

Дипломный проект: Разработка онлайн-словаря вариантов немецкого языка

**Специальность: Frontend-программист “Цифровые профессии”, Geekbrains**

**Щербак Сергей Андреевич**

**Санкт-Петербург, 2024**

СОДЕРЖАНИЕ:

1. Введение.
2. Теоретическая часть.
   1. Немецкий язык
      1. Краткие сведения о немецком языке.
      2. Диалекты и региональные варианты немецкого языка.
      3. Особенности немецкого языка в ГДР.
   2. Web-приложение: теоретические основы.
   3. Инструменты проекта
      1. Язык гипертекстовой разметки HTML.
      2. Таблицы стилей CSS.
      3. Язык программирования JavaScript.
         1. Общие сведения о языке.
         2. Фреймворки и библиотеки JavaScript.
      4. СУБД данных SQLite.
      5. Программные продукты и инструменты web-разработки.
      6. Git.
3. Практическая часть. Разработка проекта.
   1. Анализ задачи.
   2. Первоначальная разработка проекта.
   3. Разработка и создание дизайна для страницы проекта.
   4. Наполнение базы данных.
   5. Тестирование сайта.
   6. Размещение проекта в интернете
4. Заключение.
5. Список использованной литературы.
   * + 1. Введение.

Тема настоящего дипломного проекта – разработка онлайн-приложения «Онлайн-словарь вариантов немецкого языка», исследование особенностей организации процесса разработки веб-приложений на примере разработки онлайн-словаря.

Целью данного дипломного проекта является изучение особенностей разработки веб-приложений и разработка проекта онлайн-приложения «Онлайн-словарь вариантов немецкого языка».

Основная цель онлайн-приложения — предоставить платформу, знакомящую пользователей со словарём вариантов и диалектов немецкого языка. Немецкий язык – основной литературный язык Германии, Австрии, Лихтенштейна, один из государственных языков Швейцарии и Люксембурга, язык

Задачи, решаемые в ходе работы над дипломным проектом:

1. Изучить литературу, касающуюся темы исследования.

2. Протестировать и отладить разработанное приложение.

Инструменты, использованные при разработке проекта:

* Git;
* JetBrains WebStorm.

Данный проект имеет важное значение для современной организации и распространения информации в сфере культуры и досуга.

Сохранение и продвижение культурного наследия является важной задачей для любого региона. Группа культурно-досуговых центров Бичурского района играет ключевую роль в организации и проведении различных культурных мероприятий и образовательных программ. Однако, отсутствие эффективной платформы для обмена информацией и взаимодействия с разными аудиториями ограничивает их потенциал и влияние.

информационного портала, который позволит улучшить коммуникацию и взаимодействие между группой культурно-досуговых центров Бичурского района и различными заинтересованными сторонами.

Задачи для достижения данной цели необходимо выполнить следующие:

* Изучение потребностей и ожиданий аудитории группы культурно-досуговых центров Бичурского района.
* Анализ информационных потребностей и существующих проблем обмена информацией между организацией и заинтересованными сторонами.
* Проектирование и разработка информационной архитектуры портала.
* Создание привлекательного и функционального дизайна портала, соответствующего имиджу и целям группы культурно-досуговых центров Бичурского района.
* Реализация основных функций портала, включая публикацию анонсов мероприятий, расписание, новости, галерею и возможности взаимодействия с аудиторией.
* Тестирование и отладка разработанного портала.
* Оценка эффективности портала на основе обратной связи от пользователей и сбора данных о его использовании.

Разработка информационного портала группы культурно-досуговых центров Бичурского района позволит повысить эффективность коммуникации с аудиторией, а также увеличить видимость и привлекательность культурных мероприятий и программ, предоставляемых этими центрами. Благодаря нашему проекту, жители и посетители Бичурского района смогут узнавать о последних событиях, расписании мероприятий и с легкостью взаимодействовать с организаторами.

В данном проекте опробовал следующие роли:

* Дизайнер;
* Fullstack-разработчик;
* Верстальщик;
* Программный архитектор;

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

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

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

* Веб-скрэпинг: агрегатор будет собирать данные из различных книжных онлайн-магазинов. Данные будут включать названия книг, имена авторов, описания, вид обложки и цену;
* 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;
* 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;
* 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:

* 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;
* 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;
* Использование fetch, Puppeteer и Cheerio для получения данных из книжных интернет-магазинов, которые не предоставляют API;
* 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;
* 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:

* 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;
* 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;
* 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;
* 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;
* UX-дизайнеры могут помочь с разработкой удобного и визуально привлекательного интерфейса для приложения. Они также могут предоставить рекомендации по передовым методам использования и доступности;
* 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 книг, которые хотят сравнить цены и наличие в нескольких книжных интернет-магазинах.

Состав команды: ФИО (Тестировщик)

* + - 1. Теоретическая часть.
  1. Немецкий язык.
     1. Краткие сведения о немецком языке.

Немецкий язык (нем. *«Deutsch, Deutsche Sprache»*) принадлежит к западногерманской группе германских языков и является одним из самых распространенных языков в мире. Германская языковая семья, к которой относится немецкий, известна своей богатой историей и значительным влиянием на культуру и науку. Немецкие авторы, философы и ученые внесли значительный вклад в мировую литературу, философию и науку, и их работы переведены на множество языков, включая русский.

Одной из особенностей немецкого языка является его грамматическая структура. Немецкий язык обладает грамматическим родом, числовыми и падежными формами, а также строгими правилами согласования между подлежащим и сказуемым. Благодаря этим особенностям, немецкий язык оправдывает репутацию языка, в котором словообразование и синтаксис должны быть внимательно проверены и соблюдены.

Немецкий язык также известен своим богатым, обширным словарным запасом, состоящим из слов различного происхождения. Он включает основной немецкий лексикон, наследованный от германской языковой группы, а также лексику, заимствованную из других языков, таких как латинский, французский, английский и другие. В немецком языке есть множество слов и выражений, которые воплощают разнообразные концепции и идеи, что отражает культурные и интеллектуальные традиции тех, кто говорит на этом языке.

Этимология слов в немецком языке может быть очень интересной, поскольку она иногда позволяет проследить исторические и культурные связи между разными языками. Например, многие слова в немецком языке имеют общую основу с английскими словами, что объясняется их общим происхождением от германского языка. Семантика — другой важный аспект лексикологии. Она изучает значения слов и способы их организации в лексической системе. В немецком языке существует множество понятийных полей, где группы слов связаны общими значениями. Например, в поле "фрукты" собраны слова, обозначающие различные виды фруктов, такие как "*Apfel*" (яблоко), "*Banane*" (банан), "*Orange*" (апельсин) и т.д. Кроме этого, в немецком языке существуют синонимы и антонимы, которые обогащают разнообразие выражения. Например, слова "fröhlich" (веселый) и "glücklich" (счастливый) имеют схожее значение, а слова "heiß" (горячий) и "kalt" (холодный) являются примером антонимов.

Также стоит отметить, что в немецком языке много сложных слов, образованных путем соединения нескольких лексических элементов. Например, слово "Dampfschifffahrtskapitän" (капитан парохода) состоит из трех основных частей, каждая из которых вносит свой смысл к общему значению слова.

Немецкий язык также богат различными группами сленга и жаргона, которые могут варьироваться в зависимости от региона и социальной группы. Это создает множество вариантов и выражений в рамках немецкого языка и позволяет говорящим выразить свою индивидуальность и принадлежность к определенной группе.

Важно отметить, что немецкий язык часто используется в научной и академической среде. Многие важные научные работы и исследования изначально публикуются на немецком языке, и изучение немецкого может открыть дополнительные возможности для научного и интеллектуального развития. Также стоит отметить богатство лексического состава немецкого языка. В немецком есть множество слов, которые имеют свои аналоги в других языках, но при этом они часто обладают уникальными нюансами и оттенками значения. Это делает немецкий язык интересным для изучения и позволяет расширить словарный запас.

