Научная статья
УДК 004.891.2
DOI: https://doi.org/10.18127/j19998554-202506-02

Микросервисная архитектура веб-сервиса с использованием нейросетей и алгоритмов компьютерного зрения для онлайн обработки контента

А.И. Карпухин[1], **Е.А. Вайсберг**[2], **Н.А. Данилкина**[3]
[1] Финансовый университет при Правительстве Российской Федерации (Москва, Россия)
[2] ООО «Открытая мобильная платформа» (Москва, Россия)
[3] АО «Россельхозбанк» (Москва, Россия)
[1] aikarpukhin@fa.ru, [2] 11lizik.lizik@gmail.com, [3] n-danilkina@list.ru

Аннотация

Постановка проблемы. Предложенная в работе архитектура веб-сервиса, основанная на микросервисном подходе, предназначена для решения проблемы онлайн-обработки цифрового контента (изображения, видео и связных данных) с использованием алгоритмов глубокого обучения и компьютерного зрения.
Цель. Повысить эффективность обработки различных видов контента в онлайн-сервисе, в том числе распознавания, классификации и интерпретации пользовательских изображений с дальнейшим формированием рекомендаций для пользователя и его работы с контентом.
Результаты. Представлено описание реализованной на практике архитектуры веб-сервиса с использованием нейросетей и алгоритмов компьютерного зрения, включающего в себя обученную авторами модель глубокого машинного обучения.
Практическая значимость. Предложенный подход является во многом универсальным с точки зрения создания различных конфигураций микросервисов и веб-приложений для решения широкого круга задач, включая онлайн-обработку контента в экспертных системах, интерпретацию результатов анализа, а также генерацию и обобщение производных данных и формирования рекомендаций эксперта в рамках подготовки экспертных заключений.

Ключевые слова

Компьютерное зрение, микросервисная архитектура, глубокое машинное обучение, рекомендательные системы, интеграция моделей глубокого обучения, кастомные модели глубокого обучения в микросервисной архитектуре, микросервисы, экспертные системы

Для цитирования

Карпухин А.И., Вайсберг Е.А., Данилкина Н.А. Микросервисная архитектура веб-сервиса с использованием нейросетей и алгоритмов компьютерного зрения для онлайн обработки контента // Нейрокомпьютеры: разработка, применение. 2025. Т. 27. № 6. С. 17–23. DOI: 10.18127/j19998554-202506-02

A brief version in English is given at the end of the article

Введение

Представленная в данной работе архитектура веб-сервиса основана на микросервисном подходе и предназначена для онлайн-обработки цифрового контента (изображения, видео и связных данных) с использованием алгоритмов глубокого обучения и компьютерного зрения. Эта архитектура разработана для решения проблемы эффективной обработки различных видов контента в онлайн-сервисе, в том числе распознавания, классификации и интерпретации пользовательских изображений с дальнейшим формированием рекомендаций для пользователя и его работы с контентом.

В работе дано описание реализованной на практике архитектуры веб-сервиса с использованием нейросетей и алгоритмов компьютерного зрения, включающего в себя обученную авторами модель глубокого машинного обучения.

Система реализована в виде нескольких связных модулей:

веб-приложение (Web Application), обеспечивающее серверный рендеринг;

взаимодействие с пользователем и загрузку контента;

клиентское одностраничное приложение (Single-page application), исполняемое на стороне клиента;

API-приложение (API Application) – серверное API-приложение, реализующее основную бизнес-логику;

приложение с нейросетевым модулем (Neural Model Application) – модуль обработки изображений, использующий нейросетевую модель YOLO для многоклассовой классификации объектов;

база данных (Database) – базы данных, хранящей контент и метаданные.

В данной работе показано, что предложенный подход является во многом универсальным с точки зрения создания различных конфигураций микросервисов и веб-приложений для решения широкого круга задач, включая онлайн-обработку контента, интерпретации результатов анализа, а также генерации производных данных (например, формирование текстовых рекомендаций на основе результатов распознавания и классификации объектов в изображениях и видео).

