Secure Microservices Architecture for ICC-Topo: Cybersecurity Applied to a Graph-Based Curricular Analytics System

Anthony Pachay Espinoza and Alicia Andrade Vera(✉)

Universidad Estatal Península de Santa Elena (UPSE), La Libertad, Ecuador
{aespinoza,aandrade}@upse.edu.ec

Abstract. This paper presents the design and preliminary validation of a secure microservices architecture for ICC-Topo, a graph-based curricular analytics system for higher education. The objective was to develop an initial architecture that integrates DevSecOps principles and selected OWASP controls without compromising ICC computation. The study followed a quantitative, non-experimental, cross-sectional approach framed by Design Science Research and combined STRIDE and LINDDUN risk modeling, architectural design, prototype implementation, and functional, security, and exploratory performance assessment. The main results were the integration of six core components, end-to-end verification of the principal analytical flows and key security controls, successful processing of 52 of 57 curricular meshes, and a controlled concurrent-load characterization over representative analytical scenarios. These findings provide preliminary evidence that secure curricular analytics can be operationalized in a distributed academic platform, although broader privacy validation, ASVS coverage, and advanced DevSecOps automation remain open.

Keywords: Access control · DevSecOps · higher education security

1 Introduction

The digital transformation of higher education institutions has increased dependence on platforms that process curricular information and student trajectories to support academic decision-making, institutional evaluation, and program redesign. Within this context, graph-based curricular analytics has become a useful approach for representing curricula as prerequisite networks, identifying bottleneck courses, estimating structural complexity, and analyzing academic progression [23,26,28–30,37].

However, when these analytical capabilities are deployed as distributed services, the challenge is no longer exclusively mathematical or curricular. It also becomes a cybersecurity problem. The literature on microservices shows

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026 Workshops, LNCS 16762, pp. 159–176, 2027.
https://doi.org/10.1007/978-3-032-30524-4_11

that functional decomposition improves modularity, scalability, and independent deployment, but it also enlarges the attack surface, multiplies APIs, increases authentication and authorization complexity, and makes it more difficult to maintain coherent security policies across services [4,8,15,17]. In academic settings, this complexity is especially sensitive because the managed data include curricular structures, histories, trajectories, and patterns that may influence institutional decisions and therefore require protection in terms of confidentiality, integrity, availability, privacy, and traceability [16,23–25].

Recent literature has advanced on several related fronts, although usually in a fragmented manner: inter-service security and threat mitigation in microservices [3,15,32], Zero Trust and differentiated protection of North-South and East-West traffic [21], API gateway hardening through JWT and rate limiting [13,22,27,36], decoupled authorization through RBAC and token-based mechanisms [5,18,34], structured evaluation with OWASP ASVS and security maturity approaches [9,35], DevSecOps as a verifiable lifecycle property [1,2,6,7,10,11,14,33], and graph-oriented educational modeling [12,20,31]. Despite these advances, a clear integration gap remains because most studies do not converge into a single architecture for the academic domain that combines graph-based curricular analytics, distributed security, access control, student privacy, and reproducible validation [15,20,23,34].

From that gap, ICC-Topo becomes justified. The research proposal is aligned with the research line of Information Technology and Information Systems and with the subline of Networks and Information Security because it designs a secure microservices architecture for a system that processes institutional academic information and incorporates secure development practices, OWASP controls, and privacy-oriented mechanisms.

The scientific problem addressed in this paper is how to design and validate a secure microservices architecture for the ICC-Topo system that integrates DevSecOps practices and selected OWASP controls while protecting academic information. Accordingly, the objective of the article is to develop and validate such an architecture without compromising ICC computation. Unlike prior studies that address curricular analytics, API security, or DevSecOps practices in a fragmented manner, ICC-Topo integrates graph-based curricular analysis, distributed academic microservices, OWASP-aligned controls, STRIDE/LINDDUN risk modeling, and observable security validation into a unified architecture for higher-education curricular analysis. Its originality therefore lies in the operational integration of these dimensions within an academic analytical environment rather than in proposing an isolated security mechanism. The main contributions are fourfold: organizing the state of the art around security-aware curricular analytics, presenting a distributed architecture with implemented controls, documenting functional, security, and exploratory performance evidence, and explicitly identifying the elements that still require consolidation for advanced validation.

