов)

    if (login.value.length < 4 || password.value.length < 6 || group.value.length < 4 || email.value.length < 8 || numberPhone.value.length < 10) {

        // вставка текста в элемент

        document.getElementById('loginErrorEmpty').textContent = errText

        if (!document.getElementById('loginErrorEmpty').hasAttribute("open")) {

            // если мало и нет атрибута опен

            return document.getElementById('loginErrorEmpty').toggleAttribute("open")

        }

    } else {

        // если подходит

        document.getElementById('loginErrorEmpty').removeAttribute("open")

    }

    // вслучае коректной работы всего выше

    if (error === false) {

        // добавление в массив

        users.push({

            login: login.value,

            password: password.value,

            group: group.value,

            email: email.value,

            phone: numberPhone.value

        })

        // сохранение в хранилище

        localStorage.setItem("users", JSON.stringify(users))

        // по аналогии с ошибкочной сделана удачная регистрация

        errText = "Регистрация произведена успешно"

        document.getElementById('loginErrorEmpty').style.background = "green"

        document.getElementById('loginErrorEmpty').textContent = errText

        document.getElementById('loginErrorEmpty').toggleAttribute("open")

        // поле отображения успешной регистрации закрывается через время

        setTimeout(() => {

            document.getElementById('loginErrorEmpty').removeAttribute("open");

            document.getElementById('loginErrorEmpty').style.background = "darkred"

        }, 2000)

    } else {

        // иная ситуация завершения кода

        return

    }

})

// вызов функции после загрузки страницы и вызывается функция

document.addEventListener('DOMContentLoaded', () => {

    scriptInclude()

    // скрытие блока

    document.getElementById('authForm').style.display = "none"

    // проверка элемента в хранилище

    if (JSON.parse(localStorage.getItem("auth")) === true) {

        document.getElementById('authForm').style.display = "none"

        document.getElementById('regForm').style.display = 'none'

        document.getElementById('regAccordeonH1').textContent = "Личный кабинет"

        document.getElementById('authButton').textContent = ''

    }

})

document.getElementById('authButton').addEventListener('click', (event) => {

    // отмена отправки формы

    event.preventDefault()

    // при скрытии regForm такие измения

    if (document.getElementById('regForm').style.display === 'none') {

        document.getElementById('authButton').textContent = 'Уже есть аккаунт?'

        document.getElementById('regForm').style.display = 'flex'

        document.getElementById('regAccordeonH1').textContent = "Регистрация"

        document.getElementById('authForm').style.display = "none"

    } else {

    // Иначе такие

        document.getElementById('authButton').textContent = 'Регистрация'

        document.getElementById('regForm').style.display = 'none'

        document.getElementById('regAccordeonH1').textContent = "Авторизация"

        document.getElementById('authForm').style.display = "flex"

    }

})

// 3

document.forms.authForm.submit.addEventListener('click', (event) => {

    // отмена отпраки формы

    event.preventDefault()

    // достаем значения users из локального хранилища

    let users = JSON.parse(localStorage.getItem("users"))

    // получение значений

    let loginAuth = document.forms.authForm.login

    let passwordAuth = document.forms.authForm.password

    let account = null

    // переборка значений для проверки наличия пользователя с такими данными

    users.forEach((el) => {

        if (el.login === loginAuth.value) {account = el}

    })

    // если значения логина одинаковые происходит проверка пароля

    if (loginAuth.value === account.login && passwordAuth.value === account.password) {

        // если успешно тто вывод уведомления

        alert('Вход выполнен успешно')

        // установка значений в хранилище

        localStorage.setItem('auth', true)

        localStorage.setItem('currentUser', JSON.stringify(account))

        // скрытие элементов

        document.getElementById('authForm').style.display = "none"

        document.getElementById('regForm').style.display = 'none'

        document.getElementById('regAccordeonH1').textContent = "Личный кабинет"

        document.getElementById('authButton').textContent = ''

