|  |  |
| --- | --- |
| *voenmeh* | МИНОБРНАУКИ РОССИИ  Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»**  **(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)** |
| БГТУ.СМК-Ф-4.2-К5-01 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Факультет |  | И |  | Информационные и управляющие системы |
|  |  | шифр |  | Наименование |
| Кафедра |  | И9 |  | Систем управления и компьютерных технологий |
|  |  | шифр |  | Наименование |
| Дисциплина |  | Научно-исследовательская работа в семестре | | |

КУРСОВАЯ РАБОТА

на тему

|  |
| --- |
| Выбор языка программирования для |
| написания динамических веб-страниц |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнила магистрант группы | | | |  | | И9М33 |
| Гаврютина А.А. | | | | | | |
| Фамилия И.О. | | | | | | |
| **ПРЕПОДАВАТЕЛЬ** | | | | | | |
| Каминский В.Н. | |  |  | | | |
| Фамилия И.О. Подпись | | | | | | |
| Оценка |  | | | |  | |
| «\_\_\_\_\_» |  | | | | 2018 г. | |

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc513205418)

[ВИДЫ ВЕБ-СТРАНИЦ 4](#_Toc513205419)

[ЯЗЫКИ, ИСПОЛЬЗУЕМЫЕ ДЛЯ СОЗДАНИЯ ВЕБ-СТРАНИЦ 10](#_Toc513205420)

[1. PHP 11](#_Toc513205421)

[2. Ruby 13](#_Toc513205422)

[3. Perl 16](#_Toc513205423)

[4. Node.js 17](#_Toc513205424)

[5. C# 19](#_Toc513205425)

[КРИТЕРИИ ДЛЯ ВЫБОРА ЯЗЫКА 24](#_Toc513205426)

[ФОРМАЛИЗАЦИЯ КРИТЕРИЕВ. МЕТОД ЭКСПЕРТНОЙ ОЦЕНКИ. МЕТОД МНОГОКРИТЕРИАЛЬНОГО РАНЖИРОВАНИЯ КРИТЕРИЕВ 26](#_Toc513205427)

[ТЕКСТ ПРОГРАММЫ РЕАЛИЗАЦИИ АЛГОРИТМА МНОГОКРИТЕРИАЛЬНОГО РАНЖИРОВАНИЯ АЛЬТЕРНАТИВ 32](#_Toc513205428)

[ЗАКЛЮЧЕНИЕ 42](#_Toc513205429)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 43](#_Toc513205430)

ВВЕДЕНИЕ

Одной из основных задач при разработке любого программного продукта, в том числе сайта является выбор средств и инструментов разработки. При существующем многообразии языков программирования встает вопрос о выборе наиболее подходящего.

Целью курсовой работы является выбор языка программирования для создания динамических веб-страниц.

Для достижения цели были поставлены следующие задачи:

* Ввести понятие «динамической веб-страницы».
* Выдвинуть критерии, наличие которых необходимо для языка программирования.
* Выбрать и рассмотреть языки программирования, направленные на разработку динамических веб-страниц.
* Рассмотреть алгоритмы выбора на основе экспертных оценок и выбрать наиболее подходящий.

ВИДЫ ВЕБ-СТРАНИЦ

Веб-страница – это документ, хранящийся на веб-сервере, или создаваемый по запросу пользователя и имеющий уникальный адрес. Во всемирной паутине адрес называют URL (от англ. Uniform Resource Locator – унифицированный указатель ресурсов). URL главной страницы сайта называют адресом сайта. Веб-страница имеет следующие синонимы:

* страница;
* HTML-страница;
* гипертекстовый документ;
* HTML-документ;
* веб-документ.

Веб-страницы делятся на:

* статические и динамические;
* активные и пассивные.

Статическая страница хранится в готовом виде на веб-сервере и выдаётся по запросу пользователя. В самом простом случае статическая страница состоит из одного файла, содержащего текст, размеченный тегами языка HTML. В общем случае в состав страницы входят файлы с изображениями, аудио- и видеофайлы, файлы с каскадными таблицами стилей (CSS) и файлы, содержащие скрипты, выполняемые на стороне браузера. HTML-файл содержит ссылки на все остальные файлы страницы [1].

Скриптом в веб-программировании называют программу, служащую для создания или изменения страницы. Различают скрипты, выполняемые на стороне веб-сервера и на стороне браузера. Серверные скрипты обычно пишут на языках Perl, PHP, Node.js, а для браузерных используют JavaScript.

Динамическая страница создаётся (генерируется) на веб-сервере скриптом по запросу пользователя. В остальном, она ничем не отличается от статической.

Свойства страницы, поступившей с веб-сервера на браузер, не зависят от того, динамическая она или статическая. Страница, вид которой в окне браузера в течение всего времени просмотра остаётся неизменным, называется пассивной. Страница, меняющая свой вид под воздействием пользователя (движением или щелчком мышки, нажатием клавиши) называется активной.

Активная страница содержит в себе скрипт, выполняемый на стороне браузера в ответ на действия пользователя.

Итак, страница одновременно может быть:

* динамической и активной,
* динамической и пассивной,
* статической и активной,
* статической и пассивной.

Разделение страниц на статические и динамические позволяет существенно повысить производительность веб-серверов. Время выполнения запроса на выдачу статической страницы в среднем намного меньше, чем на выдачу динамической, так как для формирования последней требуется выполнить скрипт. Значительная часть времени выполнения скрипта приходится на запросы к базе данных.

Статической принято называть страницу, которая целиком хранится на сервере и показывается посетителю в своем неизменном виде (следует учесть, что статическая страница может содержать некоторые изменяемые элементы, например, баннеры, однако она все равно остается статической). Для большей ясности можно пояснить понятие следующим образом. В каждом браузере имеется возможность просмотреть так называемый исходный код страницы. Так вот, если на сервере страница хранится в точности в таком же виде, то это статическая страница. Сервер просто пересылает такую страницу браузеру, который выводит ее на экран монитора и тем самым выдает их посетителю сайта [3].

Статическим называется сайт, все страницы которого являются статическими. Таких сайтов довольно много, с них, можно сказать, начинался интернет. Статические сайты создаются, в основном, на «чистом HTML».

Динамическим следует называть любой сайт, на котором имеется хотя бы одна динамическая страница. Дело в том, что для создания даже одной динамической страницы нужно уже задействовать те механизмы, которые позволяют такие страницы создавать. И если владелец сайта такие механизмы освоил, число динамических страниц на его сайте будет обязательно увеличиваться, а доля статических страниц, соответственно, уменьшаться.

Дальнейшим развитием идеи динамического сайта являются интерактивные сайт — это сайты, который активно взаимодействуют с пользователем, выдавая информацию, различающуюся в зависимости от действий пользователя. Самый характерный пример - игры с удаленным компьютером через Интернет.

Но есть и более практические применения интерактивности: поиск, обращение к базам данных, различные регистрации, формы и прочее. Интерактивность на персональной странице позволяет привлечь больше посетителей, поскольку повышает удобство пользования информацией на сайте (примеры: смена кодировки, языка, выбор дизайна и так далее).

Очевидно, что чтобы сделать сайт динамическим, то есть чтобы у вебмастера появилась возможность создавать динамические страницы, на стороне сервера должны работать какие-то программы, которые формируют выдаваемую страничку из различных частей (создаваемых, возможно, другими программами) и обрабатывают данные, присылаемые пользователями. Следовательно, вебмастер должен при создании такого сайта использовать средства программирования. К ним относятся CGI, PHP, Perl, ASP и т.д.

Преимущества динамических сайтов:

1. Главное достоинство динамических сайтов — это возможность быстрого внесения изменений сразу во все страницы сайта.
2. Упрощение модификации и обновления страниц на сайте.
3. Разделение информации и дизайна сайта. Дизайн сайта - понятие многогранное. Можно понимать дизайн в широком и в узком смысле. Под дизайном в узком смысле понимают только внешний вид сайта: набор цветов или фоновых картинок, выбор шрифтов для текста и заголовков, шаблон страницы (выделение определенных областей страницы для тех или иных блоков, образующих страницу). Использование динамических страниц позволяет именно дизайн в этом узком смысле отделить от содержательного наполнения страницы (контента).
4. Возможность для посетителей изменять контент сайта. Преимущества, описанные в первых двух пунктах, облегчали работу создателей сайтов и снижали их трудозатраты, но для посетителей это было не очень важно. А вот возможность изменять содержание страниц отразилась непосредственно на них. Но очевидно, что если на сайте планируется какая-либо обратная связь с пользователем, например, предоставление пользователю возможности добавлять комментарии по поводу той или иной статьи, или голосование, то никак нельзя обойтись статическими документами.
5. Таким образом, видно, что обойтись одними только статическими документами при создании современного веб-сайта в подавляющем большинстве случаев невозможно. Большая часть страниц, пусть даже с неизменным содержимым, все равно должна собираться динамически, из-за наличия меняющихся меню, элементов оформления и т.д. Такой сайт скорее представляет из себя не набор документов, а программу, работающую на сервере.