В условиях бурного роста экономики искусственного интеллекта, когда непрерывно создаются, развиваются и внедряются новые модели глубокого машинного обучения, предложенный подход, обеспечивающий универсальную интеграцию таких моделей в бизнес-процессы, является актуальным.

Цель работы – повысить эффективность обработки различных видов контента в онлайн-сервисе, в том числе распознавания, классификации и интерпретации пользовательских изображений с дальнейшим формированием рекомендаций для пользователя и его работы с контентом.

Микросервисная архитектура

Созданию и развитию миркосервисных архитектур как подходу к разработке программного обеспечения, предусматривающему разбиение приложения на множество небольших, независимых и взаимодополняющих модулей (микросервисов) посвящено множество публикаций. Например, наиболее актуальные – публикации [1–5], данные о которых представлены в табл. 1 и 2.

Таблица 1. ***Данные о научных публикациях по тематике микросервисов в eLibrary***

Ключевое слово	Число упоминаний в наименованиях статей в журналах и в ключевых словах, ед.
Микросервисы	1251
Микросервисная архитектура	1229
Микросервисный подход	329

Примечание. Источник: https://elibrary.ru

Таблица 2. ***Данные о научных публикациях по тематике микросервисов в Scholar Google***

Ключевое слово	Число упоминаний в наименованиях статей в журналах и в ключевых словах, ед.
Microservices	~ 91 600
Microservice architecture	~ 53 200
Microservice approach	~ 49 200

Примечание. Источник: https://scholar.google.com/

В то же время не много научных работ посвящено проблемам интеграции моделей и методов глубокого машинного обучения в микросервисные архитектуры [6–7]. При этом наблюдается интенсивный рост количества и качества различных моделей глубокого машинного обучения и их вариации, в связи с чем возникает проблема интеграции новых моделей в уже существующие или вновь создаваемые информационные архитектуры.

Цель создания систем с интегрированными моделями глубокого машинного обучения, в том числе компьютерного зрения, на основе микросервисов – повышение качества и эффективности управления информацией, данными и знаниями, обрабатываемыми пользователями для поддержки принятия решений.

Материалы и методы решения задачи

В процессе анализа и обобщения существующих решений и данных о различных подходах к организации информационных комплексов и систем на базе микросервисной архитектуры установлено, что для создания таких систем необходимо последовательное решение следующих задач:

формирование требований к сервису;

проектирование сервиса;

разработка инструментария и модулей: DL/CV; Backend; Frontend; DevOps.

В данной работе представлена реализованная авторами микросервисная архитектура, ориентированная на использование модели компьютерного зрения YOLO для распознавания и классификации патернов на изображениях и видео.

Обобщенная схема информационной системы на основе микросервисного подхода с интегрированной моделью компьютерного зрения на примере реализованного проекта – http://snapcook.tech/ представлена на рис. 1.

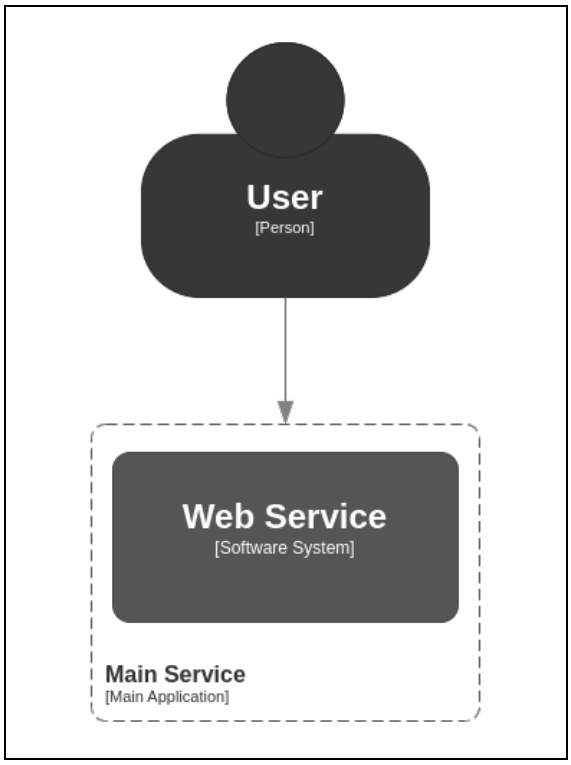

Рис. 1. Контекст проекта SnapNCook User
Fig. 1. The context of the SnapNCook User project