Еще одной интересной особенностью немецкого языка является наличие диалектов. В различных регионах Германии, Швейцарии и Австрии употребляются различные диалекты, которые могут отличаться по произношению, лексике и грамматике. Это может создавать некоторые затруднения для общения между представителями разных регионов, но в то же время придает языку большее эстетическое разнообразие.

Немецкий язык играет важную роль в международном бизнесе и туризме. Германия является крупной экономической державой, и знание немецкого языка может предоставить широкий спектр возможностей для карьерного роста и открыть двери к сотрудничеству с немецкими компаниями. В заключение, немецкий язык имеет свои уникальные особенности, которые делают его интересным для изучения и использования. Он является частью богатой культурной и языковой традиции и имеет большое значение в науке, литературе, бизнесе и международных отношениях. Изучение немецкого языка может быть увлекательным и позволить расширить границы коммуникации и культурного обмена.

* + 1. Диалекты и региональные варианты немецкого языка.

Немецкий язык обладает множеством диалектов и региональных вариантов, которые различаются по произношению, лексике и грамматике. Эти различия могут быть настолько значительными, что иногда даже представители разных регионов могут испытывать трудности во взаимопонимании.

Одним из известных диалектов немецкого языка является баварский, который наиболее распространен в южной части Германии и Австрии. Баварский диалект характеризуется особым произношением, особенно в звуковом сочетании "ch". Например, слово "ich" (я) в стандартном немецком произносится [ɪç], в то время как в баварском оно звучит как [i]. Другой известный диалект - алеманский, преимущественно употребляется в южной части Германии, Швейцарии и Австрии. Алеманский диалект также имеет свои особенности в произношении и грамматике. Например, в алеманском диалекте слово "ich" произносится как [ɪg] или [iɡ]. Также стоит упомянуть франконский диалект, распространенный в Северной Баварии и некоторых частях ФРГ, Вюртембергский диалект, используемый в Штутгарте и окрестностях, а также рейнский диалект, характерный для Рейнской области и численных приграничных регионов.

Каждый диалект имеет свои уникальные особенности, которые отражают богатство и многообразие немецкого языка. Они также могут включать в себя специфические лексические выражения, фразы и даже грамматические конструкции, которые могут отличаться от стандартного немецкого языка. Однако несмотря на эти различия, стандартный немецкий язык всегда остается общим средством коммуникации между говорящими на разных диалектах. Диалекты и региональные варианты немецкого языка придают языку его уникальность и культурное наследие. Они являются важным аспектом изучения немецкого языка и позволяют лучше понять культуру и традиции различных регионов, где используется немецкий язык.

Каждый диалект и региональный вариант немецкого языка имеет свою историю и культурные корни. Их ареалы определяются географическими, историческими и социокультурными факторами. Различные диалекты и языковые варианты часто отражают историческую сложность и разнообразие регионов, в которых они зародились и сохранились.

Диалекты немецкого языка также находят отражение в литературе, музыке и других культурных проявлениях. Множество писателей, поэтов и композиторов использовали диалекты для выражения своего личного стиля и отражения культурного наследия своих родных регионов. Это позволяет нам узнать больше о различных культурах и традициях внутри Германии и других немецкоязычных стран. Однако не следует забывать, что стандартный немецкий язык (Hochdeutsch) остается государственным и общим для всех немецкоязычных регионов. Это язык образования, делового общения и официального общения в Германии и других странах. В школах и университетах преподаются основы стандартного немецкого языка, а также ведется его стандартизация через учебники и правила грамматики. Стандартный немецкий язык также используется в СМИ и в официальных документах.

Немецкий язык богат диалектами и региональными вариантами, которые отражают культурное разнообразие и историческое наследие немецкоязычных регионов. Понимание и изучение этих различий помогает нам более глубоко погрузиться в немецкую культуру и лучше понять многообразие языка и стилей, которые сопутствуют его использованию в разных регионах. Помимо описанных ранее диалектов, существует множество других местных вариантов немецкого языка, которые имеют свою уникальность. Например, в Северной Германии распространен верхненемецкий диалект, который включает в себя региональные варианты, такие как нижненемецкий, кельнский и гессенский диалекты, а Восточная Германия характеризуется наличием сильной прослойки лексических и грамматических особенностей, которые отличаются от западных региональных вариантов.

* + 1. Особенности немецкого языка в ГДР.

Особенности немецкого языка в Германской Демократической Республике (ГДР) отличались от использования немецкого языка в других германоязычных регионах. Германская Демократическая Республика (ГДР) была социалистическим государством, созданным в 1949 году на территории восточной части Германии после окончания Второй мировой войны. ГДР существовала до 1990 года, когда объединилась с ФРГ, ставшей частью обновленной Германии. В течение своего существования ГДР была социалистическим государством, в стране проводились коллективизация сельского хозяйства, национализация промышленности, централизованное планирование экономики. Политическая система страны была однопартийной, руководящую роль играла СЕПГ – Социалистическая единая партия Германии. Падение Берлинской стены в 1989 году и объединение Германии в 1990 году означало окончание существования ГДР. Во время существования ГДР (1949-1990), язык испытал влияние социально-политических факторов, что привело к некоторым особенностям в его использовании.

1. Социалистический словарь: В ГДР стремились к созданию своего уникального лексикона, отличавшегося от использования немецкого языка на западе. Были введены новые термины и выражения, связанные с социалистической и марксистско-ленинской идеологией. Некоторые слова имели специфический семантический оттенок, относящийся к социалистической системе.

2. Использование советской нормы: Влияние советского стиля и норм русского языка также было заметно в использовании немецкого языка в ГДР. Например, применение конструкций, подвижных приставок и словосочетаний, аналогичных русским языковым образцам.

3. Отличия в произношении: Некоторые говорящие на немецком языке в ГДР могли иметь отличия в произношении некоторых звуков и интонаций из-за региональной вариации или влияния русского языка.

4. Ограничения на использование слов и выражений: В определенные периоды существования ГДР существовали цензурные ограничения на использование определенных слов и выражений. Власти стремились контролировать язык и его использование, чтобы поддерживать специфическую политическую атмосферу.

5. Влияние других региональных вариантов немецкого языка: В ГДР встречались различные диалекты и варианты немецкого языка. Некоторые из них могли иметь свои уникальные особенности в произношении, грамматике и лексике.

6. Идеологическая терминология: В ГДР было распространено использование специфической идеологической терминологии, связанной с коммунистической идеологией. Например, часто употреблялись термины, такие как "Klassenkampf" (борьба классов), "Sozialismus" (социализм) и "Realsozialismus" (реальный социализм).

7. Влияние русского языка: благодаря близким связям с СССР, русский язык оказал значительное влияние на немецкий язык в ГДР. Введение советской нормы и использование русских заимствований сделало некоторые выражения и конструкции уникальными для ГДР.

8. Административная лексика: в связи с социалистической организацией государства и экономики, в ГДР была развита специфическая административная лексика. Были введены новые термины и выражения, связанные с работой государственных органов, планированием и контролем экономики.

9. Культурные отличия: в ГДР поддерживалась специфическая культура и искусство. Это отразилось и в использовании языка. В ГДР развивалась своя литературная традиция, к которой относились такие выдающиеся писатели, как Бертольд Брехт и Штефан Гейслер. Их произведения и стиль письма имели свои особенности, которые отличались от используемого на Западе.

10. Устаревшие выражения: со временем, после объединения Германии, некоторые выражения и фразы, использовавшиеся в ГДР, стали устаревшими в современном немецком языке. Однако, они могут быть все же встречены в текстах, документах и литературе, отражая исторический контекст и время создания.

В целом, немецкий язык в ГДР имел свои уникальные особенности, вызванные социально-политическим контекстом времени и стремлением властей формировать специфическую идеологическую и культурную среду. Эти особенности создавали определенную отличительную атмосферу в использовании языка в ГДР.

Стоит отметить, что многие из этих особенностей постепенно ушли в прошлое после объединения Германии в 1990 году. Однако, ряд слов и выражений, используемых в ГДР, продолжает присутствовать в современном немецком языке, особенно в восточных регионах Германии.