Недостатки динамических сайтов

1. Появляется необходимость использования дополнительных программных средств для построения динамического сайта. На статическом сайте все страницы уже готовы, серверу остается только показать их посетителю, а на динамическом сайте необходимо вносить в них какие-то изменения, для этого требуется соответствующие программные решения.
2. В зависимости от сложности сайта, трудоемкость и стоимость разработки таких программ может очень сильно варьироваться. Сейчас существует множество готовых решений для создания сайта, в том числе и бесплатных. В сети можно найти не один десяток всевозможных бесплатных скриптов, которые позволят вам создать на сайте форум, доску объявлений, клуб знакомств, магазин и т.д. Но, если требуется что-то специфическое, все равно не обойтись без дополнительных разработок.
3. Повышение требований к аппаратным мощностям серверных систем. Этот недостаток непосредственно следует из предыдущего, т.к. теперь серверу требуется еще выполнить какую-то программу для модификации страницы сайта, а только потом выдать ее посетителю. Особенно заметной эта проблема становится на сайтах с большой посещаемостью. Часто в таких случаях приходится производить дополнительную оптимизацию кода для нормальной работы сайта.
4. Следовательно, и стоимость услуг по содержанию такого сайта, намного выше, нежели статического. Хотя сейчас даже многие бесплатные хостинги поддерживают возможность создания динамических сайтов, не говоря уже о платных, где все необходимое входит в стандартный набор услуг.
5. Третьим недостатком, также вытекающим из первого, является сложность больших структурных изменений сайта. Вы можете возмутиться, ведь выше было сказано, что динамика облегчает и упрощает модификацию. Но не спешите, давайте разберем этот вопрос подробнее. В данном случае все завязано на программное обеспечение, которое используется на сайте. Именно оно объединяет кусочки дизайна и данных в один полноценный сайт. Но у любой программы есть свои ограничения, и, если требуется получить что-то, что программно не предусмотрено, значит нужно менять программу. Либо дорабатывать старую, либо искать новую [3].

Несмотря на эти недостатки, динамических сайтов в сети становится все больше и больше, видимо, перечисленные выше преимущества перекрывают все недостатки.

Надо сказать, что сейчас все больше набирает силу гибридная схема разработки. Речь идет об AJAX, что означает: Asyncronous JavaScript And XML. Эта технология позволяет скриптам на JavaScript обращаться к какому-либо скрипту на сервере и получать информацию с сервера. Это в свою очередь, дает пользователю гибкость и позволяет перезагружать только часть содержимого страницы, а не всю ее полностью (что значительно экономит трафик).

ЯЗЫКИ, ИСПОЛЬЗУЕМЫЕ ДЛЯ СОЗДАНИЯ ВЕБ-СТРАНИЦ

Условно, разработку сайта можно разделить на две части: серверную часть и клиентскую часть. Рассмотрим каждую из них подробнее.

Клиентская часть (англ. front-end, client-side) — это все, что браузер может читать, выводить на экран и / или запускать. К средствам разработки можно отнести HTML, CSS, JavaScript, ActionScript и др.

Серверная часть (англ. back-end, server-side) — это все, что работает на сервере, то есть «не в браузере» или «на компьютере, подсоединенном к сети (обычно к Интернету), который отвечает на сообщения от других компьютеров».

Для реализации серверной части можно использовать любые инструменты, доступные сервере (который, по сути, является просто компьютером, настроенным для ответов на сообщения). К средствам разработки можно отнести такие языки, как: Ruby, PHP, Perl, Python, Java, C#, Node.js. Также можно использовать системы управления базами данных, такие как MySQL, PostgreSQL, MongoDB, Cassandra, Redis, Memcached.

На сегодняшний день существует несколько основных архитектур, определяющих, как будут взаимодействовать серверная и клиентская части.

1. Серверные приложения

В этом случае HTTP-запросы отправляются напрямую на сервер приложения, а сервер отвечает HTML-страницей.

Между получением запроса и ответом сервер обычно ищет по запросу информацию в базе данных и встраивает ее в шаблон (ERB, Blade, EJS, Handlebars).

1. Связь с использованием AJAX

Другой тип архитектуры использует для связи AJAX (англ. Asynchronous JavaScript and XML). Это означает, что JavaScript, загруженный в браузере, отправляет HTTP-запрос (XHR, XML HTTP Request) изнутри страницы и получает XML-ответ. Сейчас для ответов также можно использовать формат JSON.

Это значит, что у сервера должна быть конечная точка, которая отвечает на запросы JSON- или XML-кодом. Два примера протоколов, используемых для этого — REST и SOAP.

1. Клиентские (одностраничные) приложения

AJAX позволяет вам загружать данные без обновления страницы. Больше всего это используется в таких фреймворках, как Angular и Ember. После сборки такие приложения отправляются в браузер, и любой последующий рендеринг выполняется на стороне клиента (в браузере).

Такой клиент общается с сервером через HTTP, используя JSON- или XML-ответы.

1. Универсальные/изоморфные приложения

Некоторые библиотеки и фреймворки, например, React и Ember, позволяют вам исполнять приложения как на сервере, так и в клиенте.

В этом случае для связи клиента с сервером приложение использует и AJAX, и обрабатываемый на сервере HTML.

В данной работе, т.к. мы предполагаем, что динамическая веб-страница та, которая формируется на стороне сервера, будем рассматривать языки, которые используются для server-side программирования.

Рассмотрим поподробнее несколько языков программирования для написания динамических веб-страниц: Ruby, PHP, Perl, Python, C#, Node.js.

1. PHP

В настоящее время PHP является одним из лидеров среди серверных языков программирования, применяющихся для создания динамических веб-сайтов и веб-приложений. Большая часть коробочных систем управления сайтами написана именно на PHP, язык поддерживается подавляющим большинством хостинг-провайдеров, Язык получил широкое распространение благодаря своей простоте, скорости, мультипарадигмально-сти, богатой функциональности и кроссплатформенности.

Язык популярный, но при этом средний уровень разработчиков ката-строфически низкий: программистами на PHP называют себя даже начинаю-щие HTML-верстальщики. Это в значительной степени определят плохое качество PHP-проектов.

Фреймворки достаточно разрозненные — их много, явного лидера нет, поэтому число разработчиков на отдельно взятом PHP-фреймворке может быть существенно ниже, чем количество разработчиков на том же Ruby on Rails (хотя сам язык программирования Ruby значительно менее распро-странён, нежели PHP).

Также сам язык обладает достаточно большим количеством недостатков, которые проявляются при разработке сложных решений. Во многом, эти недостатки — следствие эволюционного развития PHP и следования обратной совместимости.

Преимущества PHP

* Практичность

Язык программирования должен предоставить программисту средства для решения поставленной задачи. Благодаря своей богатой функциональности PHP отлично подходит для решения широкого спектра задач.

* Простота в изучении

Отсутствие строгой типизации, такой как в Java или C++, делает этот язык простым в изучении. Например, при создании переменной Вам совершенно необязательно указывать её тип или беспокоиться об её усечении или переполнении.

* Традиционность

Изначально PHP создавался как настройка на Perl, поэтому язык сочетает в себе достоинства Perl и С. Код PHP очень похож на написанный на Cи, что заметно снижает усилия при изучении этого языка.

* Эффективность

Один из важнейших факторов при выборе языка программирования. Благодаря своему “движку”, сценарии в PHP выполняются с большой скоростью, что позволяет создавать на PHP серьезные WEB-приложения.

* Гибкость

Так как PHP является встраиваемым языком, это дает чрезвычайную гибкость в процессе разработки. Чаще всего сценарии PHP интегрируются в HTML страницы, но при необходимости могут встраиваться и в JavaScript, WML, XML и другие языки.

* Безопасность

В PHP реализованы гибкие и эффективные средства безопасности, в том числе ряд надежных механизмов шифрования. Кроме того, так как сценарии компилируются на стороне сервера, их исходный текст нельзя просмотреть в браузере. Мелочь, а приятно. Ваши гениальные сценарии не подсмотрит пытливый глаз продвинутого пользователя.

* Базы данных

Одним из самых больших преимуществ PHP является поддержка большого числа баз данных (более 20 видов). Кроме того, PHP поддерживает DBX для работы на абстрактном уровне и ODBC, что позволяет работать с любой базой данных, поддерживающей эти стандарты.

Многие злоупотребляют его гибкостью и стараются применять его повсеместно. К чему это приводит? Только к усложнению структуры кода и некорректному выполнению программы. Область применения PHP, главным образом, сфокусирована на написании серверных сценариев.

