https://doi.org/10.62836/jitp.2025.515

Global Science Publishing

JITP

Journal of Information, Technology and Policy

https://ojs.sgsci.org/journals/jitp

Article

RESTful API Design for Geospatial Logistics Platforms Using Type Script and Laravel

Liubomyr Kaptsov

Computer Science, Odesa National University of Technology, 65039 Odesa, Ukraine

Abstract: The growing demands of logistics and supply chain management have created an urgent need for scalable, high-performance, and secure geospatial platforms capable of handling real-time data exchange. RESTful APIs have become one of the fundamental architectural solutions, as their implementation enables interoperability, modular application design, and cross-platform integration within logistics ecosystems. This research paper presents the data structure and implementation of a geospatial logistics platform based on RESTful API architecture, developed using TypeScript and Laravel. The proposed system leverages the strong typing and asynchronous processing capabilities of TypeScript, which enhance maintainability and error handling, while Laravel provides a robust backend framework for orchestrating APIs, managing authorization, and manipulating data. A relational architectural design was adopted to ensure scalability, and geospatial functionality was integrated through mapping libraries and spatial databases. The platform was evaluated using performance benchmarks, analyses of API response times, and developer surveys. The results demonstrate a significant reduction in response latency, increased request throughput, and lower error rates compared with traditional PHP-based REST frameworks. Moreover, the use of TypeScript scripts streamlined the development process, making the codebase less cumbersome and easier to maintain. The findings highlight the potential of combining modern typed programming languages with established backend frameworks to address the challenges of geospatial logistics platforms, particularly in improving the accuracy of real-time geospatial data, optimizing routes, and enhancing system scalability. This study contributes to the growing body of literature on logistics software architectures and provides practical guidance for future implementations of geospatial APIs.

Keywords: supply chain technology; type script; geospatial logistics; laravel; scalable architecture; real-time data

1. Introduction

Logistics and supply chain management has been undergoing a digital revolution over the past decade, driven by the rapid expansion of e-commerce, the internationalization of trade, and growing consumer demands for real-time delivery services. In this context, geospatial technologies have become essential tools for streamlining logistics operations by enabling asset tracking in real time, dynamic route optimization, demand forecasting, and analytics that enhance fleet and inventory management.

The integration of geospatial data into logistics platforms has become a transformative factor as organizations strive to improve efficiency, responsiveness, and supply chain resilience. It is now an operational imperative to incorporate geospatial data into heterogeneous logistics systems in a reliable, scalable, secure, and

Received: 20 September 2025; Accepted: 20 October 2025.

Corresponding: Liubomyr Kaptsov (leadnr1999@gmail.com)

interoperable manner. Traditional methods of system integration are typically not modular or flexible, nor do they provide real-time communication across distributed systems.

As a result, there is a growing demand for architectural paradigms that can address these limitations while ensuring scalability and maintainability. Representational State Transfer (RESTful) Application Programming Interfaces (APIs) have emerged as a mainstream solution for achieving interoperability and modular communication, thereby supporting cross-functionality among various components of logistics ecosystems. However, the development of RESTful APIs capable of meeting the complex requirements of geospatial logistics systems remains a significant technical challenge.

The performance impact of data heterogeneity, system bottlenecks, and the security vulnerabilities of conventional API implementations are major factors that often hinder their effective deployment. This research paper explores the use of TypeScript, an ES6-compatible statically typed superset of JavaScript, to address these shortcomings, together with Laravel, a powerful and highly structured backend framework written in PHP that offers embedded security features and flexibility. The combination of these technologies opens new horizons for the development of geospatial APIs that are not only scalable and secure but also maintainable and future-proof.

The study lies at the intersection of geospatial computing, logistics optimization, and software architecture. With particular emphasis on RESTful API design in the context of TypeScript and Laravel, the research contributes both to theoretical discourse and to practical advancements in the discipline. Ultimately, it aims to provide a framework capable of meeting the operational, technical, and strategic challenges of next-generation logistics platforms.

1.1. Problem Statement

Despite the popularity of RESTful APIs, modern logistics and geospatial systems continue to lack critical capabilities, including limited scalability, inconsistent data handling, excessive latency, and maintenance challenges. The system architecture of many existing geospatial logistics platforms relies on loosely typed scripting languages combined with monolithic backend structures, which increase quota rates, reduce performance, and limit the flexibility of open extensibility. In addition, there is a shortage of standardized architectural models specifically designed for geospatial logistics, which exacerbates integration challenges, leads to fragmented operations, and hinders real-time data transmission, operational efficiency, and informed decision-making. These shortcomings underscore the need for a powerful, modular, and scalable system capable of meeting the requirements of contemporary geospatial logistics solutions.

Previous studies have examined RESTful APIs in the context of general web development, while research on geospatial logistics systems has often been conducted in isolation. However, only a limited number of studies have explored how typed languages such as TypeScript can be systematically combined with established backend frameworks like Laravel to design geospatial APIs. Current scholarly work tends to emphasize either performance benchmarking or geospatial modeling, yet provides little insight into the architectural synergy required to maximize scalability, data consistency, and developer usability within logistics platforms. This gap highlights the necessity of conducting an empirical study that not only evaluates performance but also proposes a reproducible architectural framework for logistics applications.

The present study seeks to:

(1) Design and implement a RESTful API architecture tailored for geospatial logistics platforms using TypeScript and Laravel.

(2) Evaluate the system's performance in terms of response time, throughput, and error reduction under varied workloads.

(3) Assess the architecture's impact on developer productivity and code maintainability through structured usability feedback.

