

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.
      1. Frontend веб-приложения, его особенности.
      2. Backend веб-приложения, его особенности.
      3. Язык гипертекстовой разметки HTML.
      4. Таблицы стилей CSS.
      5. Язык программирования JavaScript.
         1. Общие сведения о языке.
         2. Фреймворки и библиотеки JavaScript.
      6. Язык программирования Python.
         1. Общие сведения о языке.
         2. Фреймворки и библиотеки Python.
      7. СУБД данных SQLite.
      8. Программные продукты и инструменты web-разработки.
      9. Git.
3. Практическая часть. Разработка проекта:
   1. Этап планирования разработки.
   2. Этап разработки проекта.
   3. Этап тестирования.
   4. Этап публикации.
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». Немецкий язык – основной литературный язык Германии, Австрии, Лихтенштейна, один из государственных языков Швейцарии и Люксембурга, язык культурной, научной и деловой жизни, играющий важную роль даже не в немецкогоговорящих странах мира. Данный проект преследует образовательные цели и предназначен для широкого круга изучающих и просто интересующихся немецким языком.

Основная цель и планируемый итоговый результат проекта — предоставить платформу, знакомящую пользователей со словарём вариантов и диалектов немецкого языка и разработка соответствующего веб-сайта, который собирает и представляет информацию о книгах из различных русскоязычных книжных интернет-магазинов.

Задачи, решаемые в ходе работы над дипломным проектом:

Изучение теории, касающейся темы исследования.

Разработка привлекательного и функционального дизайна приложения.

Проектирование и разработка веб-приложения, реализация его основных функций.

Тестирование и отладка разработанного приложения.

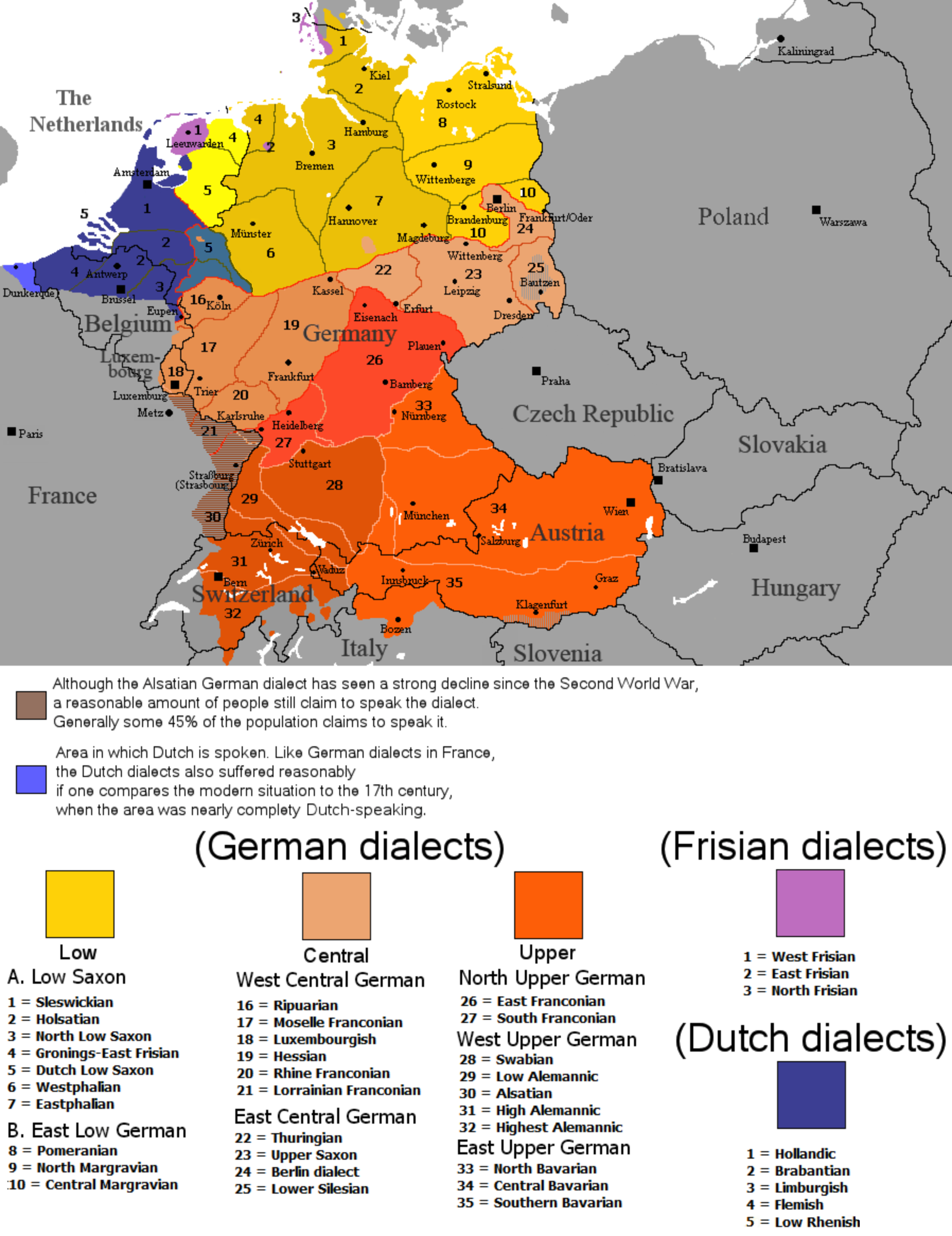
В ходе работы над проектом веб-приложения с использованием HTML, CSS, Python и Flask планируется уделить внимание следующим ключевым компонентам:

1. Разработке пользовательского интерфейса (UI) и взаимодействие с пользователем (UX) приложения с помощью HTML, CSS. Пользовательский интерфейс должен быть визуально привлекательным и простым в навигации. Пользовательский интерфейс: пользовательский интерфейс веб-сайта должен быть разработан таким образом, чтобы обеспечить удобство для посетителей.
2. Работе с базой 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;

Инструменты, используемые при разработке проекта:

1. Git (GitHub);
2. JetBrains WebStorm;
3. JetBrains PyCharm.

Проект будет разрабатываться собственными силами. В ходе работы над приложением будет возможность опробовать на практике следующие роли:

1. Frontend-разработчик – специалист, обладающий знаниями в области HTML, CSS и JavaScript, способный создавать внешний пользовательский интерфейс и добавлять интерактивность в приложение с помощью таких различных фреймворков (например, Vue.js) и иных инструментов;
2. Backend-разработчик – специалист с опытом работы с инструментами для разработки серверной части веб-приложений (например, фреймворки Django, Flask, Node.js, Express, FastAPI и иными), опытом работы с базами данных, способный создавать инфраструктуру на стороне сервера, конечные точки API и подключения к базе данных;
3. QA Инженер – специалист, чья деятельность связана с тестированием приложения на наличие ошибок и проблем, способный разрабатывать тестовые случаи и сценарии, чтобы убедиться, что приложение работает должным образом;
4. UX-дизайнер – специалист по разработке удобного и визуально привлекательного интерфейса для приложения;
5. Fullstack-разработчик – высококвалифицированный специалист, способный совмещать в одном лице все вышеперечисленные роли и готовый самостоятельно проектировать, разрабатывать, тестировать и публиковать в сети интернет готовый программный продукт: web-приложения, сайты различной функциональности и т.д.

*Рис. Карта вариантов немецкого языка и близкородственных идиомов.*

* + - 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. 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. Германская языковая семья, к которой относится немецкий, известна своей богатой историей и значительным влиянием на культуру и науку, но особенный вклад в мировое духовное наследие внесли именно носители английского и немецкого языка. Немецкие авторы, философы и ученые внесли значительный вклад в мировую литературу, философию и науку, и их работы переведены на множество языков, включая русский.

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.

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.

Каждый, сталкивавшийся в жизни с задачей изучения немецкого языка, обращал внимание на то, 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.

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.

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.

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.