Общение и изучение немецкого языка позволяют погрузиться в его богатую лексику и открыть для себя множество интересных слов и выражений. Лексикология немецкого языка предлагает множество увлекательных аспектов для изучения и исследования. В целом, изучение лексикологии и семантики немецкого языка позволяет развить более глубокое понимание его словарного запаса и организации слов. Кроме того, это может помочь в изучении немецкого языка и повышении своего уровня владения им.

* 1. Web-приложение: теоретические основы.

Веб-приложение: теоретические 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:

• 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;

• Социальные сети: веб-приложения могут предоставлять службы социальных сетей, которые позволяют пользователям создавать профили, обмениваться контентом и взаимодействовать с другими пользователями;

• Продуктивность. Веб-приложения могут предоставлять инструменты для совместной работы, управления задачами и планирования проектов, позволяя командам эффективно и продуктивно работать вместе;

• 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 книги 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 книг, а также базу данных с информацией о книгах 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 книг 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 книгах 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:

• Пользовательский интерфейс: пользовательский интерфейс будет визуальным представлением веб-приложения и будет включать такие элементы, как панели поиска, страницы результатов и страницы сведений о книгах;

• HTML — это язык разметки, используемый для структурирования содержимого веб-страниц. HTML используется для определения структуры пользовательского интерфейса, включая заголовки, абзацы, списки и таблицы;

• 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;

• 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;

• 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;

• 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 книги 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 книге или когда меняются цены на книги.

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 книгах, ценах и других деталей.

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

• 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;

• 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 книг, 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 книги 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.

* 1. Инструменты проекта.
     1. Язык гипертекстовой разметки HTML.

HTML (HyperText Markup Language) — это язык разметки, используемый для создания и структурирования веб-страниц. Он состоит из набора тегов, которые определяют различные элементы и их взаимодействие на странице. HTML является основным языком разметки веб-страниц и является частью разработки веб-сайтов. Он позволяет создавать структуру и содержимое страницы, определять иерархию элементов и добавлять функциональность через другие языки, такие как CSS и JavaScript. HTML (HyperText Markup Language) является языком разметки, используемым для создания веб-страниц. HTML был разработан в начале 1990-х годов Тимом Бернерсом-Ли (Tim Berners-Lee) и его коллегами в ЦЕРНе (Европейский центр физических исследований) в Женеве, Швейцария.

В то время, когда первые веб-страницы стали появляться, нужен был способ для представления текста и других элементов на странице таким образом, чтобы пользователи могли легко переходить по ссылкам и осуществлять поиск по информации. HTML был разработан как простой инструктивный язык разметки, который позволял разработчикам создавать структурированный контент с помощью тегов.

Первая версия HTML, известная как HTML 1.0, была выпущена в 1991 году. Она включала основные элементы разметки, такие как заголовки (h1, h2 и так далее), параграфы (p), списки (ul, ol), ссылки (a) и теги для форматирования текста (bold, italic). С течением времени были выпущены новые версии HTML, включая HTML 2.0, HTML 3.2, HTML 4.01 и HTML5, каждая из которых вносила дополнительные функции и возможности. В настоящее время HTML5 является последней версией и включает в себя множество новых элементов и атрибутов, которые позволяют разработчикам создавать более интерактивные и богатые по функционалу веб-страницы. HTML стал фундаментальным языком для разработки веб-сайтов и по-прежнему широко используется веб-разработчиками по всему миру.

Рассмотрим некоторые основные элементы HTML:

1. Теги: HTML-элементы обозначаются с помощью открывающего и закрывающего тегов. Например, <p>текст параграфа</p> задает параграф, а <h1>заголовок</h1> — заголовок первого уровня.

2. Элементы: HTML поддерживает различные элементы, такие как заголовки, параграфы, списки, таблицы, изображения и формы. Например, <ul> и <li> используются для создания неупорядоченного списка, а <img> для вставки изображений.

3. Атрибуты: Теги могут иметь атрибуты, которые предоставляют дополнительную информацию о элементе. Атрибуты обычно указываются в начале открывающего тега. Например, <a href="https://www.example.com"> URL -ссылка</a> задает гиперссылку с атрибутом href, указывающим адрес URL.

4. Структура: HTML-страница имеет определенную структуру. Она начинается с объявления DOCTYPE, которое указывает на версию HTML, и затем содержит <html>, <head> и <body>. <head> содержит метаданные страницы, такие как заголовок и ссылки на внешние стили CSS, а содержимое страницы размещается внутри <body>.

5. Стили и форматирование: HTML предоставляет базовую возможность стилизации с помощью атрибутов стиля, но более распространенное использование для этой цели - это использование CSS (Cascading Style Sheets). CSS позволяет изменять внешний вид HTML-элементов, определяя различные свойства, такие как цвет, шрифт, размер и расположение.

6. Верстка и компоновка: С помощью HTML вы можете создавать различные компоновки элементов на странице. Например, вы можете использовать таблицы (<table>), блочную модель (<div>) или сетку для распределения и организации содержимого.

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:

* <html>: Начальный тег для HTML-документа.
* <head>: Содержит метаданные документа, такие как заголовки страницы, ссылки на стили CSS, и другие данные, которые не отображаются на странице.
* <title>: Определяет заголовок документа.
* <body>: Определяет содержимое страницы, которое отображается в браузере.
* <div>: Определяет блок элементов на странице.
* <p>: Определяет абзац текста.
* <img>: Определяет изображение на странице.
* <a>: Определяет ссылку на другую страницу или ресурс.
* <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.

* + 1. Таблицы стилей CSS.

CSS (Cascading Style Sheets) — это язык стилей, используемый для определения внешнего вида (стилизации) элементов HTML на веб-странице. С помощью CSS вы можете изменять цвета, шрифты, размеры, раскладку и многое другое для создания эстетически приятных и привлекательных веб-страниц. CSS был разработан группой специалистов веб-разработки, включая Хакона Виума Лиебека (Håkon Wium Lie), Берта Боса (Bert Bos) и Тима Бернерс-Ли (Tim Berners-Lee). Работа по спецификации CSS началась в конце 1994 года, и первая редакция CSS была выпущена в 1996 году. С тех пор CSS претерпел множество изменений и стал одним из основных инструментов для стилизации веб-страниц.

Вот некоторые основные концепции CSS:

1. Селекторы: CSS использует селекторы для определения, к каким элементам HTML должны быть применены определенные стили. Например, вы можете выбрать все абзацы (p) или все элементы с определенным классом (#id или .class).

2. Свойства: Свойства CSS определяют внешний вид элементов. Например, свойство color устанавливает цвет текста, а font-size устанавливает размер шрифта. Существует огромное количество свойств CSS, которые позволяют настроить практически все аспекты элемента.

3. Значения: CSS-свойства имеют значения, определяющие конкретные параметры. Например: color: red; задает красный цвет текста, а font-size: 16px; устанавливает размер шрифта в 16 пикселей.

4. Каскадирование и наследование: Каскадирование в CSS означает, что стили могут быть определены несколько раз для одного элемента, и их влияние будет комбинироваться и применяться в порядке приоритета. Наследование означает, что определенные свойства стилей, примененные к родительскому элементу, также могут наследоваться дочерними элементами.

5. Единицы измерения: CSS поддерживает различные единицы измерения, такие как пиксели (px), проценты (%), em, rem и другие. Эти единицы используются для задания размеров, отступов, отступов и других значений в стилях.

6. Медиазапросы: Медиазапросы позволяют создавать стили, которые применяются только при определенных условиях экрана или устройства. Например, вы можете задать особый стиль для мобильного устройства или планшета, используя медиазапросы.

7. Бокс-модель: В CSS каждый элемент имеет бокс-модель, которая определяет пространство вокруг элемента. Бокс-модель включает в себя границу (border), отступы (margin) и внутренние отступы (padding). Вы можете использовать свойства CSS, чтобы настроить размеры и внешний вид каждого компонента бокс-модели.

