УДК 004.7 DOI 10.18522/2311-3103-2021-4-256-266

И.В. Родыгина, А.В. Наливайко

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ТЕХНОЛОГИЙ ДЛЯ РАЗРАБОТКИ СЕРВЕРНОЙ ЧАСТИ СИСТЕМЫ УПРАВЛЕНИЯ ПРОДАЖАМИ

Данная статья посвящена важной теме при разработке веб-приложений: выбор технологии для написания серверной части приложения. Вопрос выбора правильного языка и фреймворка для реализации серверной части приложения всегда актуален, поскольку от этого зависит его качество работы: сможет ли сервер обработать большое количество запросов, насколько быстро будет произведена обработка данных и выдача их пользователю - что особенно важно для системы управления продажами, поскольку она предполагает работу с большим количеством данных. Большинство современных веб-приложений написаны с использованием таких языков как PHP, NodeJS, поскольку они обеспечивают разработчику высокую скорость написания кода. Метод разработки с помощью PHP позволяет писать блокирующий и не блокирующий код, который, в любой реализации, при большом количеств запросов будет значительно нагружать систему. NodeJS позволяет реализовывать асинхронный не блокирующий код, но отсутствие типизации может значительно снизить качество работы над проектом при его масштабировании. В таком случае следует рассмотреть Java - его фреймворки и библиотеки, которые позволят выполнить задачу. В статье описан принцип взаимодействия клиентской и серверной части веб-приложения. По выделенным критериям проведено сравнение таких технологий как Java, PHP, платформы NodeJS, а также рассмотрен принцип работы бессерверной архитектуры с помощью сервиса Google Firebase, Рассмотрены фреймворки и библиотеки для создания серверной части приложения. Также важным этапом при сравнении будет проведение тестов производительности, которые покажут, какую нагрузку могут выдержать фреймворки, какое количество запросов могут обработать, задержка между ними. В результате исследования, на основе проведенного исследования, будет выбрана оптимальная технология, которая будет использована для разработки серверной части системы для управления продажами.

Backend; java vert.x; php; node.js; веб-приложение; SAAS.

I.V. Rodygina, A.V. Nalivayko

COMPARATIVE ANALYSIS OF TECHNOLOGIES FOR DEVELOPING THE SERVER SIDE OF THE SALES MANAGEMENT SYSTEM

This article focuses on a topic when developing web applications: the choice of technology for writing the server side of the application. The question of the correct language and framework for the implementation of the server side of the application is always relevant, since its quality of work depends on this: whether the server will process a large number of requests, how quickly the data will be processed and issued to the user - which is especially important for a sales management system, since it assumes work with a large amount of data. Most modern web applications are written using PHP, NodeJS. The method of development with PHP allows us to write blocking and non-blocking code, which in any implementation, with a large number of requests, will significantly load the system. NodeJS allows to implement asynchronous non-blocking code, but the lack of typing can reduce the work on the project when it scales with quality. Java and its frameworks and libraries will allow completing this task. The article describes the principle of interaction between the client and server parts of a web application. Based on the selected criteria, a compared were the technologies such as Java, PHP, NodeJS, and also the principle of operation of a serverless architecture using the Google Firebase service was considered, frameworks and libraries for creating the server side of an application were considered. Also, an important stage in the comparison will be the execution of tests that will show what loads the frameworks can withstand, how many requests can be processed, the delay between them. The optimal technology will be selected, which will be used to develop the server side for sales management.

Backend; java vert.x; php; node.js; web application; SAAS.

Введение. Актуальность темы обусловлена тем, что системы, которые предполагают работу с большим количеством данных, должны обеспечивать бесперебойную работу и быструю скорость выдачи данных пользователю. Из этого следует, что перед тем, как приступить к написанию серверной части веб-приложения, необходимо провести анализ технологий, который позволят реализовать ее.

Цель работы – провести сравнительный анализ технологий для разработки серверной части системы управления продажами.

Для достижения цели были поставлены следующие задачи:

- рассмотреть понятие backend системы и ее взаимодействие с клиентской частью;
- проанализировать языки, фреймворки и библиотеки для написания backend системы;
- сравнить технологии по ключевым параметрам;
- провести тесты производительности;
- выявить оптимальный фреймворк для разработки.

Backend и его взаимодействие с браузером. Backend включает в себя код, который исполняется на сервере. Он включает в себя взаимодействие с базами данных, API, обработку входящих данных. Пользователь работает лишь с внешней частью системы – заполняя некоторую форму и подтверждая ввод, клиент отправляет введенные данные на сервер, где они затем, при необходимости, подвергаются обработке и сохраняются в базу данных. Поэтому, когда пользователь снова входит в приложение, вся информация остается на виду, где затем ее можно отредактировать, удалить и выполнит другие действия.

