

Modern web technologies – frameworks, advantages, disadvantages and optimal applications

P. SERAFIN

patryk.serafin@wat.edu.pl

Military University of Technology, Faculty of Cybernetics
Kaliskiego Str. 2, 00-908 Warsaw, Poland

Modern web applications leverage a variety of frontend and backend technologies to deliver scalable, secure, and high-performance digital experiences. This paper examines key web development frameworks, highlighting their advantages, disadvantages, and optimal use cases. Frontend frameworks such as React, Angular, and Vue.js are analyzed in terms of interactivity, modularity, and scalability, while backend solutions like Django, Node.js, Flask, and Spring Boot are evaluated for security, performance, and data management capabilities. Additionally, the paper explores technology selection criteria based on application complexity, real-time processing requirements, and scalability needs. Special attention is given to security and performance considerations, emphasizing best practices for mitigating vulnerabilities and optimizing resource efficiency. The study concludes with insights into emerging trends in web development, including microservices, serverless computing, and AI-driven applications. By providing a structured comparison of modern web technologies, this paper serves as a guide for developers and businesses seeking to make informed decisions in selecting the right stack for their specific requirements.

Keywords: Web applications, frontend frameworks, backend frameworks.

DOI: 10.5604/01.3001.0055.0854

1. Introduction

Modern web applications have become essential in today's digital landscape, providing users with interactive, scalable, and responsive platforms that cater to diverse needs. Unlike traditional software, web applications are accessed via browsers and rely on a combination of frontend and backend technologies to deliver seamless experiences. These applications range from simple informational websites to complex enterprise solutions that handle vast amounts of data and multiple user interactions simultaneously. The evolution of web technologies can be categorized into three distinct phases [3]. Web 1.0, the earliest stage, primarily consisted of static websites where users could only consume content without interactivity. These sites were built using basic HTML and CSS, offering limited engagement [26]. The transition to Web 2.0 introduced dynamic and interactive web applications powered by JavaScript, AJAX, and early content management systems [9]. This era enabled real-time updates, user-generated content, and the rise of social media platforms. Currently, the Web 3.0 paradigm leverages artificial intelligence,

blockchain technology, and decentralized applications to enhance user experiences and data security.

Selecting the right web development framework is crucial for ensuring an application's success [11]. The chosen technologies directly impact performance, scalability, security, maintainability, and development efficiency. Frontend frameworks like React, Angular, and Vue.js influence user experience and interface responsiveness, while backend technologies such as Node.js, Django, and Spring Boot dictate data processing capabilities and system reliability. Additionally, considerations such as cost, developer expertise, and business requirements must be evaluated to make informed technology choices that align with project goals [24].

As web applications continue to evolve, developers must stay updated with emerging technologies and best practices to build secure, high-performing, and scalable solutions [5]. The choice of framework and architecture determines not only the application's efficiency but also its long-term viability in an increasingly competitive digital environment.

2. Categories of Web Applications

Web applications can be categorized based on their complexity, functionality, and intended user base. Each category has distinct technological requirements that determine how they are built and maintained. Selecting the right approach depends on factors such as scalability, security, and interactivity.

Simple informational websites serve as digital brochures, providing users with static content without requiring significant interactivity [6]. Examples include personal blogs, corporate landing pages, and online portfolios. These websites are typically built using HTML, CSS, and basic JavaScript, with content often managed through CMS platforms like WordPress. Since these applications do not process dynamic user input extensively, they do not require robust backend infrastructures and can be hosted with minimal server resources [25].

More complex web applications, such as Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) systems, integrate multiple functionalities to streamline business operations [10]. These applications require advanced backend frameworks like Django, Spring Boot, or Node.js to manage large datasets and facilitate seamless multi-user interactions. They often include real-time data processing, authentication mechanisms, and API integrations to ensure a smooth workflow across different departments within an organization [20].

Security-sensitive applications, such as online banking systems and medical records platforms, demand rigorous security protocols to protect sensitive user data. Technologies like Django, Spring Boot, and ASP.NET incorporate built-in security measures, including encryption, access control, and compliance with industry regulations. These applications must implement multi-factor authentication, secure data storage, and real-time threat monitoring to prevent unauthorized access and breaches.

Single Page Applications (SPAs) provide highly interactive user experiences by dynamically updating content without requiring full page reloads [2]. Popular examples include Gmail and Trello, which rely on frontend frameworks like React.js, Angular, and Vue.js. These applications improve responsiveness and reduce server load by handling most rendering processes on the client side, ensuring a smooth and engaging user experience.

Progressive Web Apps (PWAs) bridge the gap between traditional web applications and

native mobile apps by offering offline functionality and enhanced performance [15]. Services like Twitter Lite and Starbucks PWA leverage technologies such as Service Workers and IndexedDB to enable caching and background synchronization, allowing users to access content even in limited connectivity scenarios. These applications prioritize fast loading times, making them ideal for mobile-first solutions.

Applications that handle large databases, such as social media platforms and e-commerce websites, require scalable architectures capable of processing vast amounts of user-generated data [13]. Technologies like Node.js, Spring Boot, MongoDB, and PostgreSQL are commonly used to ensure high availability and efficient data retrieval. These applications often incorporate distributed computing, cloud-based hosting, and load-balancing strategies to manage heavy traffic loads effectively.

The diverse nature of web applications underscores the importance of selecting the right technologies based on their specific use cases. Whether developing a simple informational site, an interactive business platform, or a security-critical system, understanding these categories helps guide architectural decisions and optimize performance for the target audience [16].

3. Overview of Web Development Technologies

The development of modern web applications relies on a diverse set of technologies, each serving a specific purpose in creating seamless, efficient, and scalable digital solutions. Web technologies can be broadly categorized into frontend, backend, API communication, and cloud deployment tools. Understanding these components is crucial for selecting the right stack for a given application.

Frontend Technologies

The frontend of a web application is the part that users interact with directly. It consists of markup languages, styling tools, and scripting technologies that define the visual presentation and behavior of a webpage [17], [18], [19].

- **HTML (HyperText Markup Language):** The foundation of web pages, providing structural elements such as headings, paragraphs, links, and forms.
- **CSS (Cascading Style Sheets):** Responsible for styling and layout, allowing developers to control fonts, colors, spacing, and responsiveness.

- JavaScript: A scripting language that adds interactivity, enabling features such as animations, event handling, and dynamic content updates.
- Frontend Frameworks: Frameworks such as React.js, Angular, and Vue.js help streamline frontend development by providing reusable components, state management, and efficient rendering techniques.

Backend Technologies

The backend of a web application handles data processing, authentication, and server-side logic. It ensures the smooth execution of requests between the frontend and databases [11].

- Server-Side Languages: Various programming languages, such as Python, JavaScript (Node.js), Ruby, and Java, power backend operations.
- Backend Frameworks: Django, Flask, Spring Boot, and Ruby on Rails provide structured environments for developing scalable and maintainable applications.
- Databases: Web applications use relational databases like PostgreSQL and MySQL or NoSQL solutions such as MongoDB and Firebase to store and retrieve data efficiently.
- Server Management: Web servers like Nginx and Apache facilitate request handling, caching, and security, while cloud-based solutions offer scalability and high availability.

API and Communication Protocols

APIs (Application Programming Interfaces) define the rules for communication between different software components, enabling interoperability between systems [14].

- RESTful APIs: The most common architecture for APIs, using standard HTTP methods to facilitate communication between clients and servers.
- GraphQL: A flexible alternative to REST, allowing clients to query specific data and reducing over-fetching of information.
- WebSockets: A technology that enables real-time, bidirectional communication between the client and server, commonly used in chat applications and live updates.
- gRPC: A high-performance API framework using protocol buffers, suitable for microservices and distributed systems that require efficient communication.

Cloud and Deployment Technologies

Modern web applications are often hosted on cloud platforms, leveraging containerization and continuous integration practices to ensure smooth deployment and scalability [23].

- Hosting Services: Cloud providers such as AWS, Azure, Google Cloud, Vercel, and Netlify offer scalable infrastructure for web applications.
- Containerization and Orchestration: Technologies like Docker enable applications to run in isolated environments, while Kubernetes manages containerized applications for seamless scaling.
- CI/CD Pipelines: Automated deployment and testing tools, including GitHub Actions, Jenkins, and GitLab CI/CD, streamline software development workflows and ensure rapid delivery of updates.

A well-structured technology stack incorporating these tools enables developers to build robust, maintainable, and high-performance web applications suited to modern business and user demands.

4. Frontend Frameworks and Their Use Cases

Frontend frameworks have revolutionized modern web development by providing structured approaches to building dynamic and interactive applications. These frameworks streamline the development process, improve maintainability, and enhance performance by optimizing how components render and interact with user inputs. Among the most widely adopted frontend frameworks are React.js, Angular, and Vue.js, each offering unique advantages and suited for specific use cases.

React.js (Facebook)

React.js is one of the most popular frontend frameworks, designed to develop highly interactive Single Page Applications (SPA). Its component-based architecture allows developers to build reusable UI elements, which improves maintainability and scalability. The framework utilizes a Virtual DOM, enabling efficient rendering and updates, making it an excellent choice for real-time and data-driven applications. React has a vast ecosystem and community support, ensuring continuous innovation and resource availability [12].

- Use Case: Modern, interactive SPAs such as social media platforms and real-time dashboards.

- **Advantages:**
 - Component-based architecture facilitates reusability and modular development.
 - Strong ecosystem and community-driven innovation.
 - Efficient rendering through Virtual DOM, improving performance.
- **Disadvantages:**
 - Steep learning curve for beginners due to JSX and state management concepts.
 - Requires additional libraries such as Redux for state management in complex applications.

Angular (Google)

Angular, developed by Google, is a full-fledged frontend framework primarily used for enterprise-scale applications. Unlike React, which primarily focuses on the view layer, Angular provides a complete MVC (Model-View-Controller) structure, making it suitable for applications requiring a well-structured, scalable architecture. Angular includes built-in functionalities such as dependency injection, two-way data binding, and TypeScript support, ensuring robust application development [1], [2].

- **Use Case:** Large-scale enterprise applications such as banking systems and business management software.
- **Advantages:**
 - Comprehensive framework with built-in solutions for routing, forms, and dependency injection.
 - TypeScript integration enhances maintainability and scalability.
 - Strong modularity supports structured application architecture.
- **Disadvantages:**
 - Higher complexity compared to React and Vue.js, resulting in a steeper learning curve.
 - Larger bundle sizes, which may impact performance if not optimized properly.

Vue.js

Vue.js is known for its simplicity and flexibility, offering a lightweight framework that balances ease of use and performance. It features reactive data binding and a virtual DOM, similar to React, but with a more straightforward syntax. Vue is widely adopted for smaller projects or applications that require quick deployment and easy integration with existing technologies. Its progressive framework design allows developers to incrementally adopt Vue into their projects without significant rewrites [27].

- **Use Case:** Lightweight, flexible applications such as landing pages, small e-commerce platforms, and prototyping.
- **Advantages:**
 - Simple learning curve makes it accessible for beginners and quick adoption.
 - Two-way data binding enhances reactivity, reducing the need for additional state management solutions.
 - Lightweight and optimized for performance in smaller-scale applications.
- **Disadvantages:**
 - Smaller ecosystem compared to React and Angular, resulting in fewer third-party libraries and enterprise support.
 - Less adoption in large-scale enterprise environments, making it less favored for complex applications.

Tab. 1. Comparison of Frontend Frameworks

| Framework | Best For | Key Advantage | Key Disadvantage |
|-----------|------------------|---------------------------|-----------------------------|
| React.js | SPAs, dynamic UI | Virtual DOM, ecosystem | Requires external libraries |
| Angular | Enterprise apps | Full-featured, TypeScript | High complexity |
| Vue.js | Lightweight apps | Simple and fast | Smaller ecosystem |

Choosing the right frontend framework depends on various factors, including application complexity, performance requirements, and development team expertise. React.js is ideal for highly dynamic applications with frequent UI updates, Angular excels in enterprise environments requiring robust architectural support, while Vue.js offers simplicity and speed for smaller, lightweight applications. Understanding the strengths and trade-offs of each framework helps developers make informed decisions tailored to their project needs [22].