2 Theoretical Background and Related Work

2.1 Curricular Analytics and Graph-Based Educational Modeling

Curricular analytics interprets curricula as structured networks in which courses and prerequisites can be represented through graph-based models. This perspective supports the identification of bottlenecks, the estimation of structural complexity, and the analysis of academic progression. Prior work has shown the value of course-prerequisite networks and related graph-based methods for understanding curricular behavior and supporting curriculum assessment and redesign [26,28–30,37]. Additional educational graph initiatives, including knowledge-graph approaches, reinforce the relevance of graph-centered representations for precise and scalable academic analysis [20,31].

2.2 Security Challenges in Microservice Architectures

Microservice architectures provide scalability, modularity, and independent deployment, yet they also introduce increased API exposure, token abuse, inter-service trust problems, access-control inconsistencies, denial-of-service risks, and greater operational complexity [4,8,15,17]. Zero Trust principles, differentiated traffic protection, and layered security have therefore become central in distributed environments [3,21,32]. API gateways are commonly used for authentication, request filtering, observability, and rate limiting, but gateway-only protection is insufficient without internal validation and authorization [19,22,27,36]. Likewise, token-based authentication requires complementary authorization mechanisms such as RBAC [13,18,34].

2.3 DevSecOps, OWASP ASVS, and Privacy-Oriented Security

DevSecOps frames security as a verifiable property of the software lifecycle rather than as a late-stage addition and supports secure design, automated verification, dependency review, scanning, and operational traceability [1,2,6,7,10,11,14,33]. In parallel, OWASP ASVS offers a structured model for assessing web and API security across architecture, authentication, cryptography, access control, data protection, logging, and API-specific controls [9,35]. Privacy is equally relevant in academic systems because educational platforms process trajectories, patterns, and potentially identifiable combinations of attributes [16,24,25].

3 Methodology and Approach

3.1 Research Design

The study adopted a quantitative, non-experimental, and cross-sectional approach in accordance with the academic proposal. Design Science Research served as the methodological framework because the main outcome was a technological artifact that had to be designed, implemented, and evaluated against a practical and scientific problem.

The process was organized into three phases. Phase I focused on the security model and risk analysis through STRIDE for application and network threats and LINDDUN for privacy analysis. Phase II addressed the design of the microservices architecture, the selection of OWASP ASVS-aligned controls, secret-management considerations, and REST communication over HTTPS/TLS. Phase III covered implementation and validation through unit testing, integration checks, API security validation, and basic performance assessment. In methodological terms, privacy was addressed in this article at the design and control-selection level, whereas formal re-identification experiments and empirical k-anonymity verification were explicitly delimited as a later validation stage.

3.2 System Context and Data Inputs

ICC-Topo was conceived to support curricular complexity analysis from curricula modeled as directed acyclic graphs. The system manages academic catalogs, authenticates users, enforces role- and permission-based access control, executes ICC analysis over curricular structures, and stores results for historical consultation.

According to the proposal, the study relied on three classes of inputs: curricular reference data for UPSE and UTEQ, aggregated academic data for analytical processing, and synthetic data for functional and security testing. In the implemented prototype, information is loaded through Excel files with the sheets `courses` and `prerequisites`. These sheets encode subjects, credits, semesters, and prerequisite relationships.

For batch validation, 57 curricular meshes were considered. Of these, 52 were processed successfully and 5 were excluded because of input inconsistencies, specifically credits lower than or equal to zero and duplicated course identifiers. In total, the batch comprised 2423 courses and 1353 prerequisites, with averages of 42.51 courses and 23.74 prerequisites per processed mesh.

3.3 Experimental Environment and Reproducibility

The validation environment corresponded to the workstation `DESKTOP-DEV-PAC`, running Microsoft Windows NT 10.0.26200.0 on AMD64 architecture, with processor identifier `AMD64 Family 25 Model 97 Stepping 2, AuthenticAMD` and 31.9 GB RAM. The deployed functional architecture integrated an Angular frontend, Java-based gateway and services using Spring Boot and Spring Cloud Gateway, a Python analytical backend with FastAPI, and PostgreSQL persistence. During consolidation, a compilation issue in `spring-gateway` associated with Lombok usage without an active dependency was corrected, and Maven compilation of the gateway was validated for the reported phase.