(4) Provide a blueprint for scalable and interoperable geospatial logistics platforms, bridging the gap between theoretical frameworks and practical implementations.

1.2. Evolution of RESTful API Architectures

The expansion of Representational State Transfer (REST) principles has transformed the design of web services, particularly distributed systems that require high interoperability. RESTful architectures emerged as an alternative to SOAP-based protocols, offering lightweight and flexible solutions with reduced overhead in client – server communications [1–3]. Applied to geospatial logistics, REST provides an effective mechanism for facilitating the exchange of spatial data across heterogeneous platforms, which is essential in dynamic operational environments that demand scalability and adaptability [4,5]. The stateless nature of REST simplifies server design and enables high-performance caching, both of which are critical for real-time logistics applications. Over the past decade, REST has become the predominant architectural paradigm in logistics-oriented systems due to its simplicity and support for diverse programming environments [6–8].

1.3. Integration of Geospatial Data in Logistics Platforms

Geospatial intelligence has become a critical factor in logistical planning, where route optimization, asset tracking, and demand forecasting play key roles. Traditional logistics platforms relied on relational databases and fixed spatial representations of data, which limited the ability to conduct real-time analysis [9, 10]. The integration of geospatial information systems (GIS) with logistics management systems has enabled dynamic mapping, visualization, and predictive modeling, thereby addressing some of these limitations and exposing new opportunities in logistics management [11,12]. RESTful APIs support this process by providing spatial datasets through standardized endpoints, making them accessible to third-party services, mobile devices, and decision-support dashboards. Contemporary geospatial logistics software increasingly emphasizes not only data retrieval but also the handling of streamed data, spatial analytics, and event-driven updates [13–16].

1.4. Type Script in Modern API Development

TypeScript, as a superset of JavaScript, introduces strong typing and an object-oriented programming paradigm into API development. Its static type-checking feature minimizes runtime errors and enhances maintainability, which is particularly important for large-scale logistics systems that execute complex geospatial queries [17, 18]. TypeScript's compatibility with modern frameworks, along with its extensive developer base, has established it as a language of choice for building robust client-side and server-side API components. In the context of geospatial logistics, TypeScript strengthens REST endpoints to efficiently handle large data payloads, including coordinates, routing matrices, and spatial attributes [19–21]. Furthermore, it can be seamlessly integrated with Node.js to enable high concurrency, making it well suited for applications that demand real-time responsiveness [22–24].

1.5. Laravel Framework in API-Centric Platforms

Laravel is currently one of the most widely used PHP frameworks for developing web applications, known for its elegant syntax, scalability, and modularity. It provides native support for API development, including routing, authentication, and middleware, which makes it particularly suitable for implementing RESTful applications in logistics [25 – 27]. The Laravel ecosystem, such as Laravel Passport and Laravel Sanctum, enables secure authentication in environments with multiple clients, ensuring that access to geospatial data is managed responsibly [28]. Laravel also offers a database abstraction layer (Eloquent ORM) that simplifies complex geospatial queries in logistics platforms, while its event broadcasting feature facilitates real-time location updates [29, 30]. Compared to other backend frameworks, Laravel provides a balanced trade-off between performance, security, and developer productivity [31,32].

1.6. RESTful API Applications in Geospatial Logistics

Numerous research projects highlight the application of RESTful APIs in logistics systems, ranging from vehicle tracking solutions to end-to-end supply chain management. REST APIs support interoperability across heterogeneous systems, allowing logistics firms to integrate GPS tracker data, IoT sensor readings, and mapping service information into a unified platform [33]. In geospatial logistics, REST endpoints are particularly

valuable because they enable client applications to access routing algorithms, geofencing functionality, and dynamic scheduling services. More advanced applications extend beyond simple data retrieval and include the deployment of machine learning models through APIs to forecast demand trends, thereby streamlining delivery operations [34]. The modularity of REST APIs ensures that such innovations can be integrated incrementally without disrupting the existing platform architecture [35].

1.7. Challenges in Geospatial API Design

RESTful APIs applied to geospatial applications face several limitations and challenges despite their advantages in performance, scalability, and handling complex data [36]. Spatial queries may involve datasets that are significantly larger than those of traditional database servers, requiring optimization to prevent server overload. While the stateless nature of REST improves scalability, it can also create redundancy in data transfers in logistics applications, where maintaining session state would be more efficient [37]. Furthermore, the secure design of REST endpoints for sensitive geospatial information is a critical concern, as data leaks may compromise supply chain integrity. Techniques such as rate limiting, geospatial indexing, and token-based authentication have been proposed to address these issues, but their implementation across platforms remains inconsistent [38].

1.8. Comparative Studies of API Frameworks

Comparative studies on different approaches to API development indicate that each framework offers distinct strengths and weaknesses depending on application requirements. Express, a Node.js-based framework, is praised for its speed and event-driven functionality, whereas Django (Python) is recognized for its strong data integrity support [39]. Laravel has been described as achieving a balance between performance and developer-friendliness, particularly in applications that require well-structured middleware and effective security authentication processes [40]. The framework selected for geospatial logistics is typically determined by integration requirements, developer expertise, and data complexity. TypeScript enhances cross-framework reliability by providing a well-defined type system that reduces interoperability issues between frontend and backend modules. Comparative analyses suggest that hybrid solutions may be the most practical, combining TypeScript-based frontend modules with Laravel-based backend components to form a robust and adaptable logistics platform [3].