    } else {

        alert('Не правильные данные')

    }

})

Приложение В.1. Код регистрации на js

let arr = new Array();

arr = [

    one = {

        title: 'Звуковые пластинки',

        paragraph: 'Опция даанной покупки временно преостановленна'

    },

    two = {

        title: 'VHC-Кассеты',

        paragraph: 'Опция даанной покупки временно преостановленна'

    },

    three = {

        title: 'Все биты',

        paragraph: 'Биты разных жанров можно приобрести у пяти наших производителей',

        spanOne: 'NO Beats',

        linkOne: 'https://vk.com/no\_beatz',

        spanTwo: 'KANJO',

        linkTwo: 'https://vk.com/kanjoprod',

        spanThree: 'Je Sai',

        linkThree: 'https://vk.com/jesaibeat',

        spanFour: 'SYFER',

        linkFour: 'https://vk.com/syferprod',

        spanFive: 'kenezis',

        linkFive: 'https://vk.com/kenezis27'

    },

    four = {

        title: 'Сведение или Мастеринг',

        paragraph: 'Сведение или Мастеринг можно заказать у одного из трех наших специалистов',

        spanOne: 'Юра Фауст',

        linkOne: 'https://vk.com/yurafaustgroup',

        spanTwo: 'UnknMusic',

        linkTwo: 'https://vk.com/unknmusiclabel',

        spanThree: 'PRINZ CORD',

        linkThree: 'https://vk.com/prinzcord'

    },

    five = {

        title: 'Дистрибуция трека',

        paragraph: 'За дистрибуцией трека вы можете обратиться к лейблу',

        spanOne: 'FUJIPROD',

        linkOne: 'https://vk.com/fujiprod'

    },

    six = {

        title: 'Дистрибуция альбома',

        paragraph: 'За дистрибуцией альбома вы можете обратиться к данному лейблу',

        spanOne: 'DistrCompFl',

        linkOne: 'https://vk.com/distributiontracks'

    },

    seven = {

        title: 'Студия звукозаписи в',

        paragraph: 'Данный вид товара рекламирует чужие студии звукозаписи в разных городах, наша платформа не поддерживает покупку данного товара',

        paragraphTwo:  'находится по адресу',

    },

    eight = {

        nameOne: 'SoundRise Records',

        cityOne: 'Москва',

        addressOne: 'малый кисельный переулок 7/10',

        nameOne: '88 Records',

        cityTwo: 'Санкт-Петербург',

        addressTwo: 'кадетская линия 5 корп 2д 88',

        nameOne: 'Город. А',

        cityThree: 'Сочи',

        addressThree: 'донская улица 3',

        nameOne: 'Showtime Records',

        cityFour: 'Уфа',

        addressFour: 'улица юрия гагарина 41/2'

    }

];

// Переменнные для нахождения

let dialog = document.getElementById("dialog");

let h = document.getElementById("h");

let pOne = document.getElementById("pOne");

let pTwo = document.getElementById("pTwo");

let aOne = document.getElementById("aOne");

let aTwo = document.getElementById("aTwo");

let aThree = document.getElementById("aThree");

let aFour = document.getElementById("aFour");

let aFive = document.getElementById("aFive");

// Открытие модальных окон

let btnOne = document.getElementById("btnOne").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${one.title}`;

    pOne.innerText = `${one.paragraph}`;

});

let btnTwo = document.getElementById("btnTwo").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${two.title}`;

    pOne.innerText = `${two.paragraph}`;

});

let btnThree = document.getElementById("btnThree").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${three.title}`;

    pOne.innerText = `${three.paragraph}`;

    aOne.setAttribute('href', '`${three.linkOne}`');

    aOne.innerText = `${three.spanOne}`;

    aTwo.setAttribute('href', '`${three.linkTwo}`');

    aTwo.innerText = `${three.spanTwo}`;

    aThree.setAttribute('href', '`${three.linkThree}`');

    aThree.innerText = `${three.spanThree}`;

    aFour.setAttribute('href', '`${three.linkFour}`');

    aFour.innerText = `${three.spanFour}`;

    aFive.setAttribute('href', '`${three.linkFive}`');

    aFive.innerText = `${three.spanFive}`;

