Managing authorization in government management systems: the experience of SIARE-Artefactos

Rafael Palau Heikel
Department of Electronics and Informatics
Universidad Católica "Nuestra Señora de la Asunción"
Ministry of Economy and Finance, Paraguay
ORCID: 0000-0002-5813-3506

Magalí Gonzalez
Department of Electronics and Informatics
Universidad Católica "Nuestra Señora de la Asunción", Paraguay
ORCID: 0000-0002-5368-443X

and

Luca Cernuzzi
Department of Electronics and Informatics
Universidad Católica "Nuestra Señora de la Asunción", Paraguay
ORCID: 0000-0001-7803-1067

Abstract

Public administration software systems are increasingly adopting microservices architectures to achieve scalability, flexibility, and resilience. However, the complexity of distributed systems poses challenges to access authorization management. This study presents the SIARE-Artefactos solution, which is designed to automate the registration and authorization of resources within Paraguay's Integrated State Resource Management System (SIARE). SIARE-Artefactos leverages OAuth 2.0 and Spring Boot Starter to ensure secure and efficient authorization processes. SIARE-Artefactos has been implemented and used by public employees in Paraguay, enabling validation with real users. For the validation we adopted a mixed-methods, combining quantitative with qualitative data sources, and a longitudinal observational design. The convergence of evidence allowed us to correlate the objective reduction in deployment times (from minutes to seconds) with the subjective perception of 'system predictability' reported by the DevOps team. Validation of the proposal demonstrates the solution's replicability to other cases with similar challenges and resources, as well as its ability to significantly improve system reliability and reduce configuration time by 74%. This highlights its potential to transform large-scale public administration systems using modern DevOps practices and agile methodologies.

Keywords: Microservices, Public Sector, Authorization, DevOps, OAuth 2.0

1 Introduction

Contemporary applications require innovative approaches to address challenges inherent to traditional software development, such as scalability and rapid application delivery. The adoption of microservices architecture has emerged as an effective solution, enabling the implementation of distributed and independently interconnected systems [1]. In this context, the Java programming language, together with the Spring Boot framework, has gained prominence in the development of REST services for large-scale systems [2].

The adoption of microservices architectures in public sector software systems offers opportunities to improve scalability, flexibility, and resilience. However, it also presents significant challenges in access authorization management and deployment efficiency due to the inherent complexity of large-scale distributed systems, such as those used in public administration.

Traditional approaches to resource management and authorization are often based on manual processes that are time-consuming, error-prone, and difficult to scale [3]. When implementing microservices architectures, the management of authorization and authentication for resource access becomes a critical concern, as the number and complexity of communication points between solution components increase. In response to this demand, OAuth 2.0[1] has emerged as an industry standard for open authorization in web applications, providing specific authorization flows for web services [4] and reducing the diversity and complexity of integrating multiple solutions. Nevertheless, authorization management and configuration remain persistent challenges.

To manage authentication and authorization for resource access in a microservices environment, application programming interface (API) managers play a fundamental role by operating in conjunction with identity services [5]. These API managers, such as WSO2 API Manager (WSO2 AM)[2], provide a comprehensive and fine-grained view of the resources exposed by microservices, adding layers of security, analytics, and consumption limits [6], complemented by identity servers such as WSO2 Identity Server (WSO2 IS)[3], which are responsible for authenticating, authorizing, and verifying the integrity of OAuth access tokens.

Even when an API management solution and its lifecycle are already in place through an API Manager, together with an identity manager such as WSO2 IS, authorization management gains increased relevance due to the large number of resources exposed by the API Manager to a diverse universe of consumers. This aspect becomes a critical point of failure and inefficiency, negatively affecting access control configuration and verification times during the deployment of software solutions.

Currently, the SIARE (Integrated State Resource Management System, Sistema Integrado de Gestión de Recursos del Estado in Spanish) comprises more than 780 REST endpoints and 62 user profiles. Without automation, deploying a new profile requires approximately 125 minutes of manual configuration, creating a bottleneck incompatible with agile methodologies.

Based on the above, this study proposes a response to these challenges through the development and implementation of SIARE-Artefactos, an automated approach for the registration and configuration of fine-grained access authorization to resources within Paraguay's Integrated State Resource Management System (SIARE). Built upon the SIARE Autodiscovery system[4], which automates API lifecycle management [7], this solution extends its functionality to include automatic authorization configuration mechanisms, streamlining access control configuration for large-scale systems. SIARE-Artefactos leverages modern technologies, including OAuth 2.0 for secure authorization and Spring Boot Starter[5] for rapid integration and deployment.

Unlike traditional solutions focused on access control through manual configurations or generic authentication tools, SIARE-Artefactos introduces an automated approach for large-scale microservices-based systems in general, and particularly for public-sector environments that require traceability, automation, and regulatory compliance. Its main contribution lies in enabling centralized and automatic authorization configuration with sufficient granularity to cover multiple functional contexts without compromising the technical autonomy of consuming systems. This proposal stands out both for its direct applicability within the SIARE ecosystem and for its efficient integration with widely adopted tools (OAuth 2.0, WSO2, and Spring Boot), as well as for its successful deployment in real production environments, demonstrating its feasibility and scalability. It is worth noting that SIARE-Artefactos has been implemented and is currently used by Paraguay's Ministry of Economy and Finance in cross-cutting systems that affect the entire public sector, enabling validation with real actors.

This study is an expanded version of the paper presented at CLEI 2025 [8] and a poster presented at CIbSE[9]. The new contributions focus on the following: i) a critical discussion of the role of microservices and how they can promote design flexibility and scalability in large-scale systems, such as SIARE; ii) a discussion of the lessons learned regarding the technical aspects of implementing the proposal; iii) an extension of the validation including a qualitative and economic analysis that further reinforce the promising results obtained; and, iv) a threat to validity

[1] https://datatracker.ietf.org/doc/html/rfc6749
[2] https://wso2.com/
[3] https://wso2.com/es/identity-server/
[4] https://gitlab.com/dgic.mh.py/SIARE
[5] https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter

analysis to discuss the limitations of the scope of the experimental results. Accordingly, we have partially reformulated the conclusions and future work.

The remainder of this paper is structured as follows: Section 2 presents related work; Section 3 describes the challenges that motivated the SIARE-Artefactos proposal, as well as the system architecture and key implementation aspects; Section 4 presents the validation in a public management environment together with the obtained results; and Section 5 discusses the main conclusions and outlines possible directions for future work.

2 Related Work

In the early stages of microservices architectures, authorization implementation relied on simple approaches such as access control lists (ACLs) or custom authentication tokens [10]. These methods provided basic control over resource access but lacked the flexibility and scalability required for complex environments. Among their main drawbacks are the lack of fine-grained access control, the difficulty of orchestrating permissions across multiple microservices, and the complexity of managing vulnerabilities against targeted security attacks. In addition, session-based authentication introduced additional latency in distributed systems due to the need to synchronize session stores, resulting in high server resource consumption and degraded performance as the user base grew [11].

Custom authentication tokens emerged as an evolution of ACL-based approaches in microservices architectures. These tokens are created and managed by the application or microservice itself, enabling user authentication and authorization. Although they provide greater control, their management can be complex and they do not scale well in environments with multiple microservices.

Role-based access control (RBAC) systems also exist, in which permissions are assigned to specific roles, simplifying authorization management [10]. However, RBAC can be inflexible in scenarios that require more fine-grained permissions.