Reproducibility was supported through observable artifacts and versions in the workspace. The frontend `front-icc` uses Angular 18.2.x, PrimeNG 18.0.0, Chart.js 4.5.1, and TypeScript 5.5.2. The services `spring-auth` and `spring-core` use Spring Boot 3.2.4 on Java 17, whereas `spring-gateway` uses

Spring Boot 3.2.4 with Spring Cloud Gateway 2023.0.0. The analytical engine `fastapi-icc` declares FastAPI 0.104.1, Uvicorn 0.23.2, Pydantic 2.5.3, NetworkX 3.2.1, Pandas 2.1.4, and Pytest 8.3.5. Persistence relies on PostgreSQL 15-alpine as declared in `docker-compose.yml`.

The deployment structure was documented in `docker-compose.yml`, where `postgres-db`, `spring-gateway`, `spring-auth`, `spring-core`, and `fastapi-icc` were explicitly defined, together with the reverse proxy and development frontend. Reproducibility was further supported by preserved validation artifacts, including the input template `template_malla.xlsx` and the generated output files `analyze_mallas_20260327_030726.json`, `analyze_mallas_20260327_171508.json` and `load_test_analyze_demo_20260327_173644.json`. The explicit exclusion criteria for meshes were credits lower than or equal to zero and duplicated course identifiers. A stable repository release identifier was not included in the validation artifact; reproducibility therefore relies on the reported versions, declared modules, and referenced artifacts.

3.4 Risk Modeling and Security Framing

Phase I formalized the security and risk-management model. STRIDE supported the identification of threats related to spoofing, tampering, repudiation, information disclosure, denial of service, and privilege escalation. LINDDUN was applied to analyze privacy risks such as re-identification, excessive traceability, and improper disclosure of student trajectories.

The resulting prioritization focused on unauthorized access to academic information, token theft or abuse, privilege escalation, massive data extraction, denial of service against APIs, and student re-identification through combinations of attributes and queries. The formal model also considered three dominant threat profiles: an automated external attacker, a curious or malicious authenticated user, and an insider or compromised administrator. Quantitative pre-control prioritization highlighted massive data extraction and token abuse as the most relevant risks, followed by re-identification, privilege escalation, and denial of service.

3.5 Architectural Design

In accordance with the proposal, the architecture integrates the following technological components: an Angular single-page application frontend, a security and gateway layer in Spring Boot and Spring Cloud Gateway, an authentication backend in Spring Boot, an analytical backend in Python with FastAPI, and PostgreSQL as the database engine.

The repository materializes this design in five modules: `front-icc`, `spring-gateway`, `spring-auth`, `spring-core`, and `fastapi-icc`. The implemented functional flow is:

```
front-icc -> spring-gateway -> spring-auth/spring-core ->
                fastapi-icc -> PostgreSQL
```

Under this design, the frontend does not consume the analytical engine directly. Instead, `spring-core` acts as the internal consumer, thereby reducing direct exposure and increasing control over internal access.

Architecturally, the public edge is located between the browser and `spring-gateway`, whereas communication among `spring-gateway`, `spring-auth`, `spring-core`, and `fastapi-icc` corresponds to the internal service plane. This separation distinguishes North-South flows for external access and East-West flows for internal service coordination.

Figure 1 formalizes the implemented architecture, trust boundaries, and associated controls introduced in this design layer.

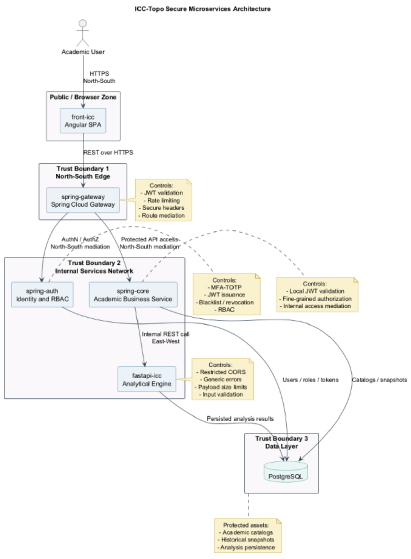

Fig. 1. Formal architecture of ICC-Topo, including components, trust boundaries, and North-South and East-West flows.