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. 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, которые можно привести в качестве иллюстративного примера является баварский, который наиболее распространен в южной части Германии и Австрии (он также непосредственно связан с австрийскими диалектами немецкого языка, составляя вместе с ними особую подгруппу, выделяемую целым рядом исследователей). 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 распространённый на территории 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 в качестве особо примечательных немецких говоров (хотя безусловно все они заслуживают внимания и пристального научного изучения) 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 нами выше 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.

Помимо собственно диалектов и региональных вариантов немецкого языка в рамках настоящей работы нас также будут интересовать особенности немецкого языка в Германской Демократической Республике (ГДР).

Немецкий язык в ГДР, находившейся в течение достаточно длительного времени в статусе суверенного государства можно рассматривать как отдельный региональный вариант языка, отличающийся от 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, став частью обновленной Германии. В течение своего существования ГДР была социалистическим государством, в стране проводились коллективизация сельского хозяйства, национализация промышленности, было введено централизованное планирование экономики. Политическая система страны была многопартийной, но руководящую роль играла СЕПГ – Социалистическая единая партия Германии. Падение Берлинской стены в 1989 году и объединение Германии в 1990 году означало окончание существования ГДР. За время существования ГДР (1949-1990), немецкий язык в ней помимо традиционных процессов, характерных для всех живых языков, испытал также влияние социально-политических факторов, что привело к некоторым особенностям в его использовании.

Мы отметим некоторые из них, чтобы удачнее пояснить нашу мысль:

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. В ГДР было весьма распространено использование специфической идеологической терминологии, связанной с коммунистической идеологией. Например, часто употреблялись термины, такие как "*Klassenkampf*" (борьба классов), "*Sozialismus*" (социализм) и "*Realsozialismus*" (реальный социализм).
2. Использование советской нормы: влияние советского стиля и норм русского языка также было заметно в использовании немецкого языка в ГДР, а благодаря близким связям с СССР, русский язык оказал значительное влияние на немецкий язык в ГДР. Например, применение конструкций, подвижных приставок и словосочетаний, аналогичных русским языковым образцам. Введение советской нормы и использование русских заимствований сделало некоторые выражения и конструкции уникальными для ГДР.
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. 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.
7. 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.
8. 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.

* 1. Web-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:

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

Применительно к дипломному проекту, онлайн-словарь вариантов немецкого языка — это тип веб-приложения, которое позволяет пользователям получать списки специфической лексики, характерной для того или иного региона либо исторической эпохи. Суть веб-приложения будет включать в себя функцию просмотра сравнительных таблиц из базы данных, а также базу данных с региональными словарями немецкого языка.

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

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.

* + 1. Frontend веб-приложения, его особенности.

Frontend — внешний (отсюда и название *frontend*) интерфейс веб-приложения, который состоит из пользовательского интерфейса, отображающего результаты взаимодействия пользователя с данными, предоставляемыми веб-приложением. 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. Пользовательский интерфейс выступает визуальным представлением веб-приложения и может включать такие элементы, как панели поиска, страницы результатов и страницы сведений о книгах;

Как правило, внешний интерфейс может создаваться на основе следующих компонентов и инструментов:

* + - 1. HTML — это язык разметки, используемый для структурирования содержимого веб-страниц. HTML используется для определения структуры пользовательского интерфейса, включая заголовки, абзацы, списки и таблицы;
      2. 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;
      3. 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;
      4. 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;
      5. 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 новая информация.

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.

* + 1. Backend веб-приложения, его особенности.

Серверная часть (Backend) веб-приложения обычно состоит из программного приложения сервера, которое может, например, обрабатывать запросы пользователей и извлекает данные из баз данных или других интернет-сервисов. Для этого требуется использование веб-протоколов, таких как HTTP и HTTPS, например, для получения данных из API-интерфейсов.

В общих случаях серверная часть может состоять из следующих компонентов:

Node.js — 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.

Серверные фреймворки на языке Python – Flask, Django и т.д. Они легко масштабируются, очень просты в освоении и обладают практически неограниченными возможностями.

Backend может собирать данные с помощью парсинга веб-страниц или вызовов API, в том числе собирать данные на регулярной основе, чтобы обеспечить их актуальность, а также обрабатывать собранные данные для извлечения необходимой информации. Backend может также управлять базой данных для хранения собранных данных. База данных должна быть спроектирована с учетом высокой производительности и масштабируемости для обработки больших объемов данных. Backend может обладать возможностью предоставлять 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, что требует использования методов анализа и нормализации данных.

Серверная часть должна иметь возможность предоставлять обновления в режиме реального времени для внешнего интерфейса при сборе новых данных или при изменении текущих. Серверная часть должна быть спроектирована с учетом масштабируемости, с возможностью обработки большого количества запросов от внешнего интерфейса и обработки больших объемов данных.

* + 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 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. 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.

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 использует теги (tags) для определения элементов страницы. Теги создаются с использованием угловых скобок <>, и обычно имеют начальный и конечный тег, где конечный тег отличается от начального символом "/". Внутри тегов могут находиться атрибуты (attributes), которые определяют дополнительные характеристики элементов, такие как их цвет, размер, ссылки на другие страницы и т.д.

Некоторые наиболее часто используемые теги в HTML:

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

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

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

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

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 условия взаимодействия пользователя с сайтом.

* + 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.

Приведём некоторые из ключевых особенностей JavaScript:

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

В данной главе мы постараемся вкратце рассмотреть основные фреймворки и библиотеки JavaScript. Наш первый пункт — **Vue.js**: прогрессивный фреймворк JavaScript, который используется для создания пользовательских интерфейсов и одностраничных приложений (SPA). Он представляет собой модульный фреймворк, который может использоваться вместе с другими библиотеками и инструментами, такими как Vuex, Vue Router, Vuetify и другие.

Основные преимущества Vue.js:

1. Легковесность - Vue.js имеет небольшой размер и быстро загружается на страницу, что позволяет создавать быстрые и отзывчивые пользовательские интерфейсы.
2. Гибкость - Vue.js можно использовать как для маленьких, простых приложений, так и для крупных и сложных проектов. Он также предоставляет широкий выбор опций и настроек, которые позволяют настроить его под различные потребности.
3. Простота в использовании - Vue.js легко устанавливается и используется даже начинающими разработчиками. Он также предоставляет простые и интуитивно понятные API и документацию.
4. Однофайловые компоненты - Vue.js поддерживает однофайловые компоненты, которые содержат в себе все необходимые HTML, CSS и JavaScript коды, что облегчает разработку и обслуживание приложений.
5. 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.
6. 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.

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

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

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

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

**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 опций, которые делают его популярным выбором для создания масштабируемых высокопроизводительных веб-приложений, в том числе:

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

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

**Express** — 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 обширный функционал, который превращает его в популярный выбор для создания веб-приложений. Среди них можно отметить:

1. Маршрутизация: Express предоставляет мощную систему маршрутизации, которая позволяет разработчикам определять маршруты для обработки входящих HTTP-запросов.
2. Промежуточное ПО (Middleware): Express предоставляет систему промежуточного ПО, которая позволяет разработчикам легко добавлять в свои приложения функциональные возможности, такие как обработка аутентификации, ведение журнала и обработка ошибок.
3. Механизмы шаблонов: Express поддерживает различные механизмы шаблонов, такие как Pug, EJS и Handlebars, что упрощает создание динамических HTML-страниц.
4. Обработка ошибок: Express предоставляет мощную систему обработки ошибок, которая позволяет разработчикам обрабатывать ошибки последовательным и удобным для пользователя способом.
5. Интеграция с другими модулями Node.js: Express можно легко интегрировать с другими модулями Node.js, такими как диспетчер пакетов Node (npm) и основные модули Node.js. Express обычно используется в сочетании с различными модулями Node.js для создания полнофункциональных веб-приложений. Он также широко используется в сочетании с интерфейсными фреймворками для создания одностраничных приложений (SPA) с внутренним API.
   * 1. Язык программирования Python.
        1. Общие сведения о языке.

Python — это высокоуровневый интерпретируемый язык программирования, созданный Гвидо ван Россумом в конце 1980-х годов. Он обладает простым и понятным синтаксисом, который делает его очень доступным для начинающих программистов, но при этом предлагает широкие возможности для разработки сложных программ и приложений.