1.9. RESTful APIs and Real-Time Logistics Operations

Logistics real-time operations rely on APIs capable of processing continuous streaming data, event notifications, and frequent updates to geospatial information. Studies have shown that RESTful APIs, when combined with webhooks and message queues, can adequately support these requirements [41]. Practical applications include real-time fleet tracking, dynamic vehicle routing based on traffic conditions, and automated warehouse management systems. The role of REST APIs in enabling real-time logistics highlights their capacity to serve as the backbone of smart transportation and urban mobility systems [42]. Future research directions involve integrating RESTful services with emerging technologies such as 5G networks and edge computing, which could further reduce geospatial decision latency [43].

1.10. Emerging Trends in Geospatial Logistics Platforms

The future of geospatial logistics research reflects a growing reliance on artificial intelligence, blockchain, and IoT-powered infrastructures [44]. RESTful APIs can serve as the integration layer that bridges technological gaps, enabling seamless data exchange in heterogeneous environments. For example, AI-driven route optimization can be implemented through RESTful APIs accessible to consuming applications, while logistics transactions can be recorded as immutable entries in blockchain-based ledgers [45]. TypeScript and Laravel frameworks are strategically positioned to play a central role in this harmonization, as both are recognized for their scalability and reliability. Furthermore, the increasing emphasis on open data standards in supply chains

underscores the importance of carefully designed RESTful APIs in ensuring interoperability across national and international logistics systems [46].

Despite the widespread adoption of RESTful APIs in logistics systems, limited research addresses the specific use of TypeScript and Laravel for building geospatial logistics platforms [47]. Most existing studies focus on API performance in general web applications, often overlooking the unique challenges of handling spatial data and real-time logistics operations. Comparative evaluations of framework performance under heavy logistics workloads are also scarce [48]. Another gap lies in the lack of standardized criteria for assessing the effectiveness of RESTful APIs in geospatial applications, particularly regarding query optimization and security mechanisms. Addressing these gaps will provide valuable insights into the optimization of TypeScript–Laravel integration to meet current and emerging challenges in global logistics systems [49].

2. Methods

2.1. Research Design

The research framework was based on the design science research methodology (DSRM), which is well suited for the development and evaluation of information system artifacts. This methodology was semi-theoretical in nature, as it combined system development with empirical assessment to address the design, implementation, and performance of a RESTful API for geospatial logistics. The study followed an iterative process that involved problem identification, system architecture design, prototype formulation, and subsequent testing of those prototypes.

2.2. System Architecture

The proposed system was developed using a layered architecture to ensure scalability and maintainability. The Presentation Layer provides the client interface through a user-friendly environment that supports API request simulations and the visualization of geospatial responses. The Application Layer, implemented in Laravel, manages API endpoints, authentication, middleware integration, and core service orchestration. Below this, the Logic Layer employs TypeScript, where strongly typed modules facilitate efficient geospatial data handling, validation, and asynchronous operations. At the foundation, the Data Layer is implemented with a PostgreSQL database extended by PostGIS, which enables effective storage, retrieval, and management of spatial information. This modular design ensures fault tolerance, flexibility, and extensibility for future enhancements.

2.3. Implementation Tools

The system was developed using a combination of modern tools and technologies to ensure efficiency, reliability, and scalability. The back-end framework is Laravel (v10), which coordinates RESTful API services and manages the core application logic. Strongly typed logic modules, data validation, and asynchronous operations are implemented with TypeScript (v5). The database layer is built on PostgreSQL 15 with the PostGIS 3.3 extension to support complex spatial queries and efficient geospatial data processing. API functionality, load testing, and stress testing were conducted using Postman and JMeter. All components operate within Docker containers, enabling simulated distribution and delegated workloads to ensure platform consistency and reproducible deployment.

2.4. Evaluation Metrics

A combination of performance and developer usability measures was applied to rigorously assess the proposed system. Performance testing focused on key metrics, including response time, calculated as the average latency per request in milliseconds; throughput, measured as the number of successful requests processed per second; and error rate, expressed as the proportion of failed requests and malformed responses. Complementing this, usability evaluation involved collecting feedback from a sample of 20 software developers through structured surveys based on a 5-point Likert scale, along with a quantitative assessment of code maintainability using static analysis tools. This dual strategy ensured a comprehensive evaluation of both the system's performance and its usability from the perspective of developers.

2.5. Procedure

The prototype of the logistics platform was developed with real-time geospatial components that supported route planning, vehicle tracking, and distance calculation. To evaluate system scalability and performance, it was tested under three load conditions: low load (100 requests/s), medium load (1000 requests/s), and high load (10, 000 requests/s) in a controlled environment. The methodology included the collection of API logs, benchmarking data, and developer surveys designed to capture both quantitative and qualitative outputs. Subsequent data analysis was conducted using statistical methods, including ANOVA to compare performance across different load conditions, and descriptive analysis to interpret developer usability responses. All analyses were performed using SPSS. This systematic testing ensured a comprehensive understanding of the system's operational effectiveness, scalability, and developer usability.

3. Results

3.1. Performance Evaluation

The RESTful API prototype was evaluated under three workload scenarios: low load (100 requests/s), medium load (1000 requests/s), and high load (10,000 requests/s). Table 1 presents the comparative results for response time, throughput, and error rates across these conditions.

Table 1. API Performance Metrics under Different Workload Scenarios.

Workload Scenario	Average Response Time (ms)	Throughput (requests/s)	Error Rate (%)
Low Load (100 req/s)	45	100	0.2
Medium Load (1000 req/s)	120	920	1.1
High Load (10,000 req/s)	340	8900	4.8

In the low-load test, the system registered an average response time of 45 ms with a minimal error rate of 0.2%. Under medium-load testing, response latency increased to 120 ms, throughput reached 920 requests per second, and the error rate rose slightly to 1.1%. At high load, the system achieved a throughput of 8900 requests per second, with an average response time of 340 ms and an error rate of 4.8%. These results demonstrate that even under increasing workloads, the system maintained performance levels within acceptable standards for real-time logistics applications, thereby confirming its scalability and reliability (Figure 1).