8. Позиционирование: CSS предоставляет различные методы позиционирования элементов на веб-странице. С помощью свойства position вы можете задать, как элемент должен быть расположен - абсолютно, относительно или фиксированно. Вы также можете использовать свойства top, left, right, bottom для точного позиционирования элементов.

9. Анимации и переходы: CSS позволяет создавать анимации и переходы для элементов на веб-странице. Вы можете задать свойства анимации, такие как animation-name, animation-duration, animation-delay и другие, чтобы создать плавные переходы между состояниями элементов.

10. Верстка на сетке: CSS предоставляет мощные инструменты для создания респонсивной верстки на сетке. Вы можете разбить страницу на горизонтальные и вертикальные столбцы, используя свойства grid или flexbox, что обеспечивает гибкость в расположении элементов на веб-странице в зависимости от размера экрана.

CSS является важной частью разработки веб-страниц и позволяет создавать привлекательный и эстетически приятный дизайн. Путем применения правильных стилей и свойств вы можете предоставить удобный и легко читаемый контент для ваших пользователей.

CSS предоставляет мощные инструменты для улучшения внешнего вида веб-страницы, делая ее более привлекательной и понятной для пользователей. Он часто используется вместе с HTML для создания полноценных и красивых веб-сайтов. 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:

Каскадность - стили могут быть определены на разных уровнях, и значения, определенные на более высоком уровне, могут быть переопределены на более низком уровне.

Наследование - стили, определенные для родительского элемента, могут быть унаследованы его дочерними элементами.

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.

* + 1. Язык программирования JavaScript.
       1. Общие сведения о языке.

JavaScript - это высокоуровневый язык программирования, который широко используется для создания веб-приложений и интерактивных веб-сайтов. Он является интерпретируемым языком, который выполняется прямо в браузере пользователя, что делает его одним из основных инструментов для Front-end разработчиков. JavaScript поддерживает объектно-ориентированное, императивное и функциональное программирование, что делает его очень гибким и универсальным языком.

JavaScript широко применяется для добавления интерактивности на веб-страницы, выполнения асинхронных запросов к серверу (AJAX), создания анимаций, игр, веб-приложений и многого другого. Он также используется на сервере (Node.js) для создания бэкенд-части приложений. JavaScript имеет обширную стандартизированную библиотеку (JavaScript Standard Library), включающую в себя множество функций для работы с DOM (Document Object Model), работой с сетью, асинхронными операциями, обработки данных и многое другое. Кроме того, существуют множество фреймворков и библиотек, таких как React, Angular, Vue.js, которые помогают упростить разработку веб-приложений на JavaScript. JavaScript является одним из самых популярных языков программирования в мире и остается одним из самых востребованных навыков в IT-индустрии.

JavaScript также отличается тем, что является интерпретируемым языком программирования, что означает, что код выполняется по мере чтения, без предварительной компиляции. Это делает его очень гибким и удобным для быстрой разработки и тестирования кода. Еще одним важным преимуществом JavaScript является его кроссплатформенность, то есть он может выполняться на различных устройствах и операционных системах без изменений. JavaScript имеет множество различных фреймворков и библиотек, которые упрощают разработку сложных веб-приложений. Например, библиотека jQuery предоставляет удобные методы для работы с DOM, упрощая манипуляции с веб-страницами. Фреймворк React позволяет создавать компоненты пользовательского интерфейса, обеспечивая модульность и переиспользование кода. Angular предоставляет инструменты для создания одностраничных приложений. JavaScript также активно используется в разработке игр, благодаря фреймворкам и библиотекам, таким как Phaser.js и Three.js. Он также находит применение в разработке мобильных приложений с использованием фреймворков Cordova и React Native.

В целом, JavaScript - мощный и универсальный язык программирования, который остается одним из ключевых инструментов для создания современных веб-приложений и игр. Его широкие возможности и активное сообщество разработчиков делают его популярным выбором для многих проектов. JavaScript также активно применяется в различных областях разработки, таких как интернет-маркетинг и аналитика. Например, с помощью JavaScript можно реализовать сбор и анализ данных на веб-сайтах, отслеживать поведение пользователей, настраивать и оптимизировать маркетинговые кампании. Благодаря постоянному развитию и обновлениям языка, JavaScript становится все более мощным и функциональным инструментом для разработчиков. Введение новых стандартов языка, таких как ECMAScript 6 (или ES6) и последующих версий, позволяет использовать более современные и продвинутые возможности языка, такие как стрелочные функции, деструктуризация объектов, классы и многое другое.

Интересно отметить, что JavaScript не ограничивается только веб-разработкой. Он может использоваться и на серверной стороне, благодаря платформе Node.js, которая позволяет выполнять JavaScript код на сервере. Это открывает новые возможности для создания полноценных веб-приложений, работающих как на клиентской, так и на серверной стороне, с общим кодом на JavaScript. JavaScript – это универсальный и многофункциональный язык программирования, который продолжает развиваться и применяться в различных областях информационных технологий. Его популярность и востребованность делают его важным инструментом для многих разработчиков и компаний.

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:

* 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.
* Асинхронное программирование: JavaScript поддерживает асинхронное программирование, что позволяет разработчикам писать код, который не блокирует выполнение другого кода.
* Кроссплатформенная совместимость: JavaScript поддерживается всеми основными веб-браузерами, что делает его популярным выбором для веб-разработки.

Расширяемость: JavaScript можно расширить с помощью библиотек и фреймворков, таких как jQuery, React и Vue.js, которые упрощают разработку сложных веб-приложений.

* + - 1. Фреймворки и библиотеки JavaScript.

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:

Легковесность - Vue.js имеет небольшой размер и быстро загружается на страницу, что позволяет создавать быстрые и отзывчивые пользовательские интерфейсы.

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.

Простота в использовании - Vue.js легко устанавливается и используется даже начинающими разработчиками. Он также предоставляет простые и интуитивно понятные API и документацию.

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 — 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:

API на основе промисов: Axios построен на основе промисов JavaScript, что позволяет разработчикам писать асинхронный код, который легко читать и понимать.

Простой API: Axios предоставляет простой и интуитивно понятный API для выполнения HTTP-запросов с методами обработки запросов GET, POST, PUT и DELETE.

Перехватчики (Interceptors): Axios предоставляет функцию, называемую перехватчиками, которая позволяет разработчикам перехватывать и изменять HTTP-запросы и ответы до их отправки или получения.

Обработка ошибок (Error handling): Axios предоставляет мощную систему обработки ошибок, которая упрощает обработку сетевых ошибок и ошибок на стороне сервера.

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.

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:

* 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.
* Высокая производительность: Node.js построен на основе движка V8, который обеспечивает высокую производительность кода JavaScript.
* Большая экосистема пакетов: Node.js имеет большую и растущую экосистему пакетов с тысячами модулей и пакетов, доступных через диспетчер пакетов Node (npm).
* Кроссплатформенная совместимость: Node.js является кроссплатформенным, что означает, что он может работать в Windows, macOS и Linux.
* 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 — 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:

* Маршрутизация: Express предоставляет мощную систему маршрутизации, которая позволяет разработчикам определять маршруты для обработки входящих HTTP-запросов.
* 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.
* Механизмы шаблонов: Express поддерживает различные механизмы шаблонов, такие как Pug, EJS и Handlebars, что упрощает создание динамических HTML-страниц.
* Обработка ошибок: Express предоставляет мощную систему обработки ошибок, которая позволяет разработчикам обрабатывать ошибки последовательным и удобным для пользователя способом.
* 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.
  + 1. СУБД SQLite.

СУБД (Система Управления Базами Данных) - это программное обеспечение, которое обеспечивает организацию, хранение, управление и доступ к данным в базе данных. СУБД предоставляет средства для создания, изменения, удаления и извлечения данных, а также для выполнения различных операций над данными, таких как сортировка, фильтрация и агрегация.

СУБД предоставляет интерфейс для взаимодействия с базой данных с помощью структурированного языка запросов, такого как SQL (Structured Query Language). SQL позволяет программистам и администраторам баз данных выполнять операции по созданию и управлению таблицами, индексами, представлениями и другими составными элементами базы данных.

