Geekbrains

Специальность: Frontend-программист “Цифровые профессии”

Дипломным проект: “Создание веб-приложения, позволяющего искать музыку по автору и(или) наименованию в различных русскоязычных музыкальных интернет-магазинах с последующим сравнением полученных результатов по цене”

Служенко Дмитрий Сергеевич

Ульяновск, 2024 год.

**Дипломным проект:** “Создание веб-приложения, позволяющего искать музыку по автору и(или) наименованию в различных русскоязычных музыкальных интернет-магазинах с последующим сравнением полученных результатов по цене”

**Содержание**

[**Введение 3**](#_nphls6uol1x9)

[**Веб-приложение и способы получения данных из внешних источников Интернета. 7**](#_v3qsmk3x0i8h)

[Веб-приложение: основы. 7](#_lc1v10509ovy)

[Frontend веб-приложения, его особенности. 9](#_pvv1bhxq5d5p)

[Backend веб-приложения, его особенности. 11](#_e7ue02fk4iwe)

[Скрэпинг (web scraping) и парсинг (web parsing). 13](#_5d42f3bz885c)

[**Программные продукты и инструменты web-разработки 17**](#_wh3svdxva66h)

[Инструменты frontend части. 21](#_uv14nzm29ak3)

[HTML 21](#_ask17rz2eb6l)

[CSS 24](#_85cn8xheknga)

[Bootstrap 26](#_hvs0zx2f9isn)

[JavaScript и Vue.js 27](#_iizaui3yetpz)

[Axios 29](#_9r67b64hncoz)

[Инструменты backend части. 31](#_yubsinsxavno)

[Node.js 31](#_xe4biuodl2t0)

[Express 32](#_far0c8urf9m6)

[Инструменты скрэпинга / парсинга. 34](#_r5h6xe6eomus)

[Fetch 34](#_9sdvd2cja3ic)

[Puppeteer 34](#_xivi0ebstgly)

[Cheerio 35](#_y2qrqifgur2d)

[**Практическая разработка и тестирование результатов работы веб-приложения 38**](#_9cxu93wvvg6h)

[Разработка backend части 38](#_8ashlxksd2m3)

[Установка Node.js 38](#_gtnjfvtn5krz)

[Установка, запуск и базовая настройка Express 38](#_mzk6pzdg7ebc)

[Установка, puppeteer и cheerio 40](#_qmkqoev1tj5r)

[Javascript код для парсинга данных с сайта музыки spotify.com 41](#_vmjkwz9tj52m)

[Javascript код для парсинга данных с сайта музыки labirint.ru 43](#_ylfl1xbndbwv)

[Javascript код для парсинга данных с сайта музыки book24.ru 45](#_ujmpihkni1ia)

[Javascript код для парсинга данных с сайта музыки chitai-gorod.ru 47](#_l49nznl942vj)

[Финальная настройка Express для обработки входящих запросов 49](#_4yoq6fb2slgc)

[Разработка frontend части 50](#_6lrpanmh65xu)

[Установка Vue CLI 50](#_2vpnrewmwbz0)

[Корневой компонент агрегатора 51](#_vst384u3ih8w)

[Дочерние компоненты агрегатора 54](#_829v57sjknnq)

[vHeader.vue - navbar сайта 54](#_ctqjym2emurf)

[vIntro.vue - описание сайта 55](#_k19v44l9j6rn)

[vSearch.vue - поиск музыки 56](#_ff3w34j48pe4)

[vShopHeader.vue - название магазина 58](#_pqb0eh73zrow)

[vArrContent.vue - вывод результатов поиска 59](#_79yi2dk9egq7)

[vSortContent.vue - сортировка по цене 61](#_r9ou0bl1ttxf)

[Тестирование работоспособности - UI/UX 62](#_uvueuwdgsbl5)

[vLoader.vue - вращающийся лоадер 65](#_fkso6m7lk07m)

[**Подведение итогов, оценка результатов и заключение 67**](#_1r7obkudnbg)

[**Список использованной литературы 69**](#_1mjamtxspbrg)

# Введение

Суть дипломного проекта по созданию веб-агрегатора информации из книжных интернет-магазинов заключается в разработке веб-сайта, который собирает и представляет информацию о музыкиах из различных русскоязычных книжных интернет-магазинов.

Основная цель проекта — предоставить платформу, облегчающую пользователям поиск и сравнение цен, наличия и другой соответствующей информации о музыкиах, в которых есть возможная заинтересованность к покупке.

Проект будет включать в себя несколько ключевых компонентов, в том числе:

1. Веб-скрэпинг: агрегатор будет собирать данные из различных книжных онлайн-магазинов. Данные будут включать названия музыки, имена авторов, описания, вид обложки и цену;
2. Пользовательский интерфейс: пользовательский интерфейс веб-сайта должен быть разработан таким образом, чтобы обеспечить удобство для посетителей. Он будет включать функции поиска, сортировки и другие функции, облегчающие пользователям поиск необходимой им информации;
3. База данных: собранные данные будут храниться в базе данных. База данных планируется к проектированию таким образом, чтобы обеспечивать эффективный поиск и извлечение информации на основе пользовательского ввода;
4. Тестирование и обслуживание. Проект планируется тщательно протестировать, чтобы убедиться, что он работает должным образом и не содержит ошибок. Для поддержания агрегатора в актуальном состоянии и бесперебойной работы потребуется техническое обслуживание и обновления.

План разработки веб-приложения с использованием HTML, CSS, Bootstrap, JS, Vue.js, Axios, Node.js, Express, Fetch, Puppeteer и Cheerio будет включать в себя:

1. Разработку пользовательского интерфейса (UI) и взаимодействие с пользователем (UX) приложения с помощью HTML, CSS. Пользовательский интерфейс должен быть визуально привлекательным и простым в навигации. Bootstrap планируется использовать для ускорения процесса разработки и создания адаптивного дизайна;
2. Использование Vue.js для добавления интерактивности и динамического контента в пользовательский интерфейс. Vue.js также можно использовать для создания компонентов и обработки пользовательского ввода, что сделает приложение более отзывчивым и удобным для пользователя;
3. Использование fetch, Puppeteer и Cheerio для получения данных из книжных интернет-магазинов, которые не предоставляют API;
4. Использование Axios для получения данных со стороны backend после парсинга и переноса их на frontend. Полученные данные также могут храниться в базе данных Firebase или локальной базе данных Lowdb;
5. Использование Node.js и Express для создания сервера, который обрабатывает HTTP-запросы и предоставляет клиенту файлы HTML, CSS и JS;

Проект будет разрабатываться собственными силами. Однако, с точки зрения создания и запуска веб-приложения, в подобных проектах могут участвовать следующие разработчики:

1. Веб-разработчики, обладающие знаниями в области HTML, CSS и JavaScript, могут помочь создать внешний пользовательский интерфейс и добавить интерактивность в приложение с помощью таких фреймворков, как Vue.js;
2. Backend разработчики с опытом работы с Node.js, Express и управлением базами данных могут помочь создать инфраструктуру на стороне сервера, конечные точки API и подключения к базе данных;
3. Инженеры данных могут помочь со сбором, очисткой и хранением данных. Они могут помочь разработать и внедрить конвейеры данных для сбора информации из различных источников, таких как книжные интернет-магазины;
4. QA Инженеры могут помочь с тестированием приложения на наличие ошибок и проблем. Они могут создавать тестовые случаи и сценарии, чтобы убедиться, что приложение работает должным образом;
5. UX-дизайнеры могут помочь с разработкой удобного и визуально привлекательного интерфейса для приложения. Они также могут предоставить рекомендации по передовым методам использования и доступности;
6. Инженеры DevOps могут помочь с развертыванием приложения в производственной среде.

Для создания успешного проекта важно, чтобы команда разработчиков обладала разнообразным набором навыков и опыта. Сотрудничество и общение между членами команды имеют решающее значение для обеспечения того, чтобы проект соответствовал требованиям и был выполнен вовремя.

Дипломный проект по созданию веб-агрегатора информации из книжных интернет-магазинов потребует сочетания навыков программирования, знаний в области веб-разработки и опыта работы с веб-скрейпингом. Конечный продукт станет полезным инструментом для покупателей музыки, которые хотят сравнить цены и наличие в нескольких книжных интернет-магазинах.

# Веб-приложение и способы получения данных из внешних источников Интернета.

## Веб-приложение: основы.

Веб-приложение — это тип программного приложения, доступ к которому и использование которого осуществляется через веб-браузер. В отличие от традиционных настольных приложений, веб-приложения не нужно устанавливать на компьютер или устройство пользователя, и они предназначены для работы на любом устройстве, подключенным к Интернету.

Содержание веб-приложения может варьироваться в зависимости от его назначения и функциональности. Веб-приложение может представлять собой как простой калькулятор или список дел, так и сложное корпоративное приложение, управляющее рабочими и бизнес-процессами. Как правило, веб-приложение состоит из внешнего пользовательского интерфейса, внутреннего сервера и базы данных, в которой хранятся данные и информация.

Принципы разработки веб-приложений включают использование веб-технологий, таких как HTML, CSS и JavaScript, для создания динамического и интерактивного пользовательского интерфейса, а также использование серверных технологий, таких как Node.js, PHP или Ruby on Rails, для обработки запросов от пользователей. Веб-приложения также используют веб-протоколы, такие как HTTP и HTTPS, для передачи данных между клиентом и сервером.

Цели веб-приложений могут различаться в зависимости от потребностей пользователя. Некоторые общие цели веб-приложений включают в себя:

1. Электронная коммерция: веб-приложения можно использовать для продажи товаров и услуг в Интернете, позволяя клиентам просматривать, сравнивать и покупать товары из любого места, где есть подключение к Интернету;
2. Социальные сети: веб-приложения могут предоставлять службы социальных сетей, которые позволяют пользователям создавать профили, обмениваться контентом и взаимодействовать с другими пользователями;
3. Продуктивность. Веб-приложения могут предоставлять инструменты для совместной работы, управления задачами и планирования проектов, позволяя командам эффективно и продуктивно работать вместе;
4. Управление информацией. Веб-приложения можно использовать для сбора, хранения и анализа данных, что позволяет организациям принимать обоснованные решения и оптимизировать свою деятельность.