Пользователь системы в рамках данного подхода – это человек, который посещает веб-интерфейс системы для использования сервиса распознавания объектов на изображениях и получения рекомендаций по рецептам. Пользователь осуществляет загрузку изображения, после чего система анализирует его содержимое и возвращает список распознанных на изображении продуктов, а также предлагает возможные варианты блюд, которые можно приготовить из этих ингредиентов.

Контекстная диаграмма позволяет скоординировать понимание системы между разработчиками, архитекторами и другими заинтересованными сторонами, обеспечивая единое представление о назначении и внешних взаимодействиях системы.

Для более детального анализа структуры программной системы используется второй уровень C4-модели – архитектура контейнеров. Данный уровень абстракции позволяет рассмотреть систему как совокупность логически и технически обособленных исполняемых компонентов (контейнеров), выполняющих конкретные задачи в рамках общей функциональности. Такой подход способствует формированию гибкой, масштабируемой и легко сопровождаемой архитектуры.

На этапе проектирования крайне важно определить границы системы, основные взаимодействия с внешними пользователями и общее назначение создаваемого сервиса. Для этих целей эффективно применяется контекстная диаграмма в рамках C4-модели, которая позволяет визуализировать систему на самом высоком уровне абстракции, фокусируясь на её связи с внешним окружением на основе C4-модель (Context, Container, Component, Code) [8–9].

В рамках C4-модели принято выделять следующие *ключевые элементы для проектирования:*

контекст – общий обзор системы, описание её окружения и взаимодействия с внешними пользователями и системами;

контейнеры – представление системы в виде контейнеров: исполняемых компонентов (веб-приложения, базы данных, API-серверов и т.д.), которые разворачиваются и взаимодействуют между собой;

компоненты – детализация одного из контейнеров на уровне логических модулей или компонентов;

код – самый низкий уровень, на котором описывается внутренняя реализация одного из компонентов.

В данном работе использовался первый и второй уровень C4-модели – первоначальная визуализация контекста и архитектура контейнеров, которые отображают разбиение системы на ключевые исполняемые части (контейнеры) и связи между ними.

На рис. 2 представлена архитектура контейнеров системы SnapNCook Detection System, включающая в себя все основные модули, участвующие в процессе взаимодействия с пользователем, обработки данных и предоставления рекомендаций.

В основу системы входит модуль SnapNCook Web Service – веб-сервис, предоставляющий пользователю доступ к функциональности системы. Данный сервис отвечает за прием изображений, их обработку и формирование итоговых рекомендаций.

Реализованная система состоит из следующих ключевых компонентов:

Web Application (Контейнер: Next.js) – серверная часть веб-приложения, реализованная с использованием фреймворка Next.js. Она отвечает за начальную загрузку, рендеринг HTML/JS-контента и базовую маршрутизацию, а также может выполнять функции SEO-оптимизации и первичной доставки клиентского интерфейса пользователю [10];

Single-page application (Контейнер: TypeScript, React) – одностраничное приложение (SPA), исполняемое на стороне клиента. Оно построено с использованием React и TypeScript, выполняет всю логику пользовательского интерфейса, обработку событий, управление состоянием и отправку запросов к серверной части системы [11];

API Application (Контейнер: Go) – серверное API-приложение, реализующее основную бизнес-логику. Оно обеспечивает функции распознавания объектов на изображениях и генерации рекомендаций по рецептам. API взаимодействует с другими компонентами через HTTP/JSON и ProtoBuf RPC [12, 13];

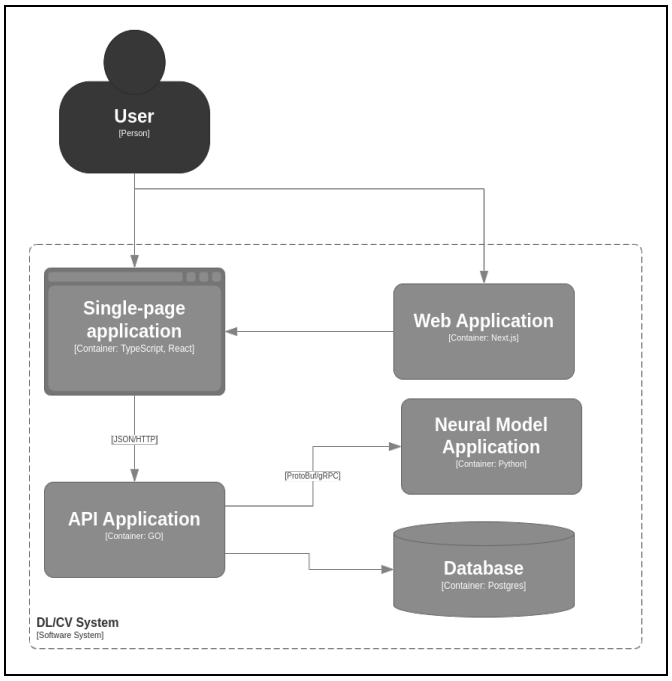

Рис. 2. Контейнер системы SnapNCook в виде микросервисной архитектуры с интегрированной моделью компьютерного зрения
Fig. 2. The SnapNCook system container in the form of a microservice architecture with an integrated computer vision model

Neural Model Application (Контейнер: Python) – модуль обработки изображений, использующий нейросетевую модель YOLO для многоклассовой классификации объектов. Он принимает изображения, выполняет их анализ и возвращает список распознанных объектов [14];

Database (Контейнер: PostgreSQL) – база данных, хранящая информацию о рецептах, продуктах и модельных классах. Она используется всеми внутренними сервисами для чтения и записи данных через PostgreSQL-драйвер [15].

Контейнерная архитектура системы обеспечивает четкое разделение обязанностей между модулями, упрощает масштабирование компонентов и повышает надёжность сервиса за счёт независимости контейнеров. Таким образом, предложенный в работе подход соответствует принципам современной микросервисной архитектуры и является хорошей основой для дальнейшего развития и внедрения новых функций, интеграции новых микросервисов и моделей глубокого машинного обучения.

Обсуждение результатов

В современных условиях микросервисные архитектуры для онлайн-обработки различных видов контента получают все большее распространение. Но главная проблема таких сервисов – необходимость подключения через API к сторонним и зачастую платным моделям глубокого машинного обучения.

Представленная авторами микросервисная архитектура с интегрированными моделями компьютерного зрения включает в себя работу с пользовательскими моделями и является во многом универсальной. Она может быть использована не только для реализации типовых потребностей пользователей в Интернет, но также в рамках многих управленческих задач в экосистеме любого предприятия и организации, где создаются специфические для компании или государственной организации модели.

Учитывая, что множество современных коммерческих и государственных организаций активно внедряют интеллектуальные системы и сервисы, предложенная в работе архитектура может быть востребована в самых разных отраслях, например, в рамках развития концепции цифровой модели эксперта.

Цифровая модель эксперта–сотрудника включает в себя набор цифровых сервисов для обеспечения деятельности экспертов, в том числе модули баз знаний и разнородного контента, модули языковых мо-

делей, мультимодальные модели, сервисы автоматической обработки текстов (рубрицирование, аннотирование, суммаризация и т.п.), модули классификации и маршрутизации документов (текстов) и т.д.

Практическая реализация цифровой модели эксперта–сотрудника организации подразумевает в том числе разработку микросерсвисной архитектуры вебсервисов, предложенной в данном исследовании.