});

let btnFour = document.getElementById("btnFour").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${three.title}`;

    pOne.innerText = `${three.paragraph}`;

    aOne.setAttribute('href', '`${three.linkOne}`');

    aOne.innerText = `${three.spanOne}`;

    aTwo.setAttribute('href', '`${three.linkTwo}`');

    aTwo.innerText = `${three.spanTwo}`;

    aThree.setAttribute('href', '`${three.linkThree}`');

    aThree.innerText = `${three.spanThree}`;

    aFour.setAttribute('href', '`${three.linkFour}`');

    aFour.innerText = `${three.spanFour}`;

    aFive.setAttribute('href', '`${three.linkFive}`');

    aFive.innerText = `${three.spanFive}`;

});

let btnFive = document.getElementById("btnFive").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${three.title}`;

    pOne.innerText = `${three.paragraph}`;

    aOne.setAttribute('href', '`${three.linkOne}`');

    aOne.innerText = `${three.spanOne}`;

    aTwo.setAttribute('href', '`${three.linkTwo}`');

    aTwo.innerText = `${three.spanTwo}`;

    aThree.setAttribute('href', '`${three.linkThree}`');

    aThree.innerText = `${three.spanThree}`;

    aFour.setAttribute('href', '`${three.linkFour}`');

    aFour.innerText = `${three.spanFour}`;

    aFive.setAttribute('href', '`${three.linkFive}`');

    aFive.innerText = `${three.spanFive}`;

});

let btnSix = document.getElementById("btnSix").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${three.title}`;

    pOne.innerText = `${three.paragraph}`;

    aOne.setAttribute('href', '`${three.linkOne}`');

    aOne.innerText = `${three.spanOne}`;

    aTwo.setAttribute('href', '`${three.linkTwo}`');

    aTwo.innerText = `${three.spanTwo}`;

    aThree.setAttribute('href', '`${three.linkThree}`');

    aThree.innerText = `${three.spanThree}`;

    aFour.setAttribute('href', '`${three.linkFour}`');

    aFour.innerText = `${three.spanFour}`;

    aFive.setAttribute('href', '`${three.linkFive}`');

    aFive.innerText = `${three.spanFive}`;

});

let btnSeven = document.getElementById("btnSeven").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${three.title}`;

    pOne.innerText = `${three.paragraph}`;

    aOne.setAttribute('href', '`${three.linkOne}`');

    aOne.innerText = `${three.spanOne}`;

    aTwo.setAttribute('href', '`${three.linkTwo}`');

    aTwo.innerText = `${three.spanTwo}`;

    aThree.setAttribute('href', '`${three.linkThree}`');

    aThree.innerText = `${three.spanThree}`;

    aFour.setAttribute('href', '`${three.linkFour}`');

    aFour.innerText = `${three.spanFour}`;

    aFive.setAttribute('href', '`${three.linkFive}`');

    aFive.innerText = `${three.spanFive}`;

});

let btnEight = document.getElementById("btnEight").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${three.title}`;

    pOne.innerText = `${three.paragraph}`;

    aOne.setAttribute('href', '`${three.linkOne}`');

    aOne.innerText = `${three.spanOne}`;

    aTwo.setAttribute('href', '`${three.linkTwo}`');

    aTwo.innerText = `${three.spanTwo}`;

    aThree.setAttribute('href', '`${three.linkThree}`');

    aThree.innerText = `${three.spanThree}`;

    aFour.setAttribute('href', '`${three.linkFour}`');

    aFour.innerText = `${three.spanFour}`;

    aFive.setAttribute('href', '`${three.linkFive}`');

    aFive.innerText = `${three.spanFive}`;