Применительно к дипломному проекту, Веб-агрегатор информации из книжных интернет-магазинов — это тип веб-приложения, которое позволяет пользователям искать музыкии и сравнивать цены и наличие в нескольких книжных онлайн-магазинах. Суть веб-приложения будет включать в себя функцию поиска, которая позволяет пользователям вводить ключевые слова или фразы для поиска музыки, а также базу данных с информацией о музыкиах и статистике минимальных цен в различных книжных онлайн-магазинах.

С точки зрения принципов разработки, веб-приложение должно быть разработано с учетом масштабируемости и удобства обслуживания, учитывая, что оно будет извлекать данные из нескольких внешних источников.

Цель веб-агрегатора информации из книжных интернет-магазинов состоит в том, чтобы предоставить пользователям удобный способ поиска и сравнения музыки в нескольких книжных онлайн-магазинах, помогая им находить лучшие предложения и экономить время в процессе поиска по отдельным сайтам.

## Frontend веб-приложения, его особенности.

Внешний (frontend) интерфейс веб-приложения будет состоять из пользовательского интерфейса, отображающего результаты поиска, информацию о музыкиах и цены в нескольких книжных интернет-магазинах.

Внешний интерфейс веб-приложения относится к той части приложения, которая видна пользователям и взаимодействует с ними напрямую. Обычно он состоит из пользовательского интерфейса, который представлен в веб-браузере и позволяет пользователям взаимодействовать с приложением.

В случае веб-агрегатора информации из книжных интернет-магазинов внешний интерфейс может состоять из следующих компонентов:

1. Пользовательский интерфейс: пользовательский интерфейс будет визуальным представлением веб-приложения и будет включать такие элементы, как панели поиска, страницы результатов и страницы сведений о музыкиах;
2. HTML — это язык разметки, используемый для структурирования содержимого веб-страниц. HTML используется для определения структуры пользовательского интерфейса, включая заголовки, абзацы, списки и таблицы;
3. CSS — это язык стилей, который используется для управления внешним видом веб-страниц. CSS используется для определения визуальных свойств пользовательского интерфейса, таких как цвет и шрифт текста, расположение элементов и общий стиль веб-приложения;
4. Bootstrap — это популярный интерфейсный фреймворк, который предоставляет предварительно разработанные компоненты и стили для создания адаптивных мобильных веб-приложений. Bootstrap используется для ускорения разработки и обеспечения единообразного внешнего вида веб-приложения;
5. Vue.js — это среда JavaScript для создания пользовательских интерфейсов и одностраничных (SPA) приложений. Vue используется для создания динамических и отзывчивых пользовательских интерфейсов, которые обновляются в режиме реального времени на основе пользовательского ввода;
6. Axios — библиотека JavaScript для выполнения HTTP-запросов со стороны клиента веб-приложения. Используется для получения данных с сервера и отображения их пользователю в режиме реального времени.

Frontend должен иметь понятный и простой в использовании интерфейс, который позволяет пользователям искать музыкии и сравнивать цены в нескольких книжных онлайн-магазинах. Пользовательский интерфейс должен быть разработан таким образом, чтобы он реагировал и адаптировался к различным размерам экрана и устройствам.

Интерфейс должен отображать обновления в реальном времени для пользователей, когда данные из бэкэнда изменяются, например, когда добавляется информация о новой музыкие или когда меняются цены на музыкии.

Пользовательский интерфейс должен быть разработан с учетом высокой производительности и быстрой загрузкой страниц, а также минимальной задержкой.

Внешний интерфейс должен быть совместим с различными веб-браузерами и операционными системами, плюс предназначен для работы на мобильных устройствах, а также на настольных компьютерах.

Пользовательский интерфейс должен быть разработан с учетом безопасности, чтобы защитить личную информацию пользователей за счет использования безопасных протоколов, таких как HTTPS.

## Backend веб-приложения, его особенности.

Серверная часть веб-приложения будет состоять из сервера, который обрабатывает запросы пользователей и извлекает данные из нескольких книжных онлайн-магазинов. Это потребует использования веб-протоколов, таких как HTTP и HTTPS, для получения данных из API-интерфейсов, предоставляемых книжными интернет-магазинами.

Веб-приложению также потребуется база данных для хранения информации о музыкиах, ценах и других деталей.

В случае веб-агрегатора информации из книжных интернет-магазинов серверная часть будет состоять из следующих компонентов:

1. Node.js — это среда выполнения JavaScript, которая позволяет запускать код JavaScript на стороне сервера веб-приложения. Он будет использоваться для обработки логики на стороне сервера и предоставления платформы для запуска кода на стороне сервера;
2. Express — это популярный веб-фреймворк для Node.js, который предоставляет набор функций и инструментов для создания веб-приложений. Его можно использовать для обработки HTTP-запросов и ответов, маршрутизации, промежуточного программного обеспечения и других задач на стороне сервера.

Backend должен будет собирать данные из нескольких книжных интернет-магазинов с помощью парсинга веб-страниц или вызовов API. Эти данные необходимо будет собирать на регулярной основе, чтобы обеспечить их актуальность.

Серверной части необходимо будет обрабатывать данные, собранные в книжных онлайн-магазинах, для извлечения соответствующей информации, такой как названия музыки, авторы, цены и иные детали

Backend должен управлять базой данных для хранения собранных данных. База данных должна быть спроектирована с учетом высокой производительности и масштабируемости для обработки больших объемов данных.

Backend должен предоставить API, который может использоваться внешним интерфейсом для доступа к собранным данным.

Также серверной части необходимо анализировать собранные данные, чтобы удалить дубликаты, исправить ошибки и обеспечить согласованность. Это потребует использования методов анализа и нормализации данных.

Серверная часть должна будет предоставлять обновления в режиме реального времени для внешнего интерфейса при сборе новых данных или при изменении цены или доступности музыкии в любом из книжных онлайн-магазинов.

Серверная часть в идеале должна быть спроектирована с учетом масштабируемости, с возможностью обработки большого количества запросов от внешнего интерфейса и обработки больших объемов данных.

## Скрэпинг (web scraping) и парсинг (web parsing).

Веб-парсинг и веб-скрэпинг — это методы, используемые для извлечения информации с веб-сайтов.

Веб-парсинг, также известный как парсинг HTML, включает в себя использование программы для извлечения определенной информации из HTML-кода веб-страницы. Этот метод включает в себя написание кода, который программно считывает HTML-код веб-страницы и извлекает определенную информацию, например заголовок страницы или содержимое определенного тега. Парсеры HTML создаются с использованием таких языков программирования, как Python или JavaScript, и библиотек, таких как BeautifulSoup или Cheerio.

Веб-скрэпинг, с другой стороны, включает в себя автоматическое извлечение данных с веб-страниц. Веб-скрэпер — это программа, которая автоматически считывает HTML-код веб-страницы и извлекает данные в соответствии с определенным набором правил или критериев. Скрэперы можно создавать с использованием таких языков программирования, как Python или Node.js, и библиотек, таких как Scrapy или Puppeteer.

Основное различие между парсингом и скрэпингом веб-страниц заключается в том, что парсинг веб-страниц включает в себя извлечение определенной информации с одной веб-страницы, а скрэпинг веб-страниц включает извлечение больших объемов данных с нескольких веб-страниц или даже целых веб-сайтов. Веб-скрэпинг часто используется для таких целей, как интеллектуальный анализ данных, исследование рынка или мониторинг цен, а веб-парсинг используется для таких задач, как извлечение метаданных или проверки веб-страниц.

В контексте веб-агрегатора информации из книжных интернет-магазинов для извлечения информации о музыкиах, ценах и иной информации из нескольких книжных онлайн-магазинов будут использоваться оба метода с использованием следующих инструментов:

1. Fetch — это веб-API для отправки HTTP-запросов со стороны клиента веб-приложения. Его можно использовать для получения данных как из внешних API, предоставляемых книжными онлайн-магазинами, так и со стороны backend веб-приложения;
2. Puppeteer — библиотека Node.js, предоставляющая высокоуровневый API для управления headless браузерами Chrome или Chromium. Его можно использовать для сбора данных с веб-страниц и извлечения информации о музыкиах, ценах и иной информации;
3. Cheerio — библиотека Node.js, которая предоставляет jQuery-подобный API для анализа и работы с HTML и DOM-структурой. Его можно использовать для извлечения информации о музыкиах из HTML-кода веб-страниц, извлеченных Puppeteer.

Для извлечения данных с веб-сайтов используются как очистка, так и синтаксический анализ, включая текст, изображения, ссылки и другие типы контента. Извлеченные данные затем можно обрабатывать и анализировать для различных целей, таких как исследование рынка, анализ данных или создание веб-приложений.

Веб-скрэпинг и парсинг можно настроить для извлечения определенных элементов данных с веб-сайта, включая данные о ценах, описания продуктов, отзывы клиентов и другие типы контента. Это позволяет пользователям собирать и анализировать данные, соответствующие их конкретным потребностям.

Однако, веб-скрэпинг и парсинг могут вызывать юридические вопросы, связанные с авторским правом, интеллектуальной собственностью и условиями использования веб-сайта. На многих веб-сайтах действуют политики, запрещающие или ограничивающие использование инструментов автоматического извлечения данных, и использование этих инструментов без разрешения может привести к судебному преследованию.

Веб-скрэпинг и парсинг могут быть технически сложными, требующими специальных знаний в области веб-технологий, языков программирования и методов обработки данных. Также может быть сложно управлять большими объемами данных и обеспечивать точность и актуальность извлеченных данных.

В целом оба метода являются мощными методами извлечения данных с веб-сайтов, но они имеют как преимущества, так и проблемы. Их можно настроить для извлечения определенных элементов данных, но они также могут вызывать юридические проблемы и требовать технических знаний для эффективной реализации. Понимая эти функции и особенности, пользователи должны принимать обоснованные решения об использовании веб-скрэпинга и парсинга для сбора и анализа данных из Интернета.