Клиентская часть приложения и сервер взаимодействуют друг с другом с помощью HTTP запросов, AJAX. Как правило, используется архитектура REST API, которая включает в себя следующие методы:

- GET – запрос данных
- POST – создание новой записи
- PUT – обновление данных
- DELETE – удаление данных [1].

В современных веб-приложениях используется AJAX-модель взаимодействия. В фоне браузер создает HTTP запрос к серверу. Сервер получает запрос, обрабатывает и возвращает ответ браузеру. Браузер получает ответ от сервера, как правило в формате JSON, результат отображается на странице без ее перезагрузки. Поскольку система будет реализована в виде Single Page Application, ее жизненный цикл продемонстрирован на рис. 1.

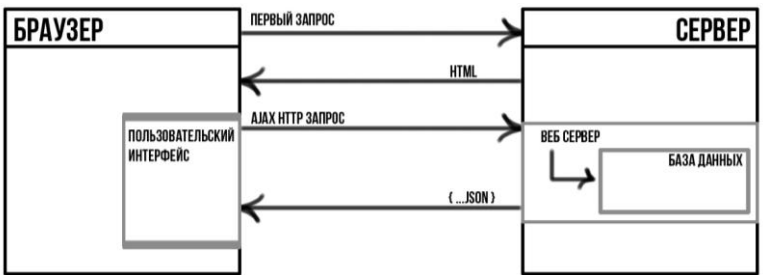

Рис. 1. Жизненный цикл SPA

Популярные технологии для реализации серверной части приложения. Наиболее популярными языками программирования при разработке северной части являются JavaScript, Java, PHP, C#. Но стоит отметить, что в чистом виде языки

используются довольно редко – на помощь приходят фреймворки и библиотеки, которые содержат в себе уже готовые блоки кода, реализующие тот или иной функционал, благодаря которому разработчик значительно экономит время [2].

Также существует понятие Serverless-архитектуры (бессерверная архитектура). В основе продукта, который предоставляет данную услугу сервер есть. Для разработчика это значит, что можно сосредоточиться на написании функционала, не тратить время на настройку базы данных, веб-сервера. Основным компонентном здесь выступает облачная функция (FaaS). Функции – это независимые единицы развертывания, которые могут выполнять разные действия, например, сохранение пользователя в базу данных, обработка файлов [3]. Один из сервисов, который позволяет воспроизвести Serverless-архитектуру – Google Firebase.

Таким образом, для проведения исследования были выбраны языки программирования и технологии, указанные в табл. 1.

Таблица 1

Технологии разработки серверной части приложений

	Языки программирования			
	Java	**JavaScript**	**PHP**	**Serverless**
Технологии	Vert.x	Node.JS	Laravel	Firebase
	Spring			

Сравнение будет проводиться по следующим параметрам (исключая Serverless облачные функции Firebase – о них будет сказано отдельно):

- поток/процесс;
- неблокирующий I/O;
- типизация;
- простота использования.

Сразу стоит отметить, что эти параметры сравнения будут относиться скорее к языкам программирования (а в случае с JavaScript – к платформе NodeJS), фреймворки и библиотеки как бы «вытекают» из области применения определенного языка программирования и являются инструментами для ускорения разработки приложений, поэтому они будут описаны дополнительно, также будут приведены результаты тестов производительности.

Потоки/процессы и неблокирующий I/O. Практически каждый сайт на PHP использует следующую модель. Браузер отправляет запрос к веб-серверу, на котором установлен Apache, который создает отдельный процесс для входящего запроса. Такой режим работы называется «Один к одному»: один пользовательский запрос – один серверный процесс для его обработки [18]. Принцип работы архитектуры продемонстрирован на рис. 2.

Такое решение крайне затратно в «high load» проектах, поскольку требует большое количество ресурсов для обработки большого количества одновременных соединений. В PHP нет потоков выполнения, поэтому выполнить задачи параллельно не получится.

Java поддерживает многопоточность, благодаря чему на каждое соединение может быть создан свой поток. Это позволяет сэкономить память, потоки могут обращаться к памяти друг друга, таким образом есть возможность получить кэшированные данные. Многопоточность в Java реализуется путем расширения класса Threads или наследованием от класса Runnable [4]. Реализация неблокирующего I/O в Java возможна с помощью пакета java.nio.