5. Backend Frameworks and Their Use Cases

Backend frameworks play a crucial role in web application development by managing server-side logic, database interactions, and application security. Choosing the right backend framework is essential for ensuring scalability,

maintainability, and optimal performance based on the application's needs. Below is an overview of some of the most widely used backend frameworks, their advantages, disadvantages, and suitable use cases.

Node.js

Node.js is a JavaScript runtime environment that enables non-blocking, event-driven server-side programming. It is particularly well-suited for real-time applications such as chat applications, video streaming services, and Internet of Things (IoT) platforms. By allowing developers to use JavaScript for both frontend and backend development, Node.js promotes a unified development experience [21].

- Use Case: Ideal for real-time applications, microservices, and lightweight APIs.
- Advantages:
 - Asynchronous, non-blocking architecture ensures high performance in concurrent operations.
 - Large ecosystem with extensive npm package support.
 - Facilitates full-stack JavaScript development, reducing context switching.
- Disadvantages:
 - Not optimized for CPU-intensive tasks such as complex calculations and data processing.
 - Callback-based asynchronous programming can lead to code complexity.

Django (Python)

Django is a high-level Python framework known for its robust security features and rapid development capabilities. It follows the "batteries-included" philosophy, providing built-in tools for authentication, database management, and security, making it a popular choice for applications handling sensitive data, such as banking systems and healthcare platforms [7].

- Use Case: Secure, scalable applications in finance, healthcare, and content management.
- Advantages:
 - Built-in security mechanisms protect against common vulnerabilities (e.g., CSRF, SQL injection).
 - Encourages rapid development with a structured framework and built-in admin panel.
 - Well-suited for monolithic applications that require robust backend functionality.
- Disadvantages:

- Can be heavy and less flexible for microservices-oriented architectures.
- Requires more server resources compared to lightweight frameworks.

Flask (Python)

Flask is a lightweight and highly flexible microframework for Python, often chosen for building RESTful APIs and small-scale applications. Unlike Django, Flask provides minimal built-in functionality, allowing developers to customize and scale their applications as needed [28], [22], [7].

- Use Case: Suitable for small-scale applications, APIs, and microservices.
- Advantages:
 - Minimalistic and modular, allowing flexibility in application structure.
 - Ideal for rapid prototyping and API development.
 - Requires fewer dependencies, making it lightweight and efficient.
- Disadvantages:
 - Lacks built-in features, requiring external libraries for advanced functionalities.
 - Less suited for large-scale, enterprise-level applications.

Ruby on Rails (Ruby)

Ruby on Rails is a full-stack web development framework that emphasizes convention over configuration, significantly reducing development time. It is widely used by startups and businesses aiming to quickly launch Minimum Viable Products (MVPs) [4].

- Use Case: Best for startups, MVPs, and small-to-medium business applications.
- Advantages:
 - Rapid development cycle due to built-in solutions for common web development needs.
 - Follows a structured, convention-based approach, reducing the need for configuration.
 - Encourages best practices such as the MVC (Model-View-Controller) pattern.
- Disadvantages:
 - Performance limitations when handling high-scale, high-traffic applications.
 - Smaller developer community compared to Python and JavaScript frameworks.

Tab. 2. Comparison of Backend Frameworks

| Framework | Best For | Key Advantage | Key Disadvantage |
|---------------|---------------------------------|--|-------------------------|
| Node.js | Real-time apps, microservices | Asynchronous JavaScript | Not CPU-friendly |
| Django | Secure, scalable apps | Built-in security, structured architecture | Monolithic structure |
| Flask | Lightweight APIs, microservices | Flexible, minimalistic | Lacks built-in features |
| Ruby on Rails | MVPs, startups | Rapid development | Performance limitations |
| Spring Boot | Enterprise applications | Scalable, secure | High memory usage |

Spring Boot (Java)

Spring Boot is a powerful framework for developing enterprise-level applications in Java. It is designed to facilitate the creation of production-ready applications with minimal setup. Its support for microservices architecture makes it a preferred choice for large-scale, high-security applications [24].