As a mechanism to overcome the limitations of previous models, OAuth originated at Twitter with the introduction of OpenID. This standard enables secure delegated access to resources by delegating authentication to an external identity provider. OAuth 2.0 offers several advantages over basic approaches. It supports fine-grained access control by allowing the definition of specific permissions for each resource and user or user role. It is scalable with linear effort, as it adapts to complex environments with multiple microservices and users or roles, and it enables improved centralized management of security policies, reducing the risk of security attacks by delegating authentication.

These benefits are achieved without negatively impacting development time, as OAuth 2.0 facilitates the integration of authorization mechanisms into microservices in a standardized and transversal manner, abstracting security management away from development teams. In recent years, the use of OAuth 2.0 in microservices architectures has become the de facto standard for authorization management. Nevertheless, the continuous evolution of this domain presents new challenges and opportunities. Secure OAuth 2.0 implementation requires careful microservices architecture design, with particular attention to the distribution of access controls, the enforcement points of security policies, and the encryption of communication channels through which tokens are transmitted in untrusted networks (Zero Trust Networking). While OAuth 2.0 addresses authorization from a protocol perspective, it is tightly coupled with authentication flows and token management, which often obscures the distinction between identity validation and authorization configuration in practice.

Attribute-based access control (ABAC) mechanisms also exist, enabling the definition of more flexible and dynamic authorization policies based on user, resource, or contextual attributes [12]. This approach is particularly suitable given the diversity of users and resources that share similar authorization requirements.

Authorization and authentication using the Sidecar Pattern also represent an efficient solution for microservices. This software design technique allows auxiliary functionalities to be implemented alongside a primary microservice. In the context of authorization and authentication, the sidecar encapsulates security logic, separating it from the main microservice and providing benefits such as modularity, reusability, scalability, and resilience [13].

Identity management systems, particularly modern Identity Providers (IdPs), play a fundamental role in resource management. In this regard, originally monolithic systems whose challenges were addressed by traditional identity

servers such as LDAP (Lightweight Directory Access Protocol) and Active Directory were primarily limited to identity and authentication management in centralized environments [6]. However, their rigidity and limited scalability make them unsuitable for the requirements of microservices architectures.

Between 2016 and 2018, IdPs emerged as a standard for delegated authorization. Notable examples include Auth0, Keycloak, and Azure AD B2C. Such providers offer greater flexibility and scalability, enabling identity and authentication management in distributed environments.

The trend toward modularity and scalability led to the adoption of microservices architectures within identity server implementations. This shift enabled the decomposition of functionalities such as user registration, authentication, and authorization management into independent microservices, improving system flexibility and resilience.

These tools have also demonstrated the core characteristics of modern identity servers for microservices, being highly scalable and capable of handling large volumes of users (up to 5,000) and authentication requests efficiently. They support federation, allowing authentication and authorization across multiple identity providers and promoting interoperability [10], while implementing robust security measures to protect sensitive user information.

To better understand the limitations of existing approaches, it is useful to compare authorization mechanisms across three key dimensions: control model, granularity, and operational complexity. From a comparative perspective, early approaches such as ACLs and custom tokens can be characterized by their low abstraction level and strong coupling to individual services. However, they exhibit significant limitations in distributed environments, particularly in terms of scalability and centralized governance. Their management becomes increasingly complex as the number of services grows, requiring manual synchronization of permissions across components.

From a comparative standpoint, RBAC improves manageability through role abstraction, but still lacks the flexibility required for dynamic authorization scenarios. OAuth 2.0, together with Identity Providers, represents a major advancement by enabling delegated and centralized authorization. These approaches support fine-grained access control and federation across distributed systems. However, they primarily address authentication and token validation, rather than the operational lifecycle of permission configuration.

Similarly, architectural patterns such as the Sidecar Pattern improve modularity and separation of concerns by externalizing security logic. Nevertheless, they require containerized environments and orchestration platforms, which are not always available in institutional contexts such as on-premise public systems.

Traditional identity systems such as LDAP and Active Directory provide centralized identity management but lack the flexibility required for modern microservices environments. Although modern Identity Providers extend these capabilities, they still depend on manual or semi-automated processes for defining and maintaining authorization rules.

Therefore, a key limitation across all these approaches is not the lack of secure authorization mechanisms, but the absence of automated, scalable, and environment-consistent processes for managing authorization configurations at the microservice level. In large-scale public systems such as SIARE, where authorization complexity is amplified by the number of services, environments, and institutional actors, this limitation becomes a critical operational bottleneck. Existing tools provide the necessary infrastructure for secure authorization but do not address the automation of resource registration and permission configuration within the software lifecycle.

The reengineering project of the Integrated State Resource Management System (SIARE) of Paraguay's Ministry of Economy and Finance aims to transform a quasi-monolithic Oracle-based architecture into a microservices-based solution comprising more than 324 components, with only 7% progress achieved to date. The current integration process, despite using Autodiscovery for managing the microservices lifecycle within the API Manager (WSO2), still relies on manual configurations for access authorization. This represents a significant obstacle for project management under agile and modern methodologies. The implementation of a Spring Boot Starter project based on Autodiscovery aims to automate access authorization management, improving efficiency and reducing the likelihood of errors in large-scale systems.

This gap motivates the need for solutions such as SIARE-Artefactos, which focus not only on authorization enforcement but also on the automation of its configuration and governance. This work shifts the focus from authorization models to authorization lifecycle automation.

3 SIARE-Artefactos: Automated Authorization for Microservices

The architecture of the SIARE-Artefactos system was carefully designed to address the unique challenges associated with authorization management in microservices-based public administration systems. By leveraging modern frameworks and best practices, the architecture ensures scalability, reliability, and ease of integration with existing systems. This section provides an overview of the challenges addressed, the proposed solution, and the technical workflow that underpins the system's operation.

The use of microservices architectures has been widely studied across various domains over the past decade, highlighting advantages such as scalability, ease of maintenance, and deployment flexibility. However, most existing studies focus on corporate or general-purpose environments, leaving a gap in the analysis of their application within Latin American governmental contexts, where conditions and constraints tend to differ significantly.

In this regard, the work of Alshuqayran et al. (2016) provides a valuable systematic review of the technical and organizational challenges associated with microservices architectures [14]. Nevertheless, there is still a lack of specific proposals for automated authorization management in public-sector systems—an especially sensitive aspect when multiple institutions with different access levels and regulatory frameworks are involved. This work seeks to contribute to this area by presenting a concrete proposal tailored to the SIARE environment.

To understand the complexity of the authorization challenges addressed by SIARE-Artefactos, it is necessary to describe the underlying architecture of the ecosystem. As detailed in previous studies regarding the Ministry's technological adaptation [15], the solution migrated from a client-server monolith to a microservices-based architecture organized into five logical layers (Fig. 1). This layered approach promotes high cohesion and low coupling, but introduces specific governance requirements for inter-component communication:

1. **Presentation Layer:** Responsible for providing a unified vision to end-users. It is decoupled from business logic using Angular to build Single Page Applications (SPA). These applications consume services exclusively via REST protocols, allowing independent scalability of the frontend.

2. **Integration Layer**: Acts as the centralized entry point for all internal and external requests. It utilizes WSO2 API Manager as an API Gateway to expose services grouped by responsibility. This layer manages the API lifecycle, traffic throttling, and initial security validations, abstracting the complexity of backend microservices from consumers.

