Торгашин А.А. К Вопросу о кроссплатформенной системе автоматизированного расчета показателя репродуктивной готовности// Международный журнал информационных технологий и энергоэффективности. – 2023. –
Т. 8 № 5(31)ч.1 с. 27–35

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.622

К ВОПРОСУ О КРОССПЛАТФОРМЕННОЙ СИСТЕМЕ АВТОМАТИЗИРОВАННОГО РАСЧЕТА ПОКАЗАТЕЛЯ РЕПРОДУКТИВНОЙ ГОТОВНОСТИ

Торгашин А.А.
ФГБОУ ВО "Сибирский государственный университет телекоммуникаций и информатики", Новосибирск, Россия (630102, Новосибирская область, Новосибирск, ул. Кирова, д. 86) e-mail: torgashin.aa@yandex.ru

В данной статье рассматривается кроссплатформенная (то есть, имеющая возможность работы на различных видах операционных систем) система, предназначенная для расчета показателя репродуктивной готовности, позволяющего оценивать уровень репродуктивной функции пациентов. В качестве инструментария для такой системы выступает набор технологий, включающий в себя язык программирования, платформу разработки, систему хранения данных, а также иные инструменты для разработки системы. Для обоснованного выбора основного языка программирования, на котором реализуется система, применен метод анализа иерархий, основанный на экспертных оценках критериев языков программирования. В результате проведенного анализа осуществлен выбор языка программирования C#, платформы .NET и фреймворка ASP.NET Core. Для хранения массива данных осуществлен выбор в пользу реляционной базы данных, а именно системы управления реляционными базами данных Microsoft SQL Server с применением технологии ORM (Object-Relational Mapping), позволяющей работать с объектами базы данных как с объектами программного кода. В качестве основного веб-сервера, отвечающим за обработку входящих запросов и возврат ответов, выбран веб-сервер Kestrel. Учитывая недостатки данного веб-сервера, дополнительно выбран веб-сервер Nginx, используемый как обратный прокси-сервер, который перенаправляет запросы в Kestrel и предоставляет дополнительные функции, такие как балансировка нагрузки и кэширование. Такая конфигурация предоставляет повышение производительности и надежность веб-приложений, в частности в сценариях с высоким трафиком.

Ключевые слова: Прогностическая модель для диагностики и лечения бесплодия, программное обеспечение, .NET, ASP.NET Core, кроссплатформенность, веб-приложение, веб-сервер, клиент-серверная архитектура, Kestrel.

ON THE ISSUE OF A CROSS-PLATFORM SYSTEM FOR AUTOMATED CALCULATION OF THE INDICATOR OF REPRODUCTIVE READINESS

Torgashin A.A.
Siberian State University of Telecommunications and Informatics, Novosibirsk, Russia (630102, Novosibirsk region, Novosibirsk, Kirova St., 86) e-mail: torgashin. aa@yandex. ru

This article discusses a cross-platform (that is, having the ability to work on various types of operating systems) system designed to calculate the indicator of reproductive readiness, which allows assessing the level of reproductive function of patients. A set of technologies acts as a toolkit for such a system, including a programming language, a development platform, a data storage system, and other tools for system development. For a reasonable choice of the main programming language in which the system is implemented, the method of analysis of hierarchies based on expert assessments of the criteria of programming languages is applied. As a

result of the analysis, the choice of the C# programming language, the .NET platform and the ASP.NET Core framework was made. To store the data array, a choice was made in favor of a relational database, namely, the Microsoft SQL Server relational database management system using ORM (Object-Relational Mapping) technology, which allows working with database objects as with program code objects. The Kestrel web server has been chosen as the main web server responsible for processing incoming requests and returning responses. Given the shortcomings of this web server, the Nginx web server was additionally chosen, used as a reverse proxy server, which redirects requests to Kestrel and provides additional features such as load balancing and caching. This configuration provides improved performance and reliability for web applications, particularly in high-traffic scenarios.

Keywords: predictive model for diagnosis and treatment of infertility, software, .NET, ASP.NET Core, cross-platform, web application, web server, client-server architecture, Kestrel.