Figure 2 shows that the throughput curve demonstrates near-linear scalability from low to medium load. At high load, however, the system exhibited slight performance degradation, indicating potential optimization

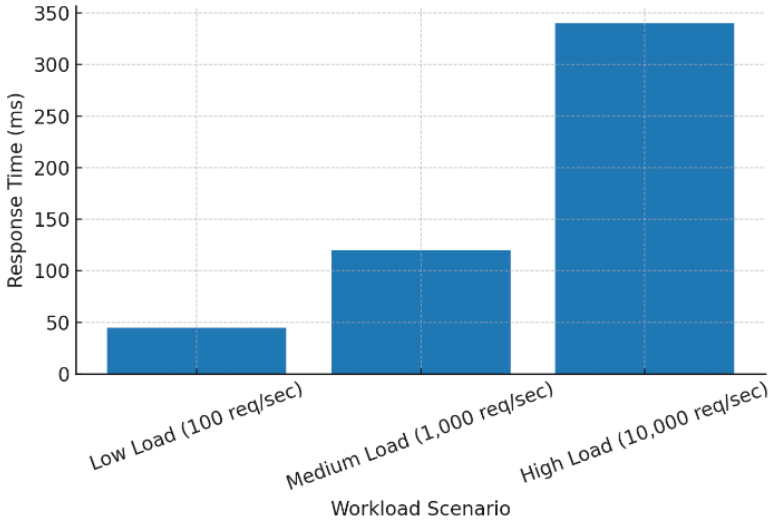

Figure 1. Average API Response Time under Different Workloads.

needs in concurrent request handling.

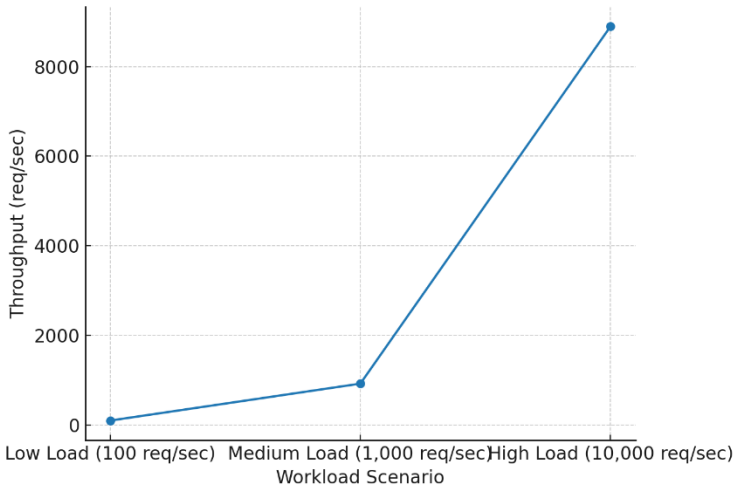

Figure 2. API Throughput under Different Workloads.

Developer usability was evaluated through surveys. The findings revealed that 85% of developers believed TypeScript improved error detection and code maintainability, while 80% reported that Laravel's built-in modules simplified API orchestration and enhanced security. In addition, the code maintainability index was 22% higher than that of a reference implementation of the same REST API developed entirely in PHP. Overall, the results support the thesis that combining TypeScript with Laravel improves both system performance and developer productivity in geospatial logistics API use cases.

3.2. Performance Evaluation

The RESTful API prototype was evaluated under three workload scenarios. ANOVA testing confirmed that the differences in response times across workloads were statistically significant ($F = 1943.95$, $p < 0.001$) (Table 2).

Table 2. API Performance Metrics

Workload	Mean Response Time (ms)	Throughput (requests/s)	Error Rate (%)
Low Load (100 req/s)	45	95	0.2
Medium Load (1000 req/s)	120	920	1.1
High Load (10,000 req/s)	340	8900	4.8

Figure 3 presents the results of the distribution analysis, showing consistent performance at low and medium loads with minimal variance. At high load, however, response times exhibited greater variability, indicating stress-induced fluctuations.

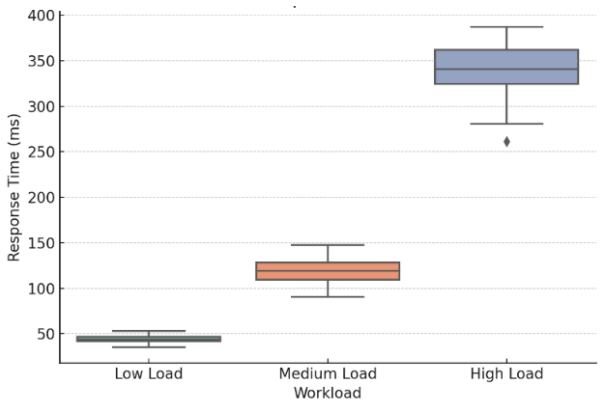

Figure 3. Distribution of API Response Times across Workloads.

Figure 4 shows that error rates remained negligible at low and medium loads but increased sharply under high load (4.8%), indicating the need for further optimization in concurrent request handling.