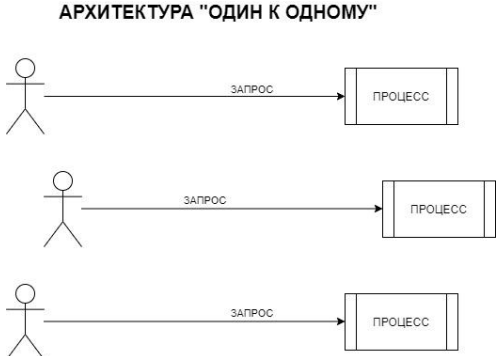

Рис. 2. Архитектура «Один к одному»

NodeJS работает в одном потоке, реализация неблокирующего I/O здесь достигается путем передачи функций обратного вызова [5]. Этот механизм называется EventLoop – цикл событий, который ставит в очередь некоторый callback (обработчик), который запускается асинхронно после завершения процесса чтения/записи – рис. 3. Пример такой реализации, на примере NodeJS и запроса к документу коллекции MongoDB, продемонстрирован в листинге 1.

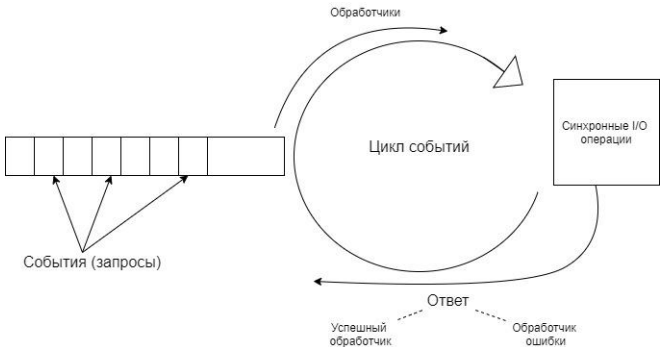

Рис. 3. Цикл событий

Листинг 1 – вызов коллбэка после запроса к базе данных

```
db.collection.find({name: “Anton”}, (err, result) => {
  if (err) return console.error(err)
  console.log(result)
})
```

Типизация. Типизация – важный аспект, который напрямую влияет на возможность поддержки проекта, когда масштабы проекта достигают среднего значения. Преимущество строгой типизации в том, что разработчик держит код под контролем: типы входящих и возвращаемых данных. Он как бы задает себе строгие рамки, за которые не может выйти, гарантируя, например, что функция будет принимать объект именно класса Car, а не Human [17]. Это полезно не только для проекта, но также и для компилятора, поскольку указание типов позволит ему делать подсказки по оптимизации. Таким образом, из перечисленных языков программирования строгую типизацию имеет лишь Java. Да, в PHP, начиная с 7 версии, также была введена типизация, но по умолчанию ее нет. Для подключения типизации необходимо использовать оператор declare() с параметром «strict_types=1» [6] – пример продемонстрирован в листинге 2.

Листинг 2 – строгая типизация для значений аргументов функции в PHP

```
declare(strict_types=1);
function multiply(int $a, int $b) {
   return $a * $b;
}
```

Простота использования. Исходя из возможностей языков можно сказать, что самым сложным вариантом написания серверной части приложения будет Java, поскольку кривая обучения здесь более крутая в силу богатой экосистемы языка. Чтобы разобраться в ней потребуется не мало времени, но при этом задача изучения не невыполнимая. Сравнивая PHP и NodeJS можно сказать, что второй использует JavaScript, код которого выглядит более лаконично и понятно. Порог входа для JS значительно ниже, поскольку отсутствует типизация, ООП в его классическом виде. Также разработчики могут использовать JavaScript не только на клиенте, но и на сервере [7].

Таким образом, в табл. 2 представлена итоговая таблица по указанным параметрам сравнения.

Таблица 2

Результаты сравнения по параметрам

	PHP	NodeJS	Java
Работа с потоками/процессами	Процессы	Потоки	Потоки
Неблокирующий I/O	Нет	Есть	Есть
Типизация	Динамическая, возможна типизация параметров функции	Динамическая	Статическая
Сложность	Средняя	Средняя	Сложная

Vert.X и Spring Boot. Сразу стоить отметить, что Eclipse Vert.X позиционирует себя не как фреймворк, а как набор инструментов, который позволяет писать асинхронный неблокирующий код [8]. Spring Boot – фреймворк, соответственно он предоставляет набор инструментов и задает разработчику некоторые архитектурные «правила» для написания приложения. Отсюда сразу можно сделать вывод, что Spring – более тяжеловесное решение, поскольку включает в себя большое количество инструментов.