Введение

В статье [1] описана прогностическая модель для диагностики и лечения бесплодия. В основе модели лежит показатель репродуктивной готовности (Indicator of Reproductive Readiness, IRR), рассчитываемый исходя из коэффициента репродуктивной активности (Coefficient of Reproductive Activity, CRA) и показателя репродуктивного здоровья (Indicator of Reproductive Health, IRH), которые в свою очередь включают в себя различные характеристики здоровья и половой активности человека (пациента). Расчет указанных показателя и коэффициента в конечном итоге дают возможность оценить уровень репродуктивной функции пациента.

Предполагается, что применение данной модели оценивания репродуктивного здоровья в первую очередь полезно для выбора направления диагностики, лечения и прогнозирования репродуктивных возможностей. Вычисление IRR может проводить как сам пациент, отвечая на вопросы специализированного программного обеспечения (ПО), так и гинеколог-репродуктолог, ведущий прием пациентов.

Учитывая многоступенчатость расчетов и наличие множества принимаемых во внимание факторов, самостоятельное проведение вычислений (например, врачом) может быть затруднено и не всегда эффективно по затратам времени. Поэтому автоматизация данного процесса является важной практической задачей.

На рынке медицинского программного обеспечения (ПО) представлено множество программных продуктов для различных направлений медицины. Такое ПО включается в себя, например, системы для поддержки принятия врачебных решений, системы для анализа медицинских изображений и прочие [9]. Данная работа направлена на разработку ПО, реализующего модель IRR, описанную в статье [1].

Такое ПО должно отвечать следующим требованиям:

- иметь централизованное (в пределах одного медицинского учреждения или сети медицинских учреждений) хранение данных, которое позволит провести интеграцию между несколькими медучреждениями, либо между пациентом и врачом, в случае самостоятельного использования ПО пациентом;
- наличие кроссплатформенности, выражающейся в возможности работы с различными операционными системами (ОС), например, Windows Server, ОС на базе ядра Linux и др.;
- наличие интерфейса, понятного как работникам медицинских учреждений, так и физическим лицам;

- наличие низкого уровня требований к аппаратным мощностям с целью расширения охвата возможных пользователей.

В настоящей статье представлен анализ выбора соответствующих инструментов и подходов для разработки такого ПО.

Архитектура программного обеспечения

Как указано выше, ПО должно иметь централизованное хранилище (базу данных). Такое требование вызвано потребностью в мобильности. Под этим подразумевается наличие возможности обмена данными, например, между отделениями в пределах одного медицинского учреждения, между несколькими медицинскими учреждениями, а также между пациентом, который может заранее провести автоматизированный расчет соответствующих показателей, и медицинским учреждением, в которое пациент может отправить свои данные.

Исходя из вышеуказанных требований, а также возможности применения ПО пациентами самостоятельно, в приоритете рассматривается клиент-серверная архитектура ПО. Такая архитектура позволяет развернуть ПО на сервере, который будет отвечать за хранение данных, обмен данными между медицинскими учреждениями или между пациентом и учреждением, а также за проведение расчетов. Помимо сервера, в клиент-серверной архитектуре присутствует «клиент», который отвечает за прием и отправку данных с сервера и на сервер, отображение информации пользователям системы, а также получение данных от пользователей.

Клиент может быть распространяемым в виде стационарного приложения, устанавливаемого на компьютеры пользователей. В настоящее время подход с установкой ПО на компьютер может быть реализован не для каждого программного продукта, а в некоторых случаях бывает излишним (например, в случаях, когда ПО используется однократно). В данной ситуации, клиентская часть выполняет лишь функцию приема и отображения данных, а применение ПО непосредственно пациентом будет единоразовым, в связи с чем установка его на компьютеры пользователей не имеет особой необходимости. В свою очередь реализация «клиента» как веб-сайта в сети интернет является наиболее выгодным решением с точки зрения простоты применения ПО как пациентами, так и медицинскими учреждениями. Фактически, пациентам будет доступна веб-версия ПО с любого устройства, будь то смартфон или персональный компьютер, а в медицинских учреждениях потребуется меньшее количество ресурсов на распространение такого программного продукта во внутренней сети.