});

let btnNine = document.getElementById("btnNine").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${three.title}`;

    pOne.innerText = `${three.paragraph}`;

    aOne.setAttribute('href', '`${three.linkOne}`');

    aOne.innerText = `${three.spanOne}`;

    aTwo.setAttribute('href', '`${three.linkTwo}`');

    aTwo.innerText = `${three.spanTwo}`;

    aThree.setAttribute('href', '`${three.linkThree}`');

    aThree.innerText = `${three.spanThree}`;

    aFour.setAttribute('href', '`${three.linkFour}`');

    aFour.innerText = `${three.spanFour}`;

    aFive.setAttribute('href', '`${three.linkFive}`');

    aFive.innerText = `${three.spanFive}`;

});

let btnTen = document.getElementById("btnTen").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${three.title}`;

    pOne.innerText = `${three.paragraph}`;

    aOne.setAttribute('href', '`${three.linkOne}`');

    aOne.innerText = `${three.spanOne}`;

    aTwo.setAttribute('href', '`${three.linkTwo}`');

    aTwo.innerText = `${three.spanTwo}`;

    aThree.setAttribute('href', '`${three.linkThree}`');

    aThree.innerText = `${three.spanThree}`;

    aFour.setAttribute('href', '`${three.linkFour}`');

    aFour.innerText = `${three.spanFour}`;

    aFive.setAttribute('href', '`${three.linkFive}`');

    aFive.innerText = `${three.spanFive}`;

});

let btnEleven = document.getElementById("btnEleven").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${four.title}`;

    pOne.innerText = `${four.paragraph}`;

    aOne.setAttribute('href', '`${four.linkOne}`');

    aOne.innerText = `${four.spanOne}`;

    aTwo.setAttribute('href', '`${four.linkTwo}`');

    aTwo.innerText = `${four.spanTwo}`;

    aThree.setAttribute('href', '`${four.linkThree}`');

    aThree.innerText = `${four.spanThree}`;

});

let btnTwelve = document.getElementById("btnTwelve").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${four.title}`;

    pOne.innerText = `${four.paragraph}`;

    aOne.setAttribute('href', '`${four.linkOne}`');

    aOne.innerText = `${four.spanOne}`;

    aTwo.setAttribute('href', '`${four.linkTwo}`');

    aTwo.innerText = `${four.spanTwo}`;

    aThree.setAttribute('href', '`${four.linkThree}`');

    aThree.innerText = `${four.spanThree}`;

});

let btnTbtnThirteenwo = document.getElementById("btnThirteen").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${four.title}`;

    pOne.innerText = `${four.paragraph}`;

    aOne.setAttribute('href', '`${four.linkOne}`');

    aOne.innerText = `${four.spanOne}`;

    aTwo.setAttribute('href', '`${four.linkTwo}`');

    aTwo.innerText = `${four.spanTwo}`;

    aThree.setAttribute('href', '`${four.linkThree}`');

    aThree.innerText = `${four.spanThree}`;

});

let btnFourteen = document.getElementById("btnFourteen").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${four.title}`;

    pOne.innerText = `${four.paragraph}`;

    aOne.setAttribute('href', '`${four.linkOne}`');

    aOne.innerText = `${four.spanOne}`;

    aTwo.setAttribute('href', '`${four.linkTwo}`');

    aTwo.innerText = `${four.spanTwo}`;

    aThree.setAttribute('href', '`${four.linkThree}`');

    aThree.innerText = `${four.spanThree}`;

});

let btnFifteen = document.getElementById("btnFifteen").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${four.title}`;

    pOne.innerText = `${four.paragraph}`;

    aOne.setAttribute('href', '`${four.linkOne}`');

    aOne.innerText = `${four.spanOne}`;

    aTwo.setAttribute('href', '`${four.linkTwo}`');

    aTwo.innerText = `${four.spanTwo}`;

    aThree.setAttribute('href', '`${four.linkThree}`');

    aThree.innerText = `${four.spanThree}`;