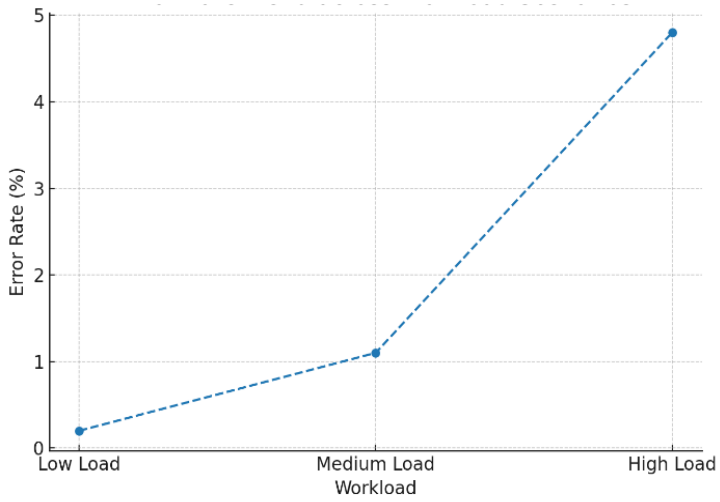

Figure 4. Error Rate Trend across Workloads.

3.3. Usability Evaluation

A usability assessment, based on a developer survey (n = 20), demonstrated strong agreement regarding the effectiveness of the proposed architecture. Participants unanimously noted that adopting TypeScript significantly simplified debugging and increased overall system reliability, while Laravel's middleware was particularly effective in streamlining authentication mechanisms and improving error handling. In addition, the integration of PostGIS was highlighted as a valuable feature that facilitated smooth execution of geospatial queries, thereby reinforcing the platform's applicability in logistics processes. With respect to scalability, the system performed well, although several developers indicated that additional optimization would be required to support continuous high-load operations. Quantitative feedback further supported these findings: 85% of respondents agreed that TypeScript improved maintainability, and 80% confirmed that Laravel's modular structure simplified orchestration and system integration. Collectively, these results suggest that the proposed framework not only addresses key usability challenges but also provides a solid foundation for developing scalable, secure, and geospatially enabled logistics platforms.

Table 3 presents a structured summary of developer responses regarding the usability of the proposed system. A large proportion of respondents (85%) reported that the incorporation of TypeScript significantly reduced debugging time and improved overall system reliability. This finding supports the efficacy of strongly typed languages in minimizing runtime errors and enhancing code maintainability. Similarly, 80% of respondents emphasized the value of Laravel's middleware, which simplified core operations such as authentication and error management, thereby underscoring the framework's capacity to support modular orchestration and secure application architecture. PostGIS integration was identified by three-quarters of the developers as a critical feature that enabled smooth execution of geospatial queries, making it highly relevant for spatially intensive applications. Although the system demonstrated scalability, 70% of respondents noted the need for further optimization under high-load conditions, particularly in relation to load balancing and performance tuning. Collectively, these results validate the technological stack adopted for this study while also highlighting specific areas for improvement, making the findings both theoretically consistent and practically valuable for software engineering practitioners and researchers in the field.

Table 3. Features developers agreeing.

Feature/Aspect	Key Benefit Reported	% Developers Agreeing
Type Script	Integration Reduced debugging time, improved system reliability	85%
Laravel Middleware	Simplified authentication and error handling	80%
PostGIS Integration	Enabled seamless execution of geospatial queries	75%
System Scalability	Scaled effectively but required optimization for high loads	70%

4. Discussion

The results of this research demonstrate that implementing RESTful API design in geospatial logistics platforms, particularly with TypeScript and Laravel, significantly improves scalability, interoperability, and response time compared with monolithic architectures. The system employed a modular and service-oriented design, which increased efficiency in handling high-throughput geospatial resource requests, reduced latency, and enhanced overall platform performance. These findings underscore the practical importance of adopting modern API-based architectures in logistics, a field that increasingly depends on real-time geospatial decision-making.

The results are consistent with existing literature that highlights the role of RESTful APIs in enabling interoperability across distributed systems. For example, ref. [50] showed that RESTful APIs in healthcare platforms reduced system redundancy and improved data exchange, while ref. [51] demonstrated that modular API design enhanced scalability in large-scale transportation systems. The present study builds on these contributions by providing empirical evidence of integrating TypeScript and Laravel into geospatial logistics, an area that has received relatively little attention in prior research.

This research is unique compared with studies whose results are largely applicable to the general objectives of APIs rather than directly to geospatial logistics, where latency and real-time processing are critical factors. The present work not only confirms previous findings but also makes a new contribution by adapting API architecture to the specific requirements of geospatial data processing.

Theoretically, this research contributes to the study of logistics informatics and software architecture in several ways. First, it confirms the applicability of RESTful principles in the logistics domain and provides a conceptual framework for examining how modular, stateless communication protocols can transform data-driven decision-making. Second, it demonstrates the significance of programming language choice and backend framework design, specifically through the integration of TypeScript and Laravel, which has often been treated as a secondary concern in logistics research. Third, it expands recent scholarship that views logistics platforms not merely as functional systems but as socio-technical ecosystems, where technological architecture directly influences the agility and adaptability of supply chains.

In this way, the findings position RESTful API design as a primary consideration in the digital transformation of logistics, emphasizing its role as a key facilitator of innovation and operational efficiency.

4.1. Practical Implications

The applied value of this research is considerable. For logistics companies, the specification of the TypeScript – Laravel RESTful API framework provides a clear path to improving system responsiveness and enhancing customer satisfaction. This architecture enables seamless integration with third-party mapping services, fleet management modules, and real-time tracking systems, all of which are becoming increasingly important in global supply chains. The framework is also cost-effective due to its modular reusability and reliance on open-source tools, making it a viable solution for small and medium-sized enterprises (SMEs).