3.6 OWASP Controls, DevSecOps Orientation, and Validation Protocol

Based on the prioritized risks, the study selected OWASP ASVS Level 2-related categories for architecture, authentication, access control, session management, error handling, data protection, and API security. In parallel, Phase II established a DevSecOps framework intended to make security a verifiable lifecycle property. The framework mapped prioritized risks to ASVS categories covering architecture, authentication, cryptography, access control, data protection, logging, and API security. It also established as a reference horizon the future integration of SAST, dependency analysis, SBOM generation, container scanning, and DAST baseline testing.

The prototype implemented the following observable controls: JWT authentication with `jti`, blacklist-based invalidation in logout and refresh, JWT validation at the gateway, additional local validation in `spring-core`, granular authorization based on role, option, and action, IP-based rate limiting, secure headers at the gateway, restricted CORS in FastAPI, generic error responses in the analytical engine, temporary lockout after failed attempts, anti-DoS limits on analytical payloads, and TOTP-based multi-factor authentication for administrators.

The hardened operational parameters were the following (Table 1).

Table 1. Implemented hardening parameters

Hardening parameter	Value
Failed attempts before lockout	5
Temporary lockout duration	15 min
Login request limit	5 per 60 s
Maximum analytical request size	1 MB
Maximum number of courses per mesh	200
Maximum number of prerequisites per mesh	800

These values defined the protective thresholds adopted for the prototype and framed the subsequent validation activities.

Phase III assessed the architecture through functional testing of demo analysis, persisted analysis using `mallaId`, and analysis history retrieval, as well as CRUD validation for the Users (`Usuarios`), Menu Options (`Opciones`), Organizations (`Organizaciones`), Programs (`Programas`), and Curricular Meshes (`Mallas`) modules. Security validation included end-to-end checks of JWT blacklist, rate limiting, granular authorization in `spring-auth` and initial granular authorization in `spring-core`, functional validation of administrator MFA, and `pytest` plus Bandit execution over `fastapi-icc`. Exploratory performance evidence relied on endpoint latency measurements, a controlled gateway-facing rate-limiting check on demo analysis, and an additional concurrent-load campaign over representative analytical scenarios. Because rate limiting was configured as a protective policy, HTTP 429 responses in the burst series were treated as evidence of expected control activation rather than as system failure.

A functional flow was considered validated when it completed the expected end-to-end path and returned the corresponding system response. A security control was considered validated when its expected activation, blocking, or rejection behavior was observed under the tested condition. Performance was treated as exploratory in this phase and was assessed through basic latency observations and controlled activation of rate limiting. Accordingly, the fourth specific objective was interpreted as prototype validation through functional and operational security evidence, while advanced privacy-risk quantification was kept outside the empirical scope of the present phase.

Table 2. Prioritized STRIDE/LINDDUN risks and controls

Model	Threat	Asset	Control	Status
STRIDE	Massive data extraction	Gateway and analysis endpoints	Rate limiting, payload limits, edge control	Implemented
STRIDE	Token theft or abuse	Auth, gateway, core	JWT with `jti`, blacklist, MFA	Implemented
STRIDE	Privilege escalation	Auth and core	RBAC and backend double validation	Partial
STRIDE	Denial of service	Gateway and analytical engine	IP-based rate limiting and request-size limits	Implemented
LINDDUN	Re-identification through combined filters	Analytical service and exports	k-anonymity threshold and query restriction	Pending
LINDDUN	Linkability and disclosure	Outputs, logs, exports	Data minimization and aggregated outputs	Partial

Table 3. Validation evidence for prioritized threats

Threat	Evidence	Status
Massive data extraction	`429` triggered in load series B and login threshold observed	Implemented
Token theft or abuse	JWT blacklist validated; `challengeToken` rejected (`401`)	Implemented
Privilege escalation	RBAC validated in `spring-auth`; initial local checks in `spring-core`	Partial
Denial of service	Request-size limit configured to 1 MB; `429` activation observed	Implemented
Re-identification through combined filters	Risk modeled; empirical re-identification tests still pending	Pending
Linkability and disclosure	Individual student queries are not exposