

Научная статья

УДК 004.4'2

DOI: 10.17072/1993-0550-2026-3-99-109

<https://elibrary.ru/bxydea>



Сравнительный анализ производительности и архитектурных характеристик фреймворков Spring Boot, ASP.NET Core и Ktor для разработки серверных приложений

Оксана Геннадиевна Старцева¹, Эрик Раилевич Фаттахов²

^{1, 2} Башкирский государственный педагогический университет им. М. Акмуллы, Уфа, Россия

¹ starcevaog@mail.ru

² erik.fattakhov.04@mail.ru

Аннотация. В статье проводится сравнительный анализ трех современных фреймворков для разработки серверных приложений: Spring Boot (Java/Kotlin), ASP.NET Core (C#) и Ktor (Kotlin). Актуальность исследования обусловлена необходимостью обоснованного выбора технологического стека при создании высоконагруженных корпоративных систем. Научная новизна заключается в разработке методики сравнения производительности различных реализаций. В статье представлены качественные критерии сравнения: архитектурные особенности, модульность, экосистема, безопасность, мониторинг и количественные: время сборки проекта, время холодного старта, объем кода, а также метрики производительности под нагрузкой – средняя латентность (p50, p95, p99), пропускная способность и потребление памяти. Эксперимент проведен на идентичных тестовых сценариях (CRUD-микросервис с подключением к одинаковой базе данных) с фиксацией условий окружения (CPU, RAM, OS, инструмент нагрузочного тестирования). Результаты показывают, что Ktor демонстрирует наилучшие показатели скорости сборки и запуска, минимальный объем кода, однако уступает ASP.NET Core по потреблению оперативной памяти, но превосходит по латентности и пропускной способности. На основе полученных данных сформулированы рекомендации по выбору фреймворка в зависимости от приоритетов проекта: скорость разработки, производительность или долгосрочной поддержки.

Ключевые слова: Ktor; Spring Boot; ASP.NET Core; сравнительный анализ; производительность; латентность; p95; p99; нагрузочное тестирование.

Для цитирования: Старцева О. Г., Фаттахов Э. Р. Сравнительный анализ производительности и архитектурных характеристик фреймворков Spring Boot, ASP.NET Core и Ktor для разработки серверных приложений // Вестник Пермского университета. Математика. Механика. Информатика. 2026. № 3(74). С. 99–109. DOI: 10.17072/1993-0550-2026-3-99-109. <https://elibrary.ru/bxydea>.

Статья поступила в редакцию 07.05.2026; одобрена после рецензирования 03.09.2026; принята к публикации 12.10.2026.



© Старцева О. Г., Фаттахов Э. Р., 2026

Лицензировано по CC BY 4.0. Чтобы посмотреть копию этой лицензии, посетите <https://creativecommons.org/licenses/by/4.0/>

Research article

Comparative Analysis of Performance and Architectural Characteristics of Spring Boot, ASP.NET Core, and Ktor Frameworks for Developing Server Applications

Oksana G. Startseva¹, Erik R. Fattakhov²^{1,2} Aknulla Bashkir State Pedagogical University, Ufa, Russia¹ starcevaog@mail.ru² erik.fattakhov.04@mail.ru

Abstract. The article provides a comparative analysis of three modern frameworks for developing server applications: Spring Boot (Java/Kotlin), ASP.NET Core (C#) and Ktor (Kotlin). The relevance of the research stems from the need for a well-founded choice of the technological stack when creating highly loaded corporate systems. The scientific novelty lies in the development of a methodology for comparing the performance of various implementations. The article presents qualitative comparison criteria (architectural features, modularity, ecosystem, security, monitoring) and quantitative criteria (project build time, cold start time, code volume), as well as performance metrics under load (average latency (p50, p95, p99), bandwidth, and memory consumption). The experiment was conducted on identical test scenarios (CRUD microservice the same database) with environmental conditions (CPU, RAM, OS, load testing tool) held constant. The results show that Ktor demonstrates the best build and startup speeds, the minimal amount of code; it is inferior to ASP.NET Core in terms of RAM usage, but surpasses it in terms of latency and bandwidth. Based on the data obtained, recommendations are formulated for choosing a framework depending on the priorities of the project: speed of development, performance, or long-term support.

Keywords: *Ktor; Spring Boot; ASP.NET Core; comparative analysis; efficiency; latency; p95; p99; load testing.*

For citation: Startseva, O. G. and Fattakhov, E. R. (2026), "Comparative Analysis of Performance and Architectural Characteristics of Spring Boot, ASP.NET Core, and Ktor Frameworks for Developing Server Applications", *Bulletin of Perm University. Mathematics. Mechanics. Computer Science*, no 3(74), pp. 99–109, DOI: 10.17072/1993-0550-2026-3-99-109, <https://elibrary.ru/bxydea>.

The article was submitted 07 May 2026; approved after review 03 Sep 2026; accepted for publication 12 Oct 2026.

Введение

Современная разработка серверных приложений требует от инженерных команд обоснованного выбора технологического стека, обеспечивающего баланс между скоростью разработки, производительностью и долгосрочной поддерживаемостью проекта. В течение последнего десятилетия доминирующие позиции в корпоративном сегменте занимали экосистемы на основе Java (Spring Framework / Spring Boot) [1] и C# (ASP.NET Core) [2]. Эти фреймворки характеризуются зрелой экосистемой, обширной документацией и доказанной способностью обслуживать высоконагруженные системы на протяжении многих лет.

Однако последние годы отмечены ростом популярности языка Kotlin [3], позиционируемого как современная альтернатива Java с улучшенной null-безопасностью, поддержкой функциональных конструкций и полной совместимостью с существующей JVM-экосистемой [4]. Естественным следствием стало появление фреймворка Ktor [5],

разрабатываемого компанией JetBrains и предлагающего альтернативный подход к построению серверных приложений: легкое ядро, модульную архитектуру и асинхронность на основе корутин.

Несмотря на растущее количество публикаций, посвященных отдельным аспектам работы с Ktor на сегодняшний день, отсутствуют комплексные исследования, в которых проводилось бы систематическое сравнение Ktor с устоявшимися корпоративными фреймворками по единой методологии.

Постановка задачи

Целью настоящей работы является проведение сравнительного анализа фреймворков – Spring Boot, ASP.NET Core и Ktor. Для объективного сравнения была предложена методика, включающая несколько групп критериев:

1. Архитектурные особенности и экосистемная зрелость (качественный анализ).
2. Метрики разработки (время сборки, объем кода).
3. Метрики производительности под нагрузкой (латентность, пропускная способность, потребление памяти).

Методы исследования

Для достижения поставленной цели в работе применяется разработанная методика, включающая следующие элементы:

- унифицированный тестовый сценарий (с подключением к базе данных) для всех трех фреймворков;
- расширенный набор метрик, включающий перцентили латентности (p95, p99) наряду со средними значениями;
- детальная фиксация условий эксперимента, обеспечивающая воспроизводимость результатов.

1. Качественное сравнение архитектур и экосистем

1.1. Критерии сравнения

Для сравнительного анализа были выбраны следующие характеристики, значимые с точки зрения долгосрочной поддержки корпоративных систем:

- язык разработки и экосистема;
- модель обработки запросов (синхронная / асинхронная);
- подход к конфигурации и инверсии управления;
- модульность и расширяемость;
- поддержка типовых задач (безопасность, работа с БД, документация API, мониторинг);
- кроссплатформенность и типы сборки.

1.2. Результаты качественного сравнения

Результаты сравнения по выбранным критериям представлены в табл. 1. Как следует из табл. 1, Ktor предоставляет разработчику минималистичное ядро без предустановленного набора решений. В отличие от Spring Boot, где инверсия управления и внедрение зависимостей являются фундаментальной основой фреймворка, Ktor не навязывает конкретную архитектурную парадигму [6]. В ASP.NET Core новые возможности добавляются через цепочку промежуточных обработчиков, а в Ktor – через подключаемые модули.

