Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 5, Number 1, May 2025
https://doi.org/10.47709/brilliance.v5i1.5971

Trends and Best Practices in API-Based Web Development Using Laravel and React

Fauzan Prasetyo Eka Putra[1], Reynal Widya Efendi[2*], Alief Badrit Tamam[3], Walid Agel Pramadi[4]
[1,2,3,4] Teknik Informatika, Universitas Madura, Pamekasan, Indonesia
[1]prasetyo@unira.ac.id, [2]reynalwidya91@gmail.com, [3]aliefbadrittamam@gmail.com, [4]walidagel99@gmail.com

open access

***Corresponding Author**

Article History:
Submitted: 22-05-2025
Accepted: 29-05-2025
Published: 16-06-2025

Keywords:
Laravel; React; API; WEB Architecture; Single Page Aplication.

ABSTRACT

Modern web development is undergoing a paradigm shift marked by the increasing adoption of API-based architectures that decouple frontend and backend responsibilities. This separation enables the development of modular, scalable, and maintainable applications, particularly through the combination of Laravel and React. This study aims to explore current trends and best practices in API-driven web application development by conducting a systematic literature review of academic publications and technical documentation published between 2022 and 2024.

Laravel, a robust PHP backend framework, provides a powerful foundation for implementing RESTful APIs, while React, a declarative JavaScript library, enhances user interface responsiveness through efficient state management and component-based architecture. The review identifies several best practices: implementing RESTful principles, leveraging React's useContext and Redux for state handling, and maintaining well-structured and versioned API documentation using tools like Swagger. Integration between the two technologies offers high flexibility but presents technical challenges such as handling secure authentication (e.g., using Laravel Sanctum or Passport), managing asynchronous data flow, and configuring Cross-Origin Resource Sharing (CORS) policies correctly.

Furthermore, emerging trends such as headless CMS, Single Page Applications (SPA), Server-Side Rendering (SSR), and microservices architecture have gained traction in enhancing performance, SEO, and development scalability. These trends reflect a broader movement toward decoupled and distributed systems. By synthesizing these findings, this study provides strategic insights for developers, educators, and researchers to better navigate the evolving landscape of web application development using Laravel and React in an API-centric ecosystem.

INTRODUCTION

The development of web technologies underwent a major transformation in the last decade with the increasing adoption of API-based architectures. These architectures allow the separation between the frontend and backend, providing great flexibility in system design, development, and maintenance. Laravel, as a modern PHP framework, and React, a frontend library of JavaScript, became the dominant combination in building responsive and scalable web applications (Hasanuddin, Asgar, & Hartono, 2022)(Sinlae, Irwanda, Maulana, & Syahputra, 2024).

In this context, the concept of separation of concerns gains great significance. By making APIs the main communication interface, developers can utilize the advantages of each technology without having to be directly bound (Khalfallah, 2024). Laravel provides a robust and efficient backend, while React presents a dynamic user interface. This architecture is able to increase maintainability and agility in large-scale projects (Herdiyatmoko, 2022).

Amidst the adoption of API-driven models, Single Page Application (SPA) and Headless Architecture approaches are becoming two major trends. SPA enables a smoother user experience by dynamically loading data without page reloading, while headless allows the backend and frontend to develop independently. According to Shah (2024), these approaches are becoming the preferred choice in the design of complex enterprise systems as they are able to accommodate omnichannel needs (Rueckauer et al., 2022)(Shah, 2024).

However, best practices in using Laravel and React in the context of APIs are still being explored. Several studies emphasize the importance of implementing RESTful API standards, using middleware for authorization, and implementing efficient state management in React. Good API documentation, such as using Swagger/OpenAPI, is a crucial element in team development.

Technical challenges also arose, especially in the integration between Laravel and React. One common issue was CORS (Cross-Origin Resource Sharing), as well as synchronization between React's lifecycle and asynchronous data from the API. In addition, performance optimization is a major concern when implementing server-side rendering (SSR)

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 5, Number 1, May 2025
https://doi.org/10.47709/brilliance.v5i1.5971

or static site generation (SSG), as noted by Ribeiro (Reid, 2024).

On the other hand, the transformation to a microservices-based architecture encourages the development of systems that are modular and easy to scale. Laravel is starting to be adopted as a micro-backend in larger systems, while React is used in the context of micro frontends. states that this architectural composition is well suited for organizations that embrace DevOps and Continuous Deployment approaches.

The adoption of SSR and SSG approaches is also gaining attention as they can improve the initial performance and SEO of web applications. React, through frameworks like Next.js, enables efficient implementation of SSR. While Laravel through Inertia.js or Laravel Livewire can offer hybrid SSR with a PHP-native approach. This practice supports modern needs for progressive enhancement and speed of access (Ardiyanto & Ardhianto, 2024)(H. P. Putra & Sari, 2024).

By reviewing various recent publications, this research aims to present a comprehensive summary of the trends, challenges, and best practices in developing Laravel and React-based applications within the framework of modern API architecture. This research is expected to provide valuable insights for practitioners and academics who want to adopt scalable, flexible, and future-oriented architectural approaches (Fauzan, 2024)(Prihantari, 2024).

LITERATURE REVIEW

In modern software architecture, the separation of frontend and backend components has become a fundamental practice. This decoupling enables parallel development between teams managing the user interface and those handling business logic and data storage. It results in a more modular, scalable, and maintainable application. This architectural approach supports contemporary paradigms such as Single Page Applications (SPAs), microservices, and cross-platform integrations—all essential in today's fast-evolving digital ecosystem (Hanggara, Nasrullah, & ..., 2024).

Moreover, frontend-backend separation enhances technological flexibility and security. For instance, frontend development can utilize frameworks like React or Angular, while backend logic may be implemented in Node.js, Python Flask, or Java Spring. this architecture streamlines development and improves user experience through efficient API interactions. Similarly, emphasize that a co-development strategy for frontend and backend fosters long-term maintainability and facilitates agile iterations in enterprise systems (Zahrani, 2024).

Core Concepts of API Architecture and HTTP Protocol

In modern web development, the Application Programming Interface (API) plays a crucial role in enabling communication between client-side and server-side systems. APIs abstract the complexity of server operations and allow frontend applications to perform essential functions like data retrieval, updates, and deletion without direct interaction with databases. RESTful APIs, in particular, follow specific architectural principles that promote scalability, simplicity, and stateless interactions through standardized HTTP methods such as GET, POST, PUT, and DELETE (Hadinata & Stianingsih, 2024).

