Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра комп'ютерно-інтегрованих технологій, автоматизації, робототехніки та безпекової інженерії

I Всеукраїнська конференція «Інтелектуальні технології цивільної безпеки та робототехнічні системи аварійно-рятувальних робіт»

I All-Ukrainian Conference “Intelligent Civil Safety Technologies and Robotic Systems for Emergency and Rescue Operations”

ICSTRO

2026

I All-Ukrainian Conference

February 12 - 13, 2026

Kharkiv

УДК: 005:004.896:62-65:338.3

Інтелектуальні технології цивільної безпеки та робототехнічні системи аварійно-рятувальних робіт 2026: матеріали І-ої Всеукраїнська конференція, Харків, 12-13 лютого 2026 р.: тези доповідей / [редкол. І.Ш. Невлюдов (відповідальний редактор)].-Харків: [електронний друк], 2026. – 192 с.

У збірник включені тези доповідей, які присвячені сучасним тенденціям розвитку технологій та засобів моделювання, прогнозування та управління ризиками у сфері цивільної безпеки; техногенна та виробнича безпека: технічні засоби, оцінка ризиків, експертиза; інтелектуальні та робототехнічні системи аварійно-рятувальних робіт; кіберфізичні системи, інформаційна безпека та цифровий захист виробництв; інформаційно-комунікаційні технології в системах управління та моніторингу надзвичайних ситуацій; сталий розвиток, екологічна безпека та соціальна відповідальність у сфері цивільної безпеки; інтелектуальні системи прийняття рішень у сфері цивільного захисту.

Intelligent Civil Safety Technologies and Robotic Systems for Emergency and Rescue Operations 2026: Proceedings of I st All-Ukrainian Conference, Kharkiv, February 12 - 13, 2026: Thesises of Reports / [Ed. I.Sh. Nevlyudov (chief editor).] .- Kharkiv .: [electronic version], 2026. - 192 p.

The collection includes the thesises of reports on devoted to current trends in the development of technologies and tools for modeling, forecasting, and risk management in the field of civil safety; industrial and technological safety, including technical means, risk assessment, and expert evaluation; intelligent and robotic systems for emergency and rescue operations; cyber-physical systems, information security, and digital protection of industrial facilities; information and communication technologies in emergency management and monitoring systems; sustainable development, environmental safety, and social responsibility in the field of civil safety; and intelligent decision-support systems in civil protection.

Харківський національний університет радіоелектроніки
Кременчуцький національний університет імені Михайла Остроградського
Національний університет «Запорізька політехніка»
Національний університет «Львівська політехніка»
Державне підприємство «Південний державний проектно-конструкторський та науково-дослідний інститут авіаційної промисловості»
Головне управління ДСНС України у Харківській області

Всеукраїнська конференція «Інтелектуальні технології цивільної безпеки та робототехнічні системи аварійно-рятувальних робіт» (ICSTRO-2026)

All-Ukrainian Conference “Intelligent Civil Safety Technologies and Robotic Systems for Emergency and Rescue Operations” (ICSTRO-2026)

ЗМІСТ

JAMSTACK ARCHITECTURE AS A SYNTHESIS OF SERVERLESS BACK-END AND DYNAMIC FRONT-END

M. Vorobyov, S. Sotnik
Kharkiv National University of Radio Electronics
Ukraine, 61166, Kharkiv, Nauky av., 14
E-mail: mykyta.vorobyov1@nure.ua

***Annotation*:** The paper examines JAMstack as an approach to building web applications, where static resources are delivered via a CDN, and business logic and integrations are handled through serverless functions and managed API services. It analyzes the typical JAMstack architecture, the role of the build process, ways to implement front-end dynamics, and the integration of headless CMS and databases. It shows how this model affects performance (TTFB, caching, server offloading), security (smaller attack surface, secret management, function isolation), and scalability (automatic scaling at the CDN and serverless levels). A comparison with traditional approaches is conducted, and practical recommendations for choosing JAMstack for different scenarios are formulated.