В рамках концепции цифровой модели эксперта–сотрудника в режиме онлайн может осуществляться распознавание и классификация объектов в изображениях и другом контенте, которые входят в состав рассматриваемых и обрабатываемых экспертом–сотрудником комплектов документов.

Таким образом, предложенная в работе микросервисная архитектура веб-сервисов для онлайн обработки контента с использованием технологий искусственного интеллекта, компьютерного зрения и других моделей, является перспективной разработкой для осуществления цифровой трансформации организаций.

Заключение

Представленная в работе микросервисная архитектура для создания веб-сервиса обеспечивает пользователям онлайн обработку различных видов контента (изображения, видео и другие виды) с использованием моделей глубокого машинного обучения и алгоритмов компьютерного зрения.

Микросервисная архитектура в реализованной системе на основе предложенного в работе подхода включает в себя следующие ключевые элементы:

Web Application;
Single-page application;
API Application;
Neural Model Application;
Database.

Данная микросервисная архитектура может быть дополнена другими микросервисами, например, в нее могут быть добавлены новые Neural Model Applications. Это обеспечивает реализацию логики мультимодульных моделей глубокого машинного обучения.

Представленная микросервисная архитектура демонстрирует функционирование системы, основанной на использовании пользовательских нейронных сетей, что подразумевает, что модели загружаются и управляются непосредственно в рамках данной архитектуры, без обращения к сторонним сервисам через API или использование ключей доступа к внешним моделям.

Список источников

1. *Niño-Martínez V.M., Ocharán-Hernández J.O., Limón X., Pérez-Arriaga J.C*. Microservice deployment, Proceedings of the institute for system programming of the RAS. University of Veracruz. Mexico. 2023. P. 57–72.
2. *Katal A., Prasanna P., Birla R., Kunal* (2025). Evolution from Monolithic to Microservices Architecture: A New Era in Software Architecture. In: Rossit, D., Torres-Aguilar, C.E., Toncovich, A.A. (eds) Advancements in Optimization and Nature-Inspired Computing for Solutions in Contemporary Engineering Challenges. Springer Tracts in Nature-Inspired Computing. Springer. Singapore. https://doi.org/10.1007/978-981-96-0706-8_12
3. *Shethiya A.S*. Building Scalable and Secure Web Applications Using .NET and Microservices. Academia Nexus Journal. 2025. 4(1). Retrieved from https://academianexusjournal.com/index.php/anj/article/view/17
4. *Maggi Kevin & Verdecchia Roberto & Scommegna Leonardo & Vicario Enrico*. Evolution of code technical debt in microservices architectures. Journal of Systems and Software. 2024. 222. 112301. 10.1016/j.jss.2024.112301
5. *Shafi N., Abdullah M., Iqbal W*. et al. DIMA: machine learning based dynamic infrastructure management for containerized applications. Computing 107, 88 (2025). https://doi.org/10.1007/s00607-025-01445-8
6. *Kaushik N., Kumar H. & Raj V*. Micro Frontend Based Performance Improvement and Prediction for Microservices Using Machine Learning. J Grid Computing 22, 44 (2024). https://doi.org/10.1007/s10723-024-09760-8
7. *Richart M., Gorricho J.-L., Baliosian J., Contreras L.M., Muñiz A. and Serrat J*. LQ-GNN: a Graph Neural Network Model for Response Time Prediction of Microservice-based Applications in the Computing Continuum. In IEEE Transactions on Parallel and Distributed Systems, doi: 10.1109/TPDS.2025.3564214
8. *Carducci M*. Documenting Architecture. In: Mastering Software Architecture. Apress, Berkeley, CA. (2025). https://doi.org/10.1007/979-8-8688-0410-6_24
9. *Brown S*. The C4 model for visualizing software architecture. Retrieved from https://c4model.com. Licensed under CC BY 4.0.
10. *Ahmad Jourji Zaidan & Dwi Fatrianto Suyatno*. (2025). Rendering Performance Analysis of Astro JS, Next JS, Nuxt JS, and SvelteKit Frameworks Using Google Lighthouse, PageSpeed Insight, and JMeter: Rendering Performance Analysis of Astro JS,