RESTful API design aligns seamlessly with both Single Page Applications (SPAs) and Server-Side Rendering (SSR) architectures.) REST APIs empower SPAs to efficiently update the user interface with minimal server interaction, thus enhancing responsiveness. Conversely, SSR can benefit from REST by delivering pre-rendered HTML content while still supporting dynamic client-server communication. RESTful communication—when optimized with HTTP caching, status codes, and payload minimization—provides significant performance gains in both client-heavy SPAs and SSR implementations, ensuring a smooth user experience across various network conditions.

Laravel as a Backend API Framework

Laravel has evolved significantly since its initial release, growing into one of the most widely adopted PHP frameworks for backend API development. Built on the principles of simplicity, elegance, and convention over configuration, Laravel offers a comprehensive suite of tools for rapid development of RESTful APIs. Features like Eloquent ORM streamline interaction with databases through object-relational mapping, while expressive routing and API Resource classes provide structured and reusable formats for JSON responses(Hanif Pratama, 2024).

Comparative studies show that Laravel offers greater efficiency and maintainability than native PHP or even other frameworks. Laravel's capability to handle microservices architectures with clear routing and modular structure, superior performance in CRUD operations and scalability in API-based applications. Additionally, Ariyanto & Rachmad (2024) provide a direct comparison between Laravel and native PHP, concluding that Laravel significantly improves developer productivity, code maintainability, and system performance in REST API development(Ariyanto, Farhan, Rachmad, & Puspitasari, 2024a).

Implementation of MVC Architecture

The Model–View–Controller (MVC) pattern divides application logic into three interconnected components. The Model handles data and business logic, the View manages UI presentation, and the Controller serves as the intermediary between them. This separation promotes clean, testable, and maintainable code(Suhendang, Pradypta, & ..., 2024).

In Laravel, MVC is integral to its structure. Models are implemented with Eloquent ORM, Views use Blade

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 5, Number 1, May 2025
https://doi.org/10.47709/brilliance.v5i1.5971

templating, and Controllers define route logic and API responses. In contrast, React—though not MVC in the traditional sense—uses component-based architecture where state and rendering logic reside together. This functional approach simplifies UI state management but requires explicit separation when working with data layers and routing, often addressed using tools like Redux and React Router(Kumar, 2024).

Authentication and Middleware in Laravel

Laravel offers multiple authentication strategies, including token-based mechanisms like Sanctum and Passport. Sanctum supports SPA authentication using cookies, while Passport uses OAuth2 for robust API security. Middleware functions in Laravel serve as request filters, enabling route protection, role checking, and session validation before controller logic executes(Sahu, 2024).

Security in web APIs often compares JWT-based and session or cookie-based authentication. JWTs offer stateless, scalable auth—especially suitable for distributed systems and mobile clients. Cookie-based authentication, on the other hand, leverages browser-native behavior for session persistence and CSRF protection, ideal for SPAs using Laravel Sanctum(Антонов, Вуколов, & Кононыхина, 2024).

Microservices in Modern Web Architecture

Microservices architecture decomposes applications into loosely coupled services, each responsible for a specific domain. This design promotes independent deployment, scalability, and improved maintainability. Laravel, being lightweight and modular, can act effectively as a micro-backend, exposing APIs that integrate with larger service ecosystems(Baehaqi, Basit, Indrajit, & ..., 2023).

React complements this architecture on the frontend via microfrontend strategies, where independent UI components are served separately and assembled at runtime. This decoupling allows teams to develop and deploy features autonomously, supporting enterprise-scale frontend systems(H. Kurniawan, Syafa'at, Budihartono, Lorosae, & ..., 2023).

React–Laravel Synchronization and Common Issues

When integrating React with Laravel APIs, technical challenges often emerge. Cross-Origin Resource Sharing (CORS) issues arise when frontend and backend reside on different origins. Laravel handles this using the barryvdh/laravel-cors package or native middleware settings(F. Kurniawan, Sitorus, Putra, & Afrizal, 2024).

Other integration pain points include asynchronous data fetching during lifecycle events, proper token handling, and ensuring secure client-server communication. Axios and React Query aid in managing API calls and caching, while middleware and headers manage token validation and refresh workflows. Proper configuration of HTTP-only cookies or local storage is essential to avoid CSRF or XSS vulnerabilities (Aji, 2023).

METHOD

The research in this article uses a type of qualitative research with a literature review approach (M. R. Kurniawan, Agoestanto, & Wijayanti, 2023). This research does not involve direct experimentation on the subject, but rather reviews, examines, and analyzes various relevant literature sources to build a theoretical framework of understanding of the topic under study. This literature approach aims to evaluate and synthesize various previous research results in order to form a strong conceptual basis to support research objectives (Mitra & Taufik, 2023).

The data sources used in this research are secondary data obtained from various scientific documents, national and international journals, academic books, and previous research results. The author filters and selects literature sources that are relevant to the topic of discussion in order to maintain the validity and reliability of the information used. The criteria for selecting sources include the relevance of the topic, the reputation of the publisher, and the time span of publication, which is focused on scientific papers in the last three years to ensure that the data used is up-to-date (Abidin, Mukhlis, & Zagladi, 2023).

Data collection techniques in this literature study were carried out through systematic searches on scientific databases, such as Google Scholar, ResearchGate, and national and international accredited journals. The collection process was carried out using certain keywords that are in accordance with the research topic. After the search, data was collected, selected, and stored for later analysis. Source validation was conducted by evaluating the credibility, relevance, and contribution of sources to the topic of study (Hasibuan, Zulaikha, Sari, & ..., 2023)(Dendi, 2023).

Data analysis procedures were carried out through content analysis techniques, which aimed to identify patterns, themes, and main arguments from each reference studied. The data collected was categorized by topic, analyzed for interrelationships, and synthesized to form an integrative understanding of the research issues. This analytical approach is particularly appropriate for literature-based research because it is able to comprehensively explore the depth of meaning from multiple sources (SITI, 2024).

As this study does not involve the direct participation of individuals, there are no empirically researched human subjects or samples. However, in the context of document selection as a unit of analysis, purposive sampling was used, which is the selection of sources based on a specific purpose, namely relevance to the research problem. This technique

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 5, Number 1, May 2025
https://doi.org/10.47709/brilliance.v5i1.5971

is commonly used in literature studies as it ensures that only quality and contextually appropriate sources are analyzed in depth (Yuniar & Wolor, 2023).

With this method, the research successfully integrated a variety of previous results and perspectives to support valid and reliable conclusions. Literature studies allow researchers to build a strong theoretical foundation and develop a comprehensive analytical framework, especially in the context of conceptual or exploratory studies. Validity is obtained through rigorous source selection, while reliability is maintained through systematic and iterative analysis procedures (Hilman, 2018).