Таблица 1. Сравнение архитектурных характеристик фреймворков

Критерий	Ktor	Spring Boot	ASP.NET Core
Язык разработки	Kotlin	Java / Kotlin	C#
Разработчик	JetBrains	VMware, сообщество	Microsoft
Год первого релиза	2017	2002 (Spring), 2012 (Spring Boot)	2014 (.NET Core), 2016 (ASP.NET Core)
Модель обработки запросов	Асинхронная (корутины)	Синхронная / Асинхронная (по выбору)	Асинхронная (async/await)
Что входит по умолчанию	Минимум – только сервер	Почти все – готовые инструменты, настройки	Настройки сервера и цепочки обработки
Управление компонентами	Встроенный механизм	Встроенный механизм	Встроенный механизм
Подход к модульности	Плагины Ktor (Gradle/Maven зависимости)	Стартеры Spring Boot	NuGet-пакеты, фреймворк сборки .NET
Поддержка OpenAPI	Отдельный модуль	Встроенный стартер	Отдельный модуль
Работа с БД	Exposed или сторонние библиотеки	Spring Data JPA / JDBC	Entity Framework Core
Тип сборки	JAR	JAR / WAR	DLL / EXE
Кросс-платформенность	Да (JVM)	Да (JVM)	Да (.NET Core / .NET 5+)

Важной особенностью Ktor является то, что разработчик самостоятельно определяет набор необходимых модулей на этапе генерации проекта [7]. Это позволяет создавать предельно легкие приложения, не несущие избыточной функциональности. Однако такая гибкость имеет и обратную сторону: требует от команды более высокой квалификации для принятия архитектурных решений.

Ktor и ASP.NET Core умеют обрабатывать много запросов одновременно без задержек. Spring Boot изначально работает по-другому – запросы обрабатываются по очереди [8]. Это может влиять на производительность при высокой нагрузке, что и проверялось в эксперименте.

Spring Boot дает разработчику почти все необходимое сразу. В нем уже есть инструменты для управления зависимостями, настройка происходит автоматически, и подклю-

чение баз данных, защиты и отслеживания работы приложения не требует дополнительных действий. Низкий порог входа для разработчиков с начальным уровнем квалификации с таким фреймворком. Но у этого подхода есть минусы: приложение дольше запускается и потребляет больше памяти.

ASP.NET Core – быстрый и работает на разных системах. Он дает гибкие настройки и удобные инструменты для работы с базами данных. При этом расходует мало памяти, поэтому подходит для проектов с ограниченными ресурсами.

2. Методология экспериментального сравнения

2.1. Тестируемое приложение

Для обеспечения сопоставимости результатов на каждом из трех фреймворков было реализовано идентичное по функциональности приложение – REST API микросервис для управления сущностью Client. Приложение поддерживает следующие эндпоинты:

- GET /client/all – получение списка всех клиентов;
- GET /client/{id} – получение клиента по идентификатору;
- POST /client – создание нового клиента;
- POST /client/search – поиск клиента по имени;
- PUT /client/{id} – обновление существующего клиента;
- DELETE /client/{id} – удаление клиента.

В качестве базы данных использовалась PostgreSQL. Выбор обусловлен распространенностью данной СУБД в промышленной разработке корпоративных приложений, а также поддержкой всеми тремя рассматриваемыми фреймворками через стандартные драйверы и ORM-решения. Экземпляр базы данных для всех приложений один. Схема базы включает таблицу clients с полями id (первичный ключ) и name (строка).

Для работы с базой данных в Ktor использовалась библиотека Exposed вместе с системой управления соединениями HikariCP. Максимальное количество одновременных подключений было ограничено десятью, время ожидания соединения не превышало 30 секунд, а неактивные подключения закрывались через 10 минут. В Spring Boot аналогичная система HikariCP работает по умолчанию с похожими настройками [9]. В ASP.NET Core использовалась встроенная система управления подключениями NpgsqlDataSource.