***Key words*:** JAMstack, serverless, CDN, headless CMS, CI/CD, API.

АРХІТЕКТУРА JAMSTACK ЯК СИНТЕЗ БЕЗСЕРВЕРНОГО BACK-END ТА ДИНАМІЧНОГО FRONT-END

М. К. Воробйов, С. В. Сотник
Харківський національний університет радіоелектроніки
Україна, 61166, Харків, пр. Науки 14
E-mail: mykyta.vorobyov1@nure.ua

Анотація. У роботі розглянуто JAMstack як підхід до побудови вебзастосунків, у якому статичні ресурси віддаються через CDN, а бізнес-логіка та інтеграції виносяться у безсерверні функції та керовані API-сервіси. Проаналізовано типову архітектуру JAMstack, роль build-процесу, способи реалізації динаміки у front-end та підключення headless CMS і баз даних. Показано, як така модель впливає на продуктивність (TTFB, кешування, розвантаження сервера), безпеку (менша площа атаки, керування секретами, ізоляція функцій) та масштабованість (автоматичне масштабування на рівні CDN і serverless). Проведено порівняння з традиційними підходами та сформульовано практичні рекомендації щодо вибору JAMstack для різних сценаріїв.

Ключові слова: JAMstack, serverless, CDN, headless CMS, CI/CD, API.

Informatization, as the process of transforming information into a key resource for societal development, defines new requirements for the speed, reliability, and flexibility of digital solutions [1]. Modern web systems increasingly operate under high loads, have integrations with dozens of services, and must be updated quickly without downtime [2-6]. Under such conditions, the classic scheme, where a single server simultaneously renders pages, processes business logic, and manages data access, becomes a bottleneck: caching is more difficult, scaling is harder, and the attack surface grows. This is why JAMstack architecture comes to the forefront as a synthesis of serverless back-end and dynamic front-end. It offers a radical separation of concerns: the front-end, compiled into optimized static files (HTML, CSS, JS), is delivered through a Content Delivery Network (CDN), providing instant speed and immunity to common DDoS attacks. All dynamics and business logic are delegated to serverless functions and third-party APIs, which scale automatically and «by default». This not only eliminates the single point of failure problem but also allows developers to choose the best tools for each task, quickly integrating new services.

The relevance of JAMstack lies precisely in its alignment with the requirements of the modern digital economy, where speed, security, scalability, and flexibility are competitive advantages. This architecture is a direct response to the challenges of deep informatization across all spheres of life, when web applications transform into complex, multi-tier systems that must operate seamlessly on a global scale and constantly evolve. JAMstack not only solves technical scaling problems but also organizes the development process itself around the principles of Continuous Integration and Continuous Delivery (CI/CD), making digital products more adaptive to rapid market changes.

For a complete understanding of how JAMstack achieves its advantages, it is worth focusing on its fundamental principles. The departure from monolithic logic to an architecture built around these principles is the key to speed, security, and scalability.

The JAM abbreviation in the name emphasizes three basic ideas: JavaScript as the mechanism for dynamics in the browser, API as the sole method of accessing data and services, and Markup as static pages that can be delivered without server-side computations. In practice, this means that most pages are generated in advance through Static Site Generation (SSG) or updated incrementally (ISR), while interactivity is added through JavaScript and API calls. The JAMstack architecture is shown in Figure 1.