RESULT

Trends in the Use of Laravel and React in API-Driven Architectures

The Laravel framework consistently occupies an important position in modern backend API development. Literature studies show that Laravel is widely used to build RESTful APIs and implement headless architectures, separating the backend and frontend through API interfaces (Novita, 2024). On the frontend side, React is the main choice for building SPA (Single Page Applications) due to its flexibility and ability to handle asynchronous data through APIs (Prihantari, 2024). The "separation of concerns" architecture pattern has become a dominant trend, allowing for modular, scalable, and independently deployable development. In addition to being a popular choice, Laravel and React are often paired in modern application development as both support the principle of service-based architecture. Laravel provides features such as Laravel Sanctum, Passport, and API Resources that simplify API data authorization and serialization. Meanwhile, React is often combined with libraries such as Axios or React Query to manage HTTP requests and asynchronous state (Iswari, 2022)(Kristianto, 2022).

This adoption trend is supported by the increasing need for interactive and real-time applications that are client-side and server-side separated, which allows for a more efficient deployment pipeline and scaling. This architecture also facilitates cross-platform team development (backend and frontend) in parallel.

Table 1. Comparison of Laravel and React Features for API Architecture

Aspects	Laravel (Backend)	React (Frontend)
Architecture	RESTful API, Headless Backend	SPA (Single Page Application), Component-Based
Data Communication	API Resources, Eloquent, Middleware	Axios, React Query, Fetch API
Authentication & Security	Laravel Sanctum / Passport	JWT Handling, Token Storage
Scalability	Modular melalui Service Provider & Route Groups	Reusable Components, Virtual DOM
API Documentation	Laravel Swagger / Scribe	Integrasi mudah dengan dokumentasi berbasis JSON
Ecosystem Support	Composer Packages, Laravel Ecosystem	NPM Packages, React Ecosystem

The table above shows how Laravel and React complement each other in building an efficient and modern API architecture. Laravel provides a strong backend foundation through features like API Resources, Eloquent ORM, and authentication systems like Sanctum and Passport that support data security in client-server communication (Branco, Aversa, & Venticinque, 2023). API documentation support such as Scribe or Swagger also facilitates cross-team integration and endpoint validation. Laravel's ability to facilitate modular development through service providers and route groups makes it ideal for large-scale systems that require high scalability (Pradana, 2022).

On the other hand, React excels in building interactive and responsive user interfaces. With its component-based approach and asynchronous data management using Axios or React Query, React is able to deliver a seamless SPA (Single Page Application) experience. This combination reinforces the principle of "separation of concerns" that enables parallel frontend and backend development, increasing the efficiency of cross-functional teams (Iswari, 2022). The simultaneous adoption of Laravel and React not only reflects technology trends, but also responds to the needs of today's applications that demand speed, modularity, and ease of cross-platform integration (Bismoputro, Huda, & Brata, 2024).

Best Practices in API Creation with Laravel

API development with Laravel has become a top choice in various software projects due to its ability to provide a structured, efficient and maintainable architecture. Numerous scientific studies and publications in the last three years highlight the following best practices:

a) Modular Architecture and Use of MVC

Laravel implements the Model-View-Controller (MVC) architecture pattern which is helpful in separating business logic, views, and data. This makes the code more structured, easy to develop, and maintain. Comparison studies between Laravel and native PHP show that Laravel excels in code efficiency, URL routing structure, and

Brilliance
Research of Artificial Intelligence
E-ISSN : 2807-9035
Volume 5, Number 1, May 2025
https://doi.org/10.47709/brilliance.v5i1.5971

project architecture model. The use of Eloquent ORM in Laravel also facilitates interaction with the database without the need to write SQL manually, making the REST API development process more modular and scalable (Ariyanto, Farhan, Rachmad, & Puspitasari, 2024b).

b) Documentation and Development Iteration

Every stage of API development, from requirements analysis, system design, development, to testing, needs to be well documented. API documentation is very important for third parties who will utilize the service. Studies of REST API development in Indonesia emphasize the importance of documentation and the use of incremental methods, where each iteration includes requirements analysis, design (class diagram and ERD), endpoint development, and testing using tools such as Postman to ensure the API meets functional requirements (Filiana, Rini, Prabawati, & Samat, 2022).

c) Authentication and Middleware

API security is a crucial aspect. Laravel provides token-based authentication support through middleware and an integrated authorization system. International research highlights the importance of implementing API authentication and middleware to maintain data security and ensure only authorized users can access certain resources. Frameworks such as Laravel also provide ease of implementation of token-based authentication, so that APIs are more secure and maintain their integrity (Khalfallah, 2024).

d) API Testing and Evaluation

Thorough API testing is highly recommended to ensure there are no bugs or errors in each CRUD (Create, Read, Update, Delete) process. Testing is done using tools such as Postman, which is capable of sending HTTP requests to API endpoints and verifying the responses received. Empirical studies show that with a good routing and controller structure in Laravel, the testing process becomes easier and more systematic (Ariyanto et al., 2024b).

e) Migration to Microservices Architecture

In the case of complex and evolving systems, migrating from a monolithic architecture to microservices becomes a best practice. Each component of the system is broken down into small services that communicate with each other via REST APIs, making development, maintenance and scaling easier. Case studies in Indonesia show that this approach, combined with Domain Driven Design, makes it easier for new developers to understand and develop the system (Maharana & Acharya, 2024).

API Consumption Best Practices in React

API consumption in React plays a very vital role in web-based application development. Efficient interaction between the frontend and backend depends not only on the tools used to submit requests, but also on how the data is managed and presented to the user. Therefore, there are several best practices that need to be considered to ensure React applications can manage APIs effectively and responsively (R. A. Putra, 2024).

a) Selection of the Right HTTP Tool

The first thing to note is the selection of tools to perform HTTP requests. React provides two main options for API consumption: Fetch API and Axios. Fetch API is a built-in JavaScript feature that allows fetching data from the server, but it has some limitations. For example, it requires manual error handling and lacks support for global configuration settings, such as base URL or authorization header settings (Yuliadarnita, Febriansyah, Wijaya, & ..., 2023).

In contrast, Axios offers a range of additional features that are more flexible. It provides a more concise syntax, supports interceptors to handle requests before they are sent or responses after they are received, and allows global configuration of things like base URLs, authentication, and headers settings. With these advantages, Axios is widely chosen in larger and complex application projects, as it allows for more structured and maintainable management of data requests and responses (Ramai, Facciorusso, DeLuca, & ..., 2022)(Rahmadhani, Wildana, & ..., 2024).

b) Organized Data State Management

Managing the state of data retrieved from APIs is an important part of React application development. This data state must be managed in an efficient way to keep the application responsive and able to handle the data optimally. In React, there are several commonly used approaches to state management (Hesti Syafitri, Hamka, & Yusuf, 2024).