3. **Business Logic Layer**: Contains the specific domain logic. It employs Spring Boot for developing independent microservices that own their data and business rules. To ensure high performance in this distributed environment, Redis is utilized for in-memory caching, while Apache Kafka serves as an event orchestrator for complex transactions requiring asynchronous communication between modules. This layer contains *SIARE-Artefactos, Back-Bienes and Back-Usuarios*

4. **Persistence Layer**: Follows the "database-per-service" pattern to ensure decoupling. While the architecture supports polyglot persistence (including PostgreSQL), it primarily leverages Oracle databases to capitalize on existing institutional knowledge and legacy data structures. Additionally, file storage is handled via filesystem repositories rather than database blobs to reduce overhead.

5. **Security and Observability**: Security is transversal, implementing OAuth 2.0 with WSO2 Identity Server for authentication and Single Sign-On (SSO). For observability, the architecture integrates the ELK Stack (Elasticsearch, Logstash, Kibana) and Filebeat, enabling centralized logging and proactive monitoring of the distributed infrastructure.

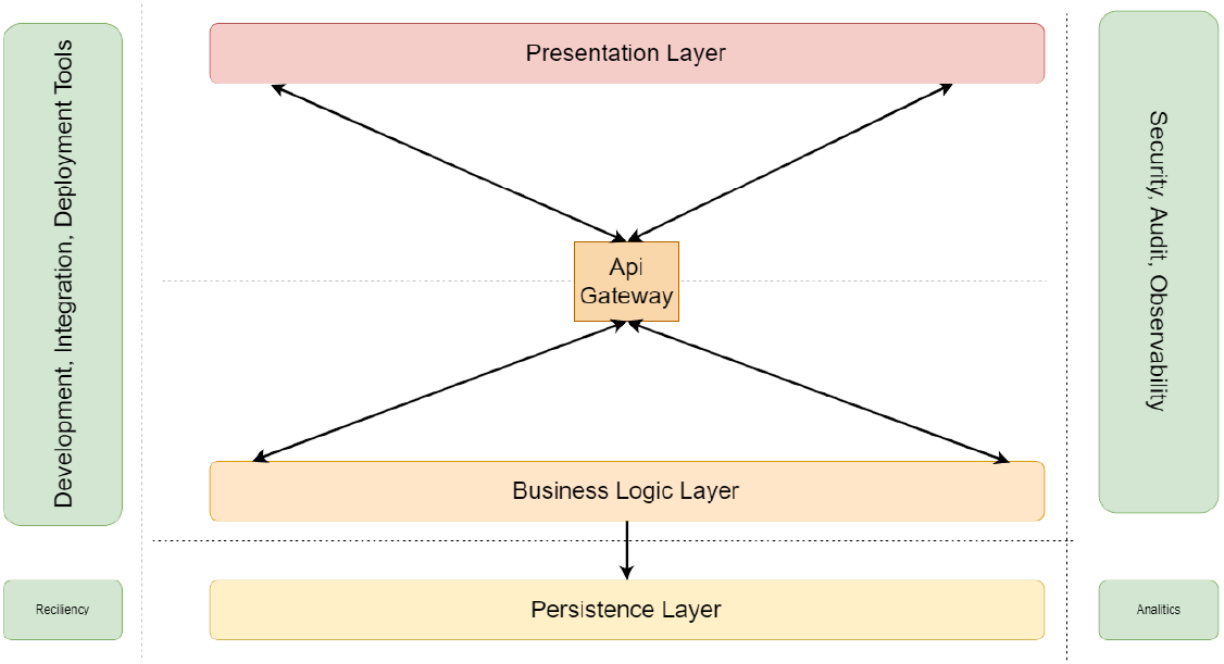

Figure 1:SIARE Architecture

3.1 Current Challenges

The SIARE system, in its current state, presents significant limitations that affect its scalability and operational efficiency. As previously mentioned, SIARE has undergone a reengineering process oriented toward a microservices architecture. The number and diversity of microservices that comprise the SIARE architecture clearly indicate the scale and complexity of the system.

Figure 2: Authorization and Access Pyramid

The system currently registers more than 2,439 artifacts (backend REST endpoints), which are grouped into 372 permissions intended to segment actions within user-oriented processes. These permissions are further disaggregated

into 92 different profiles according to users' activities within the system (Fig. 2). Depending on institutional assignment, users may hold multiple profiles. This situation represents a considerable challenge, particularly with regard to the registration and authorization of access to resources. Each new implementation or profile addition requires extensive manual effort, with an average configuration time of approximately 125 minutes per profile and between 17 and 75 minutes per backend service. This reliance on manual processes consumes valuable time, introduces inconsistencies, and increases the likelihood of human error, which may lead to more severe issues during deployment.

Prior to the implementation of SIARE-Artefactos, the authorization management process relied on an exhaustive and error-prone manual intervention. For each newly developed endpoint, the technical team had to identify the specific methods to which the permission applied and manually reference them in a web administrative interface as 'artifacts'. This registration required explicitly detailing the module and the resource URL, as well as configuring the allowed actions segregated by each HTTP verb. The complexity of this task multiplied exponentially, as this configuration procedure had to be replicated identically across each environment of the deployment lifecycle (Development, Testing, UAT, Pre-production, and Production), creating a high risk of inconsistencies between environments

The manual nature of these tasks significantly impacts system agility, delaying updates and preventing a rapid response to emerging changes and requirements. Furthermore, the inherent complexity of managing multiple endpoints and user profiles within a distributed system exacerbates these problems, creating a bottleneck that restricts the system's ability to scale efficiently. This inefficiency is further intensified by the lack of standardized processes for resource registration, making consistency across implementations difficult to achieve.

Consequently, the need for an automated, reliable, and scalable solution becomes imperative to address these challenges and ensure the long-term viability of the SIARE system. In order to sustainably manage authorization processes—considering resource access by user profiles within institutions segmented into permissions that group actions—a persistence model was designed within the user management system. This model introduces the essential foundational elements upon which the proposed system improvements are built:

- **Independence between authentication and authorization processes**: In this proposal, a user corresponds to a person with their own personal data and credentials that define user identification, while authorization is handled separately through the assignment of profiles to permissions, independently configuring access grants to resources.
- **Scope identification and limitation through groups**: Scope and group-based authorization (OAuth scopes and groups) are achieved by assigning institutions to users per profile. This enables record-level access control and simulates multi-tenancy behavior. Such functionality is not feasible in the current system, where accessing another institution requires the creation of a separate user account, approaching an attribute-based access control model.
- **Scope disaggregation into permissions and actions:** This approach enables more fine-grained authorization control. While it offers clear advantages in access granting, it also increases management complexity, as discussed in the related work section.

Within the project, WSO2 Identity Server (WSO2 IS) was selected as the identity server, as it supports multiple identity providers, including the institutional LDAP directory[6], records stored in institutional databases, electronic identity systems from the Ministry of Information and Communication Technologies (MITIC), and even external, non-controlled systems such as Google or social networks. Although only the integration with the SIARE user management system is currently implemented through a custom authenticator in WSO2 API Manager, integration with MITIC's electronic identity management system is planned.

3.2 Proposed Solution: SIARE-Artefactos

To sustainably manage authorization processes in this complex environment, the solution was built upon a persistence model governed by three fundamental principles, separation of concerns with independence between authentication (Identity) and authorization (Permissions), aligned with IAM (Identity and Access Management)

[6] https://ldap.com/

principles; multi-tenancy simulation with scope limitation by institutional groups, allowing record-level access control similar to Attribute-Based Access Control (ABAC); and hierarchical disaggregation with a structure of 'Profiles → Permissions → Actions' to ensure granular control without losing management capability.

SIARE-Artefactos ensures seamless integration of automated access authorization registration for resources within the existing ecosystem by employing a modular and systematic approach. The sequence depicted in Fig. 3 can be understood as a sequential interaction among the main system components during the application startup and deployment. The process begins with annotation identification, during which the application scans the controllers at application startup during deployment time. These annotations, such as @GetMapping, identify endpoints, HTTP methods, and actions that must be registered automatically.