- Use Case: Suitable for enterprise applications, banking systems, and large-scale platforms.
- Advantages:
 - Highly scalable, supporting both monolithic and microservices architectures.
 - Robust security features, making it ideal for handling sensitive data.
 - Strong community and corporate support for enterprise-grade development.
- Disadvantages:
 - Requires higher memory consumption and longer startup times.
 - More complex than lightweight frameworks like Flask and Node.js.

The choice of backend framework depends on the specific requirements of the application. Node.js excels in handling real-time interactions, Django is well-suited for security-critical applications, Flask provides flexibility for lightweight services, Ruby on Rails accelerates startup development, and Spring Boot ensures enterprise-level scalability. Evaluating these factors helps developers select the best framework for their project’s needs.

6. Choosing the Right Technology Based on Application Needs

Selecting the appropriate technology stack for a web application is a critical decision that impacts performance, security, scalability, and development efficiency. The choice of frontend and backend frameworks should align with the application’s requirements, available resources, and long-term maintainability. Several key factors must be considered to ensure optimal performance and sustainability.

Factors Influencing Technology Choice

The selection of technologies depends on multiple aspects, including performance expectations, security needs, scalability requirements, and development speed. Each application type demands a unique combination of frameworks and infrastructure [11].

- Performance Needs: Applications requiring real-time processing, such as messaging platforms and live-streaming services, benefit from Node.js due to its non-blocking architecture. On the other hand, applications focused on batch-processing large datasets, such as analytics platforms, are better suited for Django or Spring Boot, which offer structured data handling [21].
- Security Requirements: High-security applications, such as banking systems and healthcare platforms, require robust security mechanisms. Frameworks like Django, Spring Boot, and ASP.NET include built-in security features, such as authentication modules, SQL injection protection, and encryption tools, making them well-suited for handling sensitive data [7].
- Scalability Considerations: Applications expected to scale rapidly should be built using containerized and distributed architectures. Kubernetes and Docker facilitate horizontal scalability, while

backend frameworks such as Node.js and Spring Boot allow efficient management of concurrent requests, making them ideal for high-traffic applications like e-commerce and social media platforms.

- **Development Speed and Maintainability:** Startups and companies developing Minimum Viable Products (MVPs) often prioritize frameworks that enable rapid prototyping. Ruby on Rails and Django provide out-of-the-box solutions that reduce development time, while React and Vue.js offer flexible frontend frameworks that simplify UI development [4].

The table below presents an overview of recommended technologies based on application type and requirements:

Tab. 3. Comparison of Technologies for Different Use Cases

| Application Type | Recommended Technologies |
|-----------------------------------|--------------------------------------|
| Informational Website | HTML, CSS, JavaScript, WordPress |
| Blogs, News Portals | Vue.js, Django, WordPress |
| Business Applications (CRM, ERP) | Angular, React, Node.js, Spring Boot |
| Banking, Critical Applications | Django, Spring Boot, ASP.NET |
| Social Media Platforms | React, Node.js, Django |
| PWA Applications | React, Vue.js, Angular |
| Applications with Large Databases | Spring Boot, Django, Node.js |

Additional Considerations

Beyond performance and scalability, cost and community support play a role in technology selection [8].

- **Cost of Development:** Open-source frameworks such as React, Django, and Node.js help reduce licensing costs, making them attractive for startups and mid-sized businesses. Enterprise-grade solutions like Spring Boot and ASP.NET provide extensive documentation and corporate support, ensuring long-term sustainability.
- **Community and Ecosystem:** Frameworks with large developer communities, such as React and Django, offer extensive third-party libraries and ongoing improvements. The availability of open-source

contributions enhances development efficiency, reducing the need for custom implementations.