# Программные продукты и инструменты web-разработки

Любая разработка начинается с выбора среды разработки. Интегрированные среды разработки (IDE) — это программные приложения, которые обеспечивают комплексную среду для разработки, тестирования и развертывания программного обеспечения. IDE обычно включают в себя редактор кода, компилятор или интерпретатор и инструменты отладки, а также другие функции, такие как интеграция контроля версий и инструменты управления проектами.

Разработчики часто предпочитают IDE, потому что они обеспечивают универсальное решение для всех аспектов процесса разработки. Разработчики могут писать и редактировать код, компилировать и запускать код, а также устранять ошибки в одной и той же среде. IDE также часто предоставляют дополнительные функции, такие как завершение кода, подсветка синтаксиса и инструменты отладки, которые могут ускорить и повысить эффективность разработки.

Однако некоторые разработчики предпочитают использовать текстовые редакторы и инструменты командной строки, поскольку они обеспечивают большую гибкость и могут быть настроены в соответствии с конкретными потребностями разработки. В конечном счете, выбор между использованием IDE или текстового редактора зависит от личных предпочтений и требований проекта разработки.

Наиболее популярные примеры IDE приведены далее.

Visual Studio — это интегрированная среда разработки, разработанная Microsoft, поддерживает широкий спектр языков программирования.

Eclipse — IDE с открытым исходным кодом, поддерживающая широкий спектр языков программирования, включая Java, C++ и Python.

Xcode — разработанная Apple, эта IDE используется для разработки приложений для macOS, iOS, watchOS и tvOS.

IntelliJ IDEA — разработанная JetBrains, эта IDE обеспечивает поддержку Java, Kotlin и других языков.

PyCharm — также разработанная JetBrains, эта IDE ориентирована на разработку Python.

Дипломный проект будет разрабатываться в VS Code.

Visual Studio Code (VS Code) — это бесплатный редактор кода с открытым исходным кодом, разработанный Microsoft. Он широко используется разработчиками для различных задач программирования, включая веб-разработку и машинное обучение. Он доступен для операционных систем Windows, macOS и Linux.

Одной из главных особенностей VS Code является обширный набор расширений и плагинов, которые позволяют разработчикам настраивать редактор под свои конкретные нужды. VS Code поддерживает широкий спектр языков программирования, и доступны расширения для всего: от подсветки синтаксиса и форматирования кода до отладки и интеграции с Git.

Некоторые из ключевых особенностей VS Code включают в себя:

1. IntelliSense: эта функция предоставляет интеллектуальные предложения по завершению кода по мере ввода в зависимости от языка, с которым вы работаете.
2. Встроенная поддержка Git: VS Code включает встроенную поддержку Git, что позволяет легко управлять репозиториями кода, не выходя из редактора.
3. Поддержка отладки: VS Code предоставляет встроенный отладчик для ряда языков программирования, позволяющий легко выполнять код, выявлять и исправлять ошибки.
4. Запуск задач: VS Code поддерживает создание пользовательских задач, которые могут автоматизировать общие рабочие процессы, такие как создание и тестирование кода.
5. Встроенный терминал: VS Code включает встроенное окно терминала, позволяющее выполнять команды оболочки непосредственно в редакторе.
6. Настройка темы: VS Code включает в себя ряд встроенных цветовых тем, и вы также можете загружать и устанавливать пользовательские темы, чтобы настроить внешний вид редактора.
7. Поддержка нескольких языков: VS Code поддерживает широкий спектр языков программирования, включая JavaScript, Python, Ruby и многие другие.

VS Code — очень гибкий и настраиваемый редактор кода, предоставляющий ряд мощных функций для разработчиков. Он имеет большое и активное сообщество пользователей и разработчиков, а в Интернете доступно множество ресурсов, которые помогут начать работу и научиться эффективно его использовать.

## Инструменты frontend части.

### HTML

HTML (HyperText Markup Language) является основным языком разметки веб-страниц. Он используется для создания содержимого веб-страниц, которое отображается в браузере.

HTML является стандартизированным языком, который определяет структуру веб-страницы с помощью различных элементов и атрибутов. Каждый элемент представляет определенный тип содержимого, такой как заголовки, параграфы, списки, изображения и т.д.

HTML также используется для создания ссылок на другие веб-страницы, внутренние и внешние. Он позволяет создавать формы, которые пользователи могут заполнять и отправлять, а также добавлять мультимедийный контент, такой как видео и аудио.

HTML является важным инструментом для создания веб-страниц и веб-приложений. Он обеспечивает структуру и содержимое веб-страниц, которые пользователи видят в браузере. Без HTML веб-страницы были бы просто набором несвязанных текстов и изображений.

HTML использует теги (tags) для определения элементов страницы. Теги создаются с использованием угловых скобок <>, и обычно имеют начальный и конечный тег, где конечный тег отличается от начального символом "/".

Внутри тегов могут находиться атрибуты (attributes), которые определяют дополнительные характеристики элементов, такие как их цвет, размер, ссылки на другие страницы и т.д.

Некоторые наиболее часто используемые теги в HTML:

1. <html>: Начальный тег для HTML-документа.
2. <head>: Содержит метаданные документа, такие как заголовки страницы, ссылки на стили CSS, и другие данные, которые не отображаются на странице.
3. <title>: Определяет заголовок документа.
4. <body>: Определяет содержимое страницы, которое отображается в браузере.
5. <div>: Определяет блок элементов на странице.
6. <p>: Определяет абзац текста.
7. <img>: Определяет изображение на странице.
8. <a>: Определяет ссылку на другую страницу или ресурс.
9. <form>: Определяет форму на странице, которую пользователь может заполнить и отправить.

В современной верстке для распределения элементов на странице чаще всего используются теги <div> и <section>.

<div> - это тег в HTML, который используется для определения контейнеров на веб-страницах. Он создает блоки, в которые можно помещать другие элементы, такие как текст, изображения, видео, таблицы и т.д. <div> не предназначен для определения семантики содержимого, а скорее служит для оформления и стилизации. Он не влияет на структуру документа, но может использоваться для группировки и организации элементов страницы.

Пример использования тега <div>:

|  |
| --- |
| <div>  <h1>Заголовок</h1>  <p>Это простой текст на странице.</p> </div> |

Здесь создается блок, внутри которого находится заголовок первого уровня и абзац текста. Контейнер <div> позволяет объединить эти элементы, чтобы упростить их оформление и стилизацию. <div> можно также использовать для разделения элементов страницы на группы, например для создания секций или блоков с футером и хедером. В сочетании с CSS стилями, <div> может быть мощным инструментом для создания разнообразных макетов и оформления веб-страниц.

<section> - это тег в HTML, который используется для определения разделов на веб-страницах. Он обычно содержит группы связанных элементов, таких как заголовки, параграфы, изображения и другие элементы. <section> обычно используется для структурирования документа и улучшения доступности и семантической структуры.

Пример использования тега <section>:

|  |
| --- |
| <section>  <h2>Название раздела</h2>  <p>Это текст в разделе.</p>  <img src="image.jpg" alt="Изображение"> </section> |

Здесь создается раздел страницы, содержащий заголовок второго уровня, абзац текста и изображение. Контейнер <section> позволяет группировать эти элементы, чтобы указать, что они относятся к одному разделу страницы.

<section> обычно используется в сочетании с другими семантическими тегами, такими как <article>, <header>, <footer> и <nav>, чтобы создать структурированную и понятную семантическую разметку веб-страницы. <section> также может использоваться для стилизации страницы с помощью CSS. Он позволяет определять общие стили для групп связанных элементов и упрощает оформление веб-страницы.

Понимание тегов является необходимым для создания и правильного размещения элементов на странице. Например, правильное использование тегов позволяет браузеру корректно интерпретировать содержимое страницы и отображать его правильно. Также это может повлиять на поисковую оптимизацию (SEO), что важно для любого веб-сайта.

### CSS

CSS (Cascading Style Sheets) - это язык стилей, который используется для оформления веб-страниц. Он определяет, как должны выглядеть элементы HTML на странице, включая их цвет, размер, расположение и другие характеристики.

CSS позволяет разработчикам создавать стильные и привлекательные веб-страницы, которые могут быть адаптированы под различные устройства и размеры экранов.

Основные принципы CSS:

1. Каскадность - стили могут быть определены на разных уровнях, и значения, определенные на более высоком уровне, могут быть переопределены на более низком уровне.
2. Наследование - стили, определенные для родительского элемента, могут быть унаследованы его дочерними элементами.
3. Приоритетность - стили могут иметь разную приоритетность, и значение, определенное для элемента с более высокой приоритетностью, будет использоваться вместо значения элемента с более низкой приоритетностью.

Пример CSS:

|  |
| --- |
| body {  background-color: #f0f0f0;  font-family: Arial, sans-serif; }  h1 {  color: #333;  font-size: 28px;  text-align: center; }  p {  color: #666;  font-size: 16px;  line-height: 1.5; } |

В примере выше определены стили для элементов <body>, <h1> и <p>. <body> задает фоновый цвет и шрифт для всей страницы, <h1> задает цвет, размер и выравнивание для заголовков первого уровня, а <p> задает цвет, размер и высоту строки для параграфов.

CSS может быть определен внутри HTML-документа с использованием тега <style>, или может быть вынесен в отдельный файл .css для повторного использования на других страницах. С помощью CSS можно создавать сложные макеты страниц, анимации и интерактивные элементы, чтобы улучшить пользовательский опыт на веб-сайте.

### Bootstrap

Bootstrap - это бесплатный фреймворк для разработки веб-сайтов, который содержит набор готовых компонентов, шаблонов и стилей на основе CSS и JavaScript. Он был разработан компанией Twitter и впервые был выпущен в 2011 году.

Основные преимущества Bootstrap:

1. Гибкость и масштабируемость - Bootstrap предлагает широкий выбор компонентов и стилей, которые можно использовать для быстрой разработки веб-сайтов. Он также поддерживает адаптивный дизайн, что позволяет создавать сайты, которые хорошо отображаются на любом устройстве, включая смартфоны и планшеты.
2. Кросс-браузерность - Bootstrap обеспечивает одинаковый внешний вид веб-сайта в различных браузерах, что значительно упрощает разработку и тестирование сайта.
3. Отзывчивый дизайн - Bootstrap обеспечивает адаптивный дизайн, что позволяет сайту подстраиваться под различные размеры экранов. Это делает веб-сайт более доступным для пользователей на мобильных устройствах и улучшает пользовательский опыт.
4. Широкое сообщество - Bootstrap имеет большое сообщество пользователей, которые создают и обмениваются компонентами, шаблонами и советами, что позволяет быстро решать проблемы и находить решения для разных задач.