Next JS, Nuxt JS, and SvelteKit Frameworks Using Google Lighthouse, PageSpeed Insight, and JMeter. Journal of Emerging Information Systems and Business Intelligence, 6(1), 1~13. https://doi.org/10.26740/jeisbi.v6i1.64283
11. *Narender Reddy Karka*. Best Practices for Building Scalable Single Page Applications (SPAS). International Journal of Information Technology and Management Information Systems (IJITMIS). 2025. 16(1). 1219–1241. doi: https://doi.org/10.34218/IJITMIS_16_01_087
12. *Yellavula Naren*. Hands-on RESTful Web Services with Go : Develop Elegant RESTful APIs with Golang for Microservices and the Cloud / Naren Yellavula. Second edition. Birmingham, UK: Packt Publishing, 2020. Print.
13. *Meyerson J*. The Go Programming Language. In IEEE Software. 2014. V. 31. № 5. P. 104–104. doi: 10.1109/MS.2014.127
14. *Jiang Peiyuan & Ergu Daji & Liu Fangyao & Ying Cai & Ma Bo*. A Review of Yolo Algorithm Developments. Procedia Computer Science. 2022. 199. 1066–1073. 10.1016/j.procs.2022.01.135
15. *Douglas K., Douglas S*. PostgreSQL: a comprehensive guide to building, programming, and administering PostgresSQL databases. SAMS publishing. 2003.

Информация об авторах

Александр Игоревич Карпухин – к.э.н., доцент
SPIN-код: 9538-0890
Елизавета А. Вайсберг – мл. веб-разработчик
SPIN-код: не представлен
Наталья А. Данилкина – разработчик
SPIN-код: не представлен

Статья поступила в редакцию 15.10.2025
Одобрена после рецензирования 22.10.2025
Принята к публикации 30.10.2025

Original article

Microservice architecture of a web service using neural networks and computer vision algorithms for online content processing

A.I. Karpukhin[1], **E.A. Vaysberg**[2], **N.A. Danilkina**[3]
[1] Financial University under the Government of the Russian Federation (Moscow, Russia)
[2] Open Mobile Platform LLC (Moscow, Russia)
[3] JSC «Rosselkhozbank» (Moscow, Russia)
[1] aikarpukhin@fa.ru, [2] 11lizik.lizik@gmail.com, [3] n-danilkina@list.ru

Abstract

The proposed architecture of the web service, based on a microservice approach, is designed to solve the problem of online processing of digital content (images, videos, and connected data) using deep learning algorithms and computer vision
Goal – to increase the efficiency of processing various types of content in an online service, including recognition, classification and interpretation of user images with further formation of recommendations for the user and his work with content.
The paper describes the architecture of a web service implemented in practice using neural networks and computer vision algorithms, including a deep machine learning model trained by the authors.
The paper shows that the proposed approach is largely universal in terms of creating various configurations of microservices and web applications to solve a wide range of tasks, including online content processing in expert systems, interpretation of analysis results, as well as generation and generalization of derived data and the formation of expert recommendations in the preparation of expert opinions.

Keywords

Computer vision, microservice architecture, deep machine learning, recommendation systems, integration of deep learning models, custom deep learning models in microservice architecture, microservices, expert systems

For citation

Karpukhin A.I., Vaysberg E.A., Danilkina N.A. Microservice architecture of a web service using neural networks and computer vision algorithms for online content processing. Neurocomputers. 2025. V. 27. № 6. P. 17–23. DOI: 10.18127/j19997493-202506-02 (in Russian).