Преимущества использования СУБД включают:

1. Эффективное хранение данных: СУБД обеспечивает оптимальное использование дискового пространства, благодаря чему данные занимают меньше места и обрабатываются быстрее.

2. Безопасность: СУБД предоставляет механизмы для обеспечения безопасности данных, такие как права доступа и шифрование, что позволяет ограничить доступ к данным только авторизованным пользователям.

3. Целостность данных: СУБД обеспечивает соблюдение ограничений целостности данных, таких как ограничения на уникальность значений и ссылочные целостность, что гарантирует надежность и согласованность данных.

4. Многопользовательская поддержка: СУБД позволяет нескольким пользователям одновременно работать с базой данных, выполняя операции добавления, изменения и извлечения данных, при этом обеспечивая согласованность и последовательность выполнения операций.

Примеры популярных СУБД включают MySQL, Oracle Database, Microsoft SQL Server, PostgreSQL и SQLite. Каждая СУБД имеет свои особенности и подходит для разных сценариев использования в зависимости от требований проекта.

SQLite была разработана командой программистов во главе с Ричардом Хиппом. Хипп является создателем и основным разработчиком SQLite с момента ее начала в 2000 году. SQLite изначально разрабатывалась как небольшая, эффективная, встраиваемая СУБД для использования в приложениях с ограниченными ресурсами и поддержкой доступа через SQL. Эта легковесная и легко настраиваемая СУБД была предназначена для работы в автономном режиме, с возможностью хранения данных в одном файле без необходимости в отдельном сервере. SQLite быстро стала популярной и на данный момент является одной из наиболее широко используемых СУБД в мире. Благодаря своей простоте, эффективности и надежности SQLite активно применяется в различных сферах, включая мобильные приложения, веб-сайты и встроенную электронику.

SQLite – это легковесная встроенная СУБД (система управления базами данных), которая предоставляет надежное хранение и доступ к данным. Она отличается от большинства других СУБД тем, что база данных SQLite хранится в одном файле на диске и не требует отдельного сервера для своей работы.

Особенности SQLite:

1. Простота использования: SQLite имеет простой и понятный синтаксис языка запросов SQL. Это делает ее очень легкой в использовании, даже для начинающих разработчиков.

2. Переносимость: База данных SQLite может работать на различных платформах, включая Windows, macOS, Linux и мобильные операционные системы, такие как Android и iOS. Это обеспечивает высокую степень переносимости и позволяет использовать SQLite в различных типах приложений.

3. Надежность: SQLite обеспечивает целостность данных, поддерживает атомарные операции записи и предоставляет механизмы для обеспечения безопасности базы данных. Благодаря своей встроенной природе, SQLite обеспечивает отказоустойчивость и защиту данных от потери.

4. Экономичность: SQLite не требует больших ресурсов, таких как память или процессорное время. Она имеет небольшой размер, что позволяет эффективно использовать ресурсы системы.

5. Масштабируемость: SQLite может обрабатывать базы данных разного размера - от небольших до очень больших. Она поддерживает индексы, что позволяет ускорить поиск и сортировку данных.

SQLite часто используется в мобильной разработке, особенно для создания локальных баз данных в приложениях. Она также может использоваться в небольших веб-приложениях, прототипировании, тестировании и других сценариях, где простота и независимость от сервера являются важными факторами.

* + 1. 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.

Приведём наиболее популярные примеры IDE:

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.

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. 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.

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:

IntelliSense: эта функция предоставляет интеллектуальные предложения по завершению кода по мере ввода в зависимости от языка, с которым вы работаете.

Встроенная поддержка Git: VS Code включает встроенную поддержку Git, что позволяет легко управлять репозиториями кода, не выходя из редактора.

Поддержка отладки: VS Code предоставляет встроенный отладчик для ряда языков программирования, позволяющий легко выполнять код, выявлять и исправлять ошибки.

Запуск задач: VS Code поддерживает создание пользовательских задач, которые могут автоматизировать общие рабочие процессы, такие как создание и тестирование кода.

Встроенный терминал: VS Code включает встроенное окно терминала, позволяющее выполнять команды оболочки непосредственно в редакторе.

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.

Поддержка нескольких языков: VS Code поддерживает широкий спектр языков программирования, включая JavaScript, Python, Ruby и многие другие.

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:

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.

Кросс-браузерность - Bootstrap обеспечивает одинаковый внешний вид веб-сайта в различных браузерах, что значительно упрощает разработку и тестирование сайта.

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.

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.

* + 1. Git.

1. Практическая часть.

Создание веб-страницы включает в себя несколько основных шагов. Вот общий обзор процесса:

1. Планирование: В первую очередь, перед тем как приступить к созданию веб-страницы, важно провести подробное планирование. Определите цели, аудиторию, функциональность, дизайн и структуру страницы. Разработайте концепцию и схему сайта.

2. Верстка: после планирования переходите к верстке страницы. Верстка представляет собой создание HTML-структуры страницы. Используйте язык разметки HTML для определения структуры контента и его размещения на странице. Разделите элементы на блоки, задайте им соответствующие теги и атрибуты.

3. Стилизация: Следующий шаг - стилизация страницы с помощью CSS. Используйте CSS для определения внешнего вида страницы, задания цветов, шрифтов, размеров и расположения элементов. Создавайте стилистические правила и применяйте их к соответствующим элементам на странице.

4. Добавление контента: после создания структуры и стилизации страницы добавьте контент. Это может быть текст, изображения, видео, таблицы или другие элементы, которые несут информацию или предоставляют функциональность на странице. Разместите контент в соответствующих разделах страницы.

5. Навигация и интерактивность: Добавьте навигационные элементы, такие как меню и ссылки, чтобы пользователи могли легко перемещаться по страницам и взаимодействовать с контентом. Реализуйте интерактивные элементы с помощью JavaScript, если необходимо, чтобы страница была более динамичной и функциональной.

6. Тестирование: прежде чем опубликовать веб-страницу, проведите тестирование. Проверьте, что все элементы и функциональности работают корректно в разных браузерах и на разных устройствах. Отладьте ошибки и убедитесь, что страница отображается и функционирует должным образом.

7. Развертывание и публикация: после успешного тестирования разверните веб-страницу на сервере, чтобы она была доступна в Интернете. Зарегистрируйте доменное имя, выберите хостинг-провайдера и загрузите файлы вашей страницы на сервер.

Это общий обзор процесса создания веб-страницы. Конкретные шаги и детали могут варьироваться в зависимости от требований вашего проекта.

1. Планирование: Начните с определения структуры и содержимого вашей веб-страницы. Размышляйте о различных разделах и элементах, которые должны присутствовать на странице. Например, заголовки, параграфы, списки, изображения, таблицы и т. д.

2. Создание базовой структуры: Создайте базовую структуру вашей веб-страницы с использованием HTML-тегов. Это включает в себя создание объявления DOCTYPE, открывающего и закрывающего тегов <html>, <head> и <body>. Внутри <head> вы можете добавить различные метаданные, такие как заголовок страницы, описание, ключевые слова, ссылки на CSS файлы и другие.

3. Разметка контента: Добавьте различные HTML-теги для разметки контента на вашей странице. Например, используйте <h1> - <h6> для заголовков разных уровней, <p> для параграфов, <ul> и <li> для списков, <img> для изображений, <table> для таблиц и другие теги в зависимости от ваших потребностей.

4. Форматирование и стилизация: Добавьте CSS-стили или встроенные стили для форматирования и стилизации вашей веб-страницы. Вы можете использовать классы и идентификаторы, чтобы применить стили к определенным элементам на странице. CSS позволяет задавать различные свойства, такие как цвет текста, шрифты, размеры, отступы, рамки и т. д.

5. Создание ссылок и навигации: Добавьте ссылки для перемещения по вашей веб-странице или на другие страницы. Используйте тег <a> для создания гиперссылки. Вы также можете создать навигационное меню, используя список ссылок или другие подходящие элементы.