Choosing the right technology requires a balance between current requirements and future scalability. Understanding the strengths and limitations of different frameworks allows developers to create optimized, secure, and maintainable web applications tailored to specific business needs.

7. Security and Performance Considerations

Ensuring both security and performance in web applications is critical for maintaining user trust, protecting data integrity, and optimizing system efficiency. Modern web applications face growing security threats and increasing demands for responsiveness, making it necessary to implement robust security measures alongside performance-enhancing strategies.

Security Considerations

Web applications are prime targets for various cyber threats, requiring developers to adopt proactive security mechanisms to mitigate vulnerabilities.

- **SQL Injection:** A common attack that exploits vulnerabilities in database queries. Preventative measures include using Object-Relational Mappers (ORMs) such as Django ORM and Hibernate in Spring Boot, as well as implementing parameterized queries to prevent direct execution of malicious SQL commands [7].
- **Cross-Site Scripting (XSS):** This occurs when attackers inject malicious scripts into web pages viewed by other users. Frameworks like React.js prevent XSS by escaping outputs via the Virtual DOM, while Angular incorporates built-in sanitation techniques. Additionally, enforcing a strict Content Security Policy (CSP) further minimizes the risk of XSS attacks.
- **Cross-Site Request Forgery (CSRF):** This attack tricks users into executing unauthorized actions on authenticated sites. CSRF tokens, implemented in frameworks such as Django and Spring Security, help verify the legitimacy of user requests and prevent unauthorized execution.
- **Authentication & Authorization:** Secure authentication mechanisms, such as OAuth2 and JWT (JSON Web Tokens), ensure that only authorized users access protected resources. Session-based authentication also

plays a key role in maintaining persistent user sessions with controlled access rights.

Performance Optimization Techniques

High-performing web applications require strategies that optimize speed and responsiveness while ensuring efficient resource utilization [13].

- **Efficient Caching:** Implementing caching techniques, such as Redis and Memcached, reduces database queries and improves response times by storing frequently accessed data in memory.
- **Lazy Loading:** Loading content only when needed improves initial page rendering speeds, particularly beneficial for image-heavy and multimedia applications.
- **Database Indexing:** Proper indexing in relational databases like PostgreSQL and MySQL enhances query performance by reducing lookup times, while NoSQL databases such as MongoDB leverage indexing for faster data retrieval.
- **Code Minification & Bundling:** Tools like Webpack and Terser optimize frontend performance by reducing file sizes and consolidating resources.
- **Server-Side Rendering (SSR):** SSR, used in frameworks like Next.js and Nuxt.js, enhances perceived performance by preloading content on the server, making applications more responsive and SEO-friendly.

Scalability Strategies

As applications grow in complexity and user base, scalability becomes crucial to maintaining efficiency and availability under increased loads [23].

- **Containerization:** Technologies such as Docker and Kubernetes enable applications to scale horizontally by running multiple instances efficiently across distributed environments.
- **Load Balancing:** Traffic distribution tools like Nginx, HAProxy, and AWS Load Balancer help manage requests across multiple servers, preventing bottlenecks and ensuring high availability.
- **Microservices Architecture:** Decomposing applications into smaller, independent services improves maintainability and scalability by allowing individual components to be scaled or updated without affecting the entire system.

Security & Performance Trade-Offs

Achieving both optimal security and performance often involves trade-offs, as stronger security measures can introduce additional processing overhead. Encryption protocols, for instance, enhance data protection but may slow down request processing. Similarly, performance-driven optimizations, such as aggressive caching, must be balanced against potential security risks, ensuring that outdated or unauthorized data is not served to users.

A well-balanced approach to security and performance ensures that web applications remain both resilient against threats and responsive under high workloads, ultimately enhancing user experience and system reliability.

8. Conclusion

The rapid evolution of web development technologies has led to the availability of a wide range of tools and frameworks designed to meet varying application needs. The choice of an appropriate technology stack is instrumental in ensuring optimal performance, security, and scalability, making it a critical consideration for developers and businesses alike.