Выбор инструментов для реализации программного обеспечения

Язык программирования и платформа

Основой информационных систем (ИС) является язык программирования (ЯП), на котором ведется разработка программного кода. В настоящее время представлено множество ЯП, предназначенных для разработки любых ИС, среди которых выделяют наиболее популярные: Java, C, Python, C++, C# и другие [2]. Каждый из этих языков зачастую применяются для решения определенных задач, хотя и не ограничиваются только ими, например, C и C++ часто используются для тех задач, где критичной является скорость исполнения, но в то же время жертвуется удобством работы с ЯП. Наоборот, такие ЯП, как

Java и C#, имеют более широкий уровень абстракции, что облегчает работу с ЯП, но производительность этих языков - несколько ниже [4].

Для решения задачи выбора ЯП в нашей работе использовали метод анализа иерархий (МАИ) [10]. Общий принцип применения МАИ представлен на Рисунке 1.

В данной статье отражены только основные этапы применения МАИ, которые в наибольшей степени отражают суть применяемого метода.

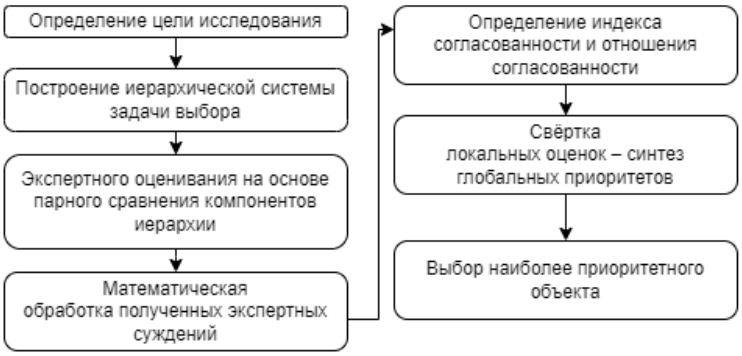

Рисунок 1 – Общий принцип применения МАИ

Путем попарного сравнения критериев выбора ЯП между собой была получена матрица парных сравнений и нормированные оценки сравниваемых критериев, которые рассчитываются по следующей формуле.

$$X = \frac{\sqrt[n]{a_1 * a_2 * \ldots * a_n}}{\sum_{i=1}^{n} (\sqrt[n]{a_1 * a_2 * \ldots * a_n})_i}, \quad (1)$$

где: X – нормированная оценка;

n – размерность матрицы;

a_n – значимость критерия.

Математической обработкой с применением формулы (1) были получены следующие нормированные оценки значимости критериев ЯП, представленные на Рисунке 2.

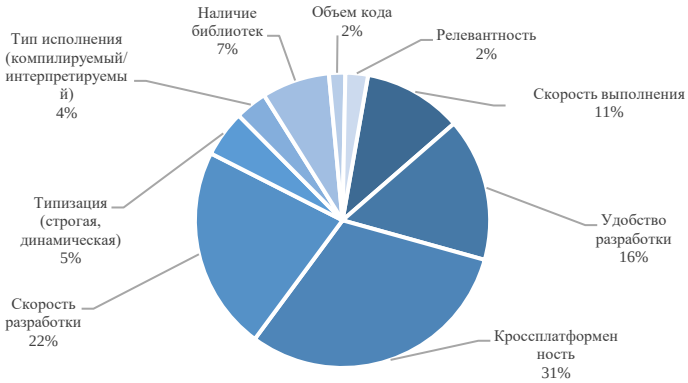

Рисунок 2 – Нормированные оценки значимости критериев ЯП

Из представленного набора критериев были отобраны четыре наиболее важных критерия, оценки значимости которых имели значение более 10% (определенный нами минимум для выбора критериев): поддержка кроссплатформенности (30,81%), скорость разработки (22,35%), удобство разработки (15,7%), скорость выполнения ПО, разработанного

на данном ЯП (10,84%). Также было проведено попарное сравнение четырех выбранных критерием между собой для актуализации значимости критериев. Были получены следующие оценки значимости (веса) критериев: поддержка кроссплатформенности – 58,11%, скорость разработки – 25,49%, удобство разработки – 11,4%, скорость выполнения – 5%.

Аналогичным образом из набора восьми наиболее популярных ЯП общего назначения (Python, C, C++, Java, C#, Visual Basic, JavaScript, PHP) были отобраны четыре языка программирования: C#, Java, C++ и Python [2].