1. Ruby

Ruby — интерпретируемый мультипарадигмальный язык программирования: динамический, объектно-ориентированный, рефлективный, императивный, функциональный. Активно используется в веб-разработке, в системном администрировании и в работе операционных систем (Mac OS X, Linux, BSD).

В Руби есть независимая от операционной системы реализация многопоточности, строгая динамическая типизация, сборщик мусора и многие другие возможности. Язык близок по особенностям синтаксиса к языкам Perl и Eiffel, по объектно-ориентированному подходу — к Smalltalk. Также некоторые черты языка взяты из Python, Lisp, Dylan и Клу. Ruby разрабатывался на Linux, но работает на многих версиях Unix, DOS, Microsoft Windows, Mac OS (где по умолчанию встроен в операционную систему), BeOS, OS/2 и т. д.

Ruby начал разрабатываться в начале 1993 и вышел в свет в конце 1995, создатель Ruby — Юкихиро Мацумото (Matz):

«Ruby родился 23 февраля 1993 года. В тот день я беседовал со своим коллегой о возможности существования объектно-ориентированного сценарного языка. Я знал Perl (Perl4, а не Perl5), но он мне не нравился — был в нём некий привкус игрушечного языка (да и поныне есть). А объектно-ориентированный интерпретируемый язык казался многообещающим. В то время я знал Python. Но он мне не нравился потому, что я не считал его настоящим объектно-ориентированным языком. Его OO свойства казались надстройкой над языком. Мне, как языковому маньяку и фанату объектно-ориентированного программирования с пятнадцатилетним стажем, очень, очень хотелось, чтобы был истинно объектно-ориентированный, простой в использовании язык. Я пытался найти такой язык, но его не было. Тогда я решил его создать. Прошло несколько месяцев, прежде чем интерпретатор заработал. Я добавил в мой язык то, что мне хотелось — итераторы, обработку исключений, автоматическую сборку мусора. Затем я переорганизовал свойства Perl и реализовал их как библиотеку классов. В декабре 1995 года я опубликовал Ruby 0.95».

Язык следует принципу «наименьшей неожиданности»: программа должна вести себя так, как ожидает программист. Однако в контексте Ruby это означает наименьшее удивление не при знакомстве с языком, а при его основательном изучении. Сам Мацумото утверждает, что целью разработки была минимизация неожиданностей при программировании для него, но после распространения языка он с удивлением узнал, что мышление программистов похоже, и для многих из них принцип «наименьшей неожиданности» совпал с его принципом.

Ruby также унаследовал идеологию языка программирования Perl в части предоставления программисту возможностей достижения одного и того же результата несколькими различными способами. Люди различны, и им для свободы необходима возможность выбирать. Одной из основных целей разработки было освобождение программистов от рутинной работы, которую вычислитель может выполнять быстрее и качественнее. Особое внимание, в частности, уделялось будничным рутинным занятиям (обработка текстов, администрирование), и для них язык настроен особенно хорошо.

В противовес машинно-ориентированным языкам, работающим быстрее, целью этой разработки был язык, наиболее близкий к человеку. Любая работа с компьютером выполняется людьми и для людей, и необходимо заботиться в первую очередь о затрачиваемых усилиях людей. Язык позволяет максимально быстро и просто для человека выполнить задачу, хотя, возможно, это и потребует дополнительного времени работы компьютера. Принципы программирования и устройства языка иногда выделяются в термин Ruby Way: «просто, но не слишком просто», «принцип наименьшего удивления», вторичность скорости работы программы, динамичность, простые строгие правила, выполнение которых не доходит до педантизма, потребность создавать полезные и красивые программы как причина программирования.

Ruby является мультипарадигменным языком: он поддерживает процедурный стиль (определение функций и переменных вне классов), объектно-ориентированный (всё — объект), функциональный (анонимные функции, замыкания, возврат значения всеми инструкциями, возврат функцией последнего вычисленного значения). Он поддерживает отражение, метапрограммирование, информацию о типах переменных на стадии выполнения (динамическая идентификация типа данных).

В языке Ruby осуществлён простой и удобный механизм для расширения языка с помощью библиотек, позволяющий легко разрабатывать дополнительные библиотеки. Для управления библиотеками и программами Ruby в виде самодостаточных пакетов предназначена система управления пакетами RubyGems (англ. gems, gem — драгоценный камень). Большинство расширений распространяются под свободными лицензиями и могут быть использованы в любом проекте практически без ограничений.

Наиболее популярным Ruby-фреймворком для разработки веб-приложений является [Ruby on Rails](https://web-creator.ru/articles/about_ruby_on_rails).

1. Perl

Язык Perl был создан в 1987 году американским программистом с лингвистическим образованием — Ларри Уоллом. Спустя год после создания, в 1988 году у языка появилась поддержка регулярных выражений, а в 1989 — обработка потока данных, представленных в двоичной форме. Работа над языком Perl не прекращается. В последних версиях Perl появилась поддержка сложных типов данных и объектная модель.

Структура Perl схожа с языком Си. Некоторые свойства языка Perl заимствованы из языков командных оболочек UNIX – систем.

Отличительная черта языка – возможность написания программ из одной строки. Они используются непосредственно в строке вызова командного интерпретатора.

Преимущества языка Perl:

* встроенные средства для работы со сложными структурами;
* свободный синтаксис (одна и та же задача может решаться разными способами);
* много готовых библиотек – модулей;
* поддержка работы с регулярными выражениями;
* простая обработка больших объемов данных;
* возможность программирования объектно-ориентированным или «функциональным» стилем;
* кросс-платформенность.

Для решения некоторых задач часть преимуществ языка Perl превращаются в недостатки. Большое количество модулей затрудняет их поиск, что не нравится программистам, которым из всего многообразия нужны лишь пара конкретных.

В языке сохраняются ранее популярные, но сейчас уже устаревшие подходы. Тем не менее, часть из них может хорошо подойти для специфических программных задач.

Язык Perl популярен среди системных администраторов UNIX и Windows. Его особенности позволяют не привлекать другие программные средства к решению административных задач. С помощью Perl реализуется управление компьютерами из командной строки, контроль одной программы над другой и другие функции.

Perl – подходящее средство автоматизации обработки электронной почты. С его помощью настраивают фильтрацию сообщений по заданным параметрам, организуют рассылку, и решают множество других задач.

Применение CGU – сценариев обусловлено необходимостью обработки данных, введенных пользователем на html–странице. На сервере запускается специальная программа, которая формирует ответ пользователю после ввода данных. Язык Perl получил распространение в этой сфере из-за простоты реализации требуемых функций.

Web-узел представляет собой хранилище текстовой информации в формате html-страниц. Язык Perl разработан с учетом необходимости обработки больших объемов текстовых данных, поэтому широко применяется в этой сфере.

1. Node.js

«Node.js — это среда выполнения JavaScript, встроенная в движок JavaScript V8 от Chrome. Node.js использует управляемую событиями, неблокирующую модель ввода-вывода, которая делает ее легкой и эффективной. Пакетная экосистема Node.js, npm, является самой большой экосистемой библиотек с открытым исходным кодом в мире».

Node js это не реализация языка (JavaScript), не библиотека, не framework и не web-сервер. Можно найти такие определения, как **environment** и **runtime**. На каком-то форуме встретилось следующее краткое определение - это event-loop c библиотеками.

Плюсы и минусы Node.js

**Плюсы**

* Node даёт вам все плюсы asynchronous non-blocking кода, такие как: web-сервер экономнее расходует ресурсы и может поддерживать больше параллельных клиентских соединений для определённого типа приложений, где активно выполняются операции ввода/вывода (I/O).
* Используется JavaScript, более-менее знакомый любому web-разработчику.
* Некоторые библиотеки копируют аналоги из Ruby/Rails стека, поэтому с ними изначально комфортно работать.
* Библиотек/инструментов много, они активно развиваются.

**Минусы**

* Нет основных библиотек/инструментов - у каждого есть множество альтернатив, плюсы/минусы которых не ясны из документации, приходится лично пробовать и выбирать. Очень много заброшенных проектов. Для многих стандартных задач нет какого-то готового и законченного решения, есть только поделки.
* Очень слабая интеграция между инструментами. Приходится при необходимости писать обвертки руками.
* Инструменты ещё не достигли той зрелости, как в стеке Ruby/Rails, они активно развиваются, но пока не достаточно функциональны для привычного Ruby-сту уровня комфорта.
* Опять JavaScript. Это не самый удачный язык и последние изменения в стандартах не очень спасают. Чтобы получить последние стандартизированные возможности, нужно добавлять транспайлер по типу babel.

Для обычного web-приложения, где активно работают с базой данных, подход Node.js обещает заметное повышение производительности.