});

let btnSixteen = document.getElementById("btnSixteen").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${four.title}`;

    pOne.innerText = `${four.paragraph}`;

    aOne.setAttribute('href', '`${four.linkOne}`');

    aOne.innerText = `${four.spanOne}`;

    aTwo.setAttribute('href', '`${four.linkTwo}`');

    aTwo.innerText = `${four.spanTwo}`;

    aThree.setAttribute('href', '`${four.linkThree}`');

    aThree.innerText = `${four.spanThree}`;

});

let btnSeventeen = document.getElementById("btnSeventeen").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${five.title}`;

    pOne.innerText = `${five.paragraph}`;

    aOne.setAttribute('href', '`${five.linkOne}`');

    aOne.innerText = `${five.spanOne}`;

});

let btnEighteen = document.getElementById("btnEighteen").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${six.title}`;

    pOne.innerText = `${six.paragraph}`;

    aOne.setAttribute('href', '`${six.linkOne}`');

    aOne.innerText = `${six.spanOne}`;

});

let btnNineteen = document.getElementById("btnNineteen").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${seven.title}` + ' ' + `${eight.cityOne}`;

    pOne.innerText = `${seven.paragraph}`;

    pTwo.innerText = 'Студия ' + `${eight.nameOne}` + `${seven.paragraphTwo}` + ' - ' + `${eight.addressOne}`;

});

let btnTwenty = document.getElementById("btnTwenty").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${seven.title}` + ' ' + `${eight.cityTwo}`;

    pOne.innerText = `${seven.paragraph}`;

    pTwo.innerText = 'Студия ' + `${eight.nameTwo}` + `${seven.paragraphTwo}` + ' - ' + `${eight.addressTwo}`;

});

let btnTwentyOne = document.getElementById("btnTwentyOne").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${seven.title}` + ' ' + `${eight.cityThree}`;

    pOne.innerText = `${seven.paragraph}`;

    pTwo.innerText = 'Студия ' + `${eight.nameThree}` + `${seven.paragraphTwo}` + ' - ' + `${eight.addressThree}`;

});

let btnTwentyTwo = document.getElementById("btnTwentyTwo").addEventListener("click", () => {

    dialog.showModal();

    dialog.classList.add("dialog");

    h.innerText = `${seven.title}` + ' ' + `${eight.cityFour}`;

    pOne.innerText = `${seven.paragraph}`;

    pTwo.innerText = 'Студия ' + `${eight.nameFour}` + `${seven.paragraphTwo}` + ' - ' + `${eight.addressFour}`;

});

let close = document.getElementById("close").addEventListener("click", () => {

    dialog.classList.remove("dialog");

    dialog.close();

    h.innerText = '';

    pOne.innerText = '';

    pTwo.innerText = '';

    aOne.innerText = '';

    aTwo.innerText = '';

    aThree.innerText = '';

    aFour.innerText = '';

    aFive.innerText = '';

});

Приложение В.1. Код всплывающих окон с текстом.

let currentSlide = 0;

function Slider() {

  let image = document.querySelectorAll(".imgSwiperOne");

  let imagesList = document.querySelector("ul");

  if (image.length === 0) {

    imagesList.style.transform = `translateX(0px)`;

    return;

  };

  imagesList.style.transform = `translateX(-${currentSlide \* 600}px)`;

  let dots = document.querySelectorAll(".dot");

  dots.forEach((dot) => {

    dot.classList.remove("active");

  });

  dots[currentSlide].classList.add("active");

  if (currentSlide === image.length - 1) {

    currentSlide = 0;

  } else {

    currentSlide++;

  };

};

setInterval(() => {

  Slider();

}, 3000);

Приложение В.3. Код автослайдера

const text1\_options = [

  "Why art is so important",

  "Why you shouldn't buy the new iPhone",

  "Is life actually real?",

  "How to learn JS in 2 months"