Исходный код всех трех реализаций доступен в репозитории [10].

2.2. Характеристики оборудования и программного обеспечения

Эксперименты проводились на следующей конфигурации: процессор: Intel Core i7-1165G7 2.80GHz, оперативная память: 16.0 ГБ, операционная система: Windows 11 22H2, версия JDK: JetBrains Runtime 21.0.9, версия .NET SDK: .NET 10, версии фреймворков: Ktor: 3.4.0, Spring Boot: 4.0.3, ASP.NET Core: 10.0.3.

Каждое приложение запускалось в изолированном окружении. Перед каждой серией измерений система перезагружалась для исключения влияния кэширования со стороны операционной системы.

2.3. Инструмент нагрузочного тестирования

Для генерации нагрузки использовался инструмент k6 [11]. Выбор обусловлен следующими характеристиками. Встроенная поддержка измерения перцентилей (p90, p95, p99) без дополнительной обработки результатов. Возможность написания сценариев на JavaScript, что обеспечивает воспроизводимость тестов.

Сценарий нагрузки эмулирует реалистичное распределение запросов к эндпоинтам: 50% операций чтения (GET /client/all, GET /client/{id}, POST /client/search), 30% операций записи (POST /client, PUT /client) и 20% операций удаления (DELETE /client/{id}). Такой профиль нагрузки приближен к типичному сценарию работы CRM-системы.

Параметры нагрузки:

- длительность теста: 60 секунд;
- количество виртуальных пользователей: 50;
- количество прогонов: 5 для каждого фреймворка;
- период прогрева: первые 10 секунд каждого прогона не учитываются в результатах.

Для каждого прогона k6 автоматически фиксирует следующие метрики:

- средняя латентность;
- p90, p95, p99 латентность;
- пропускная способность (RPS);
- процент успешных запросов (HTTP 2xx).

Потребление оперативной памяти серверным приложением измерялось отдельно с помощью команды получения ресурсов по процессу синхронно с прогонами k6.

2.4. Измеряемые метрики

В ходе эксперимента фиксировались следующие метрики.

1. Метрики сборки и запуска:
 - время полной чистой сборки проекта (мс) – от полной очистки сборки до завершения процесса сборки;
 - время холодного старта (мс) – от запуска приложения до момента готовности принять первый HTTP-запрос;
 - объем кода – количество строк и количество символов в исходных файлах (без учета автогенерируемого кода и конфигурационных файлов).
2. Метрики производительности под нагрузкой:
 - средняя латентность (мс) – среднее арифметическое времени ответа на запрос.
 - медиана (p50) – значение, ниже которого находится 50% ответов.
 - перцентили p95 и p99 – значения, ниже которых находятся 95% и 99% ответов соответственно.
 - пропускная способность (RPS) – количество запросов, обработанных за секунду.
 - потребление оперативной памяти (MB) – количество потребляемой памяти экземпляром.

Выбор перцентилей p95 и p99 обусловлен тем, что в высоконагруженных системах средняя латентность может маскировать наличие "медленных" запросов, критически влияющих на пользовательский опыт [12].

2.5. Процедура проведения измерений

Для каждого фреймворка выполнялась следующая последовательность действий:

1. Перезагрузка операционной системы.
2. Чистая сборка проекта. Фиксация времени сборки.
3. Запуск приложения. Фиксация времени холодного старта.
4. Прогрев приложения – выполнение 1000 запросов, не учитываемых в результатах.
5. Прогон нагрузки. Фиксация всех метрик.
6. Остановка приложения.
7. Повтор шагов 3–6 пять раз.

Итоговые значения для каждой метрики вычислялись как среднее арифметическое по всем прогонам.

3. Результаты эксперимента

3.1. Метрики сборки и статического анализа

Результаты измерения времени сборки, времени холодного старта и объема кода представлены в табл. 2.