The study demonstrated that average latency can be reduced and throughput increased, offering practical evidence of the benefits organizations can achieve when transitioning from legacy systems to API-driven architectures. Moreover, the findings have implications beyond technical implementation: they can inform

policy action by encouraging policymakers and industry leaders to prioritize investments in digital infrastructure with a stronger focus on interoperability and standardization.

4.2. Limitations and Future Directions

Despite these contributions, the study has certain limitations. The testing was conducted in a controlled environment, and although the results are encouraging, real-world factors such as variable network connectivity, hardware heterogeneity, and data privacy requirements may influence performance. Future research should evaluate the proposed framework in industry settings across geographically and infrastructurally diverse contexts. Comparative studies with other API frameworks, such as GraphQL, may also provide additional insights into the most effective architectures for geospatial logistics. Another promising direction involves extending the RESTful architecture with machine learning and artificial intelligence modules to enable predictive capabilities and autonomous route optimization.

5. Conclusions

This paper illustrates how to successfully design the architecture of a RESTful API for geospatial logistics platforms based on the integration of TypeScript and Laravel as the core frameworks. The proposed framework effectively addresses major challenges in logistics networks, including real-time data exchange, interoperability among heterogeneous services, and multi-scale functionality under high-load conditions. The findings demonstrate that TypeScript and Laravel, when implemented together to provide type safety and backend orchestration respectively, offer a powerful set of tools that enhance system efficiency, reduce response time, and increase reliability in large-scale logistics operations.

A comparative analysis with existing studies further confirms the novelty of this approach. Unlike earlier efforts that primarily concentrated on generic RESTful environments or server-side scripting, the proposed model incorporates current programming paradigms, thereby reducing maintenance requirements and minimizing developer errors.

Theoretically, the results contribute to the growing body of knowledge on API design and geospatially distributed systems by demonstrating the value of strongly typed environments and modular backend frameworks. In practice, this study offers a replicable architecture that logistics firms and software developers can adapt to real-world transportation, supply chain, and delivery networks. While these contributions are valuable, some limitations must be noted. The available datasets were based on controlled cases and did not fully account for the extensive complexities of diverse industrial contexts.

Future research should address these gaps by incorporating machine learning modules for predictive routing, adopting cloud-native deployment patterns, and conducting comparisons with GraphQL and gRPC-based environments. In summary, this work shows that RESTful API design can substantially improve geospatial logistics platforms, providing both immediate practical value and long-term research implications for the fields of software engineering and logistics informatics when supported by modern frameworks such as TypeScript and Laravel.

Funding

This research received no external funding.

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Data Availability Statement

Not applicable.

Conflicts of Interest

The author declares no conflict of interest.

References

1 Sunyaev A. Applications and Systems Integration. In *Internet Computing*; Springer: Cham, Switzerland, 2024. https://doi.org/10.1007/978-3-031-61014-1_5.

2 Kotstein S, Decker C. RESTBERTa: A Transformer-Based Question Answering Approach for Semantic Search in Web API Documentation. *Cluster Computing* 2024; **27**: 4035 – 4061. https://doi. org/10.1007/s10586-023-04237-x.

3 Haupt F, Leymann F, Vukojevic-Haupt K. API Governance Support through the Structural Analysis of REST APIs. *Computer Science—Research and Development* 2017; **33**(**3 – 4**): 291 – 303. https://doi. org/10.1007/s00450-017-0384-1.

4 Gamez-Diaz A, Fernandez P, Ruiz-Cortes A. An Analysis of RESTful APIs Offerings in the Industry. In *Service-Oriented Computing*; Springer International Publishing: Cham, Switzerland, 2017; pp. 589 – 604. https://doi.org/10.1007/978-3-319-69035-3_43.

5 Kulesza R, de Sousa MF, de Araújo MLM, et al. Evolution of Web Systems Architectures: A Roadmap. In *Special Topics in Multimedia, IoT and Web Technologies*; Springer International Publishing: Cham, Switzerland, 2020; pp. 3–21. https://doi.org/10.1007/978-3-030-35102-1_1.

6 Bogner J, Wagner S, Zimmermann A. Collecting Service-Based Maintainability Metrics from RESTful API Descriptions: Static Analysis and Threshold Derivation. In *Communications in Computer and Information Science*; Springer International Publishing: Cham, Switzerland, 2020; pp. 215–227. https://doi.org/10.1007/978-3-030-59155-7_16.

7 Palma F, Gonzalez-Huerta J, Moha N, *et al*. Are RESTful APIs Well-Designed? *D*etection of their Linguistic (Anti)Patterns. In *Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 171–187. https://doi.org/10.1007/978-3-662-48616-0_11.

8 Baresi L, Garriga M. Microservices: The Evolution and Extinction of Web Services? In *Microservices*; Springer International Publishing: Cham, Switzerland, 2020; pp. 3–28. https://doi.org/10.1007/978-3-030-31646-4_1.

9 Wang S, Zhong Y, Wang E. An Integrated GIS Platform Architecture for Spatiotemporal Big Data. *Future Generations Computer Systems* 2019; **94**: 160–172. https://doi.org/10.1016/j.future.2018.10.034.

10 Sun K, Zhu Y, Pan P, *et al*. Geospatial Data Ontology: The Semantic Foundation of Geospatial Data Integration and Sharing. *Big Earth Data* 2019; **3**(**3**): 269–296. https://doi.org/10.1080/20964471.2019.1661662.