React Context is often used for small to medium-sized applications, where global state such as user information or preferences can be shared across application components without straining performance. However, for more complex applications, Redux Toolkit is becoming a popular choice. Redux Toolkit provides a clearer structure for state management by separating business logic and user interface. Redux offers full control over the data flow and ensures that the application state is always predictable (Kadriu & Bilalli, n.d.)(Kumar, 2024).

In addition, React Query is a library specifically designed for server state management or the status of data coming from the server. React Query not only simplifies the process of retrieving data from the API, but also handles things like caching, automatic data retrieval when the page is refocused, and real-time data synchronization. This approach reduces the need for boilerplate code and improves overall application

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 5, Number 1, May 2025
https://doi.org/10.47709/brilliance.v5i1.5971

performance, especially for applications that rely on large amounts of external data.

c) Clear API Response Status Handling

One element that is equally important is the handling of response status from APIs. User experience can be greatly affected by how the application provides feedback regarding data requests. Therefore, it is important to provide a clear indicator of the status of the request, whether it is loading, error, or success. For example, displaying a spinner or progress bar while data is being processed provides a clear visual indication that the application is working.

Another increasingly popular technique is optimistic UI, where the application displays the results of data changes as if they were successful, even before the server confirms. This technique is very useful for improving the perception of application speed, especially in applications with high latency such as e-commerce applications or real-time-based applications. By using this technique, users do not feel that the application is slow even though the requested data is still in the process of being retrieved.

d) Configuration Management with .env

In addition to the technical aspects above, it is also important to manage configuration variables related to the API separately from the source code. One of the most common ways is to use .env files, which allow the management of environment variables in a structured way. This file is used to store sensitive information such as the API base URL, authorization tokens, and other configuration parameters. For example:

```
REACT_APP_API_BASE_URL=https://api.example.com
REACT_APP_API_KEY=abcdef123456
```

With this approach, developers can easily change application configurations when moving between different environments (e.g. development, staging, and production) without the need to change program code. In addition, the use of .env files also increases application security as sensitive information is not included directly in publicly accessible code.

Overall, API consumption in React involves more than just requesting data from a server. It involves choosing the right tools, efficiently managing data state, and implementing techniques to improve user experience, such as clear response state handling and optimistic UI. In addition, configuration management with .env files provides the flexibility and security required in the development of scalable and manageable applications.

By adhering to these best practices, developers can build React applications that are not only efficient and responsive, but also secure and easy to maintain in the long run.

With this style, the material flows more in the form of easy-to-follow text, but still contains structured elements so that the reader is not confused. If you need further explanation or additional code examples, just let me know!

DISCUSSION

Common Challenges Found in the Literature

In the integration process between React as a frontend framework and Laravel as a backend framework, the most common technical challenge encountered is the Cross-Origin Resource Sharing (CORS) issue (Kumaladewi, Refardi, & ..., 2023). React, which by default runs on localhost on a different port than Laravel, requires access to cross-domain APIs. If CORS headers such as Access-Control-Allow-Origin, Access-Control-Allow-Methods, and Access-Control-Allow-Headers are not explicitly configured on Laravel, then HTTP requests from React will be blocked by the browser. This problem can be solved through Laravel middleware or CORS configuration in the cors.php file, but often misconfiguration causes developers difficulties in the initial phase of integration (Astowo, 2024)(Purnomo, 2024).

From the React side, the challenge arises in how the application handles asynchronous processes. React utilizes hooks such as useEffect to execute processes such as retrieving data from the API. However, without sufficient understanding of the component lifecycle and re-rendering principles, novice developers are prone to race conditions. For example, when two asynchronous processes run concurrently without proper control, the incoming data can get out of sync with the application state. Therefore, a clear state management strategy is needed, such as the use of useReducer, global context, or libraries like Redux to handle data flow more stably (Yuanita, Wijayanto, & ..., 2022)(Pramadipta, 2024).

Brilliance
Research of Artificial Intelligence
E-ISSN : 2807-9035
Volume 5, Number 1, May 2025
https://doi.org/10.47709/brilliance.v5i1.5971

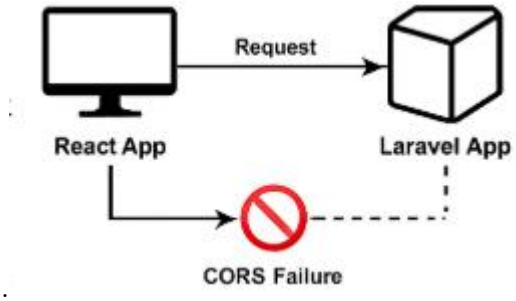

Fig. 1 Illustration of CORS Failure

Authentication synchronization is one of the most crucial areas in fullstack development. Laravel provides authentication solutions such as Sanctum and Passport that rely on tokens based on HTTP-only cookies or Bearer Tokens (Ratino, Astri, & Anggraini, 2023). However, from the React side, developers must ensure tokens are stored securely, for example by avoiding storing them in localStorage which is vulnerable to XSS attacks. Challenges also arise when managing token refresh, session expiration, and user login status in real-time, which demands the implementation of coordinated authentication logic between the two systems (Fernandes, 2024)(Ray, 2025).

Emerging Architectural Trends

As the need for flexibility and scalability in modern web application development increases, monolithic architectures are being replaced by more modular architectural approaches. One prominent trend is the shift towards headless architecture, where the backend and frontend are completely separated. Laravel, in this context, is no longer used as an all-in-one platform, but rather as a backend service that is only tasked with providing REST or GraphQL-based APIs (Basatha et al., 2024)(Arif, 2024). The frontend is developed independently using React, allowing the development of a more dynamic user interface and can be developed in parallel.

The microservices approach is also a major concern in modern architecture. Instead of building the system as one big unit, developers now prefer to break the system into small services that communicate with each other through APIs. With Laravel acting as one service, the system becomes easier to maintain, deploy separately, and scale as needed. This implementation is often combined with the use of Docker, orchestrators such as Kubernetes, and CI/CD pipelines to support the continuous development process (Aprilia & Mulianingtyas, n.d.).

Meanwhile, the use of GraphQL as an alternative to the REST API is also gaining popularity among React developers. GraphQL allows clients to specify what data they want to retrieve in the REST API.