«Из коробки» Spring boot использует многопоточную модель, когда каждый запрос работает в своем потоке [15, 16]. Абстрактно, запросы быть «медленные», которые требуют большее количество времени на выполнение, и «быстрые», время выполнение которых значительно меньше. При большом количестве «медленных» запросов, «быстрые» запросы все равно будут ожидать очереди выполнения первых, даже если на выполнение им требуется минимальное время, в результате чего образуется «простой». Увеличение потоков отчасти решает эту проблему, но в то же время будет значительно нагружать систему.

Vert.X же использует асинхронный неблокирующий I/O, который работает средствами EventLoop. Таким образом, вместо того, чтобы блокировать поток при выполнении операции ввода-вывода, будет осуществлен переход к следующей операции, которая готова к выполнению, а работа с начальной задачей будет возобновлена после ее завершения. В этом случае получается возможна обработка большего количества запросов при меньшей нагрузке [19, 20].

Vert.X полезно использовать в приложениях, где необходимо обрабатывать большое количество одновременных запросов. С одной стороны может показаться, что библиотеку стоит использовать в исключительно крупных проектах, но это не так. Vert.X также хорошо работает и с более простыми веб-приложениями, поскольку рано или поздно количество трафика будет расти. В таком случае уже будет существовать асинхронная обработка событий, скорость работы не пострадает. Библиотека является полиглотом. Она поддерживает достаточно много языков: Java, Groovy, Scala, Kotlin, JavaScript, Ruby.

Laravel. Laravel – один из самых популярных фреймворков PHP. Он следует модели MVC и значительно облегчает разработку веб-приложений благодаря инструментам роутинга, аутентификации, сеансов [9].

MVC расшифровывается как «Модель – представление – контроллер» [12]. Модель представляет из себя данные, с которыми работает приложение. Например, список товаров в базе данных и категории, за которыми они закреплены. Модель взаимодействует с контроллером. Если пользователь запрашивает страницу со списком товаров, то роль контроллера в том, чтобы выдать в шаблон именно эти данные, а не другие. Если пользователь хочет создать новый товар, то он обращается к контроллеру создания. Шаблон отвечает за показ пользователю данных в оформленном виде. Общая схема работа продемонстрирована на рис. 4.

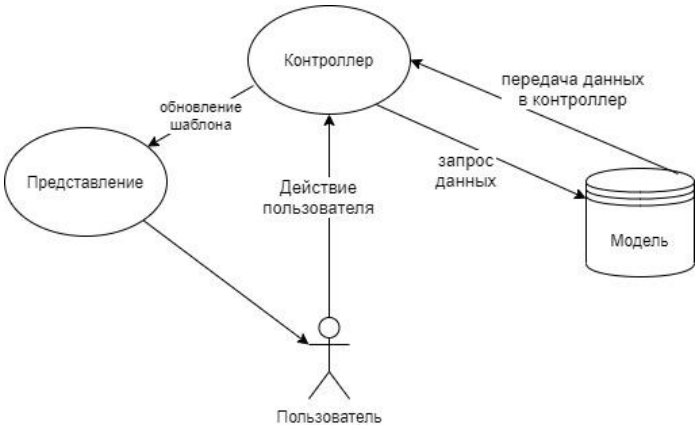

Рис. 4. Схема работы MVC

Бенчмарки. Бенчмарк – он же тест производительности – необходим для определения качества работы технологий по тем или иным параметрам.

В процессе принятия решения о выборе стэка технологии для будущей системы, стоит опираться на результаты тестирования производительности. Производительность напрямую влияет на скорость работы, возможность обработать большое количество запросов.

Результаты тестирования должны быть максимально объективными, поскольку иначе результат может быть ошибочным или не корректным. TechemPower предоставляет данные тестирования по большому количеству фреймворком и языков [11]. Их преимущество состоит в том, что, во-первых, тесты для каждой технологии пишут сами разработчики.

Это важно, поскольку они знают, как работает их технология, какие есть нюансы при написании кода – каждый пользователь может написать тест по-своему, и каждый получит разный результат. Во-вторых, написанные тесты запускаются в одном окружении – за счет чего полученные результаты имеют справедливый характер, поскольку проходили работу в одинаковых условиях.

Тест на выдачу чистого текста. На рис. 5 продемонстрированы результаты теста на выдачу чистого текста, который заключается в составлении фреймворком ответа в виде простого текста «Hello, World» на запрос. Здесь запускается шесть тестов при 256, 1024, 4096, 16384 параллельных потоков – лучший результат у Vert.X, худший – Laravel.