Summary of Key Findings

Modern web applications require a combination of frontend and backend technologies that work in harmony to deliver seamless user experiences. Different frameworks cater to distinct use cases: frontend frameworks like React, Angular, and Vue.js offer dynamic and interactive user interfaces, while backend technologies such as Django, Spring Boot, and Node.js ensure efficient server-side logic and data processing. The selection of a framework depends on multiple factors, including security requirements, scalability needs, development speed, and long-term maintainability.

Applications that prioritize real-time interactions often benefit from frameworks like Node.js, which excels in handling concurrent connections. Meanwhile, highly secure applications, such as banking and healthcare platforms, require robust frameworks like Django and Spring Boot, which integrate advanced security mechanisms. Scalable architectures, particularly in large-scale enterprise environments, rely on microservices and containerization solutions such as Docker and Kubernetes.

Future Trends in Web Development

Web development continues to evolve, driven by advancements in cloud computing, artificial intelligence, and cybersecurity. Several key trends are shaping the future of web applications:

- Serverless Architectures: The adoption of AWS Lambda, Google Cloud Functions, and Azure Functions is enabling developers to build cost-effective, scalable applications without the need for managing traditional server infrastructure.
- AI and Automation: The integration of artificial intelligence is transforming web applications, enabling personalized user experiences, automated customer interactions, and predictive analytics.
- WebAssembly (Wasm): Providing near-native execution speeds for web applications, WebAssembly allows developers to run complex computations directly in the browser, improving performance-intensive applications such as gaming and data visualization.
- Increased Focus on Security: As cyber threats continue to rise, web development practices are increasingly incorporating enhanced security measures such as Zero Trust architecture, stricter authentication protocols, and end-to-end encryption.

Final Thoughts

Selecting the right web technology is fundamental to the success of any application. A well-balanced approach that considers performance, security, and scalability enables developers to build efficient and resilient web applications. As web technologies continue to evolve, staying informed about emerging trends and best practices will be crucial in developing modern, high-quality web applications that meet the growing demands of users and businesses alike.

9. Bibliography

- [1] Angular, <https://v17.angular.io/docs> (Accessed: 11 Feb. 2025).
- [2] Angular SPA: Why Single Page Applications?, Angular University Blog, 20 Apr. 2017, <https://blog.angular-university.io/why-a-single-page-application-what-are-the-benefits-what-is-a-spa/>, (Accessed: 11 Feb. 2025).
- [3] Ardalis, Common Client-Side Web Technologies, Learn.microsoft.com, <https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-client-side-web-technologies> (Accessed: 11 Feb. 2025).
- [4] Price E., Aroraa G., *HANDS-on. MICROSERVICES with C# 8 and .NET CORE 3*, Packt Publishing, 2020.
- [5] Barolli L., *Advances in Internet, Data & Web Technologies*. Springer, 2025.
- [6] CSS-Tricks, <https://css-tricks.com> (Accessed: 11 Feb. 2025).
- [7] Django, <https://docs.djangoproject.com/en/5.1/> (Accessed 11 Feb. 2025).
- [8] „15 Best and Most Popular CMS Platforms in 2025 (Compared)”, <https://www.wpbeginner.com/showcase/best-cms-platforms-compared/> (Accessed: 11 Feb. 2025).
- [9] Elliott E., *Programming JavaScript Applications*, O’Reilly Media, Inc., 2014.
- [10] Four Architecture Choices for Application Development in the Digital Age, <https://www.ibm.com/cloud/blog/four-architecture-choices-for-application-development> (Accessed: 11 Feb. 2025).
- [11] Fowler M., *Patterns of Enterprise Application Architecture*. Boston, Mass. ; Munich, Addison-Wesley, 2002.
- [12] Getting Started – React, Reactjs.org, 2024, <https://legacy.reactjs.org/docs/getting-started.html> (Accessed: 11 Feb. 2025).
- [13] Goniwada S.R., *Cloud Native Architecture and Design: A Handbook for Modern Day Architecture and Design with Enterprise-Grade Examples*, Apress, 2021.
- [14] Introduction to the server side, https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/First_steps/Introduction (Accessed: 11 Feb. 2025).
- [15] Kaushik G., “Progressive Web App – the Future of Web Development”, *International Journal for Research in Applied Science and Engineering Technology*, vol. 7, no. 7, 495–498 (2019).
- [16] Khalifa Z., “SPA, MPA, or a Hybrid?”, *The Startup*, <https://medium.com/swlh/spa-mpa-or-a-hybrid-42fdf6b3415c> (Accessed: 11 Feb. 2025).
- [17] MDN Contributors, JavaScript, <https://developer.mozilla.org/en-US/docs/Web/javascript>, (Accessed: 11 Feb. 2025).
- [18] Minnick J.L., *Responsive Web Design with HTML5 & CSS*, Cengage, 2020.