Bootstrap включает в себя готовые компоненты, такие как меню навигации, кнопки, формы, карусели, модальные окна и многие другие, которые можно использовать для создания веб-сайтов. Он также содержит набор классов, которые можно использовать для быстрого и простого применения стилей к элементам на странице.

Bootstrap можно использовать как самостоятельный фреймворк или в сочетании с другими технологиями, такими как HTML, CSS, JavaScript, jQuery и другими. Он также поддерживает различные плагины и инструменты, такие как Sass и Less, которые облегчают разработку и обслуживание веб-сайтов.

### 

### JavaScript и Vue.js

JavaScript — это язык программирования, обычно используемый для создания динамических веб-страниц и веб-приложений. Это язык высокого уровня, поддерживающий объектно-ориентированный, императивный и функциональный стили программирования. JavaScript можно использовать для различных задач, таких как добавление интерактивных функций на веб-страницы, создание браузерных игр, создание веб-приложений и мобильных приложений, а также разработка серверных приложений.

Некоторые из ключевых особенностей JavaScript включают в себя:

1. Программирование, управляемое событиями: JavaScript позволяет разработчикам создавать интерактивные веб-страницы, которые реагируют на действия пользователя, такие как нажатие кнопки, прокрутка страницы или ввод данных в форму.
2. Асинхронное программирование: JavaScript поддерживает асинхронное программирование, что позволяет разработчикам писать код, который не блокирует выполнение другого кода.
3. Кроссплатформенная совместимость: JavaScript поддерживается всеми основными веб-браузерами, что делает его популярным выбором для веб-разработки.
4. Расширяемость: JavaScript можно расширить с помощью библиотек и фреймворков, таких как jQuery, React и Vue.js, которые упрощают разработку сложных веб-приложений.

Vue.js - это прогрессивный фреймворк JavaScript, который используется для создания пользовательских интерфейсов и одностраничных приложений (SPA). Он представляет собой модульный фреймворк, который может использоваться вместе с другими библиотеками и инструментами, такими как Vuex, Vue Router, Vuetify и другие.

Основные преимущества Vue.js:

1. Легковесность - Vue.js имеет небольшой размер и быстро загружается на страницу, что позволяет создавать быстрые и отзывчивые пользовательские интерфейсы.
2. Гибкость - Vue.js можно использовать как для маленьких, простых приложений, так и для крупных и сложных проектов. Он также предоставляет широкий выбор опций и настроек, которые позволяют настроить его под различные потребности.
3. Простота в использовании - Vue.js легко устанавливается и используется даже начинающими разработчиками. Он также предоставляет простые и интуитивно понятные API и документацию.
4. Однофайловые компоненты - Vue.js поддерживает однофайловые компоненты, которые содержат в себе все необходимые HTML, CSS и JavaScript коды, что облегчает разработку и обслуживание приложений.

Vue.js предоставляет набор компонентов и директив, которые можно использовать для создания пользовательского интерфейса, а также API для управления состоянием приложения. Он также поддерживает использование шаблонов, фильтров и многих других функций, которые упрощают разработку.

Vue.js можно использовать как самостоятельный фреймворк, так и в сочетании с другими технологиями, такими как HTML, CSS, JavaScript, Bootstrap и другие. Он также может быть использован с различными инструментами и плагинами, такими как Vuex, Vue Router, Vuetify и другие, что позволяет создавать более сложные приложения и улучшать производительность.

### Axios

Axios — это популярная библиотека JavaScript, используемая для выполнения HTTP-запросов из веб-приложения. Он основан на интерфейсе XMLHttpRequest, встроенном в веб-браузеры, и предоставляет простой и удобный в использовании API для отправки и получения данных с сервера. Axios широко используется в современных веб-приложениях, особенно в приложениях, созданных с помощью Vue.js, React и других интерфейсных фреймворков.

Axios предоставляет ряд функций, которые делают его популярным выбором для HTTP-запросов, в том числе:

1. API на основе промисов: Axios построен на основе промисов JavaScript, что позволяет разработчикам писать асинхронный код, который легко читать и понимать.
2. Простой API: Axios предоставляет простой и интуитивно понятный API для выполнения HTTP-запросов с методами обработки запросов GET, POST, PUT и DELETE.
3. Перехватчики (Interceptors): Axios предоставляет функцию, называемую перехватчиками, которая позволяет разработчикам перехватывать и изменять HTTP-запросы и ответы до их отправки или получения.
4. Обработка ошибок (Error handling): Axios предоставляет мощную систему обработки ошибок, которая упрощает обработку сетевых ошибок и ошибок на стороне сервера.
5. Кросс-платформенная совместимость: Axios совместим как со средой браузера, так и со средой Node.js, что делает его популярным выбором для создания клиентских и серверных веб-приложений.

Axios можно использовать для различных задач, таких как получение данных с сервера, загрузка файлов и обработка аутентификации пользователей.

## Инструменты backend части.

### Node.js

Node.js — это кроссплатформенная среда выполнения JavaScript с открытым исходным кодом, которая позволяет разработчикам запускать код JavaScript вне веб-браузера. Он построен на основе движка Google V8, который представляет собой высокопроизводительный движок JavaScript, на котором работает браузер Google Chrome.

Node.js позволяет разработчикам писать серверные приложения с использованием JavaScript, что упрощает процесс разработки. Он предоставляет ряд функций, которые делают его популярным выбором для создания масштабируемых высокопроизводительных веб-приложений, в том числе:

1. Неблокирующая модель ввода-вывода (Non-blocking I/O model): Node.js использует управляемую событиями неблокирующую модель ввода-вывода, которая позволяет ему обрабатывать большое количество подключений с небольшим количеством потоков.
2. Высокая производительность: Node.js построен на основе движка V8, который обеспечивает высокую производительность кода JavaScript.
3. Большая экосистема пакетов: Node.js имеет большую и растущую экосистему пакетов с тысячами модулей и пакетов, доступных через диспетчер пакетов Node (npm).
4. Кроссплатформенная совместимость: Node.js является кроссплатформенным, что означает, что он может работать в Windows, macOS и Linux.
5. Поддержка приложений реального времени: Node.js хорошо подходит для создания приложений реального времени, таких как приложения для чата и онлайн-игровые платформы, поскольку он обеспечивает связь в реальном времени между сервером и клиентом.

Node.js обычно используется в сочетании с такими веб-фреймворками, как Express.js и Sails.js, для создания серверных приложений. Он также широко используется в сочетании с интерфейсными фреймворками, такими как Vue, React и Angular, для создания полнофункциональных веб-приложений.

### Express

Express — это популярная платформа веб-приложений с открытым исходным кодом для Node.js, которая предоставляет набор функций и инструментов для создания веб-приложений и API. Он разработан, чтобы быть гибким, масштабируемым и легким, что позволяет разработчикам создавать высокопроизводительные приложения с минимальными накладными расходами.

Express предоставляет ряд функций, которые делают его популярным выбором для создания веб-приложений, в том числе:

1. Маршрутизация: Express предоставляет мощную систему маршрутизации, которая позволяет разработчикам определять маршруты для обработки входящих HTTP-запросов.
2. Промежуточное ПО (Middleware): Express предоставляет систему промежуточного ПО, которая позволяет разработчикам легко добавлять в свои приложения функциональные возможности, такие как обработка аутентификации, ведение журнала и обработка ошибок.
3. Механизмы шаблонов: Express поддерживает различные механизмы шаблонов, такие как Pug, EJS и Handlebars, что упрощает создание динамических HTML-страниц.
4. Обработка ошибок: Express предоставляет мощную систему обработки ошибок, которая позволяет разработчикам обрабатывать ошибки последовательным и удобным для пользователя способом.
5. Интеграция с другими модулями Node.js: Express можно легко интегрировать с другими модулями Node.js, такими как диспетчер пакетов Node (npm) и основные модули Node.js.

Express обычно используется в сочетании с различными модулями Node.js для создания полнофункциональных веб-приложений. Он также широко используется в сочетании с интерфейсными фреймворками для создания одностраничных приложений (SPA) с внутренним API.

## 

## 

## Инструменты скрэпинга / парсинга.

### Fetch

fetch() — это метод JavaScript для выполнения сетевых запросов, обычно к API или серверу. Это встроенный метод, доступный в современных веб-браузерах, который позволяет выполнять HTTP-запросы с более простым и интуитивно понятным API.

Метод fetch() принимает один обязательный аргумент — URL-адрес ресурса, который необходимо получить. Он возвращает промис, который разрешается в ответ от сервера, что позволяет обрабатывать ответ различными способами. Пример того, как использовать fetch(), чтобы сделать запрос GET к API и проанализировать ответ как JSON приведен далее:

|  |
| --- |
| fetch('https://yandex.ru')  .then(response => response.json())  .then(data => {  // работа с полученной информацией  })  .catch(error => {  // обработка ошибок  }); |

### Puppeteer

Puppeteer — библиотека Node.js, предоставляющая высокоуровневый API для автоматизации работы веб-браузеров через протокол Chromium DevTools. Puppeteer поддерживается командой Chrome DevTools и может использоваться для различных целей, таких как тестирование, просмотр веб-страниц и создание PDF-файлов или снимков экрана веб-страниц.

С помощью Puppeteer возможно автоматизировать взаимодействие пользователей с веб-страницами, например переход по ссылкам, заполнение форм и создание снимков экрана. Дополнительно возможно получить доступ к модели DOM и управлять ею, а также выполнять JavaScript на странице, что позволяет собирать данные с веб-страниц.