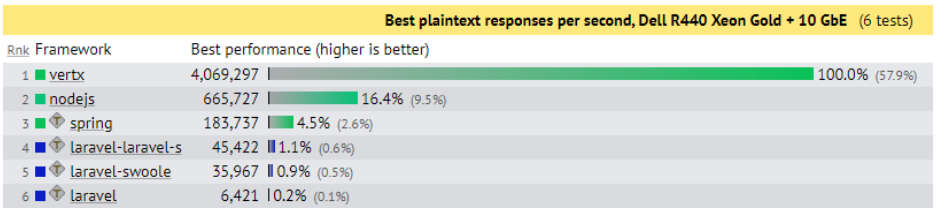

Рис. 5. Рейтинг теста на выдачу чистого текста

Сериализация. Сериализация – процесс перевода объекта в JSON строку [13]. В данном случае фреймворком ожидается объект вида «message»: {«Hello, World!»}, а сервер в свою очередь возвращает его в виде JSON строки.

Результаты теста продемонстрированы на рис. 6. Тест запускается при 16, 32, 64, 128, 256, 512 параллельных потоков. Здесь лидер снова Vert.X, при этом на рис. 7 можно заметить, что для Eclipse Vert.X задержка между ответами в среднем 4мс. Наименьшая средняя задержка у фреймворка NodeJS.

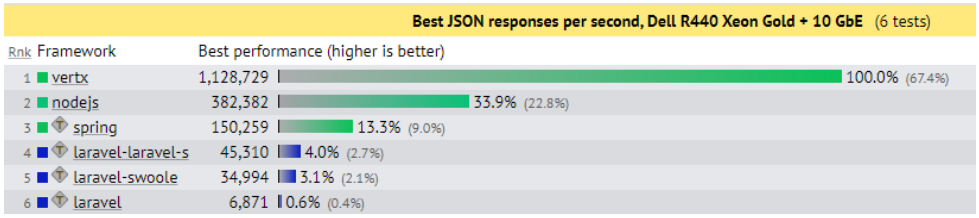

Рис. 6. Результаты теста на сериализацию объекта

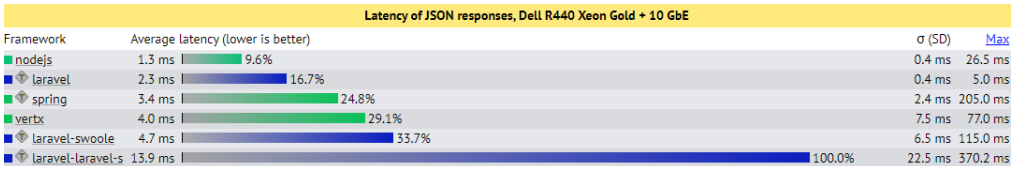

Рис. 7. Задержка между ответами

Множественные запросы к базе данных. Неотъемлемая часть любой системы – запросы к базе для получения необходимых данных. В данном бенчмарке проводится выборка множества полей из простой базы данных, а затем производится их сериализация, формируется JSON-ответ. Тест запускает 1, 5, 10, 15 и 20 запросов к базе данных за один запрос от клиента. Все они проводятся при 512 параллельных потоках.

По результатам, продемонстрированным на рис. 8 можно сделать вывод, что Vert.X может обработать большее количество запросов. Здесь он работает совместно с БД Postgres. Средняя задержка для него также минимальная из всех выбранных технологий – 16мс, максимальная задержка составила 76.2 мс.

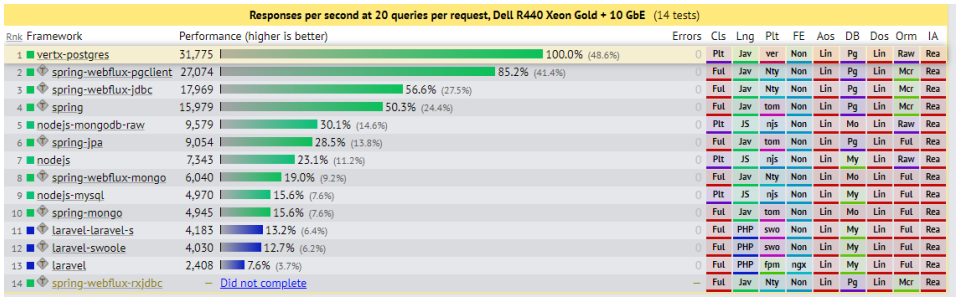

Рис. 8. Результаты теста на множественные запросы к базе данных

Firebase. Firebase отлично подходит для создания прототипов приложений, поскольку позволяет быстро строить бэкэнд. Данный сервис предоставляет большое количество инструментов – от облачной базы данных и облачных функций, то аналитики. Облачные функции позволяют разработчикам перенести API приложения в облако и вызывать их в нужных местах [10].