The next stage involves configuration validation. Centralized properties are retrieved from the Config Server, which obtains environment-specific configurations from a Git-based configuration repository (ConfigRepo)[7]. This step ensures that all configurations are consistent, secure, and tailored to the corresponding deployment environment, whether development, testing, or production. By adopting this centralized configuration approach, the system reduces the complexity of managing multiple environments and eliminates the need for manual adjustments.

To optimize performance and reduce database load, the workflow incorporates authorization management—after identity verification at the identity server—through caching mechanisms using Redis[8]. Instead of querying the database for each authorization request, the system stores the most frequently accessed authorization data in Redis. This approach significantly improves response times and ensures that the authorization process remains efficient, even under high traffic conditions.

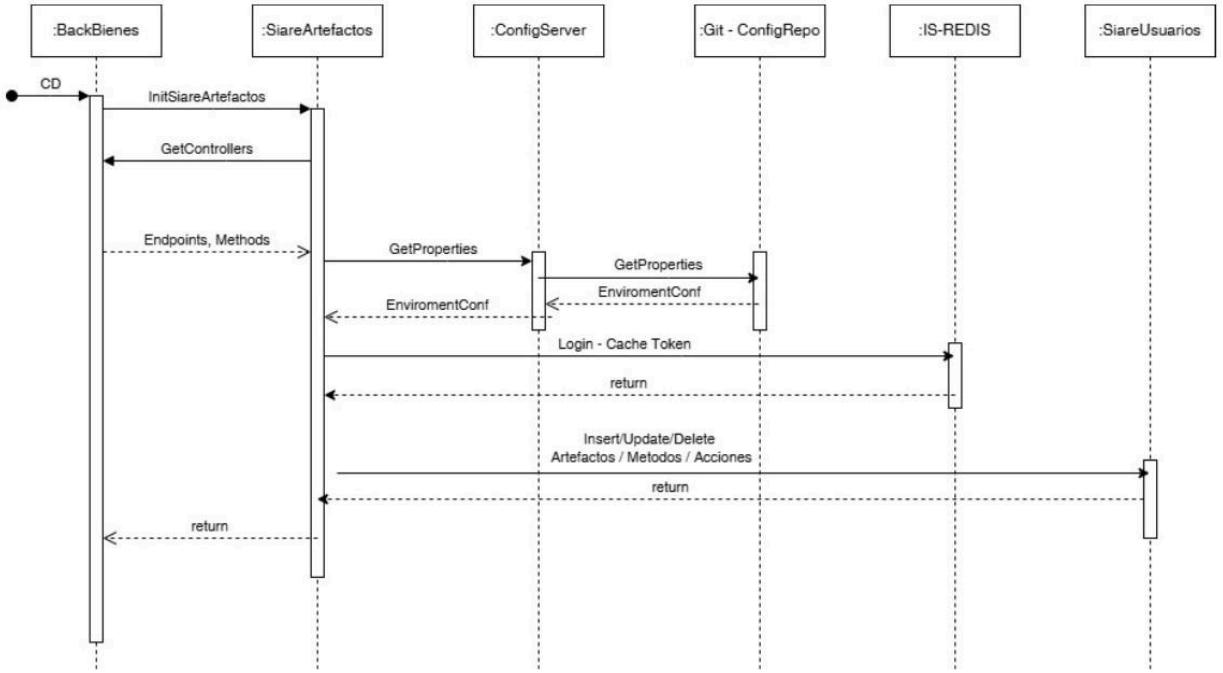

Figure 3: SIARE-Artefactos sequence

Once controller annotations have been processed, the system proceeds to the synchronization phase with the Authorization Management system (SIARE-Usuarios). During this stage, the identified resources are registered in the user backend, ensuring that all endpoints, HTTP methods, and associated actions are correctly identified, documented, and authorized. This synchronization guarantees a coherent and centralized registry of all microservices and their corresponding authorizations, minimizing discrepancies across implementations and deployments.

[7] https://git-scm.com/
[8] https://redis.io/

The final component of the workflow is integration with continuous integration and continuous deployment (CI/CD) pipelines. By embedding the automated resource registration process directly into the deployment lifecycle, the system ensures that each new deployment is consistent and free of authorization configuration errors. This integration not only reduces manual intervention but also aligns with DevOps best practices, promoting a more agile and reliable deployment process.

Overall, the technical workflow of SIARE-Artefactos is designed to optimize authorization management in microservices architectures. By automating resource registration, centralizing configuration management, and optimizing authorization processing, the system provides a robust foundation for scalable and reliable infrastructures in public administration environments.

3.3 Design Rationales and Constraints

Critically, the adoption of a microservices architecture was not merely a technical upgrade but a necessity to overcome the rigidity of the legacy monolithic system (based on Oracle Forms). The legacy system required creating multiple user accounts for a single civil servant holding roles in different institutions. The proposed microservices architecture, by decoupling the authorization logic into the 'Provider Layer', allowed for a simulated multi-tenant environment. This enables a single user identity to seamlessly switch between institutional contexts (scopes) within the SIARE ecosystem, a flexibility that was structurally impossible in the previous monolithic architecture.

SIARE-Artefactos was designed as a transitional solution adapted to a centralized on-premise environment. While cloud-native patterns like the Sidecar Pattern are effective for handling cross-cutting concerns (like authorization) in Kubernetes environments, they were not viable due to the institution's infrastructure constraints.

Therefore, the architecture focuses on logic abstraction via libraries (Spring Boot Starter) rather than container orchestration. This decision allowed the solution to introduce automated governance without requiring a complete infrastructure overhaul. The architecture separates responsibilities into four layers (Client, Startup, Provider, Configuration), proving that scalability in public systems depends as much on technological choices as on the ability to adapt to legacy constraints

3.4 Implementation

The implementation of SIARE-Artefactos was based on the principles of modularity, scalability, and ease of integration, ensuring that the solution integrates seamlessly into the existing SIARE system ecosystem. To achieve this, the system architecture and its components were divided into four main layers, each serving a specific purpose to improve maintainability and operational efficiency.

Figure 4 can be interpreted as a component-level view centered on the *SIAREArtefactosApplication*, which orchestrates the interaction between startup, configuration, client, and provider modules. The initialization process is triggered by the *SIAREArtefactosStarter* and supported by the *SIAREndpointRegister* for endpoint registration. Configuration is managed through PropertiesConfig, while external interactions are handled via BackUsuariosRestClient and the Provider Layer (ApiProvider and BackApisProvider), with ApiRegistryException ensuring controlled error handling. This structure reflects a modular design that enables clear separation of responsibilities and facilitates scalability and integration.