Пример того, как использовать Puppeteer, чтобы открыть веб-страницу и сделать снимок экрана приведен далее:

|  |
| --- |
| const puppeteer = require('puppeteer');  (async () => {  const browser = await puppeteer.launch();  const page = await browser.newPage();  await page.goto('https://yandex.ru');  await page.screenshot({ path: 'yandex.png' });  await browser.close(); })(); |

В приведенном выше примере Puppeteer используется для запуска нового экземпляра Chromium, создания новой страницы, перехода на https://yandex.ru и создания снимка экрана страницы, который сохраняется в файл.

Puppeteer также предоставляет множество возможностей гибкости и настройки, таких как настройка размеров области просмотра, эмуляция мобильных устройств и вставка пользовательских скриптов на страницу.

### Cheerio

Cheerio — это быстрая и простая библиотека, похожая на jQuery, для анализа и обработки документов HTML и XML в Node.js. Cheerio предоставляет простой и знакомый синтаксис для работы с данными HTML и XML в JavaScript, что делает его популярным выбором для задач извлечения данных.

Одним из ключевых преимуществ использования Cheerio является его скорость и эффективность. Cheerio использует простой и легкий API, который предоставляет минимальный и эффективный набор методов для анализа и управления документами HTML и XML. Он также использует очень эффективный механизм обхода DOM, который намного быстрее, чем традиционные API DOM JavaScript.

Cheerio упрощает выбор HTML-элементов и управление ими с помощью селекторов CSS, аналогично тому, как это делается в jQuery.

Пример того, как использовать Cheerio для анализа HTML-документа и извлечения всех ссылок приведен ниже:

|  |
| --- |
| const cheerio = require('cheerio'); const axios = require('axios');  axios.get('https://yandex.ru')  .then((response) => {  const $ = cheerio.load(response.data);  const links = $('a');  links.each((i, link) => {  console.log($(link).attr('href'));  });  })  .catch((error) => {  console.error(error);  }); |

В приведенном примере Axios используется для выполнения запроса GET к веб-странице, а затем используется Cheerio для анализа ответа HTML и извлечения всех ссылок с помощью селектора CSS. Затем происходит перебор ссылок и вывод атрибута href для каждой из них.

Cheerio также можно использовать для управления документами HTML, добавления или удаления элементов и изменения атрибутов. Это делает его очень полезным инструментом для скрэпинга, поскольку он позволяет легко извлекать и обрабатывать данные из документов HTML и XML в Node.js.

# Практическая разработка и тестирование результатов работы веб-приложения

## Разработка backend части

### Установка Node.js

Node.js позволяет разработчикам использовать JavaScript для создания серверных приложений, взаимодействия с базами данных, разработки API и других задач. Node.js скачивается и устанавливается по ссылке ниже: <https://nodejs.org/en/download/>

Node.js так же включает в себя npm (Node Package Manager) - менеджер пакетов для языка программирования JavaScript. Он позволяет легко управлять зависимостями и библиотеками, используемыми в проекте. С помощью npm можно устанавливать, обновлять и удалять пакеты JavaScript из центрального репозитория npm или из локального каталога. Каждый пакет в npm содержит файл package.json, который определяет его зависимости, версии и другую информацию.

В директории созданного проекта необходимо проинициализировать npm командой:

|  |
| --- |
| npm init |

### Установка, запуск и базовая настройка Express

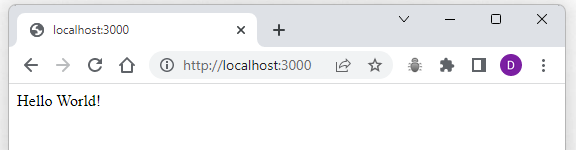
Установка express происходит через npm командой:

|  |
| --- |
| npm install express |

Стартовый код файла **index.js** для проверки успешного запуска express имеет следующий вид:

|  |
| --- |
| import express from 'express' const app = express() const port = 3000  app.get('/', (req, res) => {  res.send('Hello World!') })  app.listen(port, () => {  console.log(`Example app listening on port ${port}`) }) |

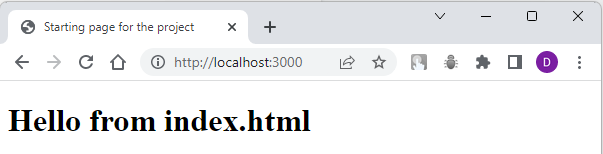
В случае успешного запуска в браузере по ссылке будет представлен следующий вид:



Чтобы указать путь до html файла, исходный код нужно изменить следующим образом:

|  |
| --- |
| import express from 'express' import path from 'path'  const app = express() const port = 3000  const \_\_dirname = path.resolve(); app.use(express.static(path.resolve(\_\_dirname, 'assets/html')))  app.get('/request', (req, res) => {  res.send('Hello World!') })  app.listen(port, () => {  console.log(`Example app listening on port ${port}`) }) |

Создав в указанной папке **assets/html** стандартный index.html, вывод будет следующим:



### Установка, puppeteer и cheerio

Установка puppeteer происходит через npm следующей командой:

|  |
| --- |
| npm i puppeteer |

Установка cheerio происходит через npm следующей командой:

|  |
| --- |
| npm i cheerio |

После чего они появятся в зависимостях в файле package.json

|  |
| --- |
| "dependencies": {  "cheerio": "^1.0.0-rc.12",  "express": "^4.18.2",  "puppeteer": "^19.7.1"  } } |

В рамках дипломного проекта планируется парсить данные со следующих сайтов:

* Spotify
* Aplle music
* Google Play Music
* Tidal

Данные магазины - крупнейшие площадки в сегменте по прослушиванию музыки, которые покрывают 90% всех запросов по поиску музыки.

### Javascript код для парсинга данных с сайта музыки my-shop

|  |
| --- |
| import \* as cheerio from 'cheerio'; import puppeteer from 'puppeteer';  export async function getSpotify(request) {  const shopTitle = 'Spotify'  const link = `https://spotify.com/shop/search/a/sort/a/page/1.html?f14\_39=3&f14\_16=4&f14\_6=${request}&next=1&p38=1`    const bookTitle = 'div > div.wrap.product-list > div > div.list-pane > div.article > div > div > div > div.top > div > div.item\_\_title\_\_container > a.item\_\_title'  const bookPrice = 'div > div > div.wrap.product-list > div > div.list-pane > div.article > div > div > div > div.top > div > div.item\_\_price > div > span.price.rubl'  const bookLink = 'div > div > div.wrap.product-list > div > div.list-pane > div.article > div > div > div > div.top > div > div.item\_\_title\_\_container > a.item\_\_title'  const bookImage = 'div > div > div.wrap.product-list > div > div.list-pane > div.article > div > div > div > div.top > div.item\_\_img > a > picture > img'   const browser = await puppeteer.launch();  const page = await browser.newPage();  await page.goto(link)   const htmlContent = await page.content()   const $ = cheerio.load(htmlContent);  await browser.close();   // Получаю массив названий музыки  const bookTitleArr = ($(bookTitle).text().split("\n").map(name => name.trim()))  bookTitleArr.shift();   // Получаю массив цен музыки  const bookPriceArr = []  for (let price of $(bookPrice)) {  bookPriceArr.push(parseInt((price.children[0].data).split(/\s+/).join("")))  }   // Получаю массив ссылок музыки  const bookLinkArr = []  for (let link of $(bookLink)) {  bookLinkArr.push('https://spotify.com' + link.attribs.href)  }   // Получаю массив изображений музыки  const bookImageArr = []  for (let image of $(bookImage)) {  bookImageArr.push('https:' + image.attribs.src)  }   //Финальный массив со всеми данными  const finalArr = []  for (let i = 0; i < bookTitleArr.length; i++) {  finalArr.push([bookTitleArr[i], bookPriceArr[i], bookLinkArr[i], bookImageArr[i], shopTitle])  }   // Сортировка массива по цене  finalArr.sort((a, b) => a[1] - b[1])   return finalArr }; |

## 

### Javascript код для парсинга данных с сайта музыки labirint.ru

|  |
| --- |
| import \* as cheerio from 'cheerio'; import puppeteer from 'puppeteer';  export async function getLabirint(request) {  const shopTitle = 'Labirint'  const link = `https://www.labirint.ru/search/${request}/?stype=0&display=table&available=1&paperbooks=1`   const bookTitle = 'div.b-search-page-content > div.content-block > table > tbody > tr > td.col-sm-4 > div > a'  const bookPrice = 'div.b-search-page-content > div.content-block > table > tbody > tr > td.products-table\_\_price.col-sm-1 > div > div > span.price-val > span'  const bookImage = 'div.b-search-page-content > div.content-block > table > tbody > tr > td.products-table\_\_buy.col-sm-2 > div > div > div.fleft.product-icons-outer > div > div > a.icon-compare.track-tooltip.js-open-actions-block'   const browser = await puppeteer.launch();  const page = await browser.newPage();  await page.goto(link)    const htmlContent = await page.content()   const $ = cheerio.load(htmlContent);   await browser.close();   // Массив названий музыки  const bookTitleArr = []  for (let title of $(bookTitle)) {  bookTitleArr.push(title.attribs.title)  }   // Массив цен музыки  const bookPriceArr = []  for (let price of $(bookPrice)) {  bookPriceArr.push(parseInt((price.children[0].data).split(/\s+/).join("")))  }   // Массив изображений и ссылок на музыкиу  const bookImageArr = []  const bookLinkArr = []  for (let image of $(bookImage)) {  bookImageArr.push(image.attribs['data-image'])  bookLinkArr.push(image.attribs['data-url'])  }   //Финальный массив со всеми данными  const finalArr = []  for (let i = 0; i < bookTitleArr.length; i++) {  finalArr.push([bookTitleArr[i], bookPriceArr[i], bookLinkArr[i], bookImageArr[i], shopTitle])  }   // Сортировка массива по цене  finalArr.sort((a, b) => a[1] - b[1])   return finalArr }; |