Основные особенности Python:

* 1. Читаемость кода: Python нацелен на читаемость кода, что делает его подходящим для разработки и поддержки крупных проектов.
  2. Динамическая типизация: Python — это язык с динамической типизацией, что означает, что типы данных определяются автоматически в процессе выполнения программы. Типизация Python так же называется «утиной», по мотивам известного афоризма – «Если что-то выглядит как утка, летает, как утка и крякает, как утка, то скорее всего это и есть утка». Точно так же «размышляет» и Python, определяя тип данных автоматически в процессе обработки готовой программы.
  3. Многообразие библиотек: Python имеет богатую экосистему библиотек и модулей, что делает его популярным выбором для разработки приложений в различных областях, таких как веб-разработка, анализ данных, машинное обучение и другие.
  4. Поддержка объектно-ориентированного программирования: Python поддерживает объектно-ориентированное программирование, что позволяет разрабатывать модульные и масштабируемые приложения.
  5. Кроссплатформенность: Python поддерживает различные платформы, такие как Windows, macOS, Linux, что делает его универсальным инструментом разработки.
  6. Коммьюнити: Python имеет активное сообщество разработчиков, которые постоянно добавляют новые функции, исправляют ошибки и создают обучающий контент.

Python широко используется в университетах, научных исследованиях, веб-разработке, автоматизации задач, разработке игр и других областях. Python — это мощный и гибкий язык программирования, который подходит как для начинающих, так и для опытных разработчиков.

Еще одним преимуществом этого языка программирования является его расширяемость. Python позволяет интегрировать код, написанный на других языках программирования, таких как C, C++ и Java, что обеспечивает гибкость при разработке проектов, требующих оптимизированных вычислений или работы с библиотеками на других языках.

Еще одной замечательной особенностью Python является его обширная стандартная библиотека. В ней содержится множество модулей и функций, которые облегчают разработку различных типов приложений без необходимости написания кода с нуля. Стандартная библиотека Python включает в себя модули для работы с сетями, файлами, базами данных, графикой, архивами, регулярными выражениями и многим другим.

Одной из ключевых особенностей Python также является его поддержка функционального программирования. Python позволяет использовать функции как объекты первого класса, что позволяет передавать функции в качестве аргументов, возвращать функции как результаты и создавать анонимные функции с помощью лямбда-выражений. Это делает Python мощным инструментом для разработки функциональных программ.

Кроме того, Python широко используется в области научных исследований благодаря библиотекам NumPy, pandas, matplotlib и другим, которые обеспечивают удобные средства для работы с данными, визуализации и анализа. Это делает Python популярным инструментом в области машинного обучения и искусственного интеллекта.

В целом, Python — это универсальный и мощный язык программирования, который подходит для широкого спектра задач, от написания скриптов и веб-приложений до разработки научных и инженерных проектов. Его простой синтаксис, богатая экосистема и гибкость делают его одним из самых популярных языков программирования в мире.

* + - 1. Фреймворки и библиотеки Python.

Фреймворк — это структура программного обеспечения, предоставляющая разработчикам готовые абстракции и инструменты для создания приложений определенного типа. Он представляет собой набор библиотек, классов, функций и методов, предназначенных для решения типовых задач в рамках конкретной области разработки. Применение фреймворка упрощает и ускоряет процесс разработки, поскольку разработчикам необходимо заниматься лишь реализацией специфичных для их приложений функций, используя готовые инструменты и компоненты, предоставленные фреймворком.

Фреймворки могут быть разработаны для различных целей и областей программирования: веб-разработки, мобильной разработки, игровой разработки, научных вычислений и т.д. Они могут включать в себя компоненты для роутинга, работы с базами данных, обработки пользовательского ввода, взаимодействия с веб-серверами и многое другое.

Использование фреймворка обеспечивает не только повышение производительности и стабильности разработки, но также способствует соблюдению лучших практик и стандартов кодирования в рамках определенной экосистемы. Это делает код более читаемым, поддерживаемым и масштабируемым.

Популярные фреймворки веб-разработки, например, включают Django для Python, Node.js и Express.js для JavaScript, Ruby on Rails для Ruby, Laravel для PHP и другие. Каждый из них предоставляет набор инструментов и практик, специально разработанных для облегчения разработки веб-приложений на соответствующем языке программирования.

Библиотека в Python — это набор предопределенных функций и классов, который предоставляет разработчику определенные возможности для выполнения конкретных задач. В отличие от фреймворка, библиотека представляет собой независимый модуль, который может использоваться для расширения возможностей стандартного языка Python.

В Python существует огромное количество библиотек для различных целей, таких как работа с математическими вычислениями (например, NumPy, SciPy), обработка данных (Pandas), визуализация данных (Matplotlib, Seaborn), разработка веб-приложений (Django, Flask), машинное обучение и искусственный интеллект (TensorFlow, PyTorch), обработка изображений и многое другое.

Использование библиотек позволяет разработчикам значительно ускорить процесс разработки, поскольку можно использовать готовые решения, вместо того чтобы писать код с нуля. Библиотеки также способствуют повышению эффективности и качества программного продукта, так как они представляют собой проверенные и оптимизированные решения для типовых задач.

Другим важным преимуществом библиотек является их сообщество разработчиков, которое поддерживает и развивает библиотеки, предоставляя обновления, исправления ошибок и новые функциональности. Это делает использование библиотек в Python очень удобным и популярным среди разработчиков.

**Django** — это мощный фреймворк для создания веб-приложений на языке программирования Python. Он разработан с учетом принципов DRY (Don't Repeat Yourself) и позволяет быстро и эффективно создавать разнообразные веб-приложения, от простых сайтов до сложных веб-сервисов.

Основные принципы, на которых основан Django, включают в себя:

* 1. MVC-подобная архитектура: Django использует концепцию Model-View-Template (шаблон) для разделения данных (модели), логики приложения (контроллеры) и представления пользовательского интерфейса (шаблоны).
  2. ORM (Object-Relational Mapping): Django предоставляет свой собственный ORM, позволяющий работать с базами данных с использованием объектно-ориентированного подхода, что делает взаимодействие с БД удобным и интуитивно понятным.
  3. Административный интерфейс: Django поставляется со встроенной административной панелью, которая автоматически создается на основе определений моделей приложения, облегчая управление данными в БД.
  4. Большое количество встроенных инструментов: Django предоставляет множество готовых модулей и инструментов для обработки URL, форм, авторизации, безопасности, кэширования и многое другое, что значительно упрощает разработку веб-приложений.
  5. Широкие возможности расширения: Django обладает обширной экосистемой сторонних библиотек и расширений, что позволяет разработчикам легко находить необходимые решения для своих проектов.
  6. В целом Django является популярным и широко используемым фреймворком, который позволяет создавать стабильные, производительные и масштабируемые веб-приложения, идеально подходящие как для начинающих, так и для опытных разработчиков.

**Flask** — это минималистичный фреймворк для создания веб-приложений на языке программирования Python. Он основан на модульности и простоте, позволяя разработчикам быстро создавать небольшие и средние веб-приложения, а также веб-API.

Среди основных особенностей и преимуществ Flask можно выделить следующие:

* 1. Легковесность: Flask предлагает минимальное число зависимостей, что делает его легким и простым в освоении. При этом он предоставляет основные инструменты для создания веб-приложений.
  2. Легкость в использовании: Flask предлагает простой API с интуитивно понятными методами для обработки запросов, управления маршрутами и отображения данных на веб-страницах.
  3. Гибкость: Flask позволяет разработчикам выбирать и подключать только необходимые расширения (плагины) для реализации конкретных функций, что делает фреймворк гибким и настраиваемым.
  4. Расширяемость: Сообщество Flask разработало множество сторонних расширений, которые позволяют значительно расширить базовый функционал фреймворка, добавить поддержку форм, авторизацию, работу с базами данных и многое другое.
  5. Шаблонизация: Flask предлагает возможность использовать различные шаблонизаторы (например, Jinja2) для создания динамических HTML-страниц, что делает процесс отображения данных более удобным и гибким.