Таблица 2. Сравнение метрик сборки и объема кода

Метрика	Ktor	Spring Boot	ASP.NET Core
Время сборки (мс)	6558	9484	6700
Время холодного старта (мс)	802	3778	1600
Количество строк кода	224	326	377

Время сборки является важным маркером при выборе фреймворка. Чем быстрее сборка, тем чаще команда может интегрировать изменения и получать обратную связь от автоматических тестов. Ktor собирается за 6.6 секунды, что на 31% быстрее Spring Boot (9.5 сек) и сопоставимо с ASP.NET Core (6.7 сек). В условиях микросервисной архитектуры, где десятки сервисов пересобираются одновременно, эта разница становится критической.

Время холодного старта важно для сценариев автоматического масштабирования (Kubernetes HPA) и частых перезапусков в разработке. Ktor запускается за 0.8 секунды – это в 4.7 раза быстрее Spring Boot (3.8 сек) и в 2 раза быстрее ASP.NET Core (1.6 сек). Если одновременно запускать несколько микросервисов, разница во времени старта может измеряться минутами, в течение которых система будет недоступна.

Количество строк кода – косвенная метрика сложности поддержки. Исследования показывают, что количество дефектов коррелирует с объемом кода [13]. Меньше кода означает:

- меньше мест для потенциальных ошибок;
- быстрее вникает новый разработчик;
- дешевле рефакторинг и добавление функциональности;
- ниже затраты на проверку кода.

Ktor требует 224 строк против 326 у Spring Boot (на 31% меньше) и 377 у ASP.NET Core (на 40% меньше). Это существенное преимущество для долгосрочной поддержки проекта.

3.2. Метрики производительности под нагрузкой

Результаты нагрузочного тестирования представлены в табл. 3. Тестирование проводилось при 50 виртуальных пользователях в течение 60 секунд.

Таблица 3. Сравнение метрик производительности

Метрика	Ktor	Spring Boot	ASP.NET Core
Оперативная память min (MB)	154.66	258.54	79.71
Оперативная память max (MB)	219.51	304.05	148.72
Средняя латентность (мс)	6.23	8.52	10.80
p90 (мс)	10.82	14.47	21.27
p95 (мс)	14.49	17.80	26.67
p99 (мс)	24.31	26.90	39.32
Пропускная способность (RPS)	411.99	400.68	392.50

Средняя латентность отражает типичное время ответа системы. Ktor показывает 6.23 мс – на 27% лучше Spring Boot (8.52 мс) и на 42% лучше ASP.NET Core (10.80 мс). Для пользователя веб-приложения разница в 4–5 миллисекунд незаметна, но при обработке миллионов запросов это перерастает в часы сэкономленного времени и меньшую нагрузку на клиентские приложения.

Перцентили p95 и p99 гораздо важнее среднего значения. Они показывают, как долго ждут ответа самые медленные запросы – именно они создают у пользователя ощущение замедленной работы системы. В высоконагруженных системах стандартом является требование p99 < 100 мс [14].

У Ktor p99 = 24.31 мс – это на 10% лучше Spring Boot (26.90 мс) и на 38% лучше ASP.NET Core (39.32 мс). Это означает, что даже в худшем случае (1% запросов) Ktor отвечает быстрее конкурентов. Для критичных по времени систем (финансовые транзакции, бронирование, голосовые вызовы) такая разница может быть определяющей при выборе фреймворка.

RPS (request per second) показывает, сколько запросов система способна обработать за секунду. Ktor обрабатывает 412 запросов в секунду, незначительно опережая Spring Boot (401) и ASP.NET Core (393). Разница в 3–5% не является решающей, но демонстрирует, что Ktor не уступает популярным зрелым решениям.

Важно отметить: при более высокой латентности у конкурентов (8.5–10.8 мс) их RPS ожидаемо ниже – это логичная взаимосвязь. Ktor выигрывает по обоим параметрам.

Оперативная память – один из основных факторов стоимости хостинга. В облачных провайдерах цена линейно растет с объемом RAM. При масштабе в сотни микросервисов разница становится существенной.