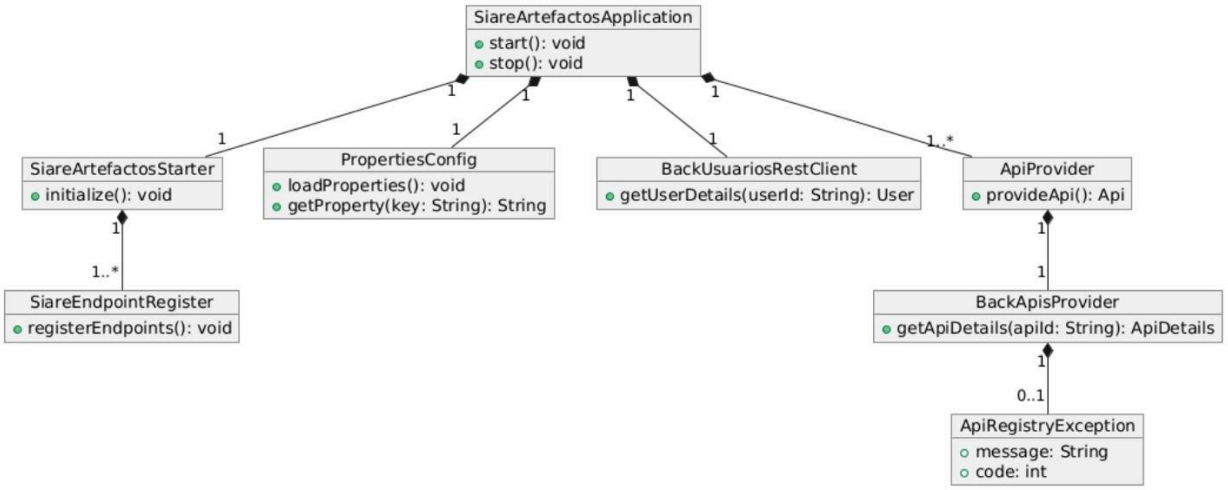

Figure 4: SIARE-Artefactos components

The first layer, the Client Layer, is responsible for managing communication with backend systems, in this case including the *BackUsuariosRestClient*. This layer comprises REST clients that interact directly with APIs to register endpoints, HTTP methods, and actions. By abstracting these interactions, the Client Layer ensures that the system remains flexible and adaptable to changes in external APIs, minimizing the need for code modifications.

The Startup Module, which constitutes the second layer, plays a fundamental role in automating the registration process. During application startup, this module initializes the automated resource registration process by scanning annotations (Fig. 5) in Spring Boot controllers across the projects in which it is invoked, including SIAREArtefactosApplication and SIAREArtefactosStarter. This abstraction of the registration process allows developers to focus on business logic rather than configuration tasks. This approach reduces manual effort and ensures both deployment consistency and automatic registration of each relevant endpoint, without requiring additional effort from developers during each deployment.

The third layer, the Provider Layer, introduces an additional level of abstraction to manage interactions with APIs. This layer includes components such as ApiProvider, BackApisProvider, and ApiRegistryException, which enable the integration of external systems with the SIARE-Artefactos module and the handling of related exceptions. By encapsulating the logic for interacting with external services, the Provider Layer ensures reusability and flexibility, allowing the system to scale as required.

Finally, the Configuration Layer manages centralized configuration for the entire system. This layer uses ConfigServer, a microservice that retrieves configurations from a Git-based configuration repository (ConfigRepo), which are stored in the PropertiesConfig object. By separating configuration management from application logic, this layer enables the system to dynamically adapt to different environments, such as development, testing, user acceptance, preproduction, and production. In addition, the Configuration Layer supports encrypted storage of sensitive data, enhancing the overall security of the solution.

```java
@GetMapping(path = "/{id}", produces = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<ResponseDto<E>> getRecordById(
                @PathVariable(value = NombreParametroPeticion.IDENTIFICADOR) K id) {

        ResponseDto<E> recordDTO;
        recordDTO = svc.getById(id);
        return new ResponseEntity<>(recordDTO, HttpStatus.OK);

}

@PostMapping(produces = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<ResponseDto<E>> createRecord(@Valid @RequestBody E entity) {

        validateBaseFields(entity);
        ResponseDto<E> persistedRecord = svc.save(entity);
        return new ResponseEntity<>(persistedRecord,
                        persistedRecord.getMessages().isEmpty() ? HttpStatus.OK : HttpStatus.PRECONDITION_FAILED);
}

@PutMapping(path = "/{id}", produces = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<ResponseDto<E>> updateRecord(
                @PathVariable(value = NombreParametroPeticion.IDENTIFICADOR) K id, @Valid @RequestBody E entity) {

        validateBaseFields(entity);
        ResponseDto<E> responsse = svc.update(id, entity);
        return new ResponseEntity<>(response,
                        responsse.getMessages().isEmpty() ? HttpStatus.OK : HttpStatus.PRECONDITION_FAILED);
}

@DeleteMapping(path = "/{id}", produces = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<Object> deleteRecordById(
                @PathVariable(value = NombreParametroPeticion.IDENTIFICADOR) K id) {

        E entity  = svc.getById(id).getResponse();
        ResponseDto<E> responsse = svc.deleteByEntity(entity);
        return new ResponseEntity<>(response,
                        responsta.getMessages().isEmpty() ? HttpStatus.OK : HttpStatus.PRECONDITION_FAILED);
}
```

Figure 5: Example of a base controller

Overall, the implementation of SIARE-Artefactos represents a significant advancement in automating authorization management for public administration systems. By adopting a modular and scalable design, the solution addresses challenges related to complexity and inefficiency, paving the way for future innovations in the management of microservices-based architectures.

3.5 Lessons Learned

Initially, an attempt was made to incorporate the authorization logic directly into the existing 'SIARE-Autodiscovery' component (used for API registration). However, implementation revealed that both processes operate at different critical moments: Autodiscovery focuses on artifact publication, while SIARE-Artefactos addresses access control. Decoupling them into separate Spring Boot Starters reduced code complexity and execution time, confirming that even related cross-cutting concerns should remain modular to facilitate maintenance.

Also, the implementation revealed that automation does not replace governance; rather, it enforces it. The efficiency achieved required strict naming conventions to maintain the traceability of automatically generated permissions. Additionally, the separation of authentication (Identity) from authorization (Permissions) proved crucial for replicating the model across diverse functional domains—from Treasury to Budgeting—without code rewriting, confirming the reusability of the modular design

4 Validation

As previously mentioned, SIARE-Artefactos has been implemented and is currently in use at Paraguay's Ministry of Economy and Finance, enabling validation with real actors. The validation of SIARE-Artefactos was an integral part

of the development process, ensuring that the proposed solution effectively and efficiently met its objectives. This section describes the validation design, highlighting the improvements achieved in terms of operational efficiency, error reduction, and resource optimization.

The validation of SIARE-Artefactos was carried out through a rigorous and systematic process, focused on three critical areas of performance and reliability, following the methodological recommendations of Runeson and Höst (2009) [17], which are commonly applied in software engineering case studies. A mixed-methods design was employed, utilizing data triangulation to combine quantitative sources (Jenkins, Matomo, and Redmine logs) with qualitative sources (semi-structured interviews and direct observation). The study followed a longitudinal observational design during the second semester of 2024, establishing a baseline with historical data to isolate the impact of the intervention. This convergence of evidence allowed us to correlate the objective reduction in deployment times (from minutes to seconds) with the subjective perception of 'system predictability' reported by the DevOps team, thereby minimizing bias inherent to a single data source.

The validation of this study case was conducted in accordance with the essential characteristics of this approach. First, the phenomenon under study—the implementation and evolution of automated access control mechanisms in public-sector systems—was analyzed within its natural context, without altering the institutional environment in which it operates and without imposing artificial or laboratory conditions. This made it possible to understand how the proposed solution interacts with the structures, processes, and constraints inherent to the SIARE ecosystem.

Regarding data collection, multiple sources were employed, including direct observations of system operation and the review of artifacts containing historical information (logs, technical documentation, and support tickets), as well as semi-structured interviews with technical stakeholders from development and development operations teams. The combination of these techniques enabled data triangulation and increased the construct validity of the study.

During the analysis, efforts were made to maintain a clear chain of evidence between the findings and the original data. Recurrent access patterns, critical exceptions, and institutional responses to changes in authorization rules were identified. Finally, the report includes representative excerpts and quantitative evidence of generated artifacts, allowing readers to follow the reasoning process and assess the robustness of the conclusions.