Несмотря на свою простоту, Flask является мощным инструментом для создания веб-приложений различного уровня сложности. Он отлично подходит для быстрого прототипирования, создания простых веб-сервисов и микросервисов, а также для обучения и экспериментов в области веб-разработки.

Библиотека **PyScript**. PyScript является библиотекой для Python, предназначенной для содействия разработке динамических и интерактивных веб-приложений. Она обеспечивает программистам возможность работать с веб-интерфейсами, включая создание визуальных элементов, обработку событий пользовательского взаимодействия и установление связи с серверной частью.

С использованием PyScript разработчики могут конструировать веб-приложения, применяя привычный для них синтаксис Python, что способствует более удобному и эффективному процессу разработки. Библиотека предлагает простое и интуитивно понятное программное API для работы с компонентами веб-страницы, такими как кнопки, поля ввода текста, изображения и прочие элементы. Из особенностей PyScript стоит выделить способность обновлять содержимое веб-страницы без полной перезагрузки страницы. Это позволяет создавать динамические приложения, реагирующие на действия пользователя в реальном времени.

PyScript также обеспечивает инструментарий для работы с AJAX запросами, что открывает возможности для обмена данными с сервером без необходимости перезагрузки страницы. Это содействует созданию более отзывчивых и производительных веб-приложений.

PyScript представляет собой ценный инструмент для разработчиков, стремящихся создать динамические и интерактивные веб-приложения с применением языка программирования Python, улучшая тем самым опыт пользователей взаимодействия с веб-интерфейсами.

Также рассмотрим библиотеку **Brython** — это интересная библиотека, которая позволяет запускать Python код прямо в браузере. Она представляет собой реализацию интерпретатора Python на JavaScript, что позволяет выполнять код Python напрямую в окружении браузера, без необходимости установки Python на сервере или компьютере пользователя.

С помощью Brython разработчики могут создавать веб-приложения, в которых Python используется как язык программирования для взаимодействия с элементами страницы, обработки событий, выполнения вычислений и многого другого. Это открывает широкие возможности для создания интерактивных и динамичных веб-приложений с использованием знакомого и удобного синтаксиса Python.

Brython позволяет использовать множество стандартных модулей Python в веб-приложениях, что делает процесс разработки более эффективным и продуктивным. Благодаря этой библиотеке разработчики могут переносить свои навыки и знания Python в область веб-разработки, не углубляясь в изучение JavaScript.

Кроме того, Brython обеспечивает возможность создания одностраничных приложений (SPA - Single Page Applications), которые загружаются и выполняются быстро, поскольку весь необходимый код находится в одном файле, а рендеринг страницы происходит динамически без перезагрузки страницы.

Однако, стоит учитывать, что Brython может быть несколько медленнее по сравнению с нативным JavaScript, так как при выполнении Python кода через интерпретатор на JS происходит дополнительный уровень абстракции. Тем не менее, для многих задач и проектов, особенно для тех, кто уже знаком с Python, использование Brython может быть привлекательным вариантом.

При желании опробовать Brython достаточно подключить эту библиотеку на страницу веб-приложения, а затем можно начинать использовать Python для создания динамичного и интерактивного веб-интерфейса.

* + 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 примеры наиболее популярных IDE:

1. Visual Studio — это интегрированная среда разработки, разработанная Microsoft, поддерживает широкий спектр языков программирования.
2. Eclipse — IDE с открытым исходным кодом, поддерживающая широкий спектр языков программирования, включая Java, C++ и Python.
3. Xcode — разработанная Apple, эта IDE используется для разработки приложений для macOS, iOS, watchOS и tvOS.
4. IntelliJ IDEA — разработанная JetBrains, эта IDE обеспечивает поддержку Java, Kotlin и других языков.
5. PyCharm — также разработанная JetBrains, эта IDE ориентирована на разработку Python.
6. Visual Studio Code (VS Code) — это наиболее широко распространённый бесплатный (что порой является немаловажным преимуществом) редактор кода с открытым исходным кодом, разработанный Microsoft. Он широко используется разработчиками для различных задач программирования, включая веб-разработку и машинное обучение.

Интерактивные среды разработки очень популярны среди разработчиков, так как предоставляют огромное количество преимуществ:

1. Они как правило доступны для самых различных операционных систем: Windows, macOS и различные вариации Linux.
2. Каждая из подобных программ— очень гибкий и настраиваемый редактор кода, предоставляющий ряд мощных функций для разработчиков.
3. Каждая IDE имеет сообщество пользователей и разработчиков (зачастую большое и активное), а в Интернете доступно множество ресурсов, которые помогут начать работу и научиться эффективно использовать программу – от различных курсов до официальной документации.
4. Одним из главных преимуществ IDE являются обширные наборы расширений и плагинов, которые позволяют разработчикам настраивать каждый редактор под свои конкретные нужды.
5. Любая IDE поддерживает широкий спектр языков программирования, например, JavaScript, Python, Ruby (и многие другие) и имеет доступные расширения для всего: от подсветки синтаксиса и форматирования кода до отладки и интеграции с Git. поддерживает широкий спектр языков программирования, включая
6. IntelliSense: функция предоставляет интеллектуальные предложения по завершению кода по мере ввода в зависимости от языка, с которым вы работаете.
7. Встроенная поддержка Git: большинство редакторов включает встроенную поддержку Git, что позволяет легко управлять репозиториями кода, не выходя из редактора.
8. Поддержка отладки: как правило, любая IDE предоставляет встроенный отладчик для ряда языков программирования, позволяющий легко выполнять код, выявлять и исправлять ошибки.
9. Запуск задач: поддержка создания пользовательских задач, которые могут автоматизировать общие рабочие процессы, такие как создание и тестирование кода.
10. Встроенный терминал: IDE включает встроенное окно терминала, позволяющее выполнять команды оболочки непосредственно в редакторе.
11. Настройки тем: любая среда разработки включает в себя ряд встроенных цветовых тем, и вы также можете загружать и устанавливать пользовательские темы, чтобы настроить внешний вид редактора.

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

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

Bootstrap включает в себя готовые компоненты, такие как меню навигации, кнопки, формы, карусели, модальные окна и многие другие, которые можно использовать для создания веб-сайтов. Он также содержит набор классов, которые можно использовать для быстрого и простого применения стилей к элементам на странице. Bootstrap можно использовать как самостоятельный фреймворк или в сочетании с другими технологиями, такими как HTML, CSS, JavaScript, jQuery и другими. Он также поддерживает различные плагины и инструменты, такие как Sass и Less, которые облегчают разработку и обслуживание веб-сайтов.

* + 1. Git.

Git представляет собой распределенную систему управления версиями, широко используемую в разработке программного обеспечения. Она обладает высокой степенью гибкости и позволяет эффективно отслеживать изменения в исходном коде проектов. Вот несколько ключевых аспектов Git:

* 1. Репозиторий: в Git проекты хранятся в репозиториях. Репозиторий содержит всю историю изменений проекта, включая все коммиты, ветки и теги. Репозиторий в Git представляет собой контейнер, хранящий всю историю изменений, включая коммиты, ветки и теги.
  2. Клонирование: Для работы с проектом в Git разработчики обычно клонируют (копируют) удаленный репозиторий на свой локальный компьютер. Клонирование репозитория позволяет разработчикам сохранить локальную копию проекта, что обеспечивает возможность работы с кодом в офлайн-режиме.
  3. Коммит: Коммит представляет собой фиксацию изменений в коде. Каждый коммит фиксирует изменения кода и содержит описание и уникальный идентификатор.
  4. Ветвление и слияние: Git поддерживает возможность создания веток (branches) для разработки новых функций или исправлений. Ветвление и слияние позволяют разработчикам работать одновременно над различными функциональными модулями и объединять их в рамках единого проекта. После завершения работы ветки разработчики могут выполнить слияние (merge) ветки со стабильной версией проекта.
  5. Откат изменений: Git позволяет откатывать изменения в коде до определенного состояния с использованием команды git reset или git revert. С помощью отката изменений можно управлять конфликтами в процессе разработки.
  6. Удаленные репозитории: Разработчики могут сотрудничать над проектом, обмениваясь коммитами через удаленные репозитории, такие как GitHub, GitLab, Bitbucket и другие. Удаленные репозитории позволяют эффективно совместно работать над проектами, обменявшись изменениями и обновлениями.
  7. Стэш: функция стэша (stash) позволяет временно сохранить незакоммиченные изменения, чтобы потом вернуться к ним. Использование функции стэша помогает временно сохранить изменения, предотвратить их потерю и вернуться к ним в будущем.