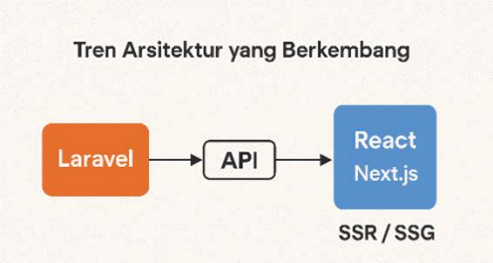

Fig. 2 Illustration of Emerging Architectural Trends

The results of this literature review indicate that the integration of Laravel as a backend framework with React as a frontend interface is becoming an increasingly widely adopted approach in the development of API-based web applications. Laravel provides a solid architecture for managing business logic, authentication, and creating RESTful and GraphQL APIs in an efficient and structured manner (Wardani, Arifianto, Sirojudin, Aziza, & Habibie, 2024)(Faturahman, Putra, & ..., 2022). On the other hand, React supports the creation of dynamic and responsive user interfaces with the principle of separate and reusable components. This combination reflects the adoption of modern architecture based on decoupled system and separation of concerns, which enables parallel development of frontend and backend and increases system scalability (Pohan, 2024)(Dawis, Rahmayanti, Rachman, Impron, & ..., 2025).

This finding is consistent with several previous studies that emphasize Laravel's advantages in terms of security, API documentation, and ease of integration with middleware for authorization. Meanwhile, React proved effective in state management, asynchronous data processing, and interface rendering optimization through the SPA (Single Page Application) approach. This research extends these findings by incorporating recent insights, including headless architecture implementation practices, the application of microservices, and the utilization of supporting technologies such as Axios, React Query, and Next.js as part of a modern ecosystem (Aji, 2023)(Sanjaya, 2025).

The simultaneous adoption of Laravel and React is inseparable from the industry's need for information systems that are modular, efficient, and easy to maintain. Laravel supports the principle of convention over configuration which

E-ISSN : 2807-9035
Volume 5, Number 1, May 2025
https://doi.org/10.47709/brilliance.v5i1.5971

speeds up the development process, while React offers great flexibility in building user interfaces integrated with APIs in real-time. In addition, this approach also supports modern DevOps practices, such as integration with Docker and implementation of CI/CD (Continuous Integration/Continuous Deployment), which are increasingly relevant in large-scale and continuous software development (Zebua, 2024)(Frabroyir, Kom, & Hasanah, n.d.).

The implications of these findings suggest that the combination of Laravel and React can provide significant work efficiency in the software development process, as well as support service-oriented system architecture. However, there are some technical challenges that need to be considered, including CORS (Cross-Origin Resource Sharing) configuration, secure authentication token management (such as JWT), and complex frontend data state management, especially in asynchronous data communication scenarios (Shofyan & Isa, 2024)(Astowo, 2024).

The limitation of this study lies in its secondary literature-based approach, without being supported by empirical studies or implementation experiments. Therefore, while the results provide a comprehensive overview of best practices and technology trends, their validity still depends on the quality and depth of the sources analyzed. In addition, most of the literature used is from the local development context (Indonesia), so generalization of the results to the global context needs to be done carefully.

Overall, this study presents a holistic analytical framework for the use of Laravel and React in API-based application development, and confirms the relevance of their combination in meeting the demands of modern software architecture. It is hoped that the results of this study can serve as a theoretical foundation for further research development, as well as a practical reference for developers in building adaptive, modular, and efficiency-oriented information systems.

CONCLUSION

The results of this literature study show that the integration between Laravel as a backend and React as a frontend in API-based application development provides an efficient and modular approach to building modern information systems. Laravel makes it easy to manage business logic, security, and build a neat and structured API architecture. React, on the other hand, enables the development of interactive and responsive user interfaces with efficient state management through a component approach. The combination of the two provides advantages in terms of separation of concerns, ease of system scalability, and support for integration with third-party services.

However, this combined implementation also has some disadvantages, such as initial complexity in configuration, especially regarding CORS settings, JWT token authentication, and the need for a deep understanding of both technologies simultaneously. In addition, poorly organized data communication between the frontend and backend can lead to inconsistencies or vulnerabilities in the system. Nonetheless, this approach offers great opportunities for continuous system development that is adaptive to future needs.

Further development possibilities include integration with other supporting technologies such as GraphQL for data efficiency, as well as the use of Docker to improve the portability and consistency of the development environment. To overcome the limitations of this research, which is literature-based, further research in the form of direct implementation in real environments, such as software project case studies or prototype system experiments, is recommended. Such research can assess the performance, security, ease of maintenance, and level of user satisfaction in using an integrated Laravel and React-based system.

REFERENCES

Abidin, H., Mukhlis, I., & Zagladi, A. N. (2023). Multi-method Approach for Qualitative Research: Literature Review with NVivo 12 PRo Mapping. *Kalam Cendekia: Jurnal Ilmiah ...*. Retrieved from https://jurnal.uns.ac.id/jkc/article/view/80748

Aji, B. S. (2023). *2023 Complete Front-End Engineer Career With ReactJS Di PT Marka Kreasi Persada*. repository.unisbablitar.ac.id. Retrieved from https://repository.unisbablitar.ac.id/id/eprint/226/1/Aji Budi Santoso TI 2020.pdf

Aprilia, A., & Mulianingtyas, O. (n.d.). ORKESTRASI CONTINUOUS INTEGRATION/CONTINUOUS DELIVERY (CI/CD) DAN AUTOMATED TESTING PADA DEVOPS MARKETPLACE .... *Journal.Uinsi.Ac.Id*. Retrieved from https://journal.uinsi.ac.id/index.php/DiJITAC/article/download/7368/2734

Ardiyanto, R., & Ardhianto, E. (2024). Analisa Performasi Metode Client Side Rendering, Server Side Rendering, dan Incremental Static Regeneration dalam Proses Website Rendering. *Computer Science (CO-SCIENCE)*. 103.75.24.116. Retrieved from http://103.75.24.116/index.php/co-science/article/view/2427

Arif, A. H. (2024). *Analisis Performa Model Protokol API REST, SOAP, GraphQL dan RPC dalam Simulasi Create, Read, Update, Delete, Sorting, dan Searching Data*. rama.unimal.ac.id. Retrieved from https://rama.unimal.ac.id/id/eprint/4772/

Ariyanto, Y., Farhan, M., Rachmad, F., & Puspitasari, D. (2024a). Issue 2 Year 2024 Pages 66-73 Jurnal Manajemen Teknologi dan Informatika. *Matrix: Jurnal Manajemen Teknologi Dan Informatika*, *14*(2), 66–73. Retrieved from https://ojs2.pnb.ac.id/index.php/MATRIX/article/view/1413