Сервис условно бесплатный, то есть до достижения определенного количества чтений и записей данных [14]. Отсюда можно сделать вывод, что необходимо правильно организовывать чтение данных, поскольку за это сервисом будет взыматься дополнительная плата.

Данные, которые хранятся здесь – привязаны к Firebase. Это проблема практически всех «Бэкэнд как сервис» поставщиков, что нет возможности и инструментов для переноса данных на другую платформу.

Вывод. Подводя итоги исследования, можно сделать вывод, что для разработки системы для управления продажами будет выбрана библиотека Java Vert.X. Она написана на Java – языке со статической типизацией. При масштабировании проекта это будет иметь значение, поскольку предоставит компилятору больше информации о коде и позволит выявить больше потенциальных ошибок, поведение приложения будет более предсказуемым.

Неблокирующий асинхронный код этой библиотеки позволит быстро обрабатывать большое количество запросов без блокирования кода и не затрачивать лишние ресурсы, так как системы управления продажами предполагают работу с большим количеством данных.

Результаты бенчмарков доказывают, что Vert.X – мощный инструмент для работы не только с большим количеством данных, но также он будет показывать хорошие результаты при выдаче статической информации.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. MDN Web Docs: Клиент-сервер. – Свободный режим доступа: https://developer. mozilla.org/ru/docs/Learn/Server-side/First_steps/Client-Server_overview (дата обращения: 24.07.2021).
2. *Ксендзовский И. Д., Калюжный И.Д., Зариковская Н.В.* Разработка серверной части приложений и систем: технологии и языки программирования // European reserch: Сб. статей XXVII Международной научно-практической конференции. В 2-х ч. Пенза, 07 июня 2020 года. – 2020. – С. 30-32.
3. *Макосий А. И., Макосий Р.* Современная облачная инфраструктура: бессерверные вычисления // Вестник Хакасского государственного университета им. Н.Ф. Катанова. – 2019. – № 2 (28). – С. 13-16.
4. *Семченко Р.В., Еровлев П.А.* Работа с многопоточностью в Java // Постулат. – 2020. – № 9 (59). – С. 45-46.
5. NodeJS: Обзор блокирующих и неблокирующих вызовов сервер. – Свободный режим доступа: https://nodejs.org/ru/docs/guides/blocking-vs-non-blocking (дата обращения: 25.07.2021).

6. PHP: Объявление типов. – Свободный режим доступа: https://www.php.net/manual/ru/language.types.declarations.php#language.types.declarations.strict (дата обращения: 25.07.2021).
7. *Гурулев Д.А.* Сравнение средств разработки PHP и node.js // Фундаментальные и прикладные разработки в области технических и физико-математических наук: Сб. научных статей по итогам работы третьего международного круглого стола, Казань, 31 июля 2018 года. – С. 88-91.
8. Vert.X: Официальная документация. – Свободный режим доступа: https://vertx.io/ (дата обращения: 27.07.2021).
9. *Насиров Э.Ф., Кириллов Д.С., Червнова М.В., Мертинс Г.Р.* Laravel-PHP-фреймворк // Прорывные научные исследования: проблемы, закономерности, перспективы: Сб. статей XV Международной научно-практической конференции: в 2 ч., Пенза, 30 декабря 2020 года. – С. 62-64.
10. *Париняк А. Ю.* Сравнительный анализ поставщиков облачных услуг для разработки backend-составляющей мобильного приложения // Научные труды магистрантов и аспирантов: Сб. научных трудов. – 2020. – С. 183-186.
11. TechEmpower: Web Framework Benchmarks – Свободный режим доступа: https://www.techempower.com/benchmarks/ (дата обращения: 28.07.2021).
12. *Бахтин И. В*. Главные принципы MVC и смысл использования в разработке программных продуктов // Форум молодых ученых. – 2020. – № 1 (41). – С. 63-65.
13. Docs Microsoft: Сериализация в .NET – Свободный режим доступа: https://docs.microsoft.com/ru-ru/dotnet/standard/serialization/ (дата обращения: 28.07.2021).
14. Firebase: Pricing. – Свободный режим доступа: https://firebase.google.com/pricing (дата обращения: 25.07.2021).
15. *Куликова Н.Н., Маширов О.А., Соломыков А.Д., Яковлев А.С*. Основы back-end разработки на spring // Фундаментальные и прикладные научные исследования: актуальные вопросы, достижения и инновации: Сб. статей XXXVI Международной научно-практической конференции: в 2 ч., Пенза, 27 июля 2020 года. – 2020. – С. 158-160.
16. *Макарова О.В*., *Машанин А.С., Ястребков А.С.* Обзор компонентов Spring Framework для разработки микросервисных приложений // Наука настоящего и будущего. – 2020. – Т. 1. – С. 179-182.
17. *Дедов С.В*., *Кирсанов О.Д., Тимошевская О.Ю.* Анализ преимуществ наиболее востребованных современных языков программирования // Актуальные вопросы современной науки : сборник статей по материалам XVII международной научно-практической конференции, Томск, 19 декабря 2018 года. – 2018. – С. 63-72.
18. *Вендров, А.М*. CASE-технологии: Современные методы и средства проектирования информационных систем. – М.: Финансы и статистика, 1998. – 176 с.
19. MDN Web Docs: Параллельная модель и цикл событий – Свободный режим доступа: https://developer.mozilla.org/ru/docs/Web/JavaScript/EventLoop (дата обращения: 20.07.2021).
20. Red Hat: Troubleshooting the Performance of Vert.x Applications, Part I – The Event Loop Model – Свободный режим доступа: https://www.redhat.com/en/blog/troubleshooting-performance-vertx-applications-part-i-%E2%80%94-event-loop-model (дата обращения: 20.07.2021).