ASP.NET Core демонстрирует наилучшие показатели: 79–149 МБ. Ktor занимает промежуточное положение: 155–220 МБ. Spring Boot потребляет больше всех: 259–304 МБ.

Тем не менее, абсолютные значения Ktor (220 МБ пик) являются приемлемыми для большинства корпоративных проектов и не должны рассматриваться как серьезный недостаток.

3.3. Ключевые выводы эксперимента

1. Ktor – показал лучший результат по скорости разработки и поддержки среди трех сравниваемых фреймворков: самая быстрая сборка, самый быстрый холодный старт, минимальный объем кода. Это снижает затраты на разработку, CI/CD и адаптацию новых разработчиков.
2. Ktor – лидер по производительности под нагрузкой: лучшие показатели латентности (средняя, p90, p95, p99) и пропускной способности. Это означает, что система на Ktor способна обработать больше запросов быстрее, особенно важно для требовательных к скорости приложений.
3. Ktor умерен в потреблении памяти: уступает ASP.NET Core, но превосходит Spring Boot. Разница в памяти с Spring Boot может давать ощутимую экономию при масштабировании.

Обсуждение

В ходе работы и тестирования сложилось несколько наблюдений, которые не отражены в количественных показателях, но важны для практического выбора фреймворка.

Удобство разработки. Разработка приложения на Ktor с языком Kotlin позволяет сократить объем шаблонного кода по сравнению с кодом на Spring с Java или на ASP.NET Core с C#. Язык Kotlin позволяет писать меньше кода и делает его понятнее – в нем уже

есть готовые инструменты для создания классов данных, встроенная защита от null-ошибок и гибкие расширения языка. В ASP.NET Core добавление новой зависимости в систему требует явного объявления интерфейса, реализации соответствующего класса и последующей регистрации этой связи в контейнере внедрения зависимостей. Это правильная архитектура, но она превращается в значительный объем шаблонного кода. Для быстрого прототипирования или стартапа Ktor выигрывает со значительным преимуществом.

Кадровая проблема. Несмотря на преимущества Ktor и его растущую популярность, компании все равно чаще выбирают Spring Boot. Причина не в технологии, а в рынке труда. Java-разработчиков с опытом Spring – значительное число. Найти команду из 10 человек, которые знают Ktor, значительно сложнее и дороже. Пока это не изменится, Spring останется корпоративным стандартом по умолчанию. Ktor – скорее выбор стартапов, внутренних инструментов и Kotlin-команд.

Легкость как архитектурный риск. Ktor намеренно минималистичен и не включает пул соединений с базой данных по умолчанию. В ходе эксперимента было обнаружено, что при отсутствии явной настройки пула (например, HikariCP) средняя латентность составляла 72 мс. После добавления пула соединений латентность снизилась до 6.23 мс – улучшение в 11.6 раза. В отличие от Ktor, Spring Boot и ASP.NET Core предоставляют пул соединений по умолчанию, что снижает риск подобных ошибок. Это требует от разработчиков на Ktor более высокой квалификации при конфигурировании приложения.

Ограничения исследования. В работе не измерялись показатели загрузки процессора и стандартное отклонение результатов, что может быть исправлено в будущих исследованиях. Также тестирование проводилось только при 50 виртуальных пользователях; при более высоких нагрузках (500–1000 виртуальных пользователей) результаты могут отличаться.

Заключение

В ходе выполнения работы была достигнута поставленная цель: проведен сравнительный анализ трех современных фреймворков для разработки серверных приложений – Spring Boot, ASP.NET Core и Ktor – по качественным и количественным критериям.

Предложенная методика, включающая качественные и количественные критерии, показала свою применимость для объективного сравнения фреймворков и позволила выявить сильные и слабые стороны каждого из них. В начале работы были сформулированы три группы критериев сравнения: архитектурные особенности, метрики разработки и метрики производительности под нагрузкой. По каждой из групп гипотеза о преимуществах Ktor подтвердилась частично: фреймворк показал лучшие результаты по скорости сборки, холодному старту, объему кода, латентности и RPS, однако уступил ASP.NET Core по потреблению памяти.