Git выделяется своей мощностью и универсальностью в контексте управления версиями программного обеспечения и является мощным инструментом для работы с проектами любого масштаба и позволяет эффективно управлять версиями кода, обеспечивая прозрачность изменений и возможность отката.

Git является распределенной системой управления версиями, широко применяемой в области разработки программного обеспечения. Ее основным принципом является отслеживание изменений в коде и сохранение истории разработки. Репозиторий Git представляет собой структуру данных, хранящую снимки проекта на различных этапах его развития. Возможность клонирования репозитория Git предоставляет разработчикам копию проекта для работы над ним локально.

Основные платформы для работы с Git включают локальные установки на компьютерах разработчиков, также платформы для хостинга репозиториев, такие как GitHub, GitLab и Bitbucket. Каждая из этих платформ предоставляет разработчикам возможность совместной работы, отслеживания изменений и управления проектами. Кроме того, существуют графические интерфейсы для Git, такие как GitKraken, которые облегчают взаимодействие с системой управления версиями и улучшают опыт разработчиков.

Вот несколько основных платформ, на которых можно использовать Git:

* 1. Локальная установка: Git можно запустить на любом компьютере, установив его локально. Это позволяет разработчикам вести управление версиями своих проектов без необходимости доступа к Интернету.
  2. GitHub: GitHub - популярный веб-сервис для хостинга репозиториев Git. Разработчики могут загружать свои проекты на GitHub, делиться ими с другими пользователями и участвовать в совместной работе над кодом.
  3. GitLab: GitLab - еще одна популярная платформа для хостинга репозиториев Git. Она предоставляет различные инструменты для управления проектами, непрерывной интеграции и построения.
  4. Bitbucket: Bitbucket - еще одна популярная платформа для хостинга репозиториев Git. Она предлагает возможности совместной работы над кодом, управления правами доступа и интеграции с другими инструментами разработки.
  5. GitKraken: GitKraken - это графический интерфейс Git, который предоставляет интуитивно понятный способ взаимодействия с репозиториями Git. Он поддерживает множество операционных систем и имеет удобные функции работы с ветвлением, слиянием и другими аспектами Git.

Эти платформы предоставляют разработчикам удобные способы работы с Git, управления кодом и совместной разработки проектов. Они облегчают процесс разработки, отслеживания изменений и совместной работы над кодом.

1. Практическая часть. Разработка проекта.

Веб-сайт – это 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 – специализированное программное обеспечение 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.

Создание веб-сайта или веб-страницы включает в себя несколько основных шагов. Попробуем разбить процесс разработки на несколько этапов, но чёткую границу между ними провести более чем затруднительно, так как все они взаимосвязаны и переходят друг в друга, конкретные шаги и детали всегда могут варьироваться в зависимости от требований и условий проекта:

1. Этап планирования: перед тем как приступить к созданию веб-страницы, важно провести подробное планирование. Определим цели, аудиторию, функциональность, дизайн и структуру страницы. Разработаем концепцию и схему сайта. Определим структуру и содержимое веб-страницы, предварительно наметим различные разделы и элементы – заголовки, параграфы, списки, изображения, таблицы и т. д. Одной из важнейших задач данного этапа является выбор инструментов разработки – определение используемых редактора кода, вида СУБД, языка программирования и фреймворков.
2. Этап вёрстки: на следующем этапе мы переходим к вёрстке страницы. Вёрстка представляет собой создание HTML-структуры страницы. С помощью языка гипертекстовой разметки HTML определим структуру контента и его размещение на странице, поделим элементы на блоки, параллельно добавив им соответствующие теги и атрибуты. Базовая структура любой интернет-странички включает в себя создание объявления DOCTYPE, открывающего и закрывающего тегов: <html>, <head> и <body>. Внутри <head> могут быть добавлены различные метаданные, такие как заголовок страницы, описание, ключевые слова, ссылки на CSS файлы и другие. Используем <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 макета нашего сайта. Параллельно либо сразу после создания структуры сайта стилизуем наши веб-страницы с помощью CSS. Настройки CSS служат для определения внешнего вида страницы, задания цветов, шрифтов, размеров и расположения элементов. Стилистические правила могут быть применены к любым элементам сайта, это могут быть как CSS-стили, так и встроенные стили для форматирования и стилизации элементов. Также можно использовать классы и идентификаторы, для применения стилей к определенным элементам на странице. CSS позволяет задавать различные свойства, такие как цвет текста, шрифты, размеры, отступы, рамки и т. д.
3. Этап наполнения веб-страницы: после создания структуры и стилизации страницы добавим в соответствующие разделы страницы контент – заголовки, тексты, изображения, видео, таблицы или другие элементы, которые несут информацию или предоставляют функциональность на сайте.
4. Этап настройки навигации и интерактивности: оформим для нашей веб-страницы навигационные элементы – меню, ссылки и кнопки, чтобы пользователи могли перемещаться по страницам и взаимодействовать с контентом. При необходимости реализуем интерактивные элементы с помощью JavaScript, если необходимо, чтобы страница была более динамичной и функциональной. Добавим ссылки для перемещения по нашей веб-странице или на другие страницы. Если необходимо создать формы, можно воспользоваться тегом <form> в HTML. Можно добавить элементы формы, такие как текстовые поля, полосы прокрутки, флажки, кнопки отправки и другие, чтобы пользователи могли отправлять данные с веб-страницы.
5. Этап тестирования: прежде чем опубликовать веб-страницу, протестируем результаты нашей работы. Проверим функциональность всех элементов, корректность работы приложения в разных браузерах и на разных устройствах. При необходимости проведём отладку, пока страница не начнёт отображаться и функционировать должным образом. Проверим также HTML-код на ошибки и правильность работы, воспользовавшись инструментами проверки валидности кода.
6. Этап публикации: после успешного тестирования опубликуем наше веб-приложение на сервере, чтобы оно было доступно в сети. Зарегистрируем доменное имя, выбрав хостинг-провайдера либо воспользуемся соответствующими интернет-сервисами и загрузим файлы на сервер.
   1. Этап планирования разработки.

Прежде, чем приступить к разработке и публикации нашего веб-приложения, нам необходимо выполнить целый ряд последовательных действий. Для начала был проведён анализ наших потребностей, исходя из предполагаемого взаимодействия пользователей с разрабатываемым нами приложением, наш сайт должен быть привлекательным и интуитивно понятным для конечного пользователя.

Далее мы определились с архитектурой сайта —способом организации страниц, доступа к ним и навигации и его структурой. Наиболее универсальным вариантом является древовидная структура сайта, при которой 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.

Далее был разработан эскиз графического оформления нашей веб-страницы. Качественный дизайна сайта подразумевает соответствие минимальным критериям:

1. элементы сайта выстроены по сетке;
2. в дизайне соблюдается стилистика;
3. цвета использованы грамотно;
4. в дизайне присутствует только одна основная шрифтовая пара;
5. иллюстрации и фото качественные, в хорошем разрешении;
6. сайт адаптирован под разные устройства;
7. механика элементов понятна и логична;

Проектирование дизайна - это процесс создания визуального и функционального концепта для продукта или проекта, который включает в себя аспекты внешнего вида, пользовательского интерфейса (User Interface, UI) и пользовательского опыта (User Experience, 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. Ключевые этапы и аспекты проектирования дизайна:

1. Исследование и анализ;
2. Разработка концепции;
3. Создание макетов;
4. Выбор цветовой палитры и типографики;
5. Работа с изображениями и графикой;
6. Адаптивный и мобильный дизайн;
7. Тестирование и рефинемент;
8. Релиз;
9. Поддержка и обновление;

Для полного обозначения разработки дизайна сайта было необходимо определиться с количеством размещаемого на сайте материала. Разграничивать материал удобно при помощи 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.