С целью подтверждения согласованности выставленных оценок при проведении попарного сравнения были рассчитаны индекс согласованности (ИС) и отношение согласованности (ОС), которые составили 5,19% и 3,58% соответственно. Если значение ИС и ОС < 10%, то оценки считаются согласованными [10].

Для выбора единственного ЯП было проведено оценивание их между собой по каждому из выбранных критериев. В результате такого оценивания получена сводная таблица значимости ЯП по каждому из критериев (Таблица 1).

Таблица 1 – Результаты анализа критериев ЯП

ЯП	**Критерии**			
	Скорость исполнения (вес критерия – 0,05)	**Удобство разработки (вес критерия – 0,11)**	**Кроссплатформенность (вес критерия – 0,58)**	**Скорость разработки (вес критерия – 0,25)**
C#	0,17	0,55	0,41	0,53
Java	0,11	0,29	0,41	0,27
C++	0,68	0,05	0,07	0,06
Python	0,04	0,11	0,12	0,14

Как можно увидеть из Таблицы 1 наиболее важным критерием был установлен критерий кроссплатформенности, что соответствует требованиям к разрабатываемому ПО, указанным во введении данной статьи.

Для нахождения глобального показателя значимости, т.е. выбора одного объекта из рассматриваемого множества (в данном случае ЯП) применяется синтез глобальных приоритетов, который производится с помощью расчета взвешенных сумм по следующей формуле:

$$A = \sum_{i=1}^{n} X_i * V_i, \quad (2)$$

где: n – количество критериев для объекта;

Xi – значимость критерия;

Vi – вес критерия.

С помощью формулы (2) были рассчитаны глобальные приоритеты языков программирования по выбранным критериям, которые составили следующие значения: C# - 0,44, Java – 0,34, Python – 0,12, C++ - 0,09.

Таким образом, выбор был сделан в пользу языка программирования C# (C Sharp), который получил наибольший глобальный приоритет, и соответствующей ему платформы .NET, которая является кроссплатформенной, то есть может исполняться в различных средах (в данном случае речь идет об операционных системах) в отличии от платформы .NET

Framework, которая предназначена исключительно для ОС Windows. Поддерживается данная платформа и язык корпорацией Microsoft, обновляется на постоянной основе - последняя версия .NET 7 обновлена в ноябре 2022 года, данные факты позволяют гарантировать актуальность выбранного языка и платформы в обозримом будущем. Принимая во внимание указанные факты: удобство и скорость работы с языком, наличие множества библиотек под различные потребности данного языка, будем считать его одним из наиболее подходящих языков для реализации соответствующего ПО.

Вместе с тем, для разработки веб-приложений в платформе .NET предусмотрен фреймворк (набор инструментов для разработки ПО) ASP.NET Core. Разработка на данном фреймворке также ведется на языке C#. Следует отметить, что и C#, и .NET, и ASP.NET Core представлены с открытым исходным кодом, что позволяет убедиться в их безопасности, отсутствии уязвимостей и иных нежелательных факторов.

Помимо ЯП для ядра системы, в роли вспомогательного ЯП был выбран язык TypeScript – относительно «молодой» ЯП (представлен в 2012 году корпорацией Microsoft), который при сборке приложения транслируется в JavaScript. Используется данный язык для управления элементами веб-страниц (например, кнопками, полями для ввода, внешним видом и т.д.), которые сами по себе являются представлением на языке разметки HTML.

Фреймворк ASP.NET Core позволяет подготовить как серверную часть приложения, так и клиентскую. Сервер в данном случае представляет собой приложение, написанное на языке программирования C#, которое принимает запросы от клиентских приложений, обрабатывает определенным образом и возвращает ответ в виде веб-страницы с определенным результатом. Клиентское приложение в данном случае — это веб-браузер, с помощью которого пользователи отправляют на сервер запросы и получают ответы, задача браузера состоит в правильном формировании и отправке таких запросов, а также браузером осуществляется отображение ответов в виде веб-страниц.

Хранение данных

Практически каждое ПО должно где-то хранить данные – это могут быть данные пользователей (такие как логин, ФИО, и другая персональная информация), данные самого ПО (например, настройки) и прочие данные, необходимые для работы ПО.

С целью хранения данных применяются различные технологии, например, хранение в базах данных, которые создаются и управляются с помощью различных систем управления базами данных (СУБД), среди которых выделяют наиболее популярные СУБД, такие как MSSQL, PostgreSQL, MySQL и др. При определенных обстоятельствах хранение данных также организуют в виде файлов в различных форматах: JSON, XML, CSV и др.