6. Работа с формами: Если вам необходимо создать формы, используйте тег <form> в HTML. Вы можете добавить элементы формы, такие как текстовые поля, полосы прокрутки, флажки, кнопки отправки и другие, чтобы пользователи могли отправлять данные с вашей веб-страницы.

7. Проверка и отладка: Проверьте свой HTML-код на ошибки и правильность работы. Воспользуйтесь инструментами проверки валидности кода и исправьте ошибки при необходимости. Также убедитесь, что ваша веб-страница корректно отображается на разных браузерах и устройствах.

* 1. Анализ задачи.

Для того, чтоб создать и опубликовать наше приложение в сети Интернет, нам необходимо выполнить целый ряд последовательных действий. Анализ потребностей пользователей основывается на потребностях клиента который состоит из:

* Рационального - безопасность, комфорт, надежность;
* Эмоционального - статусность, новизна;
* Сопряженные - “цепочка” нужной информации;
* Не сопряженные - разовый поиск информации;
* Скрытые - клиент не чувствует нужды;
* Явные - клиент осознает нужду;

Все вышеописанные потребности относятся к “Иерархии потребностей по пирамиде Маслоу”.

Дальнейшая работа была произведена следующим образом, был произведен опрос работников “Районного дома культуры” с. Бичура.

Из опроса были поставлены задачи:

* Понятный и удобный интерфейс;
* Адаптивность сайта;
* Быстрый доступ к статьям сайта;
* Гибкая и удобная с материалами сайта;
* Скорость выполнение задачи;

Сайт должен быть мобильным и интуитивно понятным для конечного пользователя.

* Для работников культуры удобным в редактирование и публикации материалов сайта.
  1. Планирование архитектуры сайта.

Архитектура сайта — это способ организации страниц, доступа к ним и навигации.(Таблица 3)

* Таблица 3 Общая архитектура сайта

|  |  |  |
| --- | --- | --- |
| **Навигация и ссылки** | **Рубрики:** | **Метки:** |
| Главная (родительская страница) | - | |
| Документы | Документы | - |
| Независимая оценка качества |
| Мероприятия | Мероприятия | #Анонс  #Кино |
| ПланМероприятий |
| Новости | Новости | #РайонныйДомКультуры |

* 1. Проектирование структуры сайта.

Структура сайта [http://cultbich.ru](http://cultbich.ru/) является древовидной, что является универсальным вариантом. Каждому пункту сайта или услуге отводится отдельная ветка: раздел или подраздел. Привычнее всего пользователям общаться именно с такими ресурсами. При древовидной структуре смысловая нагрузка делится между главной страницей и отдельными разделами, так как с ними линкуются отдельные страницы. Для продвижения это наиболее оптимальный вариант, позволяющий рекламировать сразу несколько разделов сайта.

Структура сайта [http://cultbich.ru](http://cultbich.ru/) Управление культуры МО “Бичурский район”:

* Главная
* Управление

2.1 Структура

2.2 Документы

2.2.1 Федеральные документы

2.2.2 Региональные документы

2.2.3 Муниципальные документы

2.2.4 Локальные документы

* Районный дом культуры

3.1 Сельские клубы

* Районная библиотека
* Детская школа искусств
* Туризм
* План работы РДК и СК
* План работы библиотек
* Режим работы библиотек
* Статьи
* Медиа

11.1 Фотогалерея

11.2 Видео

11.3 Файлы

11.4 Каталог сайта

* Обратная связь

12.1 Интернет приемная

12.2 Оставить отзыв

12.3 Контакты

* Противодействие коррупции

13.1 Документы

13.2 НПА

* 300-летие села Бичура

14.1 План основных мероприятий

14.2 Развлечение

14.3 Документы

14.4 Конкурсы

14.5 Как добраться

14.6 Проживание

14.7 Питание

Данная структура сайта очень обширна и громоздкая. Некоторые пункты не имеют в себе необходимую информацию либо она отсутствует.

Поставленной задаче от руководителя было принято решение по уменьшению структуры сайта и разделением учреждений. Так как по техническому заданию необходимо произвести переработку структуру сайта и минимизировать количество элементов интерфейса с которыми пользователь вынужден взаимодействовать одновременно.

Структура нового сайта:

* Главная
* Документы
* Мероприятия
* Новости
* Сельские клубы
* Галерея
* О нас
  1. Разработка и создание дизайна для страницы проекта.

Графическое оформление, представленное на этом сайте, было актуально, когда сайты смотрели только при небольшом разрешении на стационарных компьютерах.

С появлением возможности заходить на сайты с мобильных телефонов архитектура и графика сайтов изменилась в сторону унификации для одинакового отображения на различных типах мониторов и устройств.

Анализ дизайна сайта подразумевает соблюдение следующих критериев оценивания:

* Дизайн понятен целевой аудитории;
* Внимание пользователя сконцентрировано на самом важном;
* Элементы сайта выстроены по сетке;
* В дизайне соблюдается стилистика:
* Цвета использованы грамотно;
* В дизайне присутствует только одна основная шрифтовая пара;
* Иллюстрации и фото качественные;
* Сайт адаптирован под разные устройства;
* Механика элементов понятная и логичная;

Анализ сайта Управление культуры МО “Бичурский район”:

Плюсы:

* Бесплатный конструктор web-сайтов, легкий в освоении и создания сайтов;

Минусы:

* Механика элементов непонятная и загроможденная;
* Устарелый дизайн сайта;
* Бесконтрольная рекламная интеграция;
* Ограниченное предоставление ресурсов и одновременное подключение к сайту до 5 устройств;
* Не соответствует требованиям AMP;
* Плохая адаптация под мобильные устройства;
* Медиа контент в плохом качестве;
* Нет сертификатов безопасного подключения HTTPS;

Проектирование дизайна - это процесс создания визуального и функционального концепта для продукта или проекта, который включает в себя аспекты внешнего вида, пользовательского интерфейса (UI) и пользовательского опыта (UX).

Ключевые этапы и аспекты проектирования дизайна:

* Исследование и анализ;
* Разработка концепции;
* Создание макетов;
* Выбор цветовой палитры и типографики;
* Работа с изображениями и графикой;
* Адаптивный и мобильный дизайн;
* Тестирование и рефинемент;
* Релиз;
* Поддержка и обновление;

Для полного обозначения разработки дизайна сайта необходимо определиться с количеством материала. Разграничить материал в соответствии элемента “Карты сайта”.

Карта сайта - структурированное представление страниц и разделов веб-сайта, которое помогает пользователям и поисковым роботам понять организацию контента на сайте. Карта сайта представляет собой визуальное или текстовое дерево, которое показывает иерархию страниц и их взаимосвязи.

Frontend состоит из следующих составляющих:

* Верстка html/css/js;
* Текстовое наполнение;
* Визуал сайта, отображение видео и аудио контента;
* Юзабилити;
* Взаимодействие сайта с браузерами;

Графическое проектирование дизайна сайта было произведено в редакторе Figma.

Написание проекта было произведено в текстовом редакторе VS Studio Code.

Основные теги, которые будут использоваться при создании веб-сайта:

* <body> – тело документа;
* <div> – тег-контейнер для разделов HTML-документа;
* <form> – форма для сбора и отправки на сервер информации от пользователей;
* <button> – интерактивная кнопка;
* <input> – создает многофункциональные поля формы, в которые пользователь может вводить данные.

Адаптивная вёрстка сайта позволяет веб-страницам автоматически подстраиваться под экраны планшетов и смартфонов.

В данной работе будет показаны следующие виды макетов сайта:

* ПК – 1920px файл, контент под 1280 px без учета отступов на скролл и отступов справа и слева;
* Планшет 1024px без учета отступов на скролл и отступов справа и слева;
* Планшет 768px без учета отступов справа и слева;
* Смартфоны 480px без учета отступов справа и слева;
* Смартфоны 320px без учета отступов справа и слева;

Впоследствии были созданный паттерны для удобной работы с сайтом работниками культуры.

Визуальная иерархия на сайте — это организация и оформление информации таким образом, чтобы посетитель мог быстро разобраться с интерфейсом и отличить главное от второстепенного.