По результатам качественного анализа установлено, что Ktor предоставляет минималистичное ядро с модульной архитектурой, требуя от разработчика самостоятельного выбора компонентов. Spring Boot предлагает готовый набор компонентов с развитым контейнером управления зависимостями. ASP.NET Core в этом смысле занимает среднюю позицию: он имеет встроенный механизм внедрения зависимостей и конвейер обработки запросов.

Количественный эксперимент показал следующие результаты. По метрикам разработки Ktor лидирует: время сборки – 6.6 секунд (на 31% быстрее Spring Boot), время

холодного старта – 0.8 секунды (в 4.7 раза быстрее Spring Boot). По метрикам производительности под нагрузкой при 50 виртуальных пользователях Ktor также показал лучшие результаты: средняя латентность – 6.23 мс (на 27% лучше Spring Boot), р99 – 24.31 мс (на 38% лучше ASP.NET Core), пропускная способность – 412 RPS. По потреблению памяти Ktor занял промежуточное положение (155–220 MB), уступая ASP.NET Core (80–149 MB), но превосходя Spring Boot (259–304 MB).

На основе полученных данных сформулированы следующие рекомендации. Ktor рекомендуется для стартапов и Kotlin-команд, где приоритетами являются скорость разработки, быстрый старт и высокая производительность при условии высокой квалификации разработчиков. Spring Boot остается предпочтительным выбором для крупных корпораций из-за большого числа специалистов, разработчиков и зрелости экосистемы. ASP.NET Core рекомендуется для сценариев с жесткими ограничениями по оперативной памяти, особенно при масштабировании на сотни микросервисов.

Список источников

1. Лёзин И. А. Разработка веб-приложений с использованием Spring Boot: учеб. пособие / И. А. Лёзин, И. В. Лёзина. Самара: Самарский университет, 2023. 96 с.
2. Лок Э. ASP.Net Core в действии: руководство / Э. Лок; перевод с англ. Д. А. Беликова. 3-е изд. М.: ДМК Пресс, 2025. 1046 с.
3. *The RedMonk Programming Language Rankings: January 2025* [Электронный ресурс] // URL: <https://redmonk.com/sogady/2025/06/18/language-rankings-1-25/> (дата обращения: 22.11.2025).
4. Сомон П. И. Волшебство Kotlin: руководство / П. И. Сомон; перевод с англ. А. Н. Киселева. М.: ДМК Пресс, 2020. 536 с.
5. *Ktor Docs* [Электронный ресурс] // URL: <https://ktor.io/docs/welcome.html> (дата обращения: 22.11.2025).
6. *Metanit Ktor Модули* [Электронный ресурс] // URL: <https://metanit.com/kotlin/ktor/1.6.php> (дата обращения: 22.11.2025).
7. *Ktor Project Generator* [Электронный ресурс] // URL: <https://start.ktor.io/settings> (дата обращения: 22.11.2025).
8. Докука О. Практика реактивного программирования в Spring 5 / О. Докука, И. Лозинский. М.: ДМК Пресс, 2019. 508 с.
9. Булискерия Б. Б. Методики оптимизации высоконагруженных приложений на SPRING FRAMEWORK: системный подход / Б. Б. Булискерия, Д. М. Пряшников // Оптические технологии, материалы и системы (Оптотех – 2025): Междунар. науч.-техн. конф., Москва, 08–12 декабря 2025 года. М.: МИРЭА – Российский технологический университет, 2026. С. 1319–1326. EDN IWPHPG.
10. *KtorArticle* [Электронный ресурс] // URL: <https://github.com/alkmanistik/KtorArticle> (дата обращения: 25.04.2026).
11. *K6* [Электронный ресурс] // URL: <https://k6.io/> (дата обращения: 22.11.2025).
12. *Site Reliability Engineering* / ed. By B. Beyer, C. Jones, J. Petoff, N. R. Murphy. Sebastopol, CA: O'Reilly Media, Inc., 2016. 524 с.
13. Макконнелл С. Совершенный код: учеб. пособие/ Стив Макконнелл. М.: Русская Редакция, 2014. 896 с.
14. *The Site Reliability Workbook: Practical Ways to Implement SRE* / B. Beyer [et al.]. Sebastopol: O'Reilly Media, Inc., 2018. 512 p.