E-ISSN : 2807-9035
Volume 5, **Number** 1, May 2025
https://doi.org/10.47709/brilliance.v5i1.5971

Ariyanto, Y., Farhan, M., Rachmad, F., & Puspitasari, D. (2024b). Laravel Framework and Native PHP : Comparison in the Creation of Rest API. *Matrix: Jurnal Manajemen Teknologi Dan Informatika*, *14*(2), 66–73. Retrieved from https://ojs2.pnb.ac.id/index.php/MATRIX/article/view/1413

Astowo, U. B. (2024). *Desain Komunikasi dan Keamanan Data Arsitektur Aplikasi Multimasjid*. dspace.uii.ac.id. Retrieved from https://dspace.uii.ac.id/handle/123456789/51606

Baehaqi, A., Basit, M. S., Indrajit, R. E., & ... (2023). Front End Learning Management System Development Using The Nextjs Framework. *Jurnal Teknik …*. Retrieved from https://jutif.if.unsoed.ac.id/index.php/jurnal/article/view/1273

Basatha, R., MT, M., Chafid, N., Kom, S., Kom, M., Wihardjo, E., & ... (2024). *PEMROGRAMAN WEB DAN APLIKASI MOBILE*. books.google.com. Retrieved from https://books.google.com/books?hl=en&lr=&id=6ZI3EQAAQBAJ&oi=fnd&pg=PA41&dq=laravel+dan+react+dalam+kerangka+arsitektur+api+modern&ots=mWk4D5NZBA&sig=7E48onU_MPbHVJ5882aparC6jRI

Bismoputro, I., Huda, F. Al, & Brata, A. H. (2024). Pengembangan Single Page Application Berbasis Reactjs Untuk Usaha Percetakan Online (Studi Kasus: Global Grafika). *Jurnal Pengembangan Teknologi …*. Retrieved from https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/13985

Branco, D., Aversa, R., & Venticinque, S. (2023). A tool for creation of virtual exhibits presented as IIIF collections by intelligent agents. *International Conference on …*. https://doi.org/10.1007/978-3-031-28694-0_22

Dawis, A. M., Rahmayanti, D., Rachman, T., Impron, A., & ... (2025). *Pendekatan Modern Dalam Analisis Dan Desain Teknologi Informasi*. researchgate.net. Retrieved from https://www.researchgate.net/profile/Devi-Rahmayanti/publication/388381927_PENDEKATAN_MODERN_DALAM_ANALISIS_DAN_DESAIN_TEKNOLOGI_INFORMASI/links/6794dda3207c0c20fa5a3e31/PENDEKATAN-MODERN-DALAM-ANALISIS-DAN-DESAIN-TEKNOLOGI-INFORMASI.pdf

Dendi, D. A. R. (2023). *Analisis Bibliometrik Publikasi Ilmiah Tentang Pembayaran Bank Syariah Berbasis Data Scopus Periode 2010-2020*. repository.radenintan.ac.id. Retrieved from https://repository.radenintan.ac.id/id/eprint/23083

Faturahman, E. T., Putra, W. H. N., & ... (2022). Pembangunan Sistem Informasi Pemesanan Jasa Foto berbasis Web menggunakan REST API pada Heroe Photography. *... Teknologi Informasi Dan …*. Retrieved from http://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/11969

Fauzan, A. (2024). *RANCANG BANGUN APLIKASI DONASI BERBASIS ANDROID DENGAN BAHASA PEMROGRAMAN KOTLIN MENGGUNAKAN METODE AGILE UNIFIED PROCESS …*. repository.nurulfikri.ac.id. Retrieved from https://repository.nurulfikri.ac.id/id/eprint/529/

Fernandes, J. P. (2024). *Authentication API-A SSO Authentication and Authorisation Infrastructure for Web*.

Filiana, A., Rini, M. N. A., Prabawati, A. G., & Samat, R. A. (2022). Pengembangan Rest Api Untuk Informasi Pasar Tradisional Di Kota Yogyakarta Dengan Metode Incremental. *SINTECH (Science and Information Technology) Journal*, *5*(1), 10–23. https://doi.org/10.31598/sintechjournal.v5i1.1060

Frabroyir, H., Kom, S., & Hasanah, I. (n.d.). Software House: Analisis dan Implementasi Pengembangan Perangkat Lunak Sistem UTBK di PT Aksamedia Mulia Digital. *Repository.Its.Ac.Id*. Retrieved from https://repository.its.ac.id/118926/1/5025211240-Project_Report.pdf

Hadinata, W., & Stianingsih, L. (2024). Analisis Perbandingan Performa Restfull Api Antara Express.Js Dengan Laravel Framework. *Jurnal Informatika Dan Teknik Elektro Terapan*, *12*(1). https://doi.org/10.23960/jitet.v12i1.3845

Hanggara, B. T., Nasrullah, M. H., & ... (2024). Analisis Perbandingan Performa Framework NestJS dan Lumen Pada Studi Kasus Aplikasi Berbasis REST API. *J-INTECH (Journal of …*. Retrieved from https://snatika.stiki.ac.id/J-INTECH/article/view/1354

Hanif Pratama, M. (2024). *Pengembangan Application Programming Interface Untuk Layanan Booking Service Pada Pt. Tunas Dwipa Matra Menggunakan Framework Erp Odoo* (pp. 1–95). pp. 1–95. FAKULTAS MATEMATIKA DAN ….

Hasanuddin, Asgar, H., & Hartono, B. (2022). Rancang Bangun Rest Api Aplikasi Weshare Sebagai Upaya Mempermudah Pelayanan Donasi Kemanusiaan. *Jurnal Informatika Teknologi Dan Sains*, *4*(1), 8–14. https://doi.org/10.51401/jinteks.v4i1.1474

Hasibuan, N., Zulaikha, S. R., Sari, K. P., & ... (2023). Aksesibilitas Jurnal Elektronik Gale dalam Memenuhi Kebutuhan Informasi Pemustaka di Perpustakaan Politeknik Pembangunan Pertanian Yogyakarta. *Educaniora: Journal of …*. Retrieved from https://www.educaniora.org/index.php/ec/article/view/79

Herdiyatmoko, H. F. (2022). Desain Sistem Backend Berbasis Rest Api Menggunakan Framework Laravel 7. *Skanika*, *5*(2), 136–144. https://doi.org/10.36080/skanika.v5i2.2947

Hesti Syafitri, Hamka, & Yusuf. (2024). Efektivitas Model Pembelajaran React Dalam Meningkatkan Motivasi Belajar Siswa. *BEGIBUNG: Jurnal Penelitian Multidisiplin*, *2*(1), 346–355. https://doi.org/10.62667/begibung.v2i1.70