Overall, the design, execution, and documentation of the case study aim to ensure credibility, traceability, and contextual relevance, in line with best practices for empirical research in software engineering.

First, the efficiency of the solution was analyzed by measuring the average time required to register resources manually in the system, supported by the system's time-tracking instruments. By comparing automated registration processes executed through Jenkins[9] with manual registrations recorded in Redmine[10], the study identified significant improvements in operational speed and consistency. In addition, variability in registration times was evaluated to ensure that automation produced stable and predictable results, minimizing delays and inconsistencies during deployment.

Second, the validation process evaluated error rates, focusing on the accuracy of configurations during resource registration. For this purpose, frontend request logs recorded in Matomo[11] were analyzed, comparing successful requests with those resulting in authorization failures after the implementation of SIARE-Artefactos. By contrasting the outcomes of manual and automated approaches, the study demonstrated a significant reduction in configuration errors attributable to human intervention. This improvement not only increased system reliability but also optimized troubleshooting processes by providing detailed logs, as evidenced by a reduction in the number of authorization error occurrences in production environments.

Finally, the validation analyzed resource usage, specifically examining the reduction in manual effort and associated operational costs. Metrics such as the number of hours saved per deployment and the frequency of manual interventions provided a comprehensive view of the solution's impact on operational efficiency. This approach enabled a holistic understanding of the system's benefits from both technical and organizational perspectives.

[9] https://www.jenkins.io/
[10] https://www.redmine.org/
[11] https://matomo.org/

4.1 Quantitative Analysis

The results of the validation process highlighted the transformative potential of SIARE-Artefactos in authorization management for microservices-based public administration systems. To properly understand the impact of the results presented in this study, it is important to consider the growth in both the number of users and the assignment of users to institutions with their corresponding profiles. These profiles are disaggregated into permissions that group actions registered by the SIARE-Artefactos component. For illustrative purposes, an excerpt showing the evolution and growth of users and their assignment to profiles by institution during the last semester of 2024 is presented in Table 1.

Table 1: Registered users and profiles (cumulative)

Year	Month	Users Created	Profiles Assigned
2024	7	38	168
2024	8	120	378
2024	9	160	506
2024	10	193	630
2024	11	517	2,058
2024	12	835	3,624

Efficiency improvements, as expected due to the automation of the process, were particularly notable. The average time required for resource registration was reduced from a range of 17–75 minutes to an interval of 3–5 seconds. This improvement not only accelerated the deployment process but also ensured greater consistency in registrations by eliminating the variability associated with manual configurations. Table 2 shows the log timestamps between the start and end of automated endpoint registrations, including their corresponding HTTP methods and actions, within the user management system for the goods backend (SIGEBYS).

Table 2: Automatic registration – SIGEBYS

Key	Value
Start (t)	17:02:41.158
End (t)	17:02:55.470
Duration (s)	14.312
Quantity (u)	131
Artifacts per minute (u)	549.19
Jenkins task	CD-DESA-SIARE-BACK-Bienes/798

Complementarily, the validation revealed a notable 68% reduction in configuration errors, demonstrating the system's ability to improve accuracy and reliability. The integration of automated validation mechanisms and detailed logging provided developers with valuable insights for debugging and optimization, further reinforcing the robustness of the solution.

An analysis of errors recorded by the client application monitoring tool (*Matomo*) was conducted using a filter on the number of services consumed for SIGEBYS *(/accounting/))* and the number of occurrences of the error message (*{"Insufficient permissions to…"}*) detected in the production environment during the last quarter of 2024 (Fig. 6). This analysis highlights the proportion of requests affected by authorization issues and their reduction following the implementation of SIARE-Artefactos, as summarized in Table 3.

Table 3: Error reduction

Year	Month	API Calls	Authorization Errors	Error Rate	Error Reduction
2024	10	5,184	227	--	--
2024	11	45,747	4,092	8.94%	--
2024	12	177,260	5,034	2.84%	68%

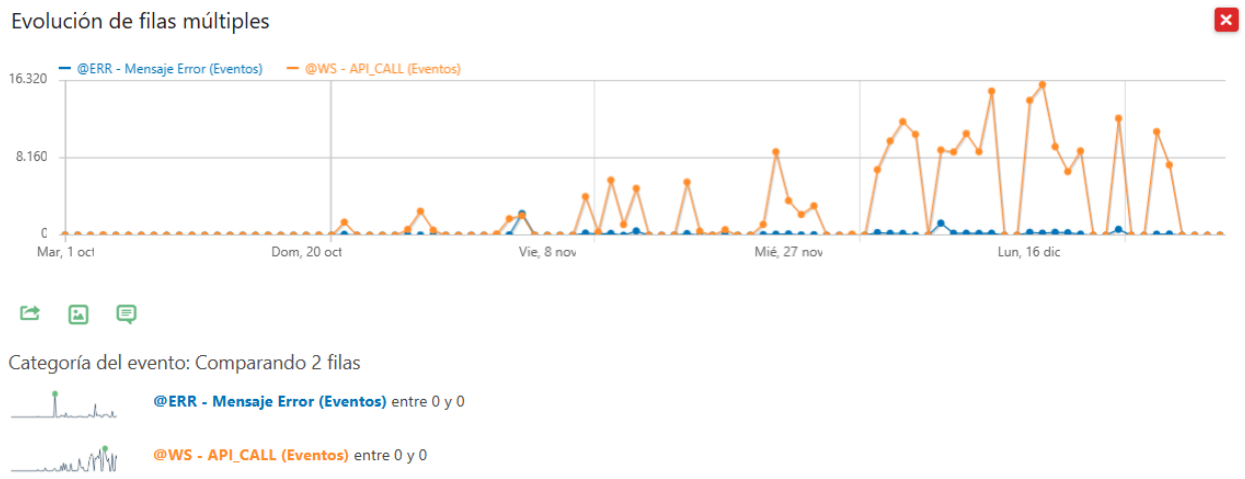

Figure 6: Evolution of requests and authorization errors

In terms of resource optimization, the implementation of SIARE-Artefactos resulted in a 74% reduction in manual effort, enabling technical teams to focus on higher-value activities such as strategic planning and system innovation.

Table 4: Monthly indicators – SIGEBYS(h)

Project / Metrics	2024-07	2024-08	2024-09	2024-10	2024-11	2024-12
SIGEBYS	30.5	48	41	7	10	14
Total	119.25	85.75	46.5	48.75	20	65.5
Quarterly Average	42.50	35.17	39.83	32.00	19.33	10.33
Variation	--	-7.3	4.66	-7.83	-12.67	-9

These results are further reflected in the analysis of time spent by the SIGEBYS– Asset Registration by Purchase project team, measured in monthly hours (Table 4), following the implementation of SIARE-Artefactos at the beginning of October 2024. The decrease in average time devoted to permission management reflects an approximate 74% reduction. The quarterly average was used because a quarter (six sprints) represents the standard product release cycle for projects, and some projects may not have releases—and therefore no artifact registration demand—in earlier periods. This calculation exclusively considered the SIGEBYS project (Fig. 7).