В современных условиях, при обилии всевозможных стационарных, портативных и мобильных устройств выходит на первый план адаптивная вёрстка сайта, которая позволяет веб-страницам автоматически подстраиваться под экраны планшетов и смартфонов. 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. Например:

1. На экране настольного компьютера либо широкоэкранного ноутбука/моноблока – контент веб-страницы будет выглядеть комфортно для просмотра в расширении минимум 1920 на 1280 пикселей;
2. На экране планшета либо широкоэкранного смартфона контент веб-страницы будет выглядеть комфортно для просмотра в разрешении минимум 768 пикселей по одной из сторон, но более комфортным будет расширение минимум 1024 рх;
3. На экране обычного смартфона (средней ценовой категории) контент веб-страницы будет выглядеть комфортно для просмотра в разрешении минимум 480 пикселей по одной из сторон;

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.

Очень важным элементом дизайна является типографика. Типографика в веб-дизайне – это свод правил оформления текста и графических элементов сайта. В данном проекте была использована следующая типографика:

1. Основной шрифт: Gothic Joker;
2. Размер шрифта: 14рх;
3. Цвет шрифта: #ffffff;

В качестве инструментов для разработки нашего проекта были выбраны:

1. Язык гипертекстовой разметки HTML – frontend-часть проекта;
2. Каскадные таблицы стилей CSS – frontend-часть проекта;
3. Фреймворк Flask – backend-часть проекта;
4. СУБД SQLite – backend-часть проекта;

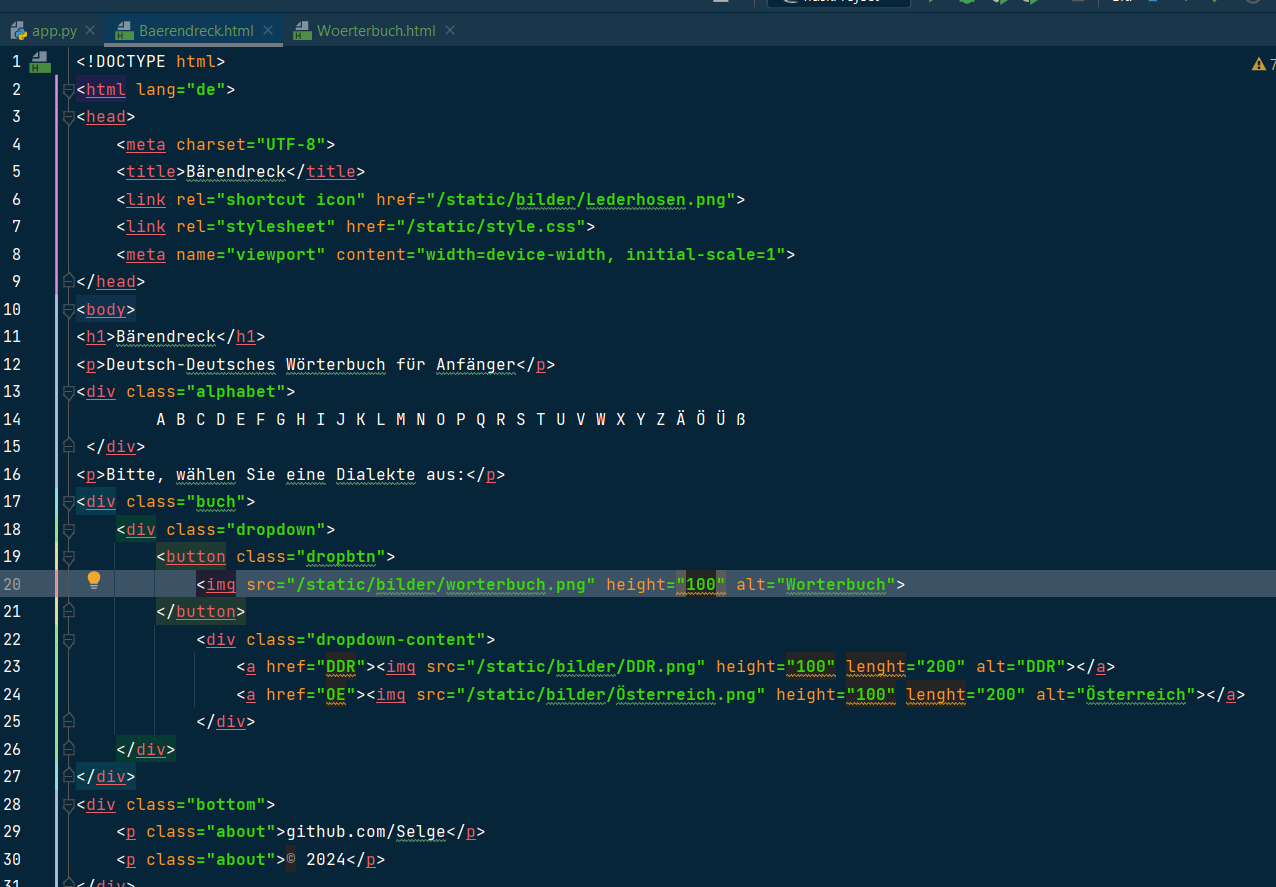
Написание проекта было произведено в интерактивных средах разработки JetBrains WebStorm (frontend-часть проекта) и JetBrains PyCharm (backend-часть проекта).

* 1. Этап разработки проекта.

На предыдущем этапе мы схематично спроектировали наш будущий веб-сайт и определили стек технологий для его создания. Процесс разработки состоит из последовательного создания сначала frontend, а затем backend частей проекта.

Процесс создания Frontend-части нашего проекта состоит из следующих элементов:

1. Верстка HTML и CSS;
2. Текстовое наполнение;
3. Взаимодействие сайта с браузерами – настройка адаптивной вёрстки;

Вёрстка веб-сайта – это комплексный процесс, объединяющий в себе разработку HTML-структуры, применение каскадных таблиц стилей (CSS) для визуального оформления элементов, а также в случае необходимости интеграцию интерактивности с использованием языка программирования JavaScript. Сначала создаются файлы HTML, содержащие семантическую

*Рис. Образец кода HTML.*

разметку контента, и файлы CSS для стилизации визуального представления сайта.

Для обеспечения удобства использования на различных устройствах и экранах, вёрстка также включает работу с медиа-элементами и реализацию адаптивного дизайна. Оптимизация кода и ресурсов, а также развертывание на хостинге будут являться завершающим этапом процесса вёрстки веб-сайта.

*Рис. Образец кода CSS*

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

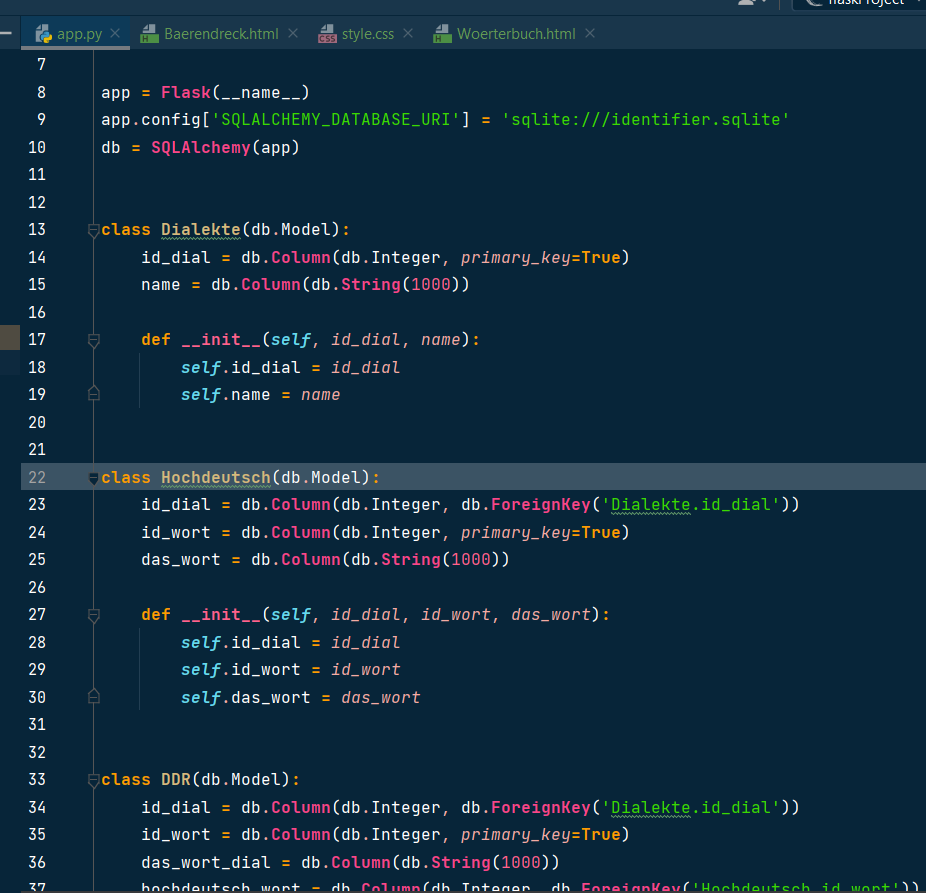
1. <head> – заголовок документа;
2. <body> – тело документа;
3. <div>, <p> – теги-контейнеры для разделов HTML-документа;
4. <button> – интерактивная кнопка;
5. <link>

После определения структуры страницы можно начинать оформлять стили веб-документа и добавлять в него контент. Это может быть текст, изображения, видео, таблицы и другие элементы, которые создаются с помощью соответствующих HTML-тегов, а также таблицы CSS. Разработчики могут создавать стили в отдельных CSS-файлах с расширением .css и затем подключать их к HTML-документам с помощью тега <link>. Также стили можно определять непосредственно внутри HTML-документа с помощью тега <style>. В нашем проекте мы будем пользоваться настройкой стилей при помощи внешних подключаемых файлов .css. Один из основных принципов CSS — это применение стилей с использованием селекторов, которые выбирают элементы на веб-странице, и свойств, которые определяют внешний вид этих элементов.

Для более гибкой и точечной стилизации различных элементов на странице мы будем применять классы и идентификаторы. Классы обозначаются символом точки . перед их именем, а идентификаторы - символом решетки #. При задании различных свойств в CSS важно учитывать единицы измерения (например, пиксели, проценты, em и т.д.), чтобы достичь нужного внешнего вида элементов и адаптировать его к различным устройствам и разрешениям экранов.

После определения правил CSS и заполнения файла .css следует применить их к соответствующим элементам на веб-странице. Это может быть сделано путем указания классов, идентификаторов или тегов элементов в CSS-правилах.

Разработка backend части нашего проекта – это разработка веб-приложения на Flask, процесс, включающий несколько этапов, начиная с настройки среды разработки и заканчивая развертыванием приложения для публичного доступа.

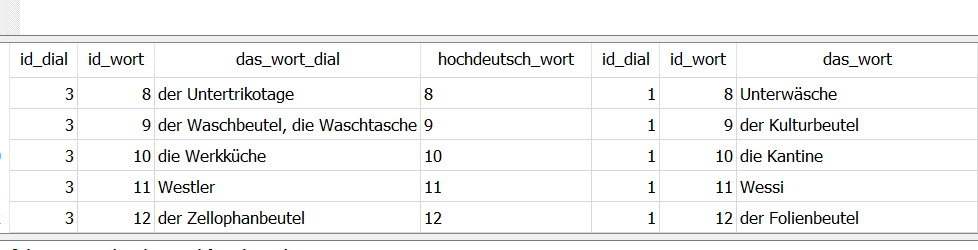
Установим Flask с помощью команды pip: **pip install Flask** и, создав рабочую директорию для нашего проекта, инициализируем виртуальное окружение. Затем создадим файл приложения, например, как в нашем случае, просто **app.py**. Фреймворк Flask был выбран для нашего проекта в том числе из-за своей простоты, лёгкости и скорости разработки, позволяющей в кратчайшие сроки развернуть простое, но функциональное приложение, что как раз удовлетворяет задаче настоящего проекта.

*Рис. Образец кода Flask*

Импортируем класс Flask, создадим экземпляр приложения, определим маршруты и создадим функцию-обработчик для отображения страниц нашего приложения.

Для работы с базами данных в Flask используют расширение SQLAlchemy или другие адаптеры, такие как Flask-SQLAlchemy. Мы определим модели данных, создадим таблицы в базе данных и настроим соединение с базой данных. Прежде чем начать заполнение базы данных, необходимо создать саму базу данных SQLite. Обычно это делается с помощью инструментов администрирования баз данных, таких как DB Browser for SQLite или средств командной строки SQLite.

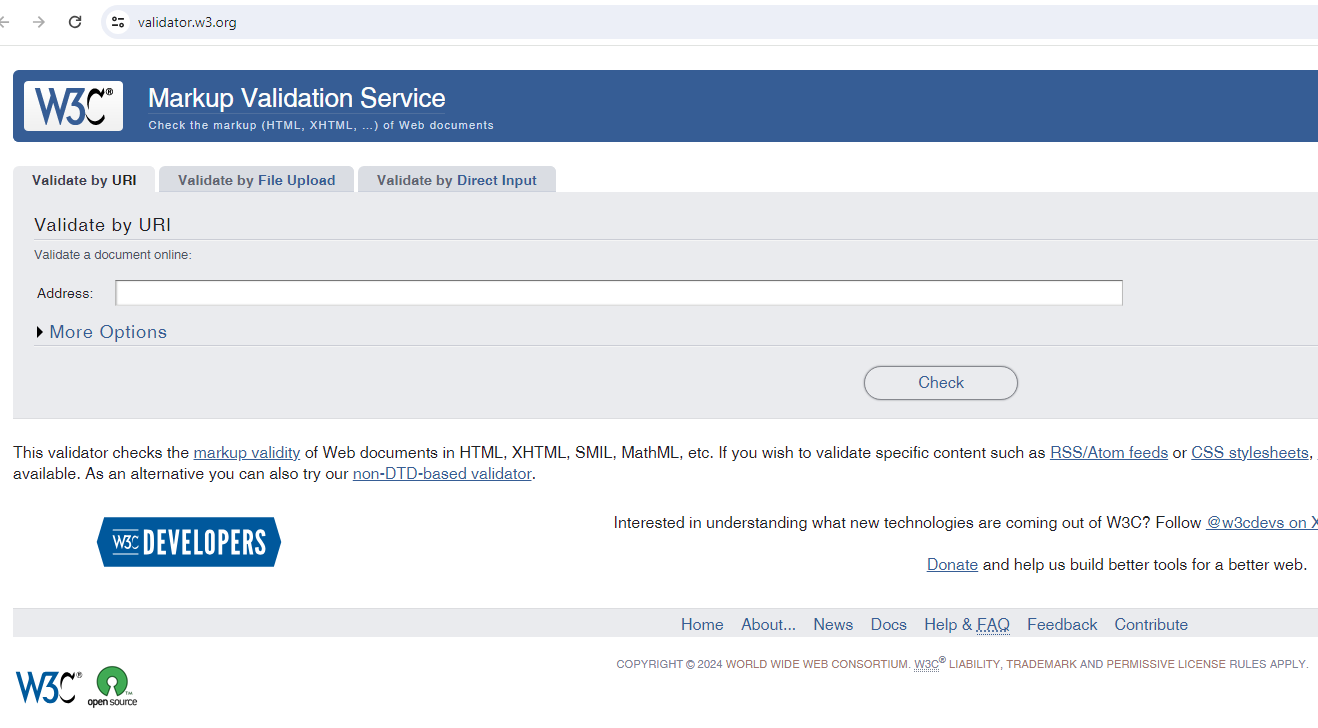
Определим необходимую структуру таблиц базы данных, включая названия таблиц, поля и типы данных для каждого поля. Это можно сделать, например с помощью SQL-запросов CREATE TABLE. Существует несколько способов заполнения данных в таблицы базы данных SQLite: можно заполнять вручную для небольших объемов данных, например, можно добавлять записи в базу данных вручную с помощью SQL-запросов INSERT, импортируя из CSV или других форматов (можно импортировать данные из файлов CSV или других форматов с помощью инструментов или библиотек Python).