REFERENCES

1. MDN Web Docs: Client-Servcer. Available at: https://developer.mozilla.org/ru/docs/Learn/Server-side/First_steps/Client-Server_overview (accessed 24 July 2021).
2. *Ksendzovskiy I. D., Kalyuzhnyy I.D., Zarikovskaya N.V.* Razrabotka servernoy chasti prilozheniy i sistem: tekhnologii i yazyki programmirovaniya [Development of the server side of applications and systems: technologies and programming languages], *European reserch: Sb. statey XXVII Mezhdunarodnoy nauchno-prakticheskoy konferentsii. V 2-kh chastyakh, Penza, 07 iyunya 2020 goda* [European reserch: Collection of articles of the XXVII International scientific and practical conference. In 2 parts, Penza, June 07, 2020], 2020, pp. 30-32.
3. *Makosiy A.I., Makosiy R.* Sovremennaya oblachnaya infrastruktura: besservernye vychisleniya [Modern cloud infrastructure: serverless computing], *Vestnik Khakasskogo gosudarstvennogo universiteta im. N.F. Katanova* [Bulletin of the Khakass State University named after N.F. Katanova], 2019, Vol 2 (28), pp. 13-16.
4. *Semchenko R.V., Erovlev P.A.* Rabota s mnogopotochnost'yu v Java [Working with multithreading in Java], *Postulat* [Postulate], 2020, Vol. 9 (59), pp. 45-46.