Паттерны сканирования на десктопах:

Паттерны сканирования: что и где подсознательно ожидает увидеть пользователь.

В первые секунды на сайте пользователь будет искать полезное содержимое на сайте, «сканируя» видимую область страницы в поисках нужной информации. Начинает он с левого верхнего угла (если он привык читать слева направо), движется направо, ожидая получить там дополнительную информацию. Затем переводит взгляд в нижний левый сектор, и быстро перескакивает в правый нижний угол.

У этого сценария, или паттерна, есть название: Диаграмма Гутенберга. Чаще всего он встречается на посадочных страницах и страницах регистрации, и именно в соответствии с ним построены начальные страницы.

**Z-паттерн.** Когда при сканировании страницы взгляду не на что опереться, включается Z-паттерн: взгляд широкими зигзагами передвигается по странице, выуживая центры внимания (Рисунок 8):

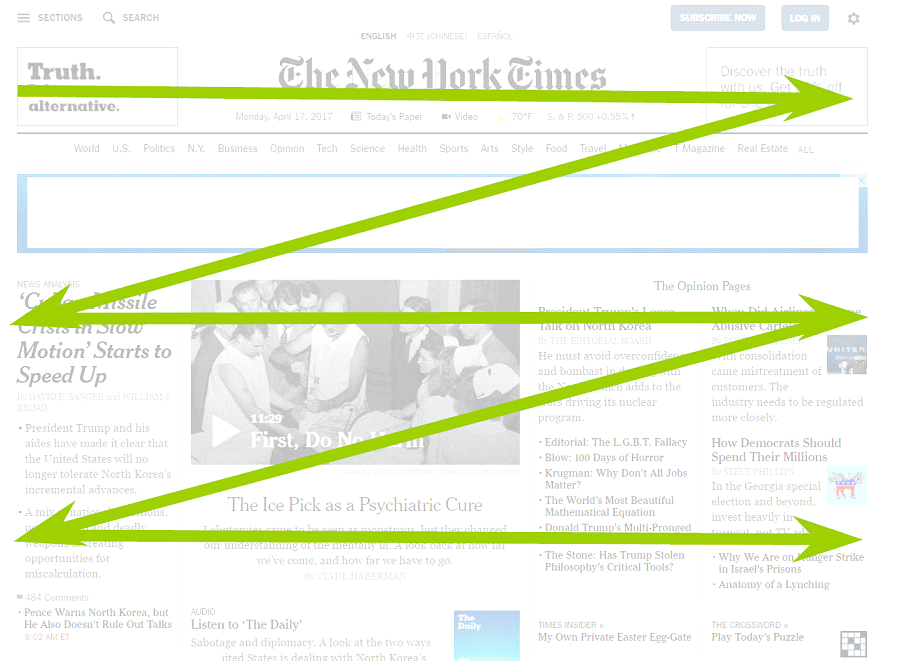


Рисунок 8 Вид Z-паттерн

* Типографика в веб-дизайне - это свод правил оформления текста и графических элементов сайта, который несет определенный посыл, воздействует на эмоции и мотивирует посетителя к совершению целевых действий.
* В данном проекте была использована следующая типографика:
* Типографика: Barlow Semi Condensed / Roboto
* Цвет шрифта: #00000;
* Шрифт для заголовков: Barlow Sermi Condensed
* Толщина шрифта: 600;
* Пример: **Привет мир!**
* Основной шрифт: Roboto;
* Толщина шрифта: 400;
* Пример: Привет мир!
* Размер шрифта: 14px;
* Обычный цвет ссылок шрифта: #046BD2;
* Пример: Привет мир!
* При наведении цвет ссылок шрифта: #045CB4;
* Пример: Привет мир!
* Высота строк: 1.5
  1. Выбор технологий для создания веб-приложения.

Для начала мы схематично спроектируем наш будущий веб-сайт и подберём технологии для его создания. Веб-сайт – одна или несколько страниц, которые имеют логическую связь между собой, имеют общий сервер и доменное имя. Сервером является специальное оборудование(компьютер), имеющее необходимые программы для работы с веб-сайтом.

Доменное имя – это адрес веб-сайта, отличающееся от других в системе доменных имен, необходим для поиска в сети.

Браузер – ПО для просмотров веб-сайта. Браузер отправляет запрос (доменное имя) серверу, с помощью HTTP-протоколов, также с помощью этих протоколов сервер отправляет ответ (документы, информацию) на запрос.

С помощью разметки, которая создана языком разметки HTML написано большинство веб-сайтов. HTML – стандартный язык разметки документов во «Всемирной паутине». Посредством данного языка разметки выделяют разные информационные блоки, их расположение на веб-странице, также оформление.

Главным плюсом HTML является одинаковое заданное расположение элементов на различных устройствах, с разными расширениями.

Теги HTML – специальные команды, которые обозначают элементы и помогают не нагружать ненужной информацией браузер. Веб-сайт компании «Paleta», будет написан на данном языке для сохранения полноты и правильности информации на любом гаджете.

Рассмотрим два способа написания сайта и выберем более подходящий для создания сайта.

Основной способ с помощью текстового редактора HTML. Для того чтобы работать в редакторе с кодом необходимо знать принципы его построения и набора вручную всех команд, также умение работать с каскадными таблицами CSS и знание JavaScript.

Другой способ легче в написании кода, некоторые команды создаются автоматически, следовательно, знаний необходимо меньше и времени тратиться соответственно меньше. Этот метод подразумевает использование движка CMS, с ним проще удалять, добавлять новую информацию на веб-сайт.

Технологии разработки сайтов открывают широкие возможности для создания web-ресурсов различного типа, сложности и целевой направленности.

Web-технологии с каждым годом совершенствуются, выходят на новый уровень. В настоящее время, доступно огромное множество разнообразных решений, благодаря которым можно разработать сайт любой сложности.

* 1. 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.

Дочерние компоненты агрегатора

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

* 1. Тестирование сайта.

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.

Тестирование производительности сайта выполняется с целью оценки и обеспечения оптимальной работы веб-ресурса. Вот несколько основных целей, для которых проводится тестирование производительности сайта:

Обеспечение удовлетворительного пользовательского опыта: Быстрая загрузка страниц и отзывчивость интерфейса сайта являются ключевыми факторами, влияющими на удовлетворение пользователей. Тестирование производительности позволяет убедиться, что сайт загружается быстро и работает плавно, что повышает удовлетворенность пользователей.

• Определение пропускной способности и масштабируемости: Тестирование помогает определить, сколько одновременных пользователей сайта он способен обслуживать без потери производительности. Это важно для сайтов, которые могут ожидать большой посещаемости.

• Выявление узких мест и оптимизация: Тестирование производительности помогает выявить узкие места в работе сайта, такие как медленные запросы к базе данных, неправильная настройка сервера или ненадежный хостинг. Эти проблемы могут быть решены для улучшения производительности.

• Оценка влияния обновлений и изменений: При внесении изменений на сайт, таких как добавление новой функциональности или обновление системы управления содержанием, тестирование производительности позволяет оценить, как эти изменения влияют на производительность сайта.

• Предотвращение потери клиентов: Медленно загружающиеся страницы и ненадежная работа сайта могут отпугнуть посетителей и потенциальных клиентов. Тестирование производительности помогает предотвратить потерю клиентов из-за проблем с производительностью.

• Соблюдение стандартов и требований: Некоторые индустриальные стандарты и законодательные требования могут обязывать веб-сайты поддерживать определенный уровень производительности и безопасности. Тестирование помогает удостовериться, что сайт соответствует этим требованиям.

Тестирование производительности сайта играет важную роль в обеспечении стабильной, быстрой и надежной работы веб-ресурса, что в свою очередь способствует удовлетворенности пользователей и успеху веб-проекта.

Тестирования сайта происходит с помощью следующего интернет ресурса, а именно: Google PageSpeed Tools. После сравнения результата тестирования мы видим прирост отклика сайта для всех устройств. Данные показатели свидетельствуют о стабильной, быстрой и надежной работы веб-ресурса, что в свою очередь способствует удовлетворенности пользователей.