### Javascript код для парсинга данных с сайта музыки book24.ru

|  |
| --- |
| import \* as cheerio from 'cheerio'; import puppeteer from 'puppeteer';  export async function getBook24(request) {  const shopTitle = 'Book24'  const link = `https://book24.ru/search/?q=${request}&available=2`   const bookTitle = 'div.product-card\_\_content > a'  const bookLink = 'article > div.product-card\_\_image-holder > a'  const bookPrice = 'article > div.product-card\_\_content > div.product-card-price.product-card\_\_price > div.product-card-price\_\_current > span'  const bookImage = 'article > div.product-card\_\_image-holder > a > picture > img'   const browser = await puppeteer.launch();  const page = await browser.newPage();  await page.goto(link)   await page.waitForSelector(bookPrice)   const htmlContent = await page.content()   const $ = cheerio.load(htmlContent);   await browser.close();   // Массив названий музыки  const bookTitleArr = []  for (let title of $(bookTitle)) {  bookTitleArr.push(title.attribs.title)  }   // Массив ссылок музыки  const bookLinkArr = []  for (let link of $(bookLink)) {  bookLinkArr.push('https://book24.ru' + link.attribs.href)  }   // Массив цен музыки  const bookPriceArr = []  for (let price of $(bookPrice)) {  let parsedPrice = (price.children[0].data).replace(/\s/g, '')  bookPriceArr.push(parseInt(parsedPrice))  }   // Массив цен музыки  const bookImageArr = []  for (let image of $(bookImage)) {  if (image.attribs['data-src'].startsWith('http')) {  bookImageArr.push(image.attribs['data-src'])  } else { bookImageArr.push('https:' + image.attribs['data-src']) }   }   //Финальный массив со всеми данными  const finalArr = []  for (let i = 0; i < bookTitleArr.length; i++) {  finalArr.push([bookTitleArr[i], bookPriceArr[i], bookLinkArr[i], bookImageArr[i], shopTitle])  }   // Сортировка массива по цене  finalArr.sort((a, b) => a[1] - b[1])   return finalArr }; |

## 

### Javascript код для парсинга данных с сайта музыки chitai-gorod.ru

|  |
| --- |
| import \* as cheerio from 'cheerio'; import puppeteer from 'puppeteer';  export async function getChitaiGorod(request) {  const shopTitle = 'Chitai Gorod'  const link = `https://www.chitai-gorod.ru/search?q=${request}&&available=1`    const bookTitle = `div.product-card\_\_text.product-card\_\_row > a > div > div.product-title\_\_head`  const bookPrice = `div.product-card\_\_price.product-card\_\_row > div > div.product-price\_\_value`  const bookLink = `div.product-card\_\_text.product-card\_\_row > a`  const bookImage = `a.product-card\_\_picture.product-card\_\_row > picture > img`   const browser = await puppeteer.launch();  const page = await browser.newPage();  await page.goto(link)   const htmlContent = await page.content()   const $ = cheerio.load(htmlContent);   await browser.close();   // Массив названия музыки  const bookTitleArr = []  for (let titleName of $(bookTitle)) {  let title = titleName.children[0].data.replace(/(\r\n|\n|\r)/gm, '').trimLeft().trimRight()  bookTitleArr.push(title)  }   // Массив цен музыки  const bookPriceArr = []  for (let price of $(bookPrice)) {  let parsedPrice = (price.children[1].data).replace(/(\r\n|\n|\r) /gm, '').trimLeft().trimRight().replace(/\s/g, '')  bookPriceArr.push(parseInt(parsedPrice))  }   // Массив ссылки на музыкиу  const bookLinkArr = []  for (let link of $(bookLink)) {  bookLinkArr.push('https://www.chitai-gorod.ru' + link.attribs.href)  }   // Массив изображения на музыкиу  const bookImageArr = []  for (let image of $(bookImage)) {  bookImageArr.push(image.attribs['data-src'])  }   //Финальный массив со всеми данными  const finalArr = []  for (let i = 0; i < bookTitleArr.length; i++) {  finalArr.push([bookTitleArr[i], bookPriceArr[i], bookLinkArr[i], bookImageArr[i], shopTitle])  }   // Сортировка массива по цене  finalArr.sort((a, b) => a[1] - b[1])   return finalArr }; |

Для того, чтобы результаты возвращались как один массив массивов, создается отдельный js файл, в который импортируются все методы парсинга, с последующим объединением в одну асинхронную функцию:

|  |
| --- |
| import { getBook24 } from './parserBook24.js' import { getChitaiGorod } from './parserСhitaigorod.js' import { getSpotify } from './parserSpotify.js' import { getLabirint } from './parserLabirint.js'  export async function getAll(request = "януш корчак") {  return await Promise.all([getBook24(request), getChitaiGorod(request), getSpotify(request), getLabirint(request)]) } |

## 

## 

### Финальная настройка Express для обработки входящих запросов

Для того, чтобы back end получал данные, в функционале Express есть возможность обратиться к встроенному объекту req.query, где по ключу first содержится пришедший запрос в виде текста.

Далее этот запрос передается в асинхронную функцию getAll()

|  |
| --- |
| import express from 'express' import path from 'path' import { getAll } from './scripts/getAllResults.js'  const app = express() const port = 3000  const \_\_dirname = path.resolve(); console.log(\_\_dirname)  app.use(express.static(path.resolve(\_\_dirname, 'html'))) app.use(express.static(path.resolve(\_\_dirname, 'scripts')))  app.get("/request", async (req, res) => {  res.header("Access-Control-Allow-Origin", "\*");  const searchReq = req.query.first;    let answer = await getAll(searchReq)  res.send(answer) })   app.listen(port, () => {  console.log(`Example app listening on port ${port}`) }) |

На моменте разработки, в целях избежания запрета CORS указано разрешение на возврат данных любым ресурсам через "Access-Control-Allow-Origin", "\*".

## 

## Разработка frontend части

Vue.js был выбран за его компонентный подход к созданию пользовательского интерфейса, использование директив и реактивных свойств для управления динамическим контентом, а также удобной интеграции с другими библиотеками и фреймворками.

Веб-приложение с использованием Vue.js состоит из компонентов, которые можно сравнить с независимыми блоками кода, отвечающими за отображение, логику и данные. Компоненты могут быть переиспользованы в разных частях приложения, что делает код более модульным и поддерживаемым.

Для создания интерфейса веб-приложения на Vue.js были использованы HTML-шаблоны, содержащие директивы Vue.js и JavaScript-код, отвечающий за логику и управление состоянием приложения.

### Установка Vue CLI

|  |
| --- |
| npm install -g @vue/cli |

Данный тип установки позволяет использовать компонентный подход в разработке веб приложения.

После установки создается папка в которой преднастроены все необходимые модули.

Запуск разработки производится командой:

|  |
| --- |
| npm run serve |

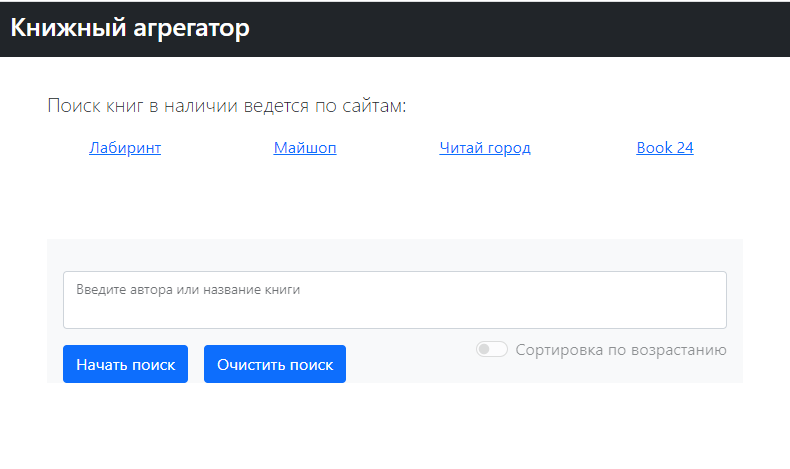
### 

### Корневой компонент агрегатора

App.vue — корневой компонент приложения Vue.js. Это начальная точка приложения, где визуализируются все остальные компоненты. Это однофайловый компонент, который служит контейнером для других компонентов и определяет базовую структуру приложения.

App.vue обычно состоит из трех основных частей: шаблона, скрипта и стиля. Раздел шаблона содержит HTML-структуру приложения, раздел сценария содержит код JavaScript, управляющий поведением приложения, а раздел стиля содержит код CSS, определяющий стиль приложения.

App.vue может также содержать другие компоненты и импортировать библиотеки по мере необходимости. Это центральная точка приложения, позволяющая легко управлять компонентами и данными приложения.



|  |
| --- |
| <template>  <vHeader />  <vIntro />  <vSearch @array-updated="handleArrayUpdate" @search-results="isOnSearchResults" @sort-change="sortChange"  ref="childComponent4Spec" />   <div v-show="showSearchResults">  <vShopHeader shopName="Book24" />  <vArrContent :myArray="parentArray[0]" @clarify-request="clarifyRequest" />   <vShopHeader shopName="Chitai-Gorod" />  <vArrContent :myArray="parentArray[1]" @clarify-request="clarifyRequest" />   <vShopHeader shopName="My-Shop" />  <vArrContent :myArray="parentArray[2]" @clarify-request="clarifyRequest" />   <vShopHeader shopName="Labirint" />  <vArrContent :myArray="parentArray[3]" @clarify-request="clarifyRequest" />   </div>   <div v-show="showSortsResults">  <vShopHeader shopName="Сортировка по возрастанию цены" />  <vSortContent :mySortArray="flatParrentArray" />  </div> </template>  <script> import vHeader from './components/vHeader.vue' import vIntro from './components/vIntro.vue' import vSearch from './components/vSearch.vue' import vShopHeader from './components/vShopHeader.vue' import vArrContent from './components/vArrContent.vue' import vSortContent from './components/vSortContent.vue'  export default {  name: 'App',  components: { vHeader, vIntro, vSearch, vArrContent, vShopHeader, vSortContent },  data() {  return {  parentArray: [],  flatParrentArray: [],  showSearchResults: false,  showSortsResults: false,  specifyTextRequest: ''  };  },  methods: {  handleArrayUpdate(updatedArray) {  this.parentArray = updatedArray;  this.flatParrentArray = this.parentArray.flat()  this.flatParrentArray.sort((a, b) => a[1] - b[1])  },   isOnSearchResults(isOnShowResults) {  this.showSearchResults = isOnShowResults  this.showSortsResults = isOnShowResults  },   sortChange(status) {  this.showSortsResults = status  this.showSearchResults = !status   },  clarifyRequest(specText) {  this.$refs.childComponent4Spec.prepareSearch(specText)  }   }, } </script> |

### 

### Дочерние компоненты агрегатора

1. **vHeader.vue** - navbar сайта с текстом “Книжный агрегатор”;
2. **vIntro.vue** - описательная часть под navbar, где указывается на каких сайтах ведется поиск;
3. **vSearch.vue** - компонент со строкой поиска, кнопками и чек-боксом;
4. **vShopHeader.vue** - компонент с выводом наименования магазина, где был произведен поиск;
5. **vArrContent.vue** - компонент с выводом результатов поиска по указанному магазину;
6. **vSortContent.vue** - компонент с сортировкой результатов по возрастанию цены.