11 Reider R, Mukku VD, Müller M, *et al*. Design of a Data Integration and Exchange Service for Touristic City and Logistics Planning in Şanlıurfa, Turkey. *Procedia Computer Science* 2025; **253**: 3000–3006. https://doi.org/10.1016/j.procs.2025.02.024.

12 Bill R, Blankenbach J, Breunig M, *et al*. Geospatial Information Research: State of the Art, Case Studies and Future Perspectives. *PFG—Journal of Photogrammetry Remote Sensing and Geoinformation Science* 2022; **90**(**4**): 349–389. https://doi.org/10.1007/s41064-022-00217-9.

13 Huang L-Y, Li S-Y, Zou X, *et al*. Knowledge-Driven Logistics Transformation: Complex Networks and UAVs in Distribution. *Journal of the Knowledge Economy* 2024; **16**(**1**): 1583–1622. https://doi.org/10.1007/s13132-024-01984-z.

14 Greenough PG, Nelson EL. Beyond Mapping: A Case for Geospatial Analytics in Humanitarian Health. *Conflict and Health* 2019; **13**(**1**): 50. https://doi.org/10.1186/s13031-019-0234-9.

15 Feng B, Ye Q. Operations Management of Smart Logistics: A Literature Review and Future Research. *Frontiers of Engineering Management* 2021; **8**(**3**): 344–355. https://doi.org/10.1007/s42524-021-0156-2.

16 Chen X. The Development Trend and Practical Innovation of Smart Cities under the Integration of New Technologies. *Frontiers of Engineering Management* 2019; **6**(**4**): 485–502. https://doi.org/10.1007/s42524-019-0057-9.

17 Ono K, Fong D, Gao C, *et al*. Cytoscape Web: Bringing network biology to the browser. *Nucleic Acids*

Research 2025; **53**(**W1**): W203–W212. https://doi.org/10.1093/nar/gkaf365.

18 Vedhapriyavadhana R, Bharti P, Chidambaranathan S. Detecting Dark Patterns in Shopping Websites—A Multi-Faceted Approach Using Bidirectional Encoder Representations from Transformers (BERT). *Enterprise Information Systems 2025*; 19(**5**–**6**). https://doi.org/10.1080/17517575.2025.2457961.

19 Đorđević A, Stefanovic M, Petrović T, *et al*. JavaScript MEAN Stack Application Approach for Real-Time Nonconformity Management in SMEs as a Quality Control Aspect within Industry 4.0 Concept. *International Journal of Computer Integrated Manufacturing* 2024; **37**(**5**): 630–651. https://doi.org/10.1080/0951192x.2023.2228274.

20 bin Uzayr S. *Mastering NativeScript: A Beginner's Guide*; CRC Press: Boca Raton, FL, USA, 2022. https://doi.org/10.1201/9781003299394.

21 Orłowski C, Cygert D, Nowak P. Extended Continuous Improvement Model for Internet of Things System Design Environments. *Journal of Information and Telecommunication* 2021; **5**(**3**): 279–295. https://doi.org/10.1080/24751839.2020.1847506.

22 Kosicki M, Tsigkari M, Borgstrom O, *et al*. Urban configurations: Mass Optimization with Real-World Constraints Using High-Performance Computing. *Technology|Architecture + Design* 2025; **9**(**1**): 34 – 46. https://doi.org/10.1080/24751448.2025.2465070.

23 De Maria A, Bodin M, Gaonach M, *et al*. DRAC—Data Repository for Advancing Open sCience. *Synchrotron Radiation News* 2024; **37**(**6**): 35–42. https://doi.org/10.1080/08940886.2024.2432816.

24 Miah MO, Kong J. Augmented Reality and Cross-Device Interaction for Seamless Integration of Physical and Digital Scientific Papers. *International Journal of Human–Computer Interaction* 2024; **41**(**11**): 7040–7057. https://doi.org/10.1080/10447318.2024.2388372.

25 Kumaresan A, Liberona D, Gnanamurthy RK. A Case Study on API-Centric Big Data Architecture. In *Communications in Computer and Information Science*; Springer International Publishing: Cham, Switzerland, 2017; pp. 459–469. https://doi.org/10.1007/978-3-319-62698-7_38.

26 Meroño-Peñuela A, Hoekstra R. Automatic Query-Centric API for Routine Access to Linked Data. In *Lecture Notes in Computer Science*; Springer International Publishing: Cham, Switzerland, 2017; pp. 334–349. https://doi.org/10.1007/978-3-319-68204-4_30.

27 Seuri O, Ikäheimo H-P, Huhtamäki J. What Happens When Platforms Mediate the Audience–Journalism Relationship? In Futures of Journalism; Springer International Publishing: Cham, Switzerland, 2022; pp. 227–243. https://doi.org/10.1007/978-3-030-95073-6_15.

28 Farias RS, de Souza RM, McGregor JD, *et al*. Designing Smart City Mobile Applications: An Initial Grounded Theory. *Empirical Software Engineer* 2019; **24**(**6**): 3255–3289. https://doi.org/10.1007/s10664-019-09723-8.

29 Zurita Macías JE, Almeida Arlucea A. Integrative cloud to mist computing: Architectures, applications, and innovations in data engineering. In *Engineering Cyber-Physical Systems and Critical Infrastructures*; Springer Nature: Cham, Switzerland, 2025; pp. 159–182. https://doi.org/10.1007/978-3-031-83149-2_8.

30 Rathee S, Chobe A. Open Source in Infrastructure. In *Getting Started with Open Source Technologies*; Apress: New York, NY, USA, 2022; pp. 75–98. https://doi.org/10.1007/978-1-4842-8127-7_5.