* 1. Публикация сайта на хостинг-провайдере.

Хостинг – это сервис, который предоставляет серверное пространство и ресурсы для размещения ваших веб-сайтов, файлов и приложений, делая их доступными для пользователей в Интернете.

Работодателю был предоставлен список VPS/VDS хостингов, где подробно были изложены преимущества и стоимость услуг для публикации сайта в Интернете. (Таблица 2)

Таблица 2 Сравнения хостингов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Хостинг** | **Год начала работы** | **Количество клиентов** | **Тестовый период** | **Стоимость, мес.** |
| Beget | 2007 | более 200 000 | 30 дней | от 165 руб. |
| Timeweb | 2006 | более 150 000 | 10 дней | от 119 руб. |
| Sprinthost | 2005 | более 450 000 | 30 дней | от 118 руб. |
| Джино | 2003 | более 300 000 | 10 дней | от 58,5 руб. |
| Fozzy | 2012 | более 35 000 | 7 дней | от 99 руб. |
| Reg.ru | 2006 | более 2 200 000 | 14 дней | от 178 руб. |
| Макхост | 2004 | более 50 000 | 30 дней | от 113,5 руб. |
| ISPserver | 1997 | – | 7 дней | от 110 руб. |
| Spaceweb | 2001 | более 100 000 | 14 дней | от 159 руб. |
| Webhost1 | 2008 | более 130 000 | 30 дней | от 75 руб. |

Выбор пал на хостинг-провайдер Бегет. Преимущества данного хостинга в том, что удобный интерфейс для работы с ресурсами, круглосуточная поддержка, работа с юридическими организациями, высокая производительность.

Преимущества:

* Максимальная надёжность благодаря качественным серверам и современному ПО.
* Поддержка популярных систем управления – Битрикс, WordPress, Joomla, др.
* Быстрая загрузка сайтов.
* Отзывчивая техподдержка.
* Постоянные доработки, усовершенствования, оперативное устранение ошибок, добавление новых функций.
* NVMe диск.

Хостинг-провайдеры, такие как Бегет, предлагают инфраструктуру и техническую поддержку для обеспечения надежности и производительности веб-проекта.

Хостинг Бегет является одним из ведущих хостинг-провайдеров на рынке и имеет множество довольных клиентов. Компания Бегет предоставляет широкий спектр услуг, от общедоступного веб-хостинга до выделенных серверов и облачного хостинга, чтобы удовлетворить потребности различных типов веб-проектов.

Обеспечивает высокую доступность вашего сайта, быструю загрузку страниц, безопасность и защиту от вредоносных атак.

Удобные инструменты управления файлами, базами данных, электронной почтой и другими функциями вашего веб-проекта. Настройка доменного имени сайта происходит автоматически с привязкой к репозиторию где находится сайт. После оплаты за регистрацию домена. Выдается в дальнейшем сертификат на владение данным доменном имнем на год. По исходу оплаченного года можно будет продлить существующий домен до следующего календарного года.

Данный проект имеет поддержку и способность оперативно предоставлять публичную отчетность для муниципальной организации, которая требуется по законодательству Российской Федерации для бюджетных организаций.

1. Заключение.

В заключение необходимо включить следующее:

1. Краткие и ёмкие теоретические и практические выводы, которые были получены во время анализа теоретической базы и практического исследования.

2. Оценка проведённого исследования, описание его результатов.

3. Практическая значимость работы, рекомендации и планы на дальнейшие исследования.

4. Общий итог — достижение цели, выполнение задач, доказательство гипотезы.

5. Предложения по совершенствованию объекта исследования.

В рамках данной дипломной работы была проведена разработка и публикация информационного портала для группы культурно-досуговых центров Бичурского района. Этот проект ставил перед собой цель создать современное и удобное средство коммуникации между организаторами мероприятий и посетителями центров, а также предоставить обширную информацию о культурной и развлекательной жизни района.

В ходе работы были решены следующие задачи:

• Анализ потребностей и ожиданий пользователей, что позволило определить функциональные требования к порталу.

• Проектирование пользовательского интерфейса, обеспечивающего легкий доступ к информации и удобство взаимодействия с порталом.

• Разработка структуры базы данных и веб-приложения, обеспечивающих хранение и обработку информации.

• Реализация возможности публикации мероприятий, новостей и обновлений для администраторов портала.

• Обеспечение механизмов фильтрации и поиска для посетителей портала, упрощающих поиск интересующей информации.

В результате проделанной работы информационный портал стал мощным инструментом для продвижения культурно-досуговых событий и инициатив в Бичурском районе. Он способствует более тесному взаимодействию между организаторами и посетителями мероприятий, а также способствует популяризации культурного наследия и разнообразия в данном районе.

Работа над данным дипломом позволила приобрести ценный опыт в области веб-разработки, проектирования пользовательских интерфейсов и анализа потребительских потребностей. Результаты данной работы могут быть использованы как основа для развития информационных порталов других культурных и развлекательных центров, а также способствовать дальнейшему укреплению культурной и социальной жизни.

Подытоживая вышеизложенное, можно утверждать, что данная дипломная работа является важным шагом в развитии культурного и информационного пространства Бичурского района и приносит пользу как его жителям, так и организаторам мероприятий.

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 книг 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.

В рамках данной дипломной работы была проведена разработка и публикация информационного портала для группы культурно-досуговых центров Бичурского района. Этот проект ставил перед собой цель создать современное и удобное средство коммуникации между организаторами мероприятий и посетителями центров, а также предоставить обширную информацию о культурной и развлекательной жизни района.

В ходе работы были решены следующие задачи:

* Анализ потребностей и ожиданий пользователей, что позволило определить функциональные требования к порталу.
* Проектирование пользовательского интерфейса, обеспечивающего легкий доступ к информации и удобство взаимодействия с порталом.
* Разработка структуры базы данных и веб-приложения, обеспечивающих хранение и обработку информации.
* Реализация возможности публикации мероприятий, новостей и обновлений для администраторов портала.
* Обеспечение механизмов фильтрации и поиска для посетителей портала, упрощающих поиск интересующей информации.

В результате проделанной работы информационный портал стал мощным инструментом для продвижения культурно-досуговых событий и инициатив в Бичурском районе. Он способствует более тесному взаимодействию между организаторами и посетителями мероприятий, а также способствует популяризации культурного наследия и разнообразия в данном районе.

Работа над данным дипломом позволила приобрести ценный опыт в области веб-разработки, проектирования пользовательских интерфейсов и анализа потребительских потребностей. Результаты данной работы могут быть использованы как основа для развития информационных порталов других культурных и развлекательных центров, а также способствовать дальнейшему укреплению культурной и социальной жизни.

Подытоживая вышеизложенное, можно утверждать, что данная дипломная работа является важным шагом в развитии культурного и информационного пространства Бичурского района и приносит пользу как его жителям, так и организаторам мероприятий.

1. Список использованной литературы.
2. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. - СПб.: Питер, 2017.
3. Владстон Ф.Ф. Теоретический минимум по Computer science. - СПб.: Питер, 2022.
4. Владстон Ф.Ф., Пиктет М. Теоретический минимум по Computer science. - СПб.: Питер, 2022.
5. Дакетт, Дж. HTML и CSS. Дизайн и построение веб-сайтов. - СПб.: Питер, 2023.
6. Крокфорд, Д. JavaScript: The Good Parts. - СПб.: Питер, 2013.
7. Мейер Э., Уэйл Э. CSS: полный справочник, 4-е изд.: Пер. с англ. - СПб.: ООО «Диалектика», 2019.
8. Роббинс, Д.Н. HTML5, CSS3 и JavaScript. Исчерпывающее руководство. 4-ое издание. – М.: ЭКСМО, 2014.
9. Учурова С. А. Лексикология немецкого языка: конспект лекций. Lexikologie der deutschen Sprache : Vorlesungsskripten / С. А. Учурова. – Екатеринбург : Изд-во Урал. ун-та, 2014.