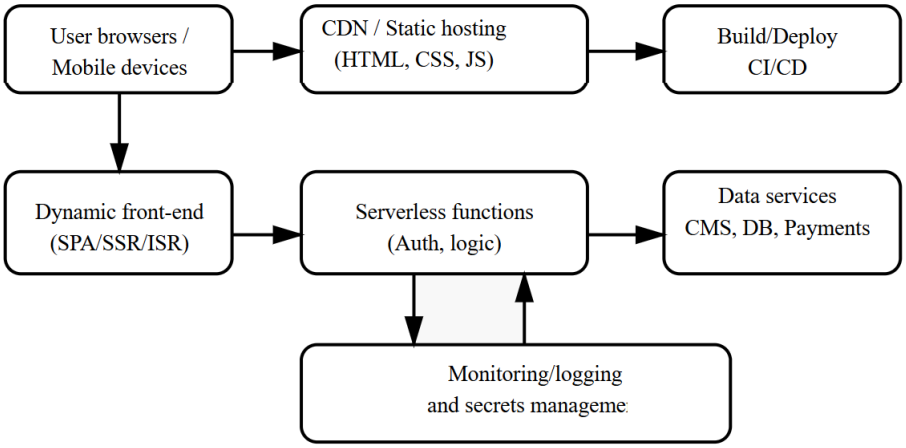

Figure 1 – JAMstack Architecture Diagram

As shown in Fig. 1, modern web architecture combines static content delivery through CDN with dynamic capabilities. The serverless back-end in JAMstack provides this dynamism through serverless functions and managed API services. Serverless back-end in JAMstack typically refers to a combination of Function as a Service (FaaS) and managed services (databases, queues, storage, authentication). Instead of a single constantly running server, we have a set of small endpoints that are invoked on demand. Typical tasks for serverless functions include: authorization, data validation and normalization, integrations with payment systems, access token generation, webhook processing, generation of signed download links, as well as lightweight computations that should not be performed in the browser. The serverless back-end provides API interfaces that are consumed by the dynamic front-end. It is on the client side that data visualization, application state management, and user interaction occur. Dynamics in JAMstack is implemented either through Single Page Application (SPA) (data from API, UI in browser), or through a hybrid approach: Server-Side Rendering (SSR)/edge and partial page updates using Incremental Static Regeneration (ISR), to avoid doing a full rebuild each time. The paper presents a comparison of JAMstack and traditional server architecture (Table 1).

Table 1 – Comparison of JAMstack and traditional server architecture

Criterion	Traditional approach (monolith/SSR)	JAMstack
Content delivery	Server generates HTML for each request (SSR) or serves pages from monolith. Response is formed dynamically during request.	Front-end (HTML, CSS, JS) is delivered as pre-rendered static files via CDN. Dynamic content is loaded separately through API.
Scalability	Requires planning: vertical (more powerful server) or horizontal (server cluster + load balancer). Complexity grows with traffic.	Automatic scaling: CDN for static content (virtually unlimited) and serverless functions on demand. Works «by default».
Security	Broader attack surface: web server, database, server sessions, admin panels require protection.	Limited attack surface: static content on CDN is practically unattackable. Threats are concentrated at API level and integrated services (isolated from each other).
Updates (deploy)	Risk of downtime: server-side deployment, DB migrations, possible conflicts. Maintenance window required.	Atomic deploy: new frontend version (build) is deployed to CDN instantly, without downtime. Backend services are updated independently.
Development flexibility	Tight coupling of frontend and backend. Often requires single technology stack. Adding new service can be complex.	Complete decoupling: frontend and backend are independent. Ability to use best tools for each task and easily integrate third-party APIs.
Performance	Depends on server capacity and DB optimization. Caching is more complex to configure. First load can be slow.	Maximum speed: static content from CDN loads from server closest to user. First load is instant. Dynamic data is loaded asynchronously.
Search Engine Optimization (SEO)	Classic SSR is usually SEO-friendly «out of the box».	Achieved through SSG or Hybrid Rendering (e.g., Next.js). Ready HTML is also indexed by search engines.

Thus, JAMstack is not just a technological replacement but a paradigm shift in development, where complexity is transferred from runtime to build time, and infrastructure tasks are delegated to specialized cloud services. This makes products faster, more secure, and more adaptive. The main performance gain in JAMstack comes from the fact that most requests do not reach the application: the CDN serves files from the nearest point of presence. This reduces TTFB and stabilizes response time even during traffic peaks. Therefore, the fundamental difference of JAMstack lies not only in technical implementation, but also in the architectural approach to security, where the key principle is minimizing the constant attack surface. From a security perspective, JAMstack reduces the number of permanently open components. If there is no monolith with admin panel, sessions, templating, and direct database access, the likelihood of typical incidents such as injections or server environment compromise is lower. A critical rule is not to transfer secrets to the client. Access to private keys, tokens, and databases should only be performed in serverless functions or managed services, while only publicly available content should be delivered to the front-end. An additional advantage is isolation. Functions have minimal permissions, a short lifecycle, and access can be restricted by policies and network rules. This aligns well with the «least privilege» principle.