- [19] Minnick J., *Web Design with HTML5 & CSS3: Comprehensive*, Cengage Learning, 2016.
- [20] MuleSoft, What Is an API? (Application Programming Interface?), <https://www.mulesoft.com/api/what-is-an-api> (Accessed: 11 Feb. 2025).
- [21] Node.js Foundation, Docs | Node.js, <https://nodejs.org/en/docs/> (Accessed: 11 Feb. 2025).
- [22] Patel J., “List of 10 Best Web Frameworks for Web App Development in 2025”, <https://www.monocubed.com/blog/most-popular-web-frameworks/> (Accessed: 11 Feb. 2025).
- [23] Ruecker B., *Practical Process Automation*. O’Reilly Media, Inc., 2021.
- [24] Spring, “Spring Projects”, <https://spring.io/projects/spring-boot> (Accessed: 11 Feb. 2025).
- [25] State of JavaScript, <https://stateofjs.com/en-US> (Accessed: 11 Feb. 2025).
- [26] Structuring Content with HTML – Learn Web Development, https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Structuring_content (Accessed: 11 Feb. 2025).
- [27] “Introduction”, <https://vuejs.org/guide/introduction.html> (Accessed: 11 Feb. 2025).
- [28] Flask, <https://flask.palletsprojects.com/en/stable/> (Accessed: 11 Feb. 2025).

Nowoczesne technologie internetowe – ramy, zalety, wady i optymalne zastosowania

P. SERAFIN

Nowoczesne aplikacje internetowe wykorzystują różnorodne technologie frontendowe i backendowe w celu zapewnienia skalowalnych, bezpiecznych i wysokowydajnych rozwiązań cyfrowych. Niniejsza praca analizuje kluczowe frameworki wykorzystywane w procesie tworzenia aplikacji internetowych, podkreślając ich zalety, wady oraz optymalne scenariusze zastosowania. Frameworki frontendowe, takie jak React, Angular i Vue.js, są oceniane pod kątem interaktywności, modułowości oraz skalowalności, natomiast technologie backendowe, w tym Django, Node.js, Flask oraz Spring Boot, są analizowane w kontekście bezpieczeństwa, wydajności oraz możliwości zarządzania danymi. Ponadto, opracowanie uwzględnia kryteria wyboru technologii w zależności od złożoności aplikacji, wymagań dotyczących przetwarzania w czasie rzeczywistym oraz potrzeb w zakresie skalowalności. Szczególna uwaga została poświęcona kwestiom bezpieczeństwa i wydajności, w tym najlepszym praktykom pozwalającym na minimalizację podatności systemów oraz optymalizację wykorzystania zasobów. Praca kończy się analizą najnowszych trendów w rozwoju aplikacji internetowych, obejmujących architekturę mikroserwisową, przetwarzanie bezserwerowe oraz zastosowanie sztucznej inteligencji. Dzięki strukturalnemu porównaniu współczesnych technologii webowych, niniejsze opracowanie stanowi przewodnik dla programistów i przedsiębiorstw, umożliwiając podejmowanie świadomych decyzji w zakresie doboru odpowiedniego stosu technologicznego do określonych wymagań.

Słowa kluczowe: Aplikacje webowe, frameworki frontendowe, frameworki backendowe.