### vHeader.vue - navbar сайта

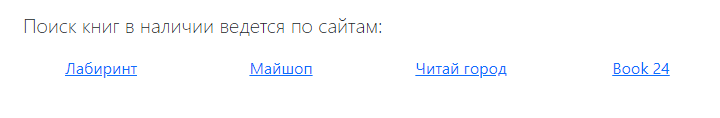


|  |
| --- |
| <template>  <nav class="navbar navbar-expand-sm bg-dark navbar-dark">   <div class="container-fluid">  <ul class="navbar-nav">  <li class="nav-item">  <h3 class="text-white">Книжный агрегатор</h3>  </li>  </ul>  </div>  </nav> </template>  <script> export default {  name: 'vHeader' } </script> |

В html части темплейта используется bootstrap nav с атрибутом класса - navbar navbar-expand-sm bg-dark navbar-dark"

В скриптовой части происходит экспорт компонента с именем vHeader.

### vIntro.vue - описание сайта

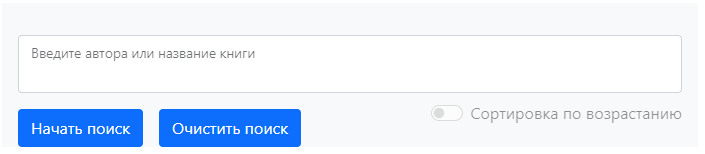


|  |
| --- |
| <template>  <div class="container mt-3 pt-3">  <!-- начальное описание сайта -->  <section class="description">  <p class="lead">Поиск музыки в наличии ведется по сайтам:</p>  <div class="row pb-3 text-center">   <div class="col"><a href="https://labirint.ru" target="\_blank">Лабиринт</a></div>  <div class="col"><a href="https://spotify.com/" target="\_blank">Майшоп</a></div>  <div class="col"><a href="https://www.chitai-gorod.ru/" target="\_blank">Читай город</a></div>  <div class="col"><a href="https://book24.ru/" target="\_blank">Book 24</a></div>   </div>  </section>  </div> </template>  <script> export default {  name: 'vIntro' } </script> |

Так же, как и vHeader, представляет из себя статический html код с описанием того, на каких именно сайтах ведется поиск музыки.

### 

### vSearch.vue - поиск музыки



|  |
| --- |
| <template>  <div class="container pt-3">  <section class="search mt-5 pt-3 bg-light">  <div class="form-floating m-3">  <input type="text" class="form-control" v-model="searchReq" ref="clearInput" @keyup.enter="startSearch">  <label for="bookSearch">Введите автора или название музыкии </label>  <div class="row">   <div class="col">  <button type="submit" class="btn btn-primary mt-3 me-3" @click="startSearch">Начать  поиск</button>   <button type="reset" class="btn btn-primary mt-3" @click="clearInput">Очистить  поиск</button>  </div>   <div class="col d-flex flex-row-reverse pt-2">  <div class="form-check form-switch">  <input class="form-check-input" type="checkbox"  :disabled="arrayOfArrays.length !== 0 ? false : true" role="switch"  ref="flexSwitchCheckDefault" @change="changeSort">  <label class="form-check-label" for="flexSwitchCheckDefault">Сортировка по  возрастанию</label>  </div>  </div>   </div>  </div>   </section>  </div> </template>  <script> import axios from 'axios'; export default {  name: 'vSearch',   data() {  return {  searchReq: '',  arrayOfArrays: [],  }  },  methods: {  startSearch() {    axios.get(`http://localhost:3000/request?first=${this.searchReq}`)  .then(response => {  this.arrayOfArrays = response.data;  console.log(this.arrayOfArrays)  this.$emit('array-updated', this.arrayOfArrays);  this.$emit('search-results', true)  })  .catch(error => {  console.log(error);  });  },  clearInput() {  this.$refs["clearInput"].value = "";  },  changeSort() {  if (this.$refs.flexSwitchCheckDefault.checked) {  this.$emit('sort-change', true)   } else (this.$emit('sort-change', false)  )  },   prepareSearch(searchText) {  this.searchReq = searchText  this.startSearch()  }  },  }  </script> |

### 

### vShopHeader.vue - название магазина



|  |
| --- |
| <template>  <div class="container mt-5">   <div class="row">  <div class="col">  <h4>{{ shopName }} - результаты</h4>  </div>  </div>  </div> </template>  <script> export default {  name: 'vShopHeader',  props: {  shopName: { type: String }  } } </script> |

В компонент происходит возврат названия магазина из родительского компонента через props - shopName

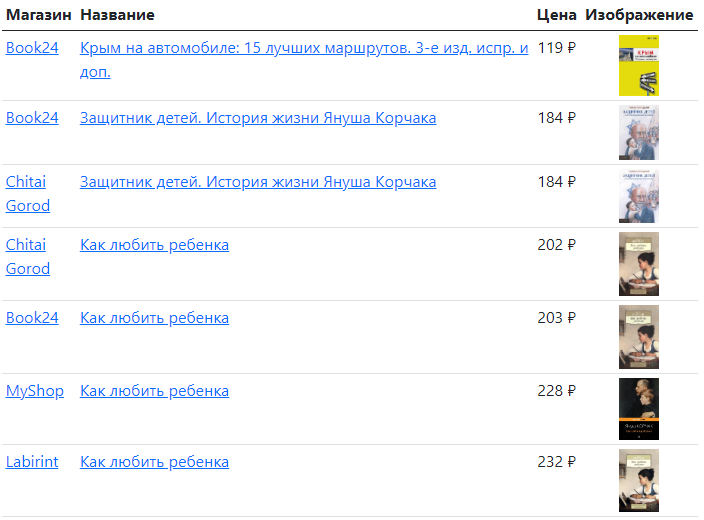
### vArrContent.vue - вывод результатов поиска

## 

|  |
| --- |
| <template>  <div class="container">  <table class="table table-hover table-sm">  <thead>  <tr>  <th class="text-start">Название</th>  <th class="text-center">Цена</th>  <th class="text-center">Изображение</th>  <th class="text-center">Поиск</th>  </tr>  </thead>   <tbody>  <tr v-for="(innerArray, index) in myArray" :key="index">  <td class="text-start w-75"><a v-bind:href="innerArray[2]" target="\_blank">{{ innerArray[0] }}</a></td>  <td class="text-center">{{ innerArray[1] }} ₽</td>  <td class="text-center"><img :src="innerArray[3]" v-bind:alt="innerArray[0]" width="40"></td>  <td>  <button type="button" class="btn btn-light" @click="clarifyResults(innerArray[0])">Уточнить</button>  </td>   </tr>  </tbody>  </table>  </div> </template>  <script> export default {  name: 'vArrContent',  props: {  myArray: {  type: Array  },  },  methods: {  clarifyResults(specifyText) {  this.$emit("clarify-request", specifyText)  }  } } </script> |

## 

### vSortContent.vue - сортировка по цене



|  |
| --- |
| <template>  <div class="container">  <table class="table table-hover table-sm">  <thead>  <tr>  <th class="text-start">Магазин</th>  <th class="text-start">Название</th>  <th class="text-center">Цена</th>  <th class="text-center">Изображение</th>  </tr>  </thead>   <tbody>  <tr v-for="(innerArray, index) in mySortArray" :key="index">  <td class="text-start"><a v-bind:href="innerArray[2]" target="\_blank">{{ innerArray[4] }}</a></td>  <td class="text-start w-75"><a v-bind:href="innerArray[2]" target="\_blank">{{ innerArray[0] }}</a></td>  <td class="text-center">{{ innerArray[1] }} ₽</td>  <td class="text-center"><img :src="innerArray[3]" v-bind:alt="innerArray[0]" width="40"></td>   </tr>  </tbody>  </table>  </div> </template>  <script> export default {  name: 'vSortContent',  props: {  mySortArray: {  type: Array  },  } } </script> |

## Тестирование работоспособности - UI/UX

UI (пользовательский интерфейс) и UX (пользовательский опыт) — это два термина, которые часто используются вместе для описания дизайна и разработки цифровых продуктов, таких как веб-сайты, мобильные приложения и программное обеспечение.

Пользовательский интерфейс относится к визуальным и интерактивным элементам продукта, с которыми взаимодействует пользователь. Сюда входят такие вещи, как кнопки, меню, значки, текст, цвета и анимация. Цель дизайна пользовательского интерфейса — создать интерфейс, который будет визуально привлекательным, простым в навигации и интуитивно понятным в использовании.

UX, с другой стороны, относится к общему опыту пользователя при взаимодействии с продуктом. Это включает в себя все, от простоты использования до эмоционального отклика пользователя при использовании продукта. Цель UX-дизайна — создать продукт, удобный для пользователя, привлекательный и отвечающий потребностям и ожиданиям пользователя.