Hilman, I. (2018). Penetapan Desa Wirausaha Dan Strategi Pengembangannya. *JIMFE (Jurnal Ilmiah Manajemen Fakultas Ekonomi)*, *3*(2), 28–36. https://doi.org/10.34203/jimfe.v3i2.644

Iswari, S. T. L. (2022). *Implementasi Mongo Db, Express Js, React Js Dan Node Js (Mern) Pada Pengembangan*

E-ISSN : 2807-9035
Volume 5, **Number** 1, May 2025
https://doi.org/10.47709/brilliance.v5i1.5971

Aplikasi Formulir, Kuis, Dan Survei Online. dspace.uii.ac.id. Retrieved from https://dspace.uii.ac.id/handle/123456789/38607

Kadriu, A., & Bilalli, S. (n.d.). Comparison of Context API and Redux Working with Complex State. *Repository.Seeu.Edu.Mk*. Retrieved from https://repository.seeu.edu.mk/sites/thesis/ThesisSharedDocs/MA_126307.pdf

Khalfallah, H. Ben. (2024). HOFA: The Path Toward Clean Architecture. *Crafting Clean Code with JavaScript and React: A …*. https://doi.org/10.1007/979-8-8688-1004-6_4

Kristianto, S. (2022). *ANALISIS DAN PENYEWAAN SISTEM INFORMASI KOST-AN DI KOTA TANGERANG*. repositori.buddhidharma.ac.id. Retrieved from http://repositori.buddhidharma.ac.id/1390/

Kumaladewi, N., Refardi, G. N., & ... (2023). Design and Build a Web-Based e-Learning System using ReactJS Framework. *2023 11th International …*. Retrieved from https://ieeexplore.ieee.org/abstract/document/10455534/

Kumar, T. (2024). *Fluent React: Build Fast, Performant, and Intuitive Web Applications*. books.google.com. Retrieved from https://books.google.com/books?hl=en&lr=&id=lI_0EAAAQBAJ&oi=fnd&pg=PT8&dq=+redux+toolkit&ots=tRp39jI62B&sig=ONDiPnHv5wgO8moZThqrcbyDk0Y

Kurniawan, F., Sitorus, Z., Putra, R. R., & Afrizal, S. (2024). *Sistem Informasi Stunting Berbasis Website: Solusi Digital Untuk Mencegah Stunting*. books.google.com. Retrieved from https://books.google.com/books?hl=en&lr=&id=FzUyEQAAQBAJ&oi=fnd&pg=PA22&dq=laravel+dan+react+dalam+kerangka+arsitektur+api+modern&ots=AoyBS6ltH8&sig=Ug4i5elXhSK4ZAmj2BWYf6NNB5o

Kurniawan, H., Syafa'at, F., Budihartono, E., Lorosae, T. A., & ... (2023). *BELAJAR WEB PROGRAMMING: Referensi Pengenalan Dasar Tahapan Belajar Pemrograman Web Untuk Pemula*. books.google.com. Retrieved from https://books.google.com/books?hl=en&lr=&id=gs3OEAAAQBAJ&oi=fnd&pg=PA31&dq=react+sebagai+frontend+framework+dan+laravel+sebagai+backend+framework&ots=7JWN1fTAP4&sig=zxqvsvSgEyUMDtjIijfWqNr7F1k

Kurniawan, M. R., Agoestanto, A., & Wijayanti, K. (2023). Systematic literature review: identifikasi kemampuan berpikir aljabar dan resiliensi matematis pada pembelajaran matematika. *Jurnal Cendekia: Jurnal …*. core.ac.uk. Retrieved from https://core.ac.uk/download/pdf/578157299.pdf

Maharana, K. C., & Acharya, S. (2024). Laravel- A Centralized framework with authentication API. *SSRN Electronic Journal*, *4*(6), 152–156. https://doi.org/10.2139/ssrn.4909110

Mitra, Y., & Taufik, T. (2023). Penerapan Model Discovery Learning (Dl) Dalam Pembelajaran Tematik Terpadu Di Kelas IV Sekolah Dasar (Studi Literatur). *E-Jurnal Inovasi Pembelajaran Sekolah Dasar*. Retrieved from https://www.academia.edu/download/100401566/4263.pdf

Novita, I. (2024). IMPLEMENTASI MODEL E-COMMERCE BERBASIS WORDPRESS PADA BENGKEL LAS TRIAN JAYA. *Jurnal Mahasiswa Ilmu Komputer*.

Pohan, H. M. (2024). DESAIN ARSITEKTUR FLEKSIBEL: MEMBANGUN LINGKUNGAN YANG RESPONSIF TERHADAP PERUBAHAN. *WriteBox*. Retrieved from http://writebox.cloud/index.php/wb/article/view/119

Pradana, L. (2022). *TA: RANCANG BANGUN WEB SERVICE API DAN DOKUMENTASI REST API WEB PORTAL UNIT KEGIATAN MAHASISWA DI POLITEKNIK NEGERI LAMPUNG*. repository.polinela.ac.id. Retrieved from https://repository.polinela.ac.id/id/eprint/2851

Pramadipta, M. B. (2024). Rancang Bangun Frontend Website Untuk Pemungutan Suara Dengan Menggunakan React.Js. *Jurnal Informatika Dan Teknik Elektro Terapan*, *12*(2). https://doi.org/10.23960/jitet.v12i2.4173

Prihantari, R. A. (2024). *RANCANG BANGUN SISTEM INFORMASI E-COMMERCE BERBASIS WEB MENGGUNAKAN FRAMEWORK SPRING BOOT PADA TOKO GIGHA STEEL*. repository.nurulfikri.ac.id. Retrieved from https://repository.nurulfikri.ac.id/id/eprint/545/

Purnomo, C. A. (2024). *PENGEMBANGAN SISTEM MONITORING PERTANIAN DALAM GREENHOUSE UNTUK MENDUKUNG SMART PRECISION FARMING 4.0: MODUL …*. digilib.uns.ac.id. Retrieved from https://digilib.uns.ac.id/dokumen/download/119489/Nzg2ODcw/Pengembangan-Sistem-Monitoring-Pertanian-dalam-Greenhouse-untuk-Mendukung-Smart-Precicion-Farming-40-Modul-Fullstack-WEB-dan-Backed-Mobile-halaman-cover.pdf

Putra, H. P., & Sari, A. P. (2024). Implementasi Server Side Rendering Pada Sistem Travel Berbasis Website. *Prosiding Seminar Nasional Informatika …*. Retrieved from https://santika.upnjatim.ac.id/submissions/index.php/santika/article/view/428