Хранение большого объема данных сопряжено с требованием в подходящем способе организации и доступа к таким данным. Наилучшим вариантом для этих целей подходит СУБД, в которой данные организованы в виде таблиц с обозначенными полями, в том числе идентификаторами записей, что позволяет оперировать такими данными с помощью фильтрации по определенным признакам.

Для реализации ПО была выбрана наиболее популярная система управлениях базами данных от корпорации Microsoft – Microsoft SQL Server (MSSQL), которая легко интегрируется с платформой .NET в целом и языком C# в частности (во многом благодаря

тому, что Microsoft SQL Server и .NET поддерживаются корпорацией Microsoft). Помимо этого, MSSQL как комплекс программ включает в себя удобную с точки зрения пользовательского интерфейса утилиту для взаимодействия с компонентами MSSQL, что ускоряет работу с базой данных и позволяет затрачивать меньшее количество ресурсов в процессе разработки.

Следует отметить, что для реализации системы также использована технология ORM (Object-Relational Mapping), в частности Entity Framework. Данная технология используется для сопоставления объектов базы данных с объектами программного кода [3]. Использование данного подхода позволяет уменьшить количество ресурсов, требуемых на разработку модуля ПО, отвечающего за работу с объектами в базе данных, а также улучшает масштабируемость программного кода в дальнейшем в связи с тем, что работа по взаимодействию с БД происходит с помощью LINQ (язык запросов в ЯП платформы .NET) и моделей данных, а не SQL с дальнейшим приведением полученных данных к типу моделей (классов, определенных в программном коде).

Web-сервер для системы

Веб-приложению для работы требуется веб-сервер – ПО, которое принимает HTTP-запросы (например, от веб-браузеров) и возвращает HTTP-ответы (зачастую вместе со страницей, построенной с использованием HTML и некоторым набором статических файлов, таких как изображения, файлы, видео и др.).

Для веб-приложений, размещаемых на серверах с ОС Windows Server, обычно используется IIS (Internet Information Server) – веб-сервер, разработанный корпорацией Microsoft и поставляемый совместно с этой ОС [5]. Учитывая тот факт, что IIS отлично подходит для размещения веб-приложений, разработанных с применением ASP.NET Core (ввиду его интегрированности с продуктами корпорации Microsoft), у данного ПО имеется существенный минус, выражающийся в том, что IIS работает только на ОС Windows Server, что не соответствует требованиям наличия кроссплатформенности, указанным выше в данной статье.

Наиболее подходящим веб-сервером в таком случае является Kestrel – веб-сервер, также разработанный корпорацией Microsoft и интегрированный в ASP.NET Core. Факт интегрированности способствует уменьшению затрат времени на развертывание системы, так как в случае использования ASP.NET Core веб-сервер уже встроен в фреймворк. Kestrel в отличии от IIS является кроссплатформенным, то есть может размещаться не только на ОС Windows Server, но также и на ОС на базе ядра Linux, MacOS и др.

Несмотря на преимущества в виде кроссплатформенности, Kestrel является простым веб-сервером и, например, не поддерживает обращение к одним и тем же IP-адресам и портам от нескольких процессов, в связи с чем рекомендуется использовать Kestrel совместно с обратным прокси-сервером, который лишен этих ограничений [6, 7]. Для реализации системы в качестве обратного прокси-сервера был использован наиболее популярный веб-сервер Nginx [8].

Заключение

В результате проведенного анализа был определен инструментарий, необходимый для реализации системы, отвечающей требованиям кроссплатформенности, что позволяет развернуть такую систему на современных ОС, при этом такая система имеет централизованное хранилище на основе СУБД, что позволяет развернуть базу данных на сервере с возможностью множественного подключения с различных устройств. Клиент-серверная архитектура, примененная при разработке, позволяет распространить ПО на большее количество пользователей ввиду простоты распространения такого ПО и поддержке работы на различных устройствах (компьютеры и смартфоны).

Благодаря использованию современных решений и технологий разработка системы требует меньшее количество ресурсов, что в перспективе удешевляет разработку, а также способствует дальнейшей поддержке системы.