5. NodeJS: Overview of Blocking vs Non-Blocking. Available at: https://nodejs.org/ru/docs/ guides/ blocking-vs-non-blocking (accessed 25 July 2021).
6. PHP: Type declarations. Available at: https://www.php.net/manual/ru/language.types. declarations.php#language.types.declarations.strict (accessed: 25 July 2021).
7. *Gurulev D.A.* Sravnenie sredstv razrabotki PHP i node.js [Comparison of PHP and node.js development tools], *Fundamental'nye i prikladnye razrabotki v oblasti tekhnicheskikh i fiziko-matematicheskikh nauk: Sbornik nauchnykh statey po itogam raboty tret'ego mezhdunarodnogo kruglogo stola, Kazan', 31 iyulya 2018 goda* [Fundamental and applied developments in the field of technical and physical and mathematical sciences: Collection of scientific articles following the results of the third international round table, Kazan, July 31, 2018], pp. 88-91.
8. Vert.X: Official documentation. Available at: https://vertx.io/ (accessed 27 July 2021).
9. *Nasirov E.F., Kirillov D.S., Chervnova M.V., Mertins G.R.* Laravel-PHP-freymvork [Laravel-php framework], *Proryvnye nauchnye issledovaniya: problemy, zakonomernosti, perspektivy: sbornik statey XV Mezhdunarodnoy nauchno-prakticheskoy konferentsii: v 2 ch., Penza, 30 dekabrya 2020 goda* [Breakthrough scientific research: problems, patterns, prospects: collection of articles of the XV International Scientific and Practical Conference: at 2 pm, Penza, December 30, 2020], pp. 62-64.
10. *Parinyak A.Y.* Sravnitel'nyy analiz postavshchikov oblachnykh uslug dlya razrabotki backend-sostavlyayushchey mobil'nogo prilozheniya [Comparative analysis of cloud service providers for the development of a backend component of a mobile application], *Nauchnye trudy magistrantov i aspirantov: Sbornik nauchnykh trudov* [Scientific works of undergraduates and graduate students: Collection of scientific papers], 2020, pp. 183-186.
11. TechEmpower: Web Framework Benchmarks Available at: https://www.techempower.com/ benchmarks/ (accessed 28 July 2021).
12. *Bahtin I.V.* Glavnye principy MVC i smysl ispol'zovaniya v razrabotke programmnykh produktov [The main principles of MVC and the meaning of using it in software development], *Forum molodykh uchenykh* [Forum of Young Scientists], 2020, Vol 1 (41), pp. 63-65.
13. Docs Microsoft: Serialization in .NET. Available at: https://docs.microsoft.com/en-us/dotnet/standard/serialization (accessed 28 July 2021).
14. Firebase: Pricing. Available at: https://firebase.google.com/pricing (accessed 25 July 2021).
15. *Kulikova N.N., Mashirov O.A., Solomykov A.D., Yakovlev A.S.* Osnovy back-end razrabotki na spring [Fundamentals of back-end development at spring]. *Fundamental'nye i prikladnye nauchnye issledovaniya: aktual'nye voprosy, dostizheniya i innovacii: sbornik statej XXXVI Mezhdunarodnoj nauchno-prakticheskoн konfereneьii: v 2 ch., Penza, 27 iyulya 2020 goda* [Fundamental and applied scientific research: current issues, achievements and innovations: collection of articles of the XXXVI International Scientific and Practical Conference: in 2 parts, Penza, July 27, 2020], 2020, pp. 158-160.
16. *Makarova O.V., Mashanin A.S., Yastrebkov A.S.* Obzor komponentov Spring Framework dlya razrabotki mikroservisnykh prilozheniy [Overview of Spring Framework components for the development of micro service applications], *Nauka nastoyashchego i budushchego* [Science of the present and the future], 2020, Vol. 1, pp. 179-182.
17. *Dedov S.V., Kirsanov O.D., Timoshevskaya Y.U.* Analiz preimushchestv naibolee vostrebovannykh sovremennykh yazykov programmirovaniya [Analysis of the advantages of the most popular modern programming languages], *Aktual'nye voprosy sovremennoy nauki: Sb. statey po materialam XVII mezhdunarodnoy nauchno-prakticheskoy konferentsii, Tomsk, 19 dekabrya 2018 goda* [Actual issues of modern science: a collection of articles based on the materials of the XVII International Scientific and Practical Conference, Tomsk, December 19, 2018], 2018, pp. 63-72.
18. *Vendrov A.M.* CASE-tekhnologii: Sovremennye metody i sredstva proektirovaniya informaeьionnykh system [CASE-technologies: Modern methods and means of designing information systems]. Moscow: *Finansy i statistika* [Finance and Statistics], 1998, 176 p.
19. MDN Web Docs: Concurrency model and the event loop. Available at: https://developer.mozilla.org/ru/docs/Web/JavaScript/EventLoop (accessed 20 July 2021).
20. Red Hat: Troubleshooting the Performance of Vert.x Applications, Part I – The Event Loop Model. Available at: https://www.redhat.com/en/blog/troubleshooting-performance-vertx-applications-part-i-%E2%80%94-event-loop-model (accessed 20 July 2021).

Статью рекомендовал к опубликованию к.т.н. А.А. Полупанов.

Родыгина Ирина Владимировна – Государственный морской университет им. адмирала Ф.Ф. Ушакова; e-mail: habarova@mail.ru; г. Новороссийск, Россия; тел.: +79282284417; кафедра радиоэлектроники и информационных технологий; к.т.н.; доцент.

Наливайко Антон Викторович – e-mail: guzurogmn@gmail.com; тел.: 89002396697; магистрант.

Rodygina Irina Vladimirovna – Admiral Ushakov Maritime State University; e-mail: habarova@gmail.ru, Novorossiysk, Russia; phone: +79282284417; the department of radioelectronics and information technologies; cand. of eng. sc.; associate professor.

Nalivayko Anton Viktorovich – e-mail: guzurogmn@gmail.com; phone: +79002396697; master's student.