References

1. *Niño-Martínez V.M., Ocharán-Hernández J.O., Limón X., Pérez-Arriaga J.C.* Microservice deployment, Proceedings of the institute for system programming of the RAS. University of Veracruz. Mexico. 2023. P. 57–72.
2. *Katal A., Prasanna P., Birla R., Kunal* (2025). Evolution from Monolithic to Microservices Architecture: A New Era in Software Architecture. In: Rossit, D., Torres-Aguilar, C.E., Toncovich, A.A. (eds) Advancements in Optimization and Nature-Inspired Computing for Solutions in Contemporary Engineering Challenges. Springer Tracts in Nature-Inspired Computing. Springer. Singapore. https://doi.org/10.1007/978-981-96-0706-8_12
3. *Shethiya A.S.* Building Scalable and Secure Web Applications Using .NET and Microservices. Academia Nexus Journal. 2025. 4(1). Retrieved from https://academianexusjournal.com/index.php/anj/article/view/17
4. *Maggi Kevin & Verdecchia Roberto & Scommegna Leonardo & Vicario Enrico*. Evolution of code technical debt in microservices architectures. Journal of Systems and Software. 2024. 222. 112301. 10.1016/j.jss.2024.112301.
5. *Shafi N., Abdullah M., Iqbal W.* et al. DIMA: machine learning based dynamic infrastructure management for containerized applications. Computing 107, 88 (2025). https://doi.org/10.1007/s00607-025-01445-8.
6. *Kaushik N., Kumar H. & Raj V.* Micro Frontend Based Performance Improvement and Prediction for Microservices Using Machine Learning. J Grid Computing 22, 44 (2024). https://doi.org/10.1007/s10723-024-09760-8
7. *Richart M., Gorricho J.-L., Baliosian J., Contreras L.M., Muñiz A. and Serrat J.* LQ-GNN: a Graph Neural Network Model for Response Time Prediction of Microservice-based Applications in the Computing Continuum. In IEEE Transactions on Parallel and Distributed Systems, doi: 10.1109/TPDS.2025.3564214.
8. *Carducci M.* Documenting Architecture. In: Mastering Software Architecture. Apress, Berkeley, CA. (2025). https://doi.org/10.1007/979-8-8688-0410-6_24
9. *Brown S.* The C4 model for visualizing software architecture. Retrieved from https://c4model.com. Licensed under CC BY 4.0.
10. *Ahmad Jourji Zaidan & Dwi Fatrianto Suyatno*. (2025). Rendering Performance Analysis of Astro JS, Next JS, Nuxt JS, and SvelteKit Frameworks Using Google Lighthouse, PageSpeed Insight, and JMeter: Rendering Performance Analysis of Astro JS, Next JS, Nuxt JS, and SvelteKit Frameworks Using Google Lighthouse, PageSpeed Insight, and JMeter. Journal of Emerging Information Systems and Business Intelligence, 6(1), 1~13. https://doi.org/10.26740/jeisbi.v6i1.64283
11. *Narender Reddy Karka*. Best Practices for Building Scalable Single Page Applications (SPAS). International Journal of Information Technology and Management Information Systems (IJITMIS). 2025. 16(1). 1219–1241. doi: https://doi.org/10.34218/IJITMIS_16_01_087
12. *Yellavula Naren*. Hands-on RESTful Web Services with Go : Develop Elegant RESTful APIs with Golang for Microservices and the Cloud / Naren Yellavula. Second edition. Birmingham, UK: Packt Publishing, 2020. Print.
13. *Meyerson J.* The Go Programming Language. In IEEE Software. 2014. V. 31. № 5. P. 104–104. doi: 10.1109/MS.2014.127.
14. *Jiang Peiyuan & Ergu Daji & Liu Fangyao & Ying Cai & Ma Bo*. A Review of Yolo Algorithm Developments. Procedia Computer Science. 2022. 199. 1066–1073. 10.1016/j.procs.2022.01.135
15. *Douglas K., Douglas S.* PostgreSQL: a comprehensive guide to building, programming, and administering PostgresSQL databases. SAMS publishing. 2003.

Information about the authors

Alexandr I. Karpukhin – Ph. D. (Ekon.), Associate Professor
Elizaveta A. Vaysberg – Junior Web Developer, Open Mobile Platform LLC
Natalya A. Danilkina – Developer

The article was submitted 15.10.2025
Approved after reviewing 22.10.2025
Accepted for publication 30.10.2025