];

const text2\_options = [

  "69 min. read",

  "7 min. read",

  "8 min. read",

  "87,658.1277 min. read"

];

const color\_options = ["#EBB9D2", "#FE9968", "#7FE0EB", "#6CE5B1"];

const image\_options = [

  "https://images.unsplash.com/photo-1524721696987-b9527df9e512?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=1190&q=80",

  "https://images.unsplash.com/photo-1556656793-08538906a9f8?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=1050&q=80",

  "https://images.unsplash.com/photo-1506073828772-2f85239b6d2d?ixlib=rb-1.2.1&auto=format&fit=crop&w=1189&q=80",

  "https://images.unsplash.com/photo-1523800503107-5bc3ba2a6f81?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=500&q=80"

];

// поменять на свои картинки и цвета и  текст

var i = 0;

const currentOptionText1 = document.getElementById("current-option-text1");

const currentOptionText2 = document.getElementById("current-option-text2");

const currentOptionImage = document.getElementById("image");

const carousel = document.getElementById("carouselText");

const mainMenu = document.getElementById("menu");

const optionPrevious = document.getElementById("previous-option");

const optionNext = document.getElementById("next-option");

currentOptionText1.innerText = text1\_options[i];

currentOptionText2.innerText = text2\_options[i];

currentOptionImage.style.backgroundImage = "url(" + image\_options[i] + ")";

carousel.style.background = color\_options[i];

optionNext.onclick = function () {

  i = i + 1;

  i = i % text1\_options.length;

  currentOptionText1.dataset.nextText = text1\_options[i];

  currentOptionText2.dataset.nextText = text2\_options[i];

  carousel.style.background = color\_options[i];

  carousel.classList.add("anim-next");

  setTimeout(() => {

    currentOptionImage.style.backgroundImage = "url(" + image\_options[i] + ")";

  }, 455);

  setTimeout(() => {

    currentOptionText1.innerText = text1\_options[i];

    currentOptionText2.innerText = text2\_options[i];

    carousel.classList.remove("anim-next");

  }, 650);

};

optionPrevious.onclick = function () {

  if (i === 0) {

    i = text1\_options.length;

  }

  i = i - 1;

  currentOptionText1.dataset.previousText = text1\_options[i];

  currentOptionText2.dataset.previousText = text2\_options[i];

  carousel.style.background = color\_options[i];

  carousel.classList.add("anim-previous");

  setTimeout(() => {

    currentOptionImage.style.backgroundImage = "url(" + image\_options[i] + ")";

  }, 455);

  setTimeout(() => {

    currentOptionText1.innerText = text1\_options[i];

    currentOptionText2.innerText = text2\_options[i];

    carousel.classList.remove("anim-previous");

  }, 650);

};

Приложение В.4. Код обеих слайдеров с текстом(отличаются только названием переменных)

const prevButton = document.getElementById("prev");

const nextButton = document.getElementById("next");

const repeatButton = document.getElementById("repeat");

const shuffleButton = document.getElementById("shuffle");

const pauseButton = document.getElementById("pause");

const playButton = document.getElementById("play");

const audio = document.getElementById("audio");

const songImage = document.getElementById("song-image");

const songName = document.getElementById("song-name");

const songArtist = document.getElementById("song-artist");

const maxDuration = document.getElementById("max-duration");

const currentTimeRef = document.getElementById("current-time");

const progressBar = document.getElementById("progress-bar");

const currentProgress = document.getElementById("current-progress");

//изначльное значение условий нажатия

let loop = true;

//индекс песни

let index;

window.onload = () => {

  //initially first song

  index = 0;

  setSong(index);

  //создание плелиста

  initializePlaylist();