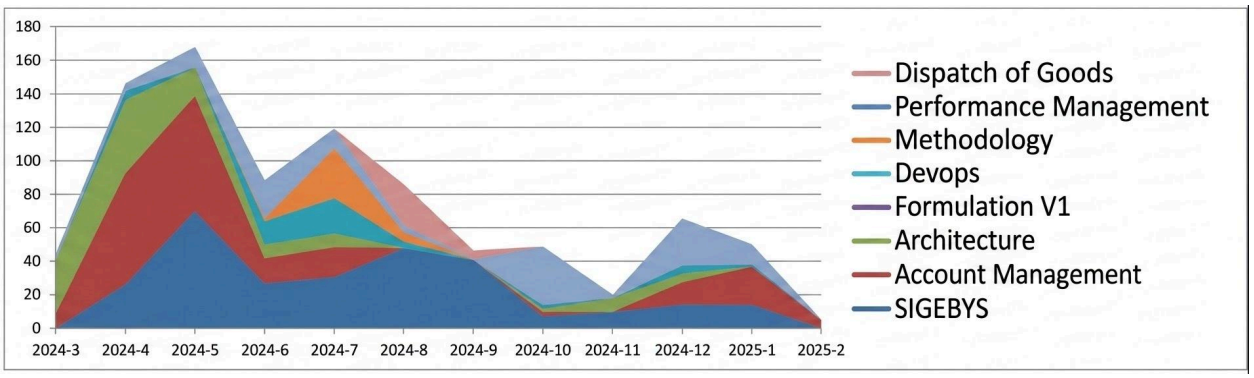

Figure 7:Manual authorization registration effort per project

The corresponding decrease in operational costs highlights the financial viability of adopting automated solutions for managing complex systems. Overall, these results demonstrate the significant advances enabled by SIARE-Artefactos, reaffirming its role as a key component in the modernization of public-sector IT infrastructures.

In a subsequent effort, SIARE-Autodiscovery was integrated into other SIARE solutions, including the budget and treasury projects, internal connectors, and external connectors, in addition to SIGEBYS. In these cases, the development effort required to add the component was only a few minutes, including automated endpoint registration testing. This suggests that the solution can be effectively scaled to other government or large-scale systems that require fine-grained access control in microservices-based architectures built on Spring Boot.

4.2 Qualitative and Economic Analysis

During a qualitative analysis and technical perception of the team, beyond performance metrics, semi-structured interviews were conducted with key technical actors (DevOps specialists, Scrum Masters, and Architects). Thematic analysis revealed that automation is perceived as a critical stability factor. Stakeholders highlighted the 'system predictability' and the elimination of recurrent human errors during deployments. A key finding was the low friction in adoption; by integrating transparently into the CI/CD pipeline, development teams did not require a steep learning curve as shown in Table 5.

Table 5: Qualitative Analysis

Thematic Category	Representative Quote	Analytical Interpretation
Operational Efficiency	**Drastic reduction in deployment time** "Previously, it took us more than an hour to register a complete service; now, with the artifact, registration is done in seconds."	Participants perceive an immediate and quantifiable improvement in deployment speed, which translates into greater operational agility and reduced waiting time for internal validations.
Operational Efficiency	**Reliable pipeline automation** "The process is predictable; there are no longer failures when running the scripts. Jenkins shows everything as green, with no intermediate errors."	Automation is recognized as a stabilizing factor in the DevOps workflow. Perceived reliability is associated with the elimination of manual steps and visual traceability provided by Jenkins.
Technical Quality of the Authorization Process	**Reduction in authorization errors** "Most of the 403 errors disappeared. Previously, logs were filled with access denied messages; now they are barely reported."	The observed reduction in error rates aligns with team perceptions. Interviewees associate this improvement with automated validation of permissions and endpoints.
Technical Quality of the Authorization Process	**Consistency between code and permissions** "The artifact takes what is defined in the code and publishes it exactly as is. It does not depend on someone copying or pasting incorrectly."	Consistency between the logical model and the authorization layer is perceived as a key technical achievement. This reinforces the causal relationship between automation and process quality.
Human Resource Utilization	**Reduction of manual effort** "The time we used to spend reviewing configurations is now used for functional testing and pipeline improvements."	The team perceives a positive reallocation of resources. Automation frees technical time and promotes a focus on higher value-added tasks.
Human Resource Utilization	**Learning and adoption without resistance** "It was easy to adopt. There was no steep learning curve; within a week we were already operating everything with artifacts."	Low adoption friction demonstrates good usability and intuitive design. This finding complements efficiency indicators with a technological acceptance component.
Human Resource Utilization	**Scalability and extension potential** "If this model is replicated in Treasury and Budget, we would have fully standardized access control."	Experience in the asset management system reinforces the perception that the model is scalable and replicable. Stakeholders identify cross-cutting benefits beyond the initial use case.

4.3 Economic Impact Analysis

Regarding resource efficiency, the analysis focused on the direct operational costs associated with manual permission management versus the automated approach. The data reveals a reduction in the average monthly hours dedicated to these tasks from 42.50 to 10.33 hours per project module (Table 4).

Based on the standard operational costs for senior technical personnel in the local public sector (approximately 14 USD / hour), it is estimated that monthly costs exclusively for configuration tasks were reduced from 595 USD to approximately 144,62 USD per module. This represents a direct efficiency gain of nearly 74%.

However, this financial calculation represents a conservative lower-bound estimate, as it excludes the indirect costs associated with "rework", Specifically, it does not quantify the savings derived from eliminating the diagnosis and correction time for the 68% of configuration errors that were prevented by the automated system.

Furthermore, the economic impact extends to the opportunity cost of human capital. By automating repetitive low-value tasks, the institution successfully reallocated highly skilled engineering hours toward strategic activities—such as architectural optimization and new feature development—thereby increasing the overall public value generated by the IT department without increasing the payroll.

4.4 Threats to Validity

This study follows the classification of validity threats proposed by Runeson and Höst (2009) [17]. The following limitations and mitigation strategies were identified:

A. Construct Validity: A primary threat lies in the solution's technological dependency on the WSO2 ecosystem, specifically the Identity Server and API Manager components. This strong coupling could potentially limit the portability of the results to environments using different API management providers. To mitigate this, the architecture implements a 'Provider Layer' that abstracts the communication logic with authorization and identity services. This design ensures that the theoretical concept—automated authorization registration—remains replicable and adaptable to other managers, such as Keycloak, Kong, or TyK, with minimal modifications.

B. Internal Validity: Simultaneous interventions in the SIARE infrastructure (e.g., security patches or infrastructure optimizations) during the observation period could introduce confounding variables affecting performance metrics. To control for these exogenous factors, the validation was isolated to the Goods Module (SIGEBYS). This module was selected due to its operational stability and high volume of deployments, providing a controlled environment. Furthermore, the study utilized historical baselines (pre-implementation data) to perform a comparative analysis, allowing improvements to be causally attributed to the SIARE-Artefactos intervention rather than external fluctuations.

C. External Validity: Regarding generalizability, it is important to acknowledge that the validation was performed in a specific on-premise environment within the Paraguayan Ministry of Economy. While this limits direct generalization to cloud-native or hybrid environments, the modular design facilitates future transition to such architectures. Additionally, the current implementation covers only backend components, excluding frontend authorization logic. However, the documented decoupled integration model allows for future extensions to frontend frameworks (e.g., Angular) and multi-cloud infrastructures without redesigning the core solution.

5 Conclusions and Future Work

This study addressed one of the most critical challenges in the technological evolution of modern public systems: access authorization management in microservices environments, characterized by high modularity and operational complexity using DevOps good practices. The SIARE-Artefactos solution has been developed, implemented, and is currently in use within the Paraguayan public sector. The validation performed in a real governmental environment demonstrated that the proposed approach is capable of operating under conditions of high authorization complexity, involving multiple institutional domains, heterogeneous user roles, and continuous deployment cycles.