Переход на Node получается быстрым и плавным, порог вхождения оказался невысоким. Основные трудности пока были связаны с выбором библиотек, их изучением и сравнением между собой.

С другой стороны, после синхронного кода всё получается несколько сложнее. Зрелость экосистемы и готовность библиотек для серверной разработки пока удручает.

1. C#

«Предком» языка C#, стал язык C, который был изобретен на основе

языка B в 1972 году. Авторами языка программирования стали Кен Томпсон, а также Д. Ритчи (Dennis Ritchie), В варианте C язык B расширился за счет явного использования типов, структур и ряда новых операций.

Дальнейшее развитие язык получил в 1984 году, Б. Страуструп (Bjarne Stroustrup, Bell Labs) выступил с проектом языка С++ -объектно-ориентированного расширения языка C, в котором вводится понятие класса как объекта данных.

Спустя долгий срок в 2000 году, корпорация Microsoft выпустила C# («Си шарп»), основным постулатом которого является высказывание: «всякая сущность есть объект». Язык основан на строгой компонентной архитектуре и реализует передовые механизмы обеспечения безопасности кода.

Так же необходимо отметить, что язык программирования С# так же унаследовал многое от языка Java, хотя тот и не входит в основную ветвь его развития, но тем не менее большая часть синтаксиса унаследована языком C# именно от Java.

Язык С# прошел долгий путь к выходу в свет. Несмотря на то, что синтаксис и особенности реализации унаследован языком программирования C# от «прародителей» (С++, Java), возможности этого языка программирования не ограничиваются ими.

К принципиально важным решениям, которые были реализованы, можно отнести следующие:

* компонентно-ориентированный подход к программированию;
* свойства как средство инкапсуляции данных;
* обработка событий (имеются расширения, в том числе в части
* обработки исключений, в частности, оператор try);
* унифицированная система типизации (соответствует идеологии Microsoft .NET в целом);
* делегаты (delegate, развитие указателя на функцию в языках C и C++);
* индексаторы (indexer, операторы индекса для обращения к
* элементам класса-контейнера);
* перегруженные операторы;
* оператор foreach (обработка всех элементов классов-коллекций);
* механизмы boxing и unboxing для преобразования типов;
* атрибуты (средство оперирования метаданными в COMмодели);
* прямоугольные массивы (набор элементов с доступом по номеру индекса и одинаковым количеством столбцов и строк).

На данный момент, С# успешно конкурирует с Java и С++ по популярности. Рассмотрим сходства этих языков.

Для начала перечислим сходства языков программирования C# и Java. Оба языка объектно-ориентированные и предполагают единственность наследования. Так же особенностями, которые делают схожими языки программирования C# и Java, являются механизмы интерфейсов, обработки исключительных ситуаций, нитей (threads). Оба языка имеют строгую типизацию и динамическую загрузку кода при выполнении программы.

От языка программирования C++, языком C# унаследованы:

механизмы:

* «перегруженные» операторы;
* небезопасные арифметические операции с плавающей точкой;
* множество других особенностей синтаксиса.

Исходя из особенностей языка программирования C#, сформулируем основные преимущества данного языка.

* язык программирования C# претендует на подлинную объектную ориентированность (всякая языковая сущность претендует на то, чтобы быть объектом);
* компонентно-ориентированный подход к программированию, способствующий меньшей машинно-архитектурной зависимости результирующего программного кода, гибкости, переносимости и легкости повторного использования (фрагментов) программ;
* ориентация на безопасность кода (в сравнении с С и С++);
* унифицированная система типизации;
* расширенная поддержка событийно-ориентированного программирования.

Несмотря на достоинства, язык С# имеет некоторые недостатки, такие как:

* довольно сложный синтаксис (75% из Java, 10% из C++, 5% из Visual Basic);
* мало свежих концептуальных идей (приблизительно менее чем 10% конструкций языка);
* относительно невысокая производительность (намного медленнее, чем язык C, но сравним с Java).

Ввиду объектно-ориентированного дизайна, C# является хорошим выбором для быстрого конструирования различных компонентов -от высокоуровневой бизнес логики до системных приложений, использующих низкоуровневый код. Также следует отметить, что C# является и Web ориентированным используя простые встроенные конструкции языка компоненты могут быть преобразованы в Web сервисы.

Дополнительными возможностями языка C# являются использование Web технологий, таких как: XML (Extensible Markup Language) и SOAP (Simple Object Access Protocol).

Среда разработки Web сервисов позволяет программисту смотреть Web приложения, как на родные C# объекты, что дает возможность разработчикам соотнести имеющиеся Web сервисы с их познаниями в объектно-ориентированном программировании

Язык C# является предпочтительным для внутреннего программирования и автоматизации в средах Windows. C# - это только один из языков .NET, но его синтаксис в стиле C гораздо более популярен среди программистов, которые привыкли к таким языкам, как Java или C++.

C# хорошо интегрируется с Windows. Для запуска программы на C# в среде Windows не требуется никаких специальных конфигураций. Будь то веб-приложение, служба Windows или настольное приложение, программы на c# легко развертываются в сети. Пока целевой сервер или рабочая станция поддерживает .NET, развертывание программы на C# должно быть плавным переходом от разработки к производству.

Если у вас есть команда разработчиков, Microsoft упростила для команды совместную работу над одним проектом. Visual Studio имеет встроенные средства для управления изменениями и слияний кода. Microsoft предлагает Team Foundation Server версии программного обеспечения и варианты предприятия для больших групп разработчиков.

При использовании для веб-сайта языка C# это означает, что код хранится на общедоступном сервере в двоичной форме. Если ваш сервер взломали, хакер не имеет доступа к исходному коду. По сравнению с другими распространенными языками, такими как PHP, хакер получает доступ к исходному коду, который затем может дать ему доступ к паролям базы данных. С C#, хакер должен декомпилировать ваш исходный код, прежде чем он сможет увидеть критические компоненты.

C# - отличный язык, но у него есть некоторые недостатки.

С кодом, написанным на C# сложнее работать, так как код должен компилироваться каждый раз, когда вносится даже незначительные изменения. При изменении одной буквы в коде необходимо перекомпилировать все приложение и развернуть его снова. Это часто приводит к добавлению ошибок, если незначительные изменения не были тщательно протестированы.

Поскольку C# является частью .NET Framework, сервер, на котором выполняется приложение, должен быть Windows. Другими словами, любое приложение .NET требует платформу Windows для выполнения. Многие новые компании работают с серверами Linux, так как это бесплатная и свободно распространяемая среда.

Microsoft прекращает поддержку старых версий .NET платформы. Например, старые серверы Windows 2000 могут поддерживать только приложения .NET 2.0. Хотя обновление операционной системы кажется хорошим делом, многие организации предприятия сохраняют старые операционные системы из-за многих проблем, которые обновление может принести на платформу. Основные обновления серверной архитектуры должны быть протестированы и одобрены перед развертыванием, что увеличивает время и затраты на разработку.

КРИТЕРИИ ДЛЯ ВЫБОРА ЯЗЫКА

Для выбора наиболее подходящего языка для написания динамических страниц требуется выработать список критериев для выбора наиболее подходящего.

Рассмотрим их подробнее.

1. Удобство работы с текстом

Работа со строковым типом данных – одна из самых важных при написании динамических страниц. HTML – язык разметки и для формирования, по сути, текстового документа с использованием данной разметки, необходимо использовать такой тип данных как «строка», применять интерполяцию и регулярные выражения.

1. Удобство встроенных инструментов для работы с БД

При генерации динамической страницы может возникнуть потребность вывести пользователю информацию, хранящуюся в базе данных.

Для этого необходимо, чтобы язык программирования имел инструменты для работы с базой данной, для совершения таких действий как: подключение к базе данных, выполнение запроса и дальнейшая его обработка (к примеру, запись полученных данных в переменную).

1. Возможность работы с сетью на уровне представления (по модели OSI)

Для формирования новой веб-страницы нередко используются данные, пришедшие от клиентской страницы. Приходят они посредством POST- или GET-запроса с последнего уровня модели OSI, уровня представления. Чтобы страница смогла сформироваться на основе полученных данных, необходимо разобрать полученный запрос на отдельные компоненты.

1. Удобство предоставляемых языком средств для обработки ошибок

При работе сервера, скрипта нередко могут возникать ошибки различного рода. Необходимо, чтобы язык имел инструменты для обработки ошибок и возможность вывода их на экран пользователя.

1. Удобство работы с DOM

DOM (англ. Document Object Model, «объектная модель документа») — это независящий от платформы и языка программный интерфейс, позволяющий программам и скриптам получить доступ к содержимому HTML-, XHTML- и XML-документов, а также изменять содержимое, структуру и оформление таких документов. Модель DOM не накладывает ограничений на структуру документа. Любой документ известной структуры с помощью DOM может быть представлен в виде дерева узлов, каждый узел которого представляет собой элемент, атрибут, текстовый, графический или любой другой объект. Узлы связаны между собой отношениями «родительский-дочерний».