References

1. Lyezin, I. A. and Lyezina, I. V. (2023), Development of web applications using Spring Boot, Samara University, Samara, Russia, 96 p.

2. Lock, A. (2025), ASP.Net Core in Action, 3rd ed., DMK Press, Moscow, Russia, 1046 p.
3. RedMonk (2025), "The RedMonk Programming Language Rankings: January 2025", available at: <https://redmonk.com/sograzy/2025/06/18/language-rankings-1-25/> (data accessed: 22.11.2025).
4. Somon, P. I. (2020), The Magic of Kotlin, DMK Press, Moscow, Russia, 536 p.
5. JetBrains (2025), "Ktor Documentation", available at: <https://ktor.io/docs/welcome.html> (data accessed: 22.11.2025).
6. Metanit (2025), "Ktor Modules", available at: <https://metanit.com/kotlin/ktor/1.6.php> (data accessed: 22.11.2025).
7. JetBrains (2025), "Ktor Project Generator", available at: <https://start.ktor.io/settings> (data accessed: 22.11.2025).
8. Dokuka, O. and Lozinsky, I. (2019), Reactive Programming Practice in Spring 5, DMK Press, Moscow, Russia, 508 p.
9. Buliskeria, B. B. and Pryashnikov, D. M. (2026), "Methods for optimizing high-load applications on the Spring Framework: a systematic approach", *Optical technologies, materials and systems (Optotech - 2025)*, MIREA — Russian Technological University, Moscow, Russia, pp. 1319-1326, EDN: IWPHPG.
10. alkmanistik (2026), "KtorArticle — source code repository", available at: <https://github.com/alkmanistik/KtorArticle> (accessed 25.04.2026).
11. Grafana Labs (2025), "k6 — Open source load testing tool", available at: <https://k6.io/> (data accessed: 22.11.2025).
12. Beyer, B., Jones, C., Petoff, J. and Murphy, N. R. (eds.) (2016), Site Reliability Engineering, O'Reilly Media, Inc., Sebastopol, CA, USA, 552 p.
13. McConnell, S. (2014), Code Complete, Russian Edition, Moscow, Russia, 896 p.
14. Beyer, B. et al. (2018), The Site Reliability Workbook: Practical Ways to Implement SRE, O'Reilly Media, Inc., Sebastopol, CA, USA, 512 p.

Информация об авторах:

О. Г. Старцева – кандидат педагогических наук; доцент кафедры информационных технологий Башкирского государственного педагогического университета им. М. Акмуллы (450077, Россия, г. Уфа, ул. Октябрьской революции, д. 3, -а), ORCID: 0000-0001-8198-7086, eLibrary AuthorID: 675876;

Э. Р. Фаттахов – студент направления "Информационные системы и технологии" Башкирского государственного педагогического университета им. М. Акмуллы (450077, Россия, г. Уфа, ул. Октябрьской революции, д. 3-а).

Information about the authors:

O. G. Startseva – Candidate of Sciences (Pedagogic); Associate Professor at the Department of Information Technology, Akmulla Bashkir State Pedagogical University (3-a, Oktyabrskoy Revolyutsii st., Ufa, 450077, Russia), ORCID: 0000-0001-8198-7086, eLibrary AuthorID: 675876;

E. R. Fattakhov – Student ("Information systems and technologies"), Akmulla Bashkir State Pedagogical University (3-a, Oktyabrskoy Revolyutsii st., Ufa, 450077, Russia).