Rather than focusing exclusively on performance optimization, the results suggest that authorization lifecycle automation can become a key enabler for governance, traceability, and operational consistency in large-scale public-sector microservices ecosystems.The module leverages code annotations to automatically identify and register resources, methods, and actions, eliminating the need for manual configuration.

The research was grounded in a rigorous mixed-methods approach, integrating technical analysis with organizational impact assessment within the Ministry of Economy and Finance of Paraguay. Following the methodological guidelines of Runeson and Höst for software engineering case studies, the validation combined

quantitative methods (operational metrics, automated logs) with qualitative methods (interviews and direct observation).

From an organizational perspective, the proposal reduced the operational dependency on manual configuration activities and minimized inconsistencies between deployment environments. This contributed not only to technical stability, but also to the standardization of deployment practices across teams, reinforcing the adoption of DevOps principles within the institutional modernization process.

Overall, this work provides empirical evidence that authorization lifecycle automation can evolve. It transitions from an operational support mechanism into a strategic governance capability for large-scale public digital ecosystems. This applies particularly to modernization initiatives based on microservices architectures and continuous delivery practices.

5.1. Scientific and Technical Contributions

This work presents a set of significant contributions to the field of automated access authorization in public management systems, grouped into three complementary axes:

• **Novel Architectural Design**: We designed a modular and scalable architecture capable of seamless integration with legacy and modern ecosystems. This architecture introduces a novel strategy based on source code annotations to automate the detection, registration, and configuration of REST endpoints. The definition of four main layers (Client, Startup, Provider, and Configuration) guarantees separation of concerns and facilitates maintainability.

• **Reusable Implementation Artifact**: We developed a reusable Spring Boot Starter module, allowing other government teams to incorporate the solution without altering business logic. The solution integrates centralized configuration tools (ConfigServer/Git) and caching (Redis), maximizing performance and consistency across environments. This component was successfully integrated into the Ministry's CI/CD pipelines, enabling automatic deployments with granular access control.

• **Empirical Validation Framework**: We designed and executed a rigorous validation process in a real production environment. The study proposes specific indicators to measure efficiency (configuration time), quality (error reduction), and resource optimization (manual effort reduction). This analysis not only validates the proposal but reinforces its practical relevance and potential for scalability in other public systems.

5.2. Future Work

As a natural continuity of this study, several lines of research and development have been identified. The first areas focus on Frontend Authorization Extension and Hybrid and Multi-Cloud Expansion. Additionally, future work will explore Predictive Analysis via Machine Learning.

The current implementation focuses on backend controllers. A critical next step is expanding the automation model to Angular-based frontend components. This involves implementing a complementary module to scan declared routes, identify protection tags or metadata, and automatically synchronize them with the central permission system. This would close the critical cycle of access control by ensuring semantic coherence between authorized views and granted permissions.

As public sector systems increasingly adopt cloud technologies, extending SIARE-Artefactos to operate efficiently in hybrid environments is essential. This would provide greater scalability and resilience while maintaining centralized control over heterogeneous infrastructures.

Finally, the incorporation of machine learning techniques is proposed to optimize resource management. ML models could analyze usage patterns and historical data to anticipate resource demand, detect security anomalies, and recommend proactive adjustments to authorization policies. This approach would provide decision-makers with actionable insights for continuous improvement.

These future enhancements aim to consolidate SIARE-Artefactos as a reference solution for the modernization of public administration systems. This consolidation ensures that they remain agile, secure, and capable of responding to evolving technological and organizational needs.

References

[1] M. Fowler and J. Lewis, "Microservices: a definition of this new architectural term," 2014. [Online]. Available: https://martinfowler.com/articles/microservices.html

[2] F. Zhang, G. Sun, B. Zheng, and L. Dong, "Design and implementation of energy management system based on Spring Boot framework," *Information*, vol. 12, no. 11, p. 457, 2021. [Online]. Available: http://dx.doi.org/10.3390/info12110457

[3] A. Granda Rivera, "Distributed authorization for microservices-based applications," Master's thesis, Universitat Oberta de Catalunya, Barcelona, Spain, 2020. [Online]. Available: http://openaccess.uoc.edu/webapps/o2/handle/10609/118346

[4] D. Hardt, Ed., "The OAuth 2.0 Authorization Framework," *IETF RFC 6749*, Oct. 2012. [Online]. Available: http://dx.doi.org/10.17487/RFC6749

[5] Gartner, "API Management: An Essential Component for Digital Business Platforms," 2019. [Online]. Available: https://www.gartner.com/

[6] C. Gardner, D. Beaton, and K. Hartig, "The Forrester Wave: API Management Solutions, Q3 2022," *Forrester Research*, 2022. [Online]. Available: https://www.forrester.com/

[7] M. B. Guayuan *et al.*, "Automatic deployment of microservices in API manager," in *Proc. 18th Iberian Conf. on Information Systems and Technologies (CISTI)*, Aveiro, Portugal, Jun. 2023. [Online]. Available: http://dx.doi.org/10.23919/CISTI58278.2023.10211516

[8] R. Palau Heikel, M. González, and L. Cernuzzi, "SIARE-Artefactos: Automatización de la gestión de autorizaciones para microservicios en sistemas de gestión gubernamental," in *Proc. 51st Latin American Informatics Conf. (CLEI)*, Valparaíso, Chile, 2025.

[9] R. Palau Heikel, M. González, and L. Cernuzzi, "Gestión de autorización de acceso a microservicios en sistemas de gestión pública: una propuesta de automatización," in *Proc. Ibero-American Conf. on Software Engineering (CIbSE 2025)*, Ciudad Real, Spain, May 2025, pp. 390-391. [Online]. Available: http://dx.doi.org/10.5753/cibse.2025.35335

[10] R. Valecha, M. Kashyap, S. Rajeev, R. Rao, and S. Upadhyaya, "An activity theory approach to specification of access control policies in transitive health workflows," in *International Conference on Information Systems*, 2014.

[11] A. Nehme and P. Jesus, "Fine-grained access control for microservices," in *Proc. 34th ACM/SIGAPP Symp. on Applied Computing (SAC)*, Limassol, Cyprus, 2019, pp. 137-144.

[12] R. Ahuja and S. K. Mohanty, "A scalable attribute-based access control scheme with flexible delegation cum sharing of access privileges for cloud storage," *IEEE Trans. Cloud Comput.*, vol. 8, no. 1, pp. 32-44, Jan.-Mar. 2020. [Online]. Available: http://dx.doi.org/10.1109/TCC.2017.2751471

[13] B. Burns, *Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services*. Sebastopol, CA, USA: O'Reilly Media, 2018. [Online]. Available: https://www.oreilly.com/library/view/designing-distributed-systems/9781491983638/

[14] N. Alshuqayran, N. Ali, and R. Evans, "A systematic mapping study in microservice architecture," in *Proc. 2016 IEEE 9th Int. Conf. on Service-Oriented Computing and Applications (SOCA)*, Macau, China, Nov. 2016, pp. 44-51. [Online]. Available: http://dx.doi.org/10.1109/SOCA.2016.15

[15] R. F. Palau Heikel, M. B. Guayuan, M. L. Cremona, H. Nemeth, and J. C. Mello-Román, "Development of Applications Based on Microservices - Case Study of Ministry of Economy and Finance of Paraguay," in *Proc. 50th Latin American Computer Conf. (CLEI)*, Buenos Aires, Argentina, 2024.

[17] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empir. Softw. Eng.*, vol. 14, no. 2, pp. 131-164, 2009. [Online]. Available: http://dx.doi.org/10.1007/s10664-008-9102-8