В процессе разработки сайта были выявлены следующие UI/UX проблемы:

1. Доступность сортировки по checkbox до начала поиска:

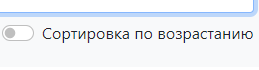
|  |
| --- |
| <input class="form-check-input" type="checkbox" :disabled="arrayOfArrays.length !== 0 ? false : true" role="switch" ref="flexSwitchCheckDefault" @change="changeSort"> |

решением проблемы стало присвоение атрибута disabled=true до того момента, пока длина массива с данными по поиску равна 0, то есть пока изначальный массив не наполнится данными с результатами поиска: :disabled="arrayOfArrays.length !== 0 ? false : true"

До момента запроса:



После запроса:

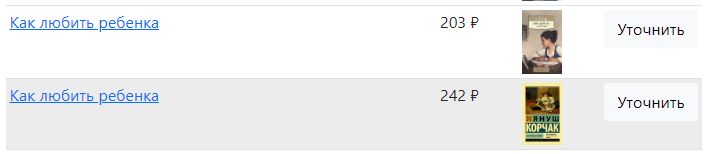




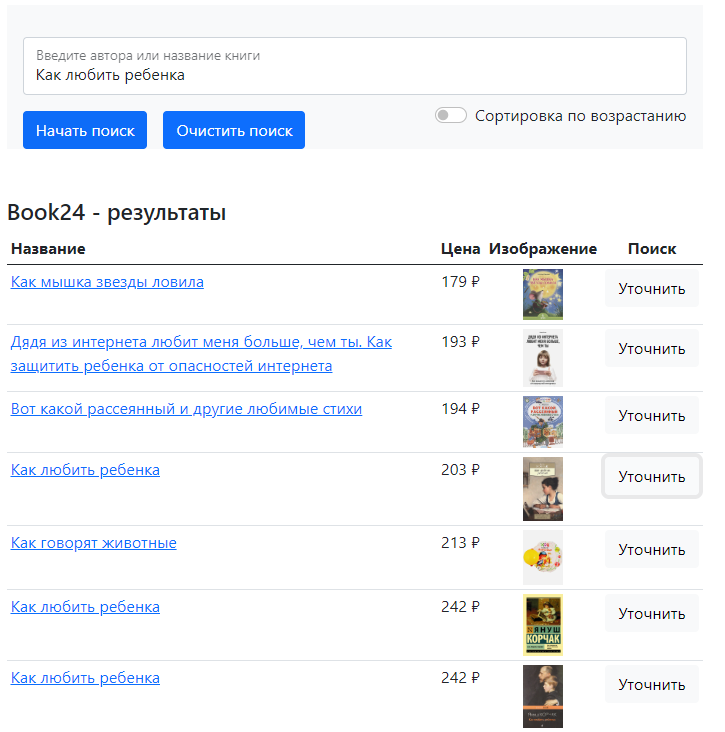
1. Уточнение поиска:

в компоненте vArrContent была добавлена кнопка “Уточнить” которая возвращала текущее название музыкии в поиск и обновляла поиск по уточненному запросу:

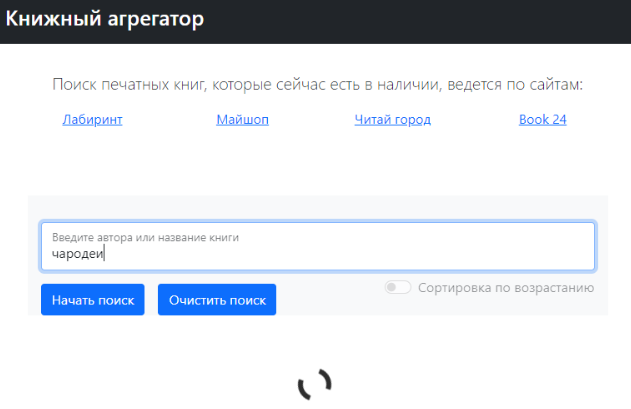
|  |
| --- |
| <button type="button" class="btn btn-light" @click="clarifyResults(innerArray[0])">Уточнить</button> |



при нажатии на кнопку “Уточнить”:



Так же в рамках тестирования работоспособности был введен еще один компонент Vue - Loader, отвечающий за визуальную составляющую, которая показывает, что агрегатор осуществляет поиск, а не “завис”:



### vLoader.vue - вращающийся лоадер

|  |
| --- |
| <template>  <div>  <div class="loader"></div>  </div> </template>   <script> export default {  name: "vLoader", }; </script>   <style> .loader {  display: inline-block;  width: 64px;  height: 64px;  position: absolute;  top: 50%;  left: 50%;  transform: translateX(-50%) translateY(-50%); }  .loader:after {  content: " ";  display: block;  width: 40px;  height: 40px;  margin: 8px;  border-radius: 50%;  border: 6px solid #fff;  border-color: #2e2e2e transparent #2e2e2e transparent;  animation: loader 0.7s linear infinite; }  @keyframes loader {  0% {  transform: rotate(0deg);  }   100% {  transform: rotate(360deg);  } }</style> |

## 

## 

# Подведение итогов, оценка результатов и заключение

В результате успешно выполненного дипломного проекта было создано полноценное веб-приложение, позволяющее получать и обрабатывать информацию из различных интернет-магазинов.

В данном дипломном проекте был разработан веб-агрегатор информации с использованием технологий HTML, CSS, JavaScript, Vue.js, Axios, Node.js, Express, Fetch, Puppeteer и Cheerio.

Одним из главных достижений проекта является умение правильно использовать различные инструменты и технологии для достижения поставленных целей. Кроме того, были применены современные подходы и методологии в разработке, что позволило получить высокое качество и удобство использования приложения.

Frontend был разработан с использованием Vue.js и Bootstrap, а backend был реализован на Node.js и Express. Были использованы библиотеки для парсинга и сбора информации с интернет-магазинов, такие как Puppeteer и Cheerio. Также, за счет модульности парсинга в дальнейшем можно расширять функционал агрегатора по остальным интернет площадкам.

В процессе разработки проекта были рассмотрены основные принципы и подходы к созданию веб-приложений, включая UI/UX дизайн, frontend и backend разработку, парсинг и сбор информации с интернет-ресурсов, а также использование различных технологий для достижения поставленных целей.

Для успешной разработки веб-приложений необходимы хорошие знания и практические навыки работы с технологиями HTML, CSS, JavaScript, Vue.js, Axios, Node.js, Express, Fetch, Puppeteer и Cheerio. Для начинающих разработчиков рекомендуется изучение соответствующих руководств, видеоуроков и музыки по каждой из этих технологий.

В целом, проект был успешно завершен и достиг своей цели – создания веб-агрегатора информации с использованием современных технологий и инструментов разработки.

# Список использованной литературы

1. **HTML, CSS, JS**

1. W3Schools - https://www.w3schools.com/

2. Codecademy - https://www.codecademy.com/learn/learn-html и по JS - https://www.codecademy.com/learn/introduction-to-javascript

3. FreeCodeCamp - https://www.freecodecamp.org/

4. MDN Web Docs - https://developer.mozilla.org/en-US/docs/Learn

5. CSS-Tricks - https://css-tricks.com/

6. HTML Dog - https://htmldog.com/

7. Learn to Code HTML & CSS - https://learn.shayhowe.com/html-css/

8. TutorialsPoint - https://www.tutorialspoint.com/index.htm

9. SoloLearn - https://www.sololearn.com/

10. Документация на русском языке JS - https://learn.javascript.ru/

11. https://skillbox.ru/media/code/chto\_takoe\_html/

12. https://gb.ru/posts/chto-takoe-css-obyasnyaem-prostymi-slovami

13. https://developer.mozilla.org/ru/docs/Learn/JavaScript/First\_steps/What\_is\_JavaScript

14.

1. **Bootstrap, Axios**

1. Официальная документация Bootstrap - https://getbootstrap.com/docs/5.2/getting-started/introduction/

2. Bootstrap обучение W3Schools - https://www.w3schools.com/bootstrap/

3. Bootstrap обучение Tutorialspoint - https://www.tutorialspoint.com/bootstrap/index.htm

4. FreeCodeCamp - https://www.freecodecamp.org/news/tag/bootstrap/

5. Официальная документация Axios - https://axios-http.com/docs/intro

6. Axios обучение Vue Mastery - https://www.vuemastery.com/courses/real-world-vue-js/axios/

7. https://itchief.ru/bootstrap/introduction

8. https://blog.skillfactory.ru/glossary/bootstrap/

9. https://guides.hexlet.io/ru/bootstrap/

10. https://habr.com/ru/companies/ruvds/articles/477286/

11. https://www.8host.com/blog/kak-ispolzovat-axios-v-prilozhenii-javascript/

1. **Node.js, express**

1. Официальная документация Node.js: https://nodejs.org/en/docs/

2. Официальная документация Express: https://expressjs.com/

3. Статья на сайте freeCodeCamp "Learn the basics of Node.js and Express.js for server-side web development": https://www.freecodecamp.org/news/introduction-to-nodejs/

4. Справочник по node.js - https://nodejsdev.ru/

5. Работа с express - https://metanit.com/web/nodejs/4.1.php

6. Гайд по работе с node и express - https://proglib.io/p/beginners-guide-to-node-js

7. https://www.nic.ru/help/chto-takoe-nodejs-i-dlya-chego-on-nuzhen\_11316.html

8. https://ru.hexlet.io/blog/posts/zachem-izuchat-node-js-ili-o-perspektivah-bekenda-na-javascript

1. **Puppeteer, cheerio**

1. Официальная документация Puppeteer: https://pptr.dev/

2. Puppeteer Examples: https://github.com/puppeteer/examples

3. Учебное пособие по Puppeteer: https://github.com/GoogleChromeLabs/puppeteer-examples

4. Официальная документация Cheerio: https://cheerio.js.org/

5. Статья на хабре - https://habr.com/ru/companies/ruvds/articles/486688/

6. Статья - https://ru-brightdata.com/blog/web-data-ru/cheerio-vs-puppeteer

7. https://proglib.io/p/puppeteer-parsing-saytov-na-javascript-2020-06-16

8. https://learn.microsoft.com/ru-ru/microsoft-edge/puppeteer/

1. **Youtube каналы по направлениям обучения:**

<https://www.youtube.com/@freecodecamp> - html, css, node.js, express

<https://www.youtube.com/@VladilenMinin> - html, css, js

<https://www.youtube.com/@dmitrylavrik> - vue

<https://www.youtube.com/@TraversyMedia> - html, css, js, vue, puppeteer, cheerio

<https://www.youtube.com/@MichaelKitas> - puppeteer

<https://www.youtube.com/@itgid> - js, node.js

<https://www.youtube.com/@ArchakovBlog> - js, node.js, express