Table 2 – Typical risks and countermeasures in JAMstack

Risk	Where it occurs	Practical countermeasure
Secrets leakage	Front-end / repository / Continuous Integration (CI)	Secrets only in vault/variables; CI checks; rotation
Cross-Site Scripting (XSS)	Client-side rendering	Content Security Policy (CSP), HTML sanitization, strict templates, dependencies under control
Excessive function permissions	Serverless	Identity and Access Management (IAM) least privilege, role separation, separate accounts/projects
Content tampering	CDN/cache	Signed builds, integrity control, domain protection, Transport Layer Security (TLS)

Alongside security, an important characteristic of modern web applications is the ability to efficiently handle increasing load. Let's consider how JAMstack ensures scalability. Scaling in JAMstack has two layers: delivery through a CDN and computation via serverless, which scales with parallel invocations without server management. However, scalability has its limits. For «long» tasks (streams, heavy Machine Learning computations, large transactions), serverless may be unprofitable due to timeouts and cold starts. In such cases, a hybrid approach is often used: JAMstack for the interface and content, and a complex backend as a separate service.

To systematize when JAMstack is the optimal choice and when other approaches are more suitable, let's consider key use cases. JAMstack demonstrates optimal suitability for marketing websites, landing pages, blogs, and documentation, where it provides maximum speed, simple deployment, and efficient content delivery through CMS integration combined with SSG, versioning, and CDN caching. For e-commerce platforms with product catalogs and admin panels with role-based access, JAMstack is partially suitable, requiring additional API integration for shopping cart and payment functionality, as well as robust authentication and access control mechanisms, often necessitating a hybrid approach. Real-time applications such as chat systems and trading platforms can be implemented with JAMstack but with certain limitations, as they require WebSocket or streaming capabilities that typically demand a separate backend service to handle persistent connections and real-time data synchronization.

JAMstack architecture transfers the main traffic to CDN and separates the application into a static part and isolated APIs/functions, which provides consistently fast content delivery. The approach enhances security through fewer permanent services, a simpler perimeter, and the ability to build access based on the principle of least privilege. Scalability is achieved through automatic scaling of CDN and serverless components, but for long or heavy tasks it is advisable to use hybrid models. Optimal implementation of JAMstack requires discipline: proper caching, dependency control, secrets management, and clear delineation of what is done at build time, in the browser, and in the backend.

REFERENCES

1. Sotnik, S. V. Support systems for robotics: principles, algorithms and development prospects. Journal of natural sciences and technologies, 2025. – 4(2). – PP. 419-430.

2. Cherednichenko, T. & et al.. Features of automatic working time control systems. Manufacturing & Mechatronic Systems 2025: Proceedings of IX st International Conference, Kharkiv, October 25-26, 2025: Thesises of Reports, 2025. – PP. 54-57

4. Nevludov, I. Sh. & et al.. Application of artificial intelligence in additive manufacturing (3D printing). Information Technologies and Automation – 2025 / Proceedings of the XVIII International

Scientific and Practical Conference. Odessa, October 30-31, 2025. – Odessa, ONUT Publishing House, 2025. – PP. 1006-1009

5. Sotnik, S. Evaluating relational database scaling strategies in web engineering. International Conference on Advanced Trends In Radioelectronics and Infocommunications (ATRIC-2025) (May 21–22, 2025), Lviv Polytechnic Publishing House, Lviv, Ukraine, 2025. – PP. 224-228

6. Andreiev, A. S. & et al.. Computer games and Web design. Proceedings of the XVII International scientific and practical conference «Information technologies and automation – 2024», 2024. – PP. 712-714

7. Rudenko, M. & et al.. Overview of approaches to scaling relational databases in development and adaptation of web applications. Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: Тези доповідей XII Міжнародної науково-практичної конференції (10-12 грудня 2024 р., м. Запоріжжя). [Електронний ресурс] /Електрон. дані. – Запоріжжя: НУ «Запорізька політехніка», 2024. – PP. 398-402

8. Sotnik, S. & et al.. Evaluating relational database scaling strategies in web engineering. International Conference on Advanced Trends In Radioelectronics and Infocommunications (ATRIC-2025) (May 21–22, 2025), Lviv Polytechnic Publishing House, Lviv, Ukraine, 2025. – PP. 224-228

9. Andreiev, A. & et al.. “Web application security: protection against modern cyber threats” Overview of key vulnerabilities (XSS, CSRF, SQL injections), protection methods, use of HTTPS, authentication, and authorization. Manufacturing & Mechatronic Systems 2025: Proceedings of IX st International Conference, Kharkiv, October 25-26, 2025: Thesises of Reports, 2025. – PP. 66-70