Это может потребоваться, если для скрипта, который формирует страницу, требуются значения или компоненты из html-страницы.

1. Удобство работы с файловой системой

Для задач, связанных с загрузкой или скачиванием файлов от языка программирования требуются инструменты для работы с файловой системой. Это включает в себя следующие возможности:

* запись файла;
* скачивание файла;
* установка прав доступа;
* работа с относительными и абсолютными путями;

ФОРМАЛИЗАЦИЯ КРИТЕРИЕВ. МЕТОД ЭКСПЕРТНОЙ ОЦЕНКИ. МЕТОД МНОГОКРИТЕРИАЛЬНОГО РАНЖИРОВАНИЯ КРИТЕРИЕВ

Формализация — представление какой-либо содержательной области (рассуждений, доказательств, процедур классификации информации и т.п.) в виде формальной системы. Формальная система — это знаковая модель, задающая множество объектов путем описания исходных объектов и правил построения новых.

В данной работе формализация критериев была произведена через экспертную оценку. Методы экспертных оценок являются частью обширной области теории принятия решений, а само экспертное оценивание — процедура получения оценки проблемы на основе мнения специалистов (экспертов) с целью последующего принятия решения (выбора).

В случаях чрезвычайной сложности проблемы, ее новизны, недостаточности имеющейся информации, невозможности математической формализации процесса решения приходится обращаться к рекомендациям компетентных специалистов, прекрасно знающих проблему, — к экспертам. Их решение задачи, аргументация, формирование количественных оценок, обработка последних формальными методами получили название метода экспертных оценок.

Существует две группы экспертных оценок: индивидуальные оценки основаны на использовании мнения отдельных экспертов, независимых друг от друга и коллективные оценки основаны на использовании коллективного мнения экспертов.

Из различных способов измерения объектов оценивания можно выделить как наиболее используемые: ранжирование, парное сравнение и непосредственная оценка. Для проводимого исследования был выбран последний, т.к. именно этот метод позволяет определить, на сколько один фактор более значим, чем другие. Диапазон изменения характеристик объекта разбивается на отдельные интервалы, каждому из которых приписывается определенная оценка (балл), например, от 0 до 10.

К критериям для отбора экспертов относятся:

1. Степень компетентности эксперта в решаемой проблеме. Для определения компетенции используются такие показатели как: уровень и профиль образования, профиль работы, стаж работы, занимаемая должность и др.
2. Уровень объективности и незаинтересованности эксперта при решении данной проблемы. (эксперт не должен быть лично заинтересован в принятии определенного решения).
3. Способность к командной работе (тогда, когда экспертный опрос проводится в группе и очно).

В качестве экспертов была взята группа из 10 человек, средний возраст группы 27 лет, средний опыт работы в web-индустрии 5 лет. Им была предложена анкета, в которой предлагалось оценить важность выработанных критериев, а также написать значение критерия для каждого из предлагаемых языков по шкале от 0 до 10. Число критериев для оценки 6, т.к. согласно данным психофизических исследований, человек уверенно различает не более 7 – 9 градаций на шкале некоторого признака. Если же она содержит большее число градаций, то соседние уровни начинают сливаться и уже не могут быть с уверенностью разграничены.

Пример анкеты приведен в таблице 1.

Для выбора наиболее предпочтительного языка программирования было принято решение ознакомиться алгоритмом многокритериального ранжирования альтернатив и создать его программную реализацию.

Предполагается, что пользователь может задавать количество альтернатив (от 2 до 10), количество критериев (2-10) и назначать коэффициенты важности критериям, определить минимальное и максимальное значение критериев. Результирующее сравнение производится после нормализации неоднородных оценок.

Таблица 1 – Вид анкеты эксперта

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Критерий | Значимость критерия  (от 0 до 10) | Оценка языка (от 0 до 10) | | | |
| Node.js | Python | PHP | Ruby |
| 1 | Удобство работы с текстом |  |  |  |  |  |
| 2 | Удобство встроенных инструментов для работы с БД |  |  |  |  |  |
| 3 | Возможность работы с сетью на уровне представления (по модели OSI) |  |  |  |  |  |
| 4 | Удобство предоставляемых языком средств для обработки ошибок |  |  |  |  |  |
| 5 | Удобство работы с DOM (Document Object Model) |  |  |  |  |  |
| 6 | Удобство работы с файловой системой |  |  |  |  |  |

**Ход работы:**

Для начала работы пользователем задаётся количество альтернатив и критериев (рисунок 1) при вводе пользователей значений не из множества [2 .. 10] выводится сообщение об ошибке (рисунок 2).

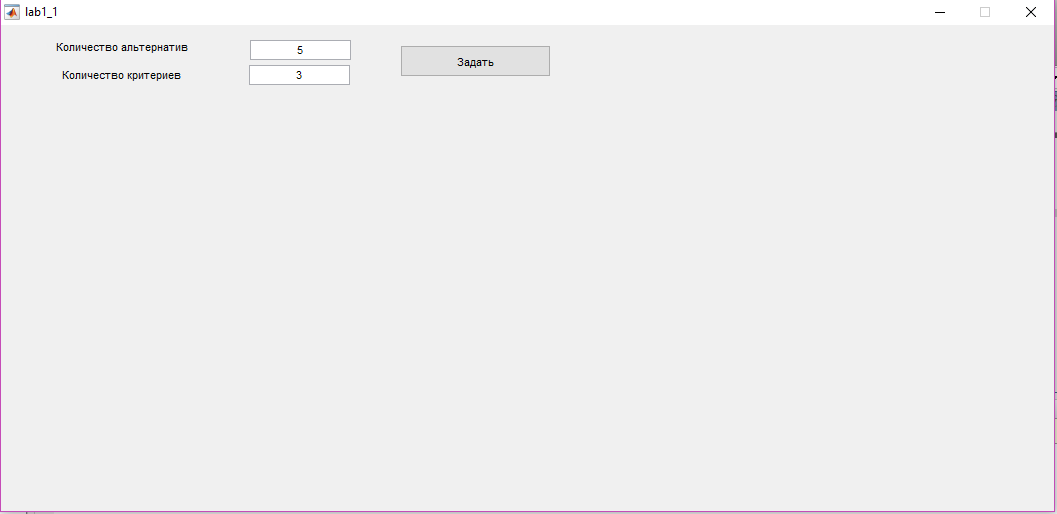


Рисунок 1 – Запрос количества альтернатив и критериев

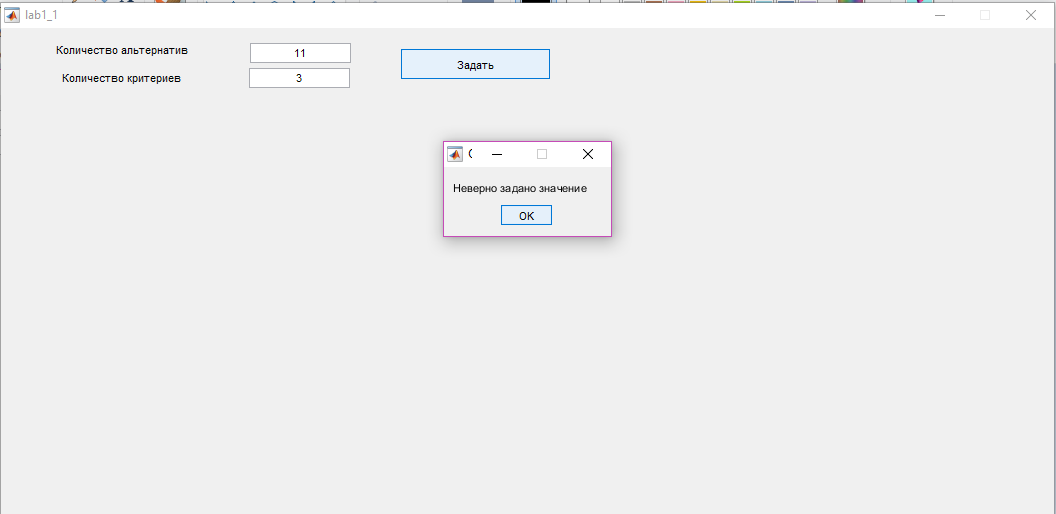


Рисунок 2 – Сообщение об ошибках при вводе количества альтернатив и критериев

На следующем этапе пользователь задаёт оценки для каждой альтернативы по каждому критерию, а также минимальное и максимальное значение оценок значения каждого критерия (рисунок 3). При ошибочном внесении данных появляются сообщения об ошибках (рисунок 4).