Putra, R. A. (2024). *PERANCANGAN APLIKASI LAYANAN PENGADUAN CUSTOMER BERBASIS WEB MENGGUNAKAN REACT JS: STUDI KASUS PADA PT INOVATI F 78*. repository.nurulfikri.ac.id. Retrieved from https://repository.nurulfikri.ac.id/id/eprint/511/

Rahmadhani, S., Wildana, D. W., & ... (2024). Penerapan React JS dan Axios untuk Pengembangan Front-end Aplikasi iCare. *Software …*. Retrieved from https://jurnal.poliwangi.ac.id/index.php/session/article/view/184

Ramai, D., Facciorusso, A., DeLuca, M., & ... (2022). Adverse events associated with AXIOS stents: Insights from the manufacturer and user facility device experience database. *Endoscopic …*. journals.lww.com. Retrieved from

E-ISSN : 2807-9035
Volume 5, **Number** 1, May 2025
https://doi.org/10.47709/brilliance.v5i1.5971

https://journals.lww.com/eusjournal/fulltext/2022/11030/Adverse_events_associated_with_AXIOS_stents_.9.aspx

Ratino, A., Astri, R., & Anggraini, P. (2023). Implementasi Framework Laravel Dalam Pengembangan Aplikasi E-Commerce Untuk Toko Jago Software. *Journal Of Informatics And Busisnes*, *01*, 33–43. Retrieved from https://jurnal.ittc.web.id/index.php/jibs/article/view/62

Ray, P. P. (2025). A Survey on Model Context Protocol: Architecture, State-of-the-art, Challenges and Future Directions. *Authorea Preprints*. https://doi.org/10.36227/techrxiv.174495492.22752319

Reid, R. L. (2024). Engineering the Arts. *Civil Engineering Magazine*, *94*(2), 38–49. https://doi.org/10.1061/ciegag.0001713

Rueckauer, B., Bybee, C., Goettsche, R., Singh, Y., Mishra, J., & Wild, A. (2022). NxTF: An API and Compiler for Deep Spiking Neural Networks on Intel Loihi. *ACM Journal on Emerging Technologies in Computing Systems*, *18*(3). https://doi.org/10.1145/3501770

Sahu, S. K. (2024). Building secure PHP applications: A comprehensive guide to protecting your web applications from threats. In *Building Secure PHP Applications: A Comprehensive Guide to Protecting Your Web Applications from Threats*. Springer. https://doi.org/10.1007/979-8-8688-0932-3

Sanjaya, I. (2025). *Pengembangan Aplikasi Website Informasi Instansi X untuk Efisiensi Pengambilan Data API dan Desain Antarmuka Responsif dengan Next. js*. repository.its.ac.id. Retrieved from https://repository.its.ac.id/116805/

Shah, H. (2024). *Navigating UI Engineering Architectures : A Deep Dive into Modern Web Application Design NAVIGATING UI ENGINEERING ARCHITECTURES : A DEEP DIVE*. (December). https://doi.org/10.13140/RG.2.2.30307.77603

Shofyan, S., & Isa, S. M. (2024). Perancangan Dasbor yang Secure Scalable dan Reusable dengan Microservices Case Study Di PT. XYZ. *Jurnal Sosial Teknologi*. Retrieved from http://sostech.greenvest.co.id/index.php/sostech/article/view/1251

Sinlae, F., Irwanda, E., Maulana, Z., & Syahputra, V. E. (2024). Penggunaan Framework Laravel dalam Membangun Aplikasi Website Berbasis PHP. *Jurnal Siber Multi Disiplin (JSMD)*, *2*(2), 119–132. Retrieved from https://creativecommons.org/licenses/by/4.0/

SITI, A. (2024). *ANALISIS ISI PESAN DAKWAH PADA AKUN VIDGRAM@ HALIMAHALAYDRUS*. repository.radenintan.ac.id. Retrieved from http://repository.radenintan.ac.id/34941/

Suhendang, D., Pradypta, A., & ... (2024). Sistem Arsip Berbasis Web Pelaporan Pajak Dan Laporan Keuangan Berbasis Web Menggunakan Framework Laravel. *INFORMATIKA SAINS* .... Retrieved from https://jurnal.uia.ac.id/index.php/INSIT/article/view/3646

Wardani, A. K., Arifianto, A. S., Sirojudin, A., Aziza, A. N., & Habibie, A. S. (2024). Implementasi Digital Twin Dengan Komunikasi Data Nirkabel pada Liquid Filling Machine. *JUKI : Jurnal Komputer Dan Informatika*, *6*(1), 46–54. https://doi.org/10.53842/juki.v6i1.457

Yuanita, H. I., Wijayanto, B., & ... (2022). Frontend Development Of Course Scheduling System Integrated Sia At Engineering Faculty University Of Jenderal Soedirman Using Devops Method. *Jurnal Teknik Informatika* .... Retrieved from https://jutif.if.unsoed.ac.id/index.php/jurnal/article/view/227

Yuliadarnita, Y., Febriansyah, M., Wijaya, A., & ... (2023). Analisis Komparatif Aplikasi Open Source Intelligence Berbasis Website Dengan Tools Osint Command Line Kominfo Bengkulu. *Jurnal Media* .... Retrieved from https://jurnal.unived.ac.id/index.php/jmi/article/view/3944

Yuniar, N. P., & Wolor, C. W. (2023). Analisis Arsip Pada Era Digital Di PT XYZ. *Bridging Journal of Islamic Digital* .... Retrieved from https://journal.alshobar.or.id/index.php/bridging/article/view/104

Zahrani, S. I. (2024). *SISTEM INFORMASI TOKO ROTI BERBASIS WEBSITE*. eprints.poltektegal.ac.id. Retrieved from http://eprints.poltektegal.ac.id/4379/

Zebua, M. (2024). Penggunaan DevOps untuk Meningkatkan Kecepatan Pengembangan Aplikasi. *Circle Archive*. Retrieved from http://circle-archive.com/index.php/carc/article/view/307

Антонов, С. А., Вуколов, А. А., & Кононыхина, К. А. (2024). ОБЗОР СОВРЕМЕННЫХ БИБЛИОТЕК ДЛЯ РАЗРАБОТКИ ИНТЕРФЕЙСА ВЕБ ПРИЛОЖЕНИЯ. *Вестник Науки*. cyberleninka.ru. Retrieved from https://cyberleninka.ru/article/n/obzor-sovremennyh-bibliotek-dlya-razrabotki-interfeysa-veb-prilozheniya