Список литературы

1. В. М. Белов, М. В. Пургина, В. В. Востриков, И. А. Чудинов, "Построение прогностической модели для диагностики и лечения бесплодия," 2022 IEEE International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON), Екатеринбург, Российская Федерация, 2022, С.1460-1464, doi: 10.1109/SIBIRCON56155.2022.10017045.
2. И.В. Коровин, И.А. Пулкин, A.S. Veranyan, Исследование скоростей выполнения базовых математических задач популярных языков программирования // Экономика и качество систем связи. – 2019. - №3. – С. 63-68.
3. Д.С. Кобылкин, О.В. Юсупова, Перспектива применения средств Entity Framework при разработке систем распределенной обработки данных на языке C# // Международный научный журнал "Символ науки". – 2021. - № 2. – С. 14-17.
4. Сравнение производительности C++ и C#. / Текст: электронный. – URL: https://habr.com/ru/post/266163 (Дата обращения: 14.04.2023).
5. Д.С. Бухаров, О некоторых особенностях веб-сервера IIS // Международный научный журнал "Инновационная наука". – 2015. - № 6. – С. 35-39.
6. When to use Kestrel with a reverse proxy / Текст: электронный. – URL: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/servers/kestrel/when-to-use-a-reverse-proxy?view=aspnetcore-7.0
7. Host ASP.NET Core on Linux with Ngin / Текст: электронный. – URL: https://learn.microsoft.com/en-us/aspnet/core/host-and-deploy/linux-nginx?view=aspnetcore-7.0&tabs=linux-ubuntu
8. Nginx, Apache, Cloudflare - statistics and overview of popular web servers / Текст: электронный. – URL: https://timeweb.com/ru/community/articles/nginx-apache-cloudflare-statistika-i-obzor-populyarnyh-veb-serverov
9. List of software registered in Russia as a medical product / Текст: электронный. – URL: https://webiomed.ru/blog/spisok-programmnogo-obespecheniia-zaregistrirovannogo-kak-meditsinskoe-izdelie
10. Thomas L. Saaty, On the measurement of the intangible. An Approach to Relative Measurements Based on the Principal Eigenvector of the Pairwise Comparison Matrix // Electronic journal Cloud of science. – 2015. - №1 – 35 p.

References

1. V. M. Belov, M. V. Purgina, V. V. Vostrikov and I. A. Chudinov, "Building a Predictive Model for the Diagnosis and Infertility Treatment," 2022 IEEE International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON), Yekaterinburg, Russian Federation, 2022, pp. 1460-1464, doi: 10.1109/SIBIRCON56155.2022.10017045.
2. I.V. Korovin, I.A. Pulkin, A.S. Veranyan, Research of speeds of perfomance of basic mathematical problems of popular programmic languages // Economics and quality of communication systems. – 2019. - №3. – pp. 63-68.
3. D.S. Kobylkin, O.V. Yusupova, The prospect of using Entity Framework tools in the development of distributed data processing systems in C# // International scientific journal "Symbol of Science". – 2021. - № 2. – pp. 14-17.
4. Performance comparison of C++ and C# [Online]. Available: https://habr.com/ru/post/266163
5. D.S. Bukharov, About some features of the iis web server // International scientific journal "Innovative science". – 2015. - № 6. – pp. 35-39.
6. When to use Kestrel with a reverse proxy [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/servers/kestrel/when-to-use-a-reverse-proxy?view=aspnetcore-7.0
7. Host ASP.NET Core on Linux with Nginx [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/host-and-deploy/linux-nginx?view=aspnetcore-7.0&tabs=linux-ubuntu
8. Nginx, Apache, Cloudflare - statistics and overview of popular web servers [Online]. Available: https://timeweb.com/ru/community/articles/nginx-apache-cloudflare-statistika-i-obzor-populyarnyh-veb-serverov
9. List of software registered in Russia as a medical product [Online]. Available: https://webiomed.ru/blog/spisok-programmnogo-obespecheniia-zaregistrirovannogo-kak-meditsinskoe-izdelie
10. Thomas L. Saaty, On the measurement of the intangible. An Approach to Relative Measurements Based on the Principal Eigenvector of the Pairwise Comparison Matrix // Electronic journal Cloud of science. – 2015. - №1 – 35 p.