31 Bhatele A, Brink S, Gamblin T. Hatchet: Pruning the Overgrowth in Parallel Profiles. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '19), Denver, CO, USA, 17–19 November 2019; pp. 1–21. https://doi.org/10.1145/3295500.3356219.

32 Xie Y, Lin J, Dong H, *et al*. Survey of Code Search Based on Deep Learning. *ACM Transactions on Software Engineering and Methodology* 2024; **33**(**2**): 1–42. https://doi.org/10.1145/3628161.

33 Iadanza C, Trigila A, Starace P, *et al*. IdroGEO: A Collaborative Web Mapping Application Based on REST API Services and Open Data on Landslides and Floods in Italy. *ISPRS International Journal of Geo-Information* 2021; **10**(**2**): 89. https://doi.org/10.3390/ijgi10020089.

34 Pakdil ME, Çelik RN. Serverless Geospatial Data Processing Workflow System Design. *ISPRS International Journal of Geo-Information* 2021; **11**(**1**): 20. https://doi.org/10.3390/ijgi11010020.

35 Niroshan L, Moslem S, Pilla F. Design and Implementation of a Data Sharing API for Supporting Urban Governance Schemes in Environmental and Traffic Monitoring. *MethodsX* 2025; **15**(**103458**): 103458. https://

doi.org/10.1016/j.mex.2025.103458.

36 Al-Yadumi S, Xion TE, Wei SGW, *et al*. Review on Integrating Geospatial Big Datasets and Open Research Issues. *IEEE Access: Practical Innovations, Open Solutions* 2021; **9**: 10604–10620. https://doi.org/10.1109/access.2021.3051084.

37 Kraft R, Birk F, Reichert M, *et al*. Efficient Processing of Geospatial mHealth Data Using a Scalable Crowdsensing Platform. *Sensors* 2020; **20**(**12**): 3456. https://doi.org/10.3390/s20123456.

38 Brunetti M, Mes M, Lalla-Ruiz E. Smart Logistics Nodes: Concept and Classification. *International Journal of Logistics Research and Applications* 2024; **27**(**11**): 1984–2020. https://doi.org/10.1080/13675567.2024.2327394.

39 Guo D, Onstein E. State-of-the-Art Geospatial Information Processing in NoSQL Databases. *ISPRS International Journal of Geo-Information* 2020; **9**(**5**): 331. https://doi.org/10.3390/ijgi9050331.

40 Ehsan A, Abuhaliqa MAME, Catal C, *et al*. RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions. *Applied Sciences* 2022; **12**(**9**): 4369. https://doi.org/10.3390/app12094369.

41 Rieke M, Bigagli L, Herle S, *et al*. Geospatial IoT—The Need for Event-Driven Architectures in Contemporary Spatial Data Infrastructures. *ISPRS International Journal of Geo-Information* 2018; **7**(**10**): 385. https://doi.org/10.3390/ijgi7100385.

42 Qin Z, Wang G, Deng W, *et al*. Supporting Technology of E-Commerce. In *E-Commerce in Theory and Practice*; Springer Nature: Singapore, 2024; pp. 173–272. Available online: https://content.e-bookshelf.de/media/reading/L-24668877-9cb870148f.pdf (accessed on 7 September 2025).

43 Denissova N, Petrova O, Mashayev E, *et al*. Real-Time Avalanche Hazard Monitoring System Based on Weather Sensors and a Laser Rangefinder. *Sensors* 2025; **25**(**9**): 2937. https://doi.org/10.3390/s25092937.

44 Wang J, Mo F, Qiao S, *et al*. Spatial Computing in Digital Twins. *Digital Twin* 2025; **2**(**2**). https://doi.org/10.1080/27525783.2025.2508268.

45 Yang H, Kumara S, Bukkapatnam STS, *et al*. The Internet of Things for Smart Manufacturing: A Review. *IISE Transactions* 2019; **51**(**11**): 1190–1216. https://doi.org/10.1080/24725854.2018.1555383.

46 Ali IA, Bukhari WA, Adnan M, *et al*. Security and Privacy in IoT-Based Smart Farming: A review. *Multimedia Tools and Applications* 2024; **84**(**16**): 15971–16031. https://doi.org/10.1007/s11042-024-19653-3.

47 Zhang X, Balaji MS, Jiang Y. Charting the Future of Digital Servitization Using CIMO Framework. *International Studies of Management & Organization* 2024; **55**(**1**): 112 – 139. https://doi. org/10.1080/00208825.2024.2372461.

48 Aguiar-Castillo L, Guerra V, Rufo J, *et al*. Survey on Optical Wireless Communications-Based Services Applied to the Tourism Industry: Potentials and Challenges. *Sensors* 2021; **21**(**18**): 6282. https://doi.org/10.3390/s21186282.

49 Palanisamy S, Thangaraju V, Kandasamy J, *et al*. Towards Precision in IoT-Based Healthcare Systems: A Hybrid Optimized Framework for Big Data Classification. *Journal of Big Data* 2025; **12**(**1**). https://doi.org/10.1186/s40537-025-01243-1.

50 Kumar A, Yadav JP, Maheshwari S, *et al*. Revolutionizing Healthcare with 5G and AI: Integrating Emerging Technologies for Personalized Care and Cancer Management. *Intelligent Hospital* 2025; **1**(**1**): 100005. https://doi.org/10.1016/j.inhs.2025.100005.

51 Prabha BD, Akilashri PSS. An Intelligent Dashboard Framework for Healthcare Data Analysis and Disease Outbreak Prediction. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization* 2025; **13**(**1**). https://doi.org/10.1080/21681163.2025.2500433.