};

const songsList = [

  {

    name: "Make Me Move",

    link: "assets/audio/Avenged Sevenfold - (O)rdinary.mp3",

    artist: "Culture Code",

    image: "make-me-move.jpg",

  },

  {

    name: "Where We Started",

    link: "assets/audio/Avenged Sevenfold - (D)eath.mp3",

    artist: "Lost Sky",

    image: "where-we-started.jpg",

  },

  {

    name: "On & On",

    link: "assets/audio/Currents - Beyond This Road.mp3",

    artist: "Cartoon",

    image: "on-on.jpg",

  },

  {

    name: "Throne",

    link: "throne.mp3",

    artist: "Rival",

    image: "throne.jpg",

  },

  {

    name: "Need You Now",

    link: "need-you-now.mp3",

    artist: "Venemy",

    image: "need-you-now.jpg",

  },

];

//Форматирование времени для отображения в time-container

const timeFormatter = (timeInput) => {

  let minute = Math.floor(timeInput / 60);

  minute = minute < 10 ? "0" + minute : minute;

  let second = Math.floor(timeInput % 60);

  second = second < 10 ? "0" + second : second;

  return `${minute}:${second}`;

};

//достаем песни из объекта

const setSong = (arrayIndex) => {

  //это извлекает все переменные из объекта

  let { name, link, artist, image } = songsList[arrayIndex];

  audio.src = link;

  songName.innerHTML = name;

  songArtist.innerHTML = artist;

  songImage.src = image;

  //продолжительность отображения при загрузке метаданных

  audio.onloadedmetadata = () => {

    maxDuration.innerText = timeFormatter(audio.duration);

  };

};

//кнопка проигрования песни

const playAudio = () => {

  audio.play();

  pauseButton.classList.remove("hide");

  playButton.classList.add("hide");

};

//реализация кнопки повтора

repeatButton.addEventListener("click", () => {

  if (repeatButton.classList.contains("active")) {

    repeatButton.classList.remove("active");

    audio.loop = false;

    console.log("repeat off");

  } else {

    repeatButton.classList.add("active");

    audio.loop = true;

    console.log("repeat on");

  }

});

//реализация кнопки следующей песни

const nextSong = () => {

  // если случайная песня

  if (loop) {

    if (index == songsList.length - 1) {

      //Если воспроизводится последняя песня

      index = 0;

    } else {

      index += 1;

    }

    setSong(index);

    playAudio();

  } else {

    //вдругом случае поиск случайного индекса песни и ее включение

    let randIndex = Math.floor(Math.random() \* songsList.length);

    console.log(randIndex);

    setSong(randIndex);

    playAudio();

  }

};

//реализация кнопки паузы

const pauseAudio = () => {

  audio.pause();

  pauseButton.classList.add("hide");

  playButton.classList.remove("hide");

};

//реализация кнопки предыдущей песни, не предусмотрена работа при случайных песнях

const previousSong = () => {

  if (index > 0) {

    pauseAudio();

    index -= 1;

  } else {

    //если играет первая песня

    index = songsList.length - 1;

  }

  setSong(index);

  playAudio();

};

//переход песни при окончании

audio.onended = () => {

  nextSong();

};

//кнопка перемешивания песен

shuffleButton.addEventListener("click", () => {

  if (shuffleButton.classList.contains("active")) {

    shuffleButton.classList.remove("active");

    loop = true;

    console.log("shuffle off");

  } else {

    shuffleButton.classList.add("active");

    loop = false;

    console.log("shuffle on");

  }

});

//кнопка проигрывания

playButton.addEventListener("click", playAudio);

//кнопка следующего трека

nextButton.addEventListener("click", nextSong);

//кнопка паузы

pauseButton.addEventListener("click", pauseAudio);

//кнопка предыдущего трека

prevButton.addEventListener("click", previousSong);

//обновлять отображение в progress bar каждую секунду

setInterval(() => {

  currentTimeRef.innerHTML = timeFormatter(audio.currentTime);

  currentProgress.style.width =

    (audio.currentTime / audio.duration.toFixed(3)) \* 100 + "%";

});

Приложение В.5. Код музыкального плеера.