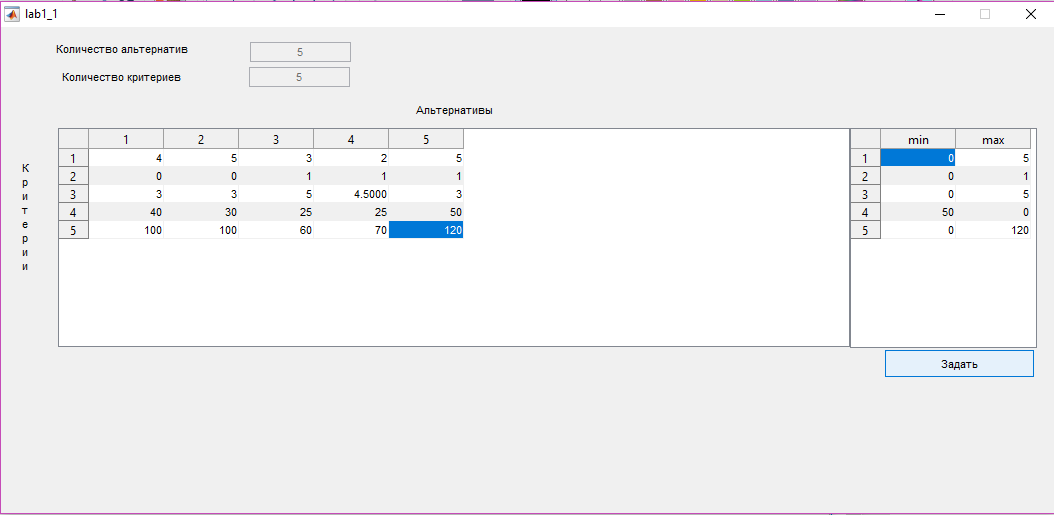
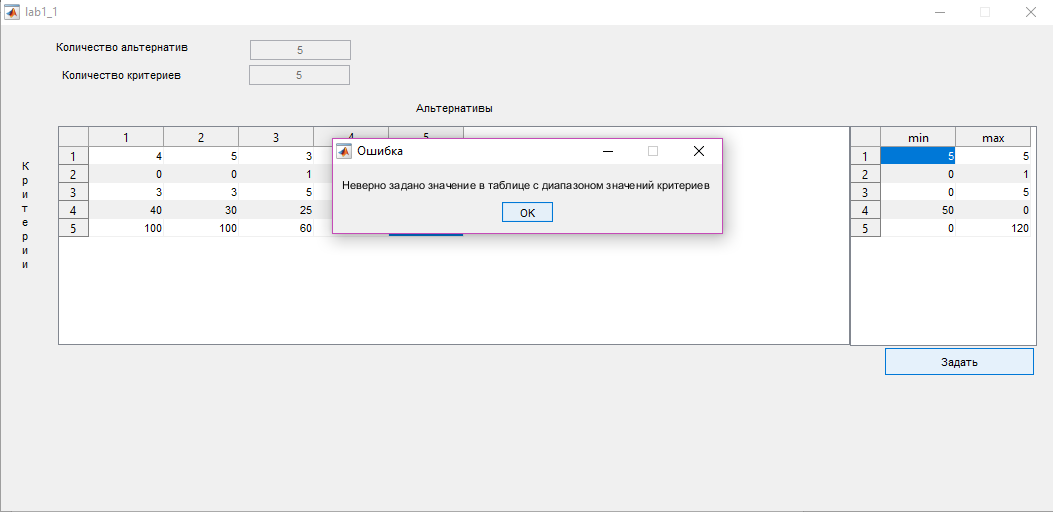


Рисунок 3 – Задание значений оценок критериев и альтернатив



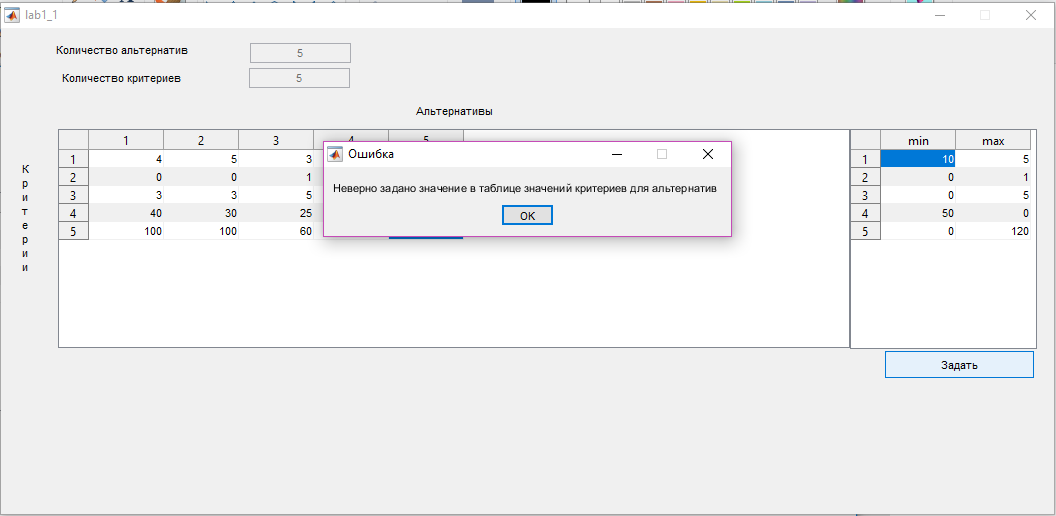


Рисунок 4 – Сообщения об ошибках при задании оценок критериев и альтернатив

Запрос значений коэффициентов важности для каждого критерия (рисунок 5). Сумма весов критериев должна равняться 1, если этого не происходит выводится сообщение об ошибке (рисунок 6).

В результате выполнения алгоритма многокритериального ранжирования альтернатив порядковый номер альтернативы выводится под номером самой альтернативы, неэффективные альтернативы имеют отрицательную нумерацию (рисунок 7).

В ходе проведенного исследования был получен список языков программирования:

1. C#
2. Perl
3. Python
4. PHP
5. Ruby
6. Node.js

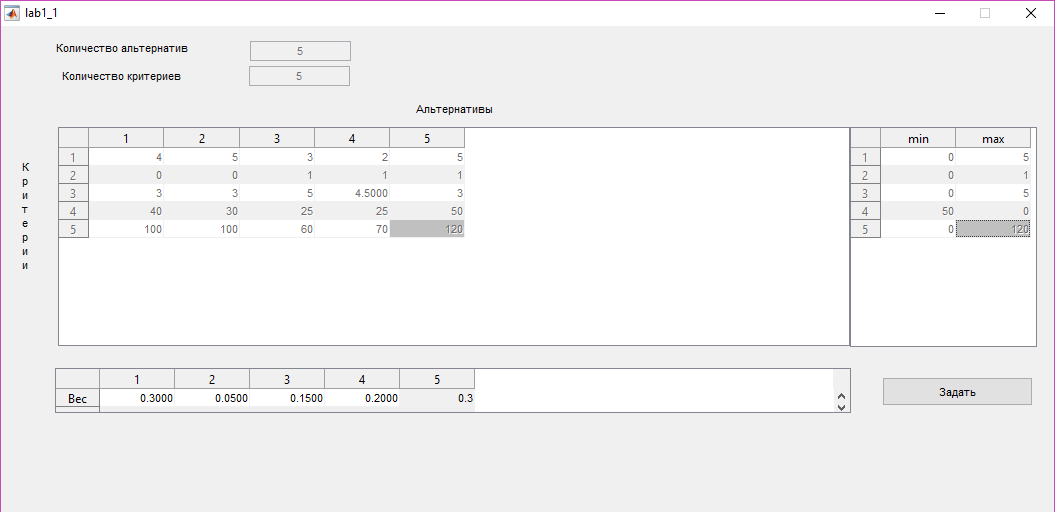


Рисунок 5 – Запрос значений коэффициента важности

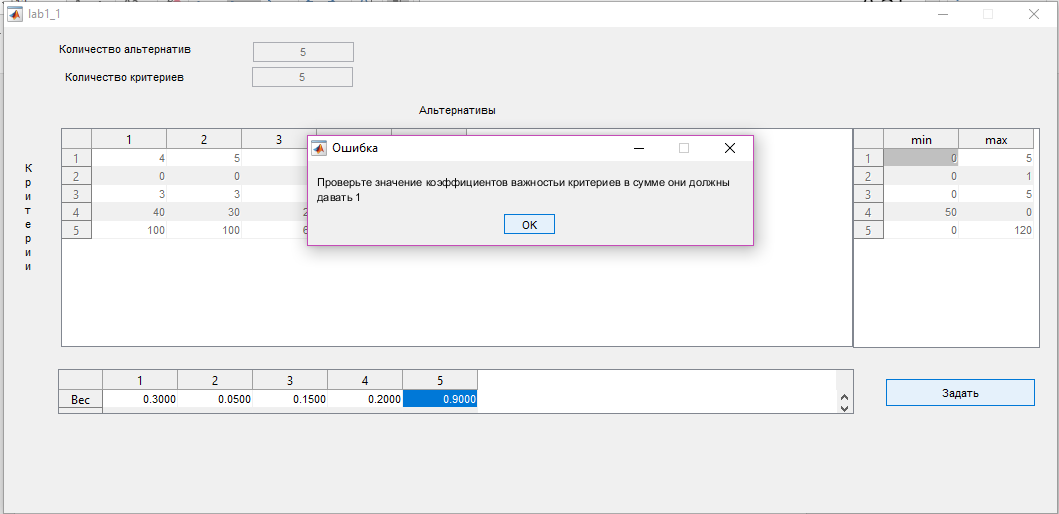


Рисунок 6 – Сообщение об ошибке при вводе коэффициентов важности критериев

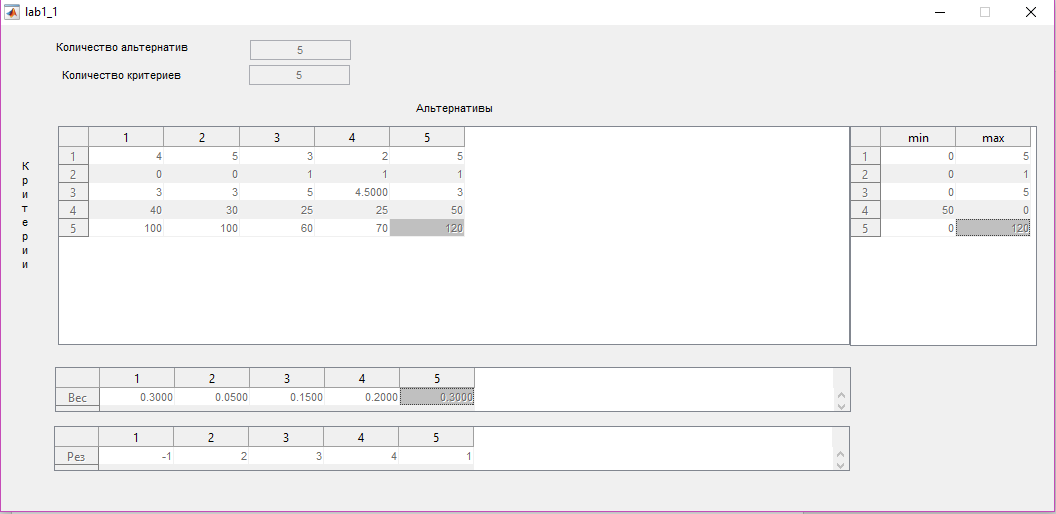


Рисунок 7 – Результаты выполнения алгоритма многокритериального ранжирования альтернатив

ТЕКСТ ПРОГРАММЫ РЕАЛИЗАЦИИ АЛГОРИТМА МНОГОКРИТЕРИАЛЬНОГО РАНЖИРОВАНИЯ АЛЬТЕРНАТИВ

function varargout = lab1\_1(varargin)

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @lab1\_1\_OpeningFcn, ...

'gui\_OutputFcn', @lab1\_1\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1}) gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout [varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else gui\_mainfcn(gui\_State, varargin{:});

end

function lab1\_1\_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

guidata(hObject, handles);

function varargout = lab1\_1\_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

function edit1\_Callback(hObject, eventdata, handles)

function edit1\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white');

end

function edit2\_Callback(hObject, eventdata, handles)

function edit2\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white');

end

%Задание количества критериев и альтернатив

function pushbutton1\_Callback(hObject, eventdata, handles)

a=str2num(get(handles.edit1,'String'));

k=str2num(get(handles.edit2,'String'));

if (isempty(a)|| a<2||a>10||isempty(k)|| k<2||k>10)

h=msgbox('Неверно задано значение','Ошибка','modal');

set(h,'WindowStyle','modal');

else

set(handles.edit1,'Enable', 'off')

set(handles.edit2,'Enable', 'off')

set(handles.pushbutton1,'Visible', 'off') % отображаем

str = cell(k,a); % создаем пустой массив данных

set(handles.uitable2,'Visible', 'on') % отображаем

set(handles.uitable2,'Data',str) % записываем в таблицу

str = cell(k,2); % создаем пустой массив данных

set(handles.uitable4,'Visible', 'on') % отображаем

set(handles.uitable4,'Data',str) % записываем в таблицу

str = cell(1,k);

set(handles.uitable6,'Data',str) % записываем в таблицу

str = cell(1,k);

set(handles.uitable7,'Data',str) % записываем в таблицу

set(handles.text4,'Visible', 'on') % отображаем

set(handles.text5,'Visible', 'on') % отображаем

set(handles.pushbutton2,'Visible', 'on') % отображаем

end

% Проверка правильности введённых значений оценок альтернатив и мин и мах значений оценок критериев

% проведение нормализации.

function pushbutton2\_Callback(hObject, eventdata, handles)

a=str2num(get(handles.edit1,'String'));

k=str2num(get(handles.edit2,'String'));

p = get(handles.uitable2, 'Data'); % считываем из таблицы

p = cell2mat(p); % преобразуем в ячейку в матрицу

minmax = get(handles.uitable4, 'Data'); % считываем из таблицы

minmax = cell2mat(minmax); % преобразуем в ячейку в матрицу

err=0;

for i=1:k

if( isempty(minmax(i,1))||isempty(minmax(i,1))||minmax(i,1)==minmax(i,2)||minmax(i,1)<0||minmax(i,2)<0)

err=1;

i=k;

h=msgbox('Неверно задано значение в таблице с диапазоном значений критериев','Ошибка','modal');

set(h,'WindowStyle','modal');

else

if (i==1) delt=[minmax(i,2)-minmax(i,1)];

else delt=[delt minmax(i,2)-minmax(i,1)];

end;

if (delt(i)>0) min=minmax(i,1); max=minmax(i,2);

else min=minmax(i,2); max=minmax(i,1);

end;

end;

for j=1:a

if( isempty(p(i,j))||(p(i,j)<min||p(i,j)>max))

err=1;

j=a;

i=k;

h=msgbox('Неверно задано значение в таблице значений критериев для альтернатив','Ошибка','modal');

set(h,'WindowStyle','modal');

end;

end;

end;

if (err==0)

set(handles.uitable2,'Enable', 'off')

set(handles.uitable4,'Enable', 'off')

set(handles.pushbutton2,'Visible', 'off') % отображаем

for i=1:k %Для всех критериев

for j=1:a% перебираем для каждой альтернативы

min=10;

for l=1:a% для перебора между альтернативами

p1=(p(i,l)-p(i,j))/delt(i);

if p1<0 p1=1; else p1=1-p1;

end;

if min>p1 min=p1;

end;

end;

if j==1 p\_norm1=[min]; else p\_norm1=[p\_norm1 min];

end;

end;

if i==1 p\_norm=[p\_norm1']; else p\_norm=[p\_norm, p\_norm1'];

end;

end;

file1 = fopen('help.xls','w');

if (file1==-1)

h=msgbox('Файл help.xls не открывается. Проверьте наличие файла.','Ошибка открытия файла');

set(h, 'WindowStyle','modal');

return;

end;

for i=1:a

for j=1:a fprintf(file1,'%f\n',p\_norm(i,j));

end

end

if (fclose(file1)==-1)

h=msgbox('Файл help.xls не удалось закрыть. ','Ошибка закрытия файла');

set(h, WindowStyle','modal');

end

set(handles.uitable6,'Visible', 'on')

set(handles.pushbutton3,'Visible', 'on') % отображаем

end;

%Считываем значение коэффициентов важности и выполняем алгоритм сравнения

function pushbutton3\_Callback(hObject, eventdata, handles)

a=str2num(get(handles.edit1,'String'));

k=str2num(get(handles.edit2,'String'));

formatSpec = '%f\t';

file1=fopen('help.xls','r');

if (file1==-1)

h=msgbox('Файл help.xls не открывается. Проверьте наличие файла.','Ошибка открытия файла');

set(h, 'WindowStyle','modal');

return;

end;

sizeA = [a a];

p\_norm = fscanf(file1,formatSpec,sizeA);

if (fclose(file1)==-1)

h=msgbox('Файл help.xls не удалось закрыть. ','Ошибка закрытия файла');

set(h, WindowStyle','modal');

end;

w = get(handles.uitable6, 'Data'); % считываем из таблицы

w = cell2mat(w); % преобразуем в ячейку в матрицу

wes=0;

for i=1:k

if( isempty(w(i))||(w(i)<=0||w(i)>=1))

h=msgbox('Неверно задано значение в таблице коэффициентов важности критериев','Ошибка','modal');

set(h,'WindowStyle','modal');

i=k;

wes=100;

else wes=wes+w(i);

end;

end;

if (wes~=1)

h=msgbox('Проверьте значение коэффициентов важностьи критериев в сумме они должны давать 1','Ошибка','modal');

set(h,'WindowStyle','modal');

else

set(handles.uitable6,'Enable', 'on')

set(handles.pushbutton3,'Visible', 'on')