Возможно также использование ORM (Object-Relational Mapping). При использовании ORM, такого как SQLAlchemy в Flask, есть возможность создавать объекты Python, которые автоматически сохраняются в базу данных. Для добавления данных в таблицы базы данных SQLite можно также использовать SQL-запросы INSERT INTO.

*Рис. Образец таблиц из базы данных проекта.*

* 1. Этап тестирования проекта.

В рамках нашего проекта стоит задача провести тестирование нашего проекта. Сюда можно включить предварительную проверку сайта на ошибки, проверку кода и оценку работоспособности UI/UX.

Оценку правильности, валидности нашего html-кода и css (валидный код сайта— это код, написанный по спецификации W3C (Консо́рциума Всеми́рной паути́ны, World Wide Web Consortium, W3C), в которой собраны стандарты и рекомендации по удобству и универсальности Всемирной сети) мы проведём с помощью онлайн-валидатора <https://validator.w3.org/> - the W3C Markup Validation Service.

*Рис. Страница онлайн-валидатора кода.*

При написании кода стоит придерживаться правил W3C. Проверка кода на валидность позволяет увидеть все подобные ошибки и исправить их. Наш код могут смотреть как другие разработчики, так и браузер, а также поисковые машины. И чтобы страница быстрее загружалась, правильнее обрабатывалась, а поисковые машины верно понимали смысл всех тегов, важно писать валидный код.

После написания CSS-стилей важно провести визуальное тестирование, чтобы убедиться, что внешний вид страницы соответствует задуманному. Различные инструменты разработчика в браузерах помогут выявить и исправить ошибки в CSS. Как только CSS-стили разработаны и применены к HTML-коду, веб-страница приобретает желаемый внешний вид и форматирование, делая пользовательский опыт более привлекательным и удобным.

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

1. 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.
2. 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.
3. 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.
4. 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.
5. 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.
6. 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. Тестирование сайта можно осуществить с помощью Google PageSpeed Tools.

* 1. Этап публикации проекта.

В рамках настоящего проекта стояла в первую очередь задача изучения теоретического материала и разработки веб-приложения, задача публикации проекта непосредственно в сети интернет не рассматривалась, но не рассмотреть этот вопрос хотя бы в теории было бы неправильно.

Существует два современных способа опубликования веб-страниц в сети: с помощью профессионального хостинга либо с помощью бесплатных сервисов типа Heroku и прочих.

Хостинг – это сервис, который предоставляет серверное пространство и ресурсы для размещения ваших веб-сайтов, файлов и приложений, делая их доступными для пользователей в Интернете. Хостинг-провайдеры предлагают инфраструктуру и техническую поддержку для обеспечения надежности и производительности веб-проекта. Как правило, хостинг-компании предоставляют широкий спектр услуг, от общедоступного веб-хостинга до выделенных серверов и облачного хостинга, чтобы удовлетворить потребности различных типов веб-проектов, удобные инструменты управления файлами, базами данных, электронной почтой и другими функциями вашего веб-проекта.

Публикация веб-сайта с использованием платформы Heroku (рассмотрим её в качестве примера) представляет собой относительно простой и удобный процесс. Вот общий шаг за шагом пример того, как это можно сделать:

* 1. Регистрация на Heroku:

Прежде всего необходимо зарегистрироваться на официальном сайте Heroku: https://www.heroku.com. После регистрации потребуется подтвердить свой аккаунт.

* 1. Установка инструментов Heroku:

После регистрации потребуется установить инструмент командной строки Heroku (Heroku CLI), который позволит управлять приложениями Heroku непосредственно из терминала компьютера.

* 1. Создание приложения на Heroku:

Далее создадим новое приложение на Heroku с помощью команды Heroku CLI. Например, можно воспользоваться командой heroku create, чтобы создать новое приложение.

* 1. Настройка репозитория:

Свяжем репозиторий с приложением Heroku. Это можно сделать через Git, а именно добавив удаленный репозиторий Heroku к проекту.

* 1. Загрузка кода на Heroku:

Используем команду git push heroku main (где main - наша основная ветка) для загрузки кода веб-сайта на Heroku.

* 1. Запуск приложения:

После успешной загрузки кода запустим приложение на Heroku с помощью команды heroku ps:scale web=1, которая укажет Heroku на то, что нужно запустить один веб-процесс.

* 1. Дополнительная настройка:

При необходимости можно провести дополнительные настройки, такие как установка переменных окружения, настройка базы данных и другое.

* 1. Просмотр приложения:

После завершения этих шагов веб-сайт должен быть доступен по URL, который будет предоставлен Heroku. Обычно это будет в формате https://название-вашего-приложения.herokuapp.com.

* 1. Поддержка и масштабирование:

Heroku предоставляет возможности масштабирования приложений, а также множество инструментов для мониторинга и управления, помогая следить за состоянием приложения и его производительностью.

Таким образом, публикация веб-сайта с помощью Heroku может быть выполнена относительно быстро и легко, с минимальными усилиями по настройке инфраструктуры.

Существует несколько альтернативных платформ, которые предлагают схожий функционал с Heroku. Вот некоторые из них:

1. Amazon Web Services (AWS): AWS предлагает широкий спектр услуг облачных вычислений, включая возможность развертывания веб-приложений. С помощью таких сервисов, как Amazon Elastic Beanstalk или AWS Elastic Beanstalk, можно развернуть и масштабировать веб-приложения с минимальными усилиями.

2. Microsoft Azure: Microsoft Azure также предлагает платформу облачных вычислений, которая включает в себя сервисы для развертывания и управления веб-приложениями. Azure App Service позволяет легко развернуть веб-приложения на платформе Azure.

3. Google Cloud Platform (GCP): GCP предоставляет ряд сервисов, таких как Google App Engine, которые помогают развертывать и масштабировать веб-приложения. GCP также предлагает инфраструктуру как услугу (IaaS) и платформу как услугу (PaaS) для хостинга веб-приложений.

4. DigitalOcean: DigitalOcean предоставляет облачные инфраструктурные услуги, включая развертывание веб-приложений на виртуальных серверах (Droplets). Они также предлагают удобные сервисы, такие как App Platform, для развертывания и управления веб-приложениями.

5. Netlify: Netlify это платформа для развертывания статических и динамических веб-сайтов. Она предлагает удобные инструменты для разработчиков, такие как автоматическая оптимизация, развертывание из Git и множество интеграций со сторонними сервисами.

Каждая из этих альтернативных платформ имеет свои уникальные особенности и преимущества, поэтому выбор конкретной платформы может зависеть от требований вашего проекта, предпочтений в разработке и бюджета.

1. Заключение.

В рамках данной дипломной работы была проведена разработка и создание онлайн-словаря вариантов немецкого языка. Этот проект ставил перед собой цель создать веб-приложение, предоставляющее пользователям доступ к таблицам вариантов лексики немецкого языка.

В ходе работы были решены следующие задачи:

1. Анализ задач по разработке веб-приложения с учётом потенциального взаимодействия пользователей с предоставляемым контентом.
2. Проектирование пользовательского интерфейса, обеспечивающего легкий доступ к информации и удобство взаимодействия с приложением.
3. Разработка структуры базы данных и веб-приложения, обеспечивающих хранение, обработку и доступ к информации.

В данном дипломном проекте был разработан онлайн-словарь с использованием технологий HTML, CSS, Flask. 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 HTML и CSS, а backend был реализован на Flask.

В результате проделанной работы было разработано лёгкое, изящное, красивое онлайн-приложение, которое может быть полезным всем интересующимся или изучающим немецкий язык.

Работа над данным дипломом позволила приобрести ценный опыт в области веб-разработки, проектирования пользовательских интерфейсов и анализа пользовательского опыта. В процессе разработки проекта были рассмотрены основные принципы и подходы к созданию веб-приложений, включая UI/UX дизайн, frontend и backend разработку, а также использование различных технологий для достижения поставленных целей.

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

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.