N0=-1;

C = N0\*eye(a);

N2=1000;

N3=500;

p\_norm=p\_norm';

for i=1:a

for j=(i+1):a

min=0;

max=0;

ravn=0;

for m=1:k

if p\_norm(i,m)>p\_norm(j,m) max=max+1; elseif p\_norm(i,m)<p\_norm(j,m) min=min+1; else ravn=ravn+1;

end;

end;

if (max==k) C(i,j)=N2; C(j,i)=0;

elseif min==k C(i,j)=0; C(j,i)=N2;

elseif ravn==k C(i,j)=1; C(j,i)=1;

elseif max==0 C(i,j)=0; C(j,i)=N3;

elseif min==0 C(i,j)=N3; C(j,i)=0;

else min=0; max=0;

for m=1:k

if (p\_norm(i,m)>p\_norm(j,m)) max=max+w(m)\*p\_norm(i,m);

elseif (p\_norm(i,m)<p\_norm(j,m)) min=min+w(m)\*p\_norm(j,m);

end;

end;

C(i,j)=max/min;

C(j,i)=1/C(i,j);

end;

end;

end;

res=zeros(1,a);

HeMeC=zeros(3,a);

a1=a;

t\_min=-1;

t\_max=1;

while (a1>1)

for i=1:a1

for j=1:a1

if C(j,i)>1 HeMeC(1,i)=HeMeC(1,i)+1; elseif C(j,i)<1 HeMeC(2,i)=HeMeC(2,i)+1;

end;

if ( HeMeC(3,i)<C(j,i)) HeMeC(3,i)=C(j,i);

end;

end;

end;

min=0;%%вторая строка

max=0;%%первая строка

ok1=0;

t=0;

tt=0;

for i=1:a1

if HeMeC(3,i)==N2 ok1=ok1+1; t=i; end;

end;

if ok1==1

tt=t;

while (t) if (res(1,ok1)==0) t=t-1; end; ok1=ok1+1;

end;

t=ok1-1;

res(1,t)=t\_min;

t\_min=t\_min-1;

a1=a1-1;

elseif ok1>1

ok1=0;

for i=1:a1

if HeMeC(3,i)==N2

if (min>HeMeC(1,i)); max=HeMeC(1,i); min=HeMeC(2,i); ok1=i;

elseif (max<HeMeC(1,i)); max=HeMeC(1,i); min=HeMeC(2,i); ok1=i;

end;

end;

end;

t=1;

tt=ok1;

while (ok1) if (res(1,t)==0) ok1=ok1-1; end; t=t+1;

end;

t=t-1;

res(1,t)=t\_min;

t\_min=t\_min-1;

a1=a1-1;

else

ok1=0;

t=0;

for i=1:a1 if HeMeC(3,i)==N3 ok1=ok1+1; t=i; end;

end;

if ok1==1

tt=t;

while (t) if (res(1,ok1)==0) t=t-1; end; ok1=ok1+1;

end;

t=ok1-1;

res(1,t)=t\_min;

t\_min=t\_min-1;

a1=a1-1;

elseif ok1>1

ok1=0;

for i=1:a1

if HeMeC(3,i)==N3

if (min>HeMeC(1,i)); max=HeMeC(1,i); min=HeMeC(2,i); ok1=i;

elseif (max<HeMeC(1,i)); max=HeMeC(1,i); min=HeMeC(2,i); ok1=i;

end;

end;

end;

t=1;

tt=ok1;

while (ok1) if (res(1,t)==0) ok1=ok1-1; end; t=t+1;

end;

t=t-1;

res(1,t)=t\_min;

t\_min=t\_min-1;

a1=a1-1;

else

min=11;%%первая строка

max=-1;%%вторая строка

t=0;

for i=1:a1

if HeMeC(2,i)>max max=HeMeC(2,i); min=HeMeC(1,i); t=i;

elseif HeMeC(2,i)==max

if HeMeC(1,i)<min max=HeMeC(2,i); min=HeMeC(1,i); t=i;

elseif HeMeC(1,i)==min

if HeMeC(3,i)<HeMeC(3,t) max=HeMeC(2,i); min=HeMeC(1,i); t=i;

end;

end;

end;

end;

ok1=1;

tt=t;

while (t) if (res(1,ok1)==0) t=t-1; end; ok1=ok1+1;

end;

t=ok1-1;

res(1,t)=t\_max;

t\_max=t\_max+1;

a1=a1-1;

end;

C(:,tt)=[];

C(tt,:)=[];

for i=1:a1+1 HeMeC(1,i)=0; HeMeC(2,i)=0; HeMeC(3,i)=0;

end;

HeMeC(:,tt)=[];

end;

end;

for i=1:k if(res(1,i)==0) res(1,i)=t\_max; end;

end;

str = cell(1,k);

set(handles.uitable7,'Data',str) % записываем в таблицу

set(handles.uitable7,'data',res)

set(handles.uitable7,'Visible', 'on')

set(handles.uitable6,'Enable', 'off')

set(handles.uitable7,'Enable', 'off')

set(handles.pushbutton3,'Visible', 'off')

end;

ЗАКЛЮЧЕНИЕ

В ходе работы было введено понятие динамической веб-страницы, web-страницы, были рассмотрены типы web-страниц, были выдвинуты критерии для выбора языка программирования, а также был выбран метод их формализации.

В ходе работы были рассмотрены наиболее используемые языки программирования для написания динамических веб-страниц. Также был получен ранжированный список языков программирования с помощью алгоритма многокритериального ранжирования альтернатив.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ford S., Wells D., Wells N. Web Programming Languages (1 September 1997) [электронный ресурс], URL - http://www.objs.com/survey/lang.htm (дата обращения 10.10.2017)
2. Эд Титтел, Джефф Ноубл. HTML, XHTML и CSS для чайников, 7-е издание = HTML, XHTML & CSS For Dummies, 7th Edition. — М.: «Диалектика», 2011. — 400 с.
3. Стивен Шафер. HTML, XHTML и CSS. Библия пользователя, 5-е издание = HTML, XHTML, and CSS Bible, 5th Edition. — М.: «Диалектика», 2010. — 656 с.
4. TC39. 4. Overview // ECMAScript Language Specification. — 5th ed. — 2009. — P. 1, 2. http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf
5. Флэнаган Д. JavaScript. Карманный справочник. Сделайте веб-страницы интерактивными! / Перевод А.Г. Сысонюк. — Москва.: Издательский дом "Вильямс", 2015. — С. 320
6. Джейсон Ленгсторф. PHP и jQuery для профессионалов = Pro PHP and jQuery. — М.: «Вильямс», 2010. — С. 352
7. Кузнецов Максим, Симдянов Игорь. PHP на примерах. — 2-е изд. перераб. и доп. — СПб.: «БХВ-Петербург», 2011. — С. 400.
8. Рэндал Л. Шварц, Том Феникс, Брайан Д. Фой. Изучаем Perl. — М.: O’Reilly, «Символ», 2009. — С. 377
9. Кристиансен, Торкингтон. Perl. Сборник рецептов. Для профессионалов = Perl Cookbook. — М.: O’Reilly, «Питер», 2004. — 928 с
10. Итан Браун. Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript = Web Development with Node and Express / Итан Браун. — Санкт-Петербург: Питер, 2017. — 336 с.
11. Martin Brampton. PHP5 CMS Framework Development. — PACKT publishing, June 2008. — 328 p.
12. Github Language Stats [Электронный ресурс] – URL: https://madnight.github.io/githut/#/pull\_requests/2017/4 (дата обращения 06.03.2018)
13. Гуцыкова С.В. Метод экспертных оценок. Теория и практика: учебное пособие/ Гуцыкова С.В.— М.: Институт психологии РАН, 2011 — 144 c.
14. Методы экспертных оценок / Хабрахабр [Электронный ресурс] – URL: https://habrahabr.ru/post/189626/ (дата обращения 10.03.2018)
15. Миллер Дж. Магическое число семь плюс или минус два. О некоторых пределах нашей способности перерабатывать информацию / Дж. Миллер // Инженерная психология. – М : Прогресс, 1964. – С. 192-225
16. З.Г. Джабраилова, С.Р. Нобари. Метод многокритериального ранжирования для решения задач управления персоналом / З.Г. Джабраилова, С.Р. Нобари // Штучний інтелект. — 2009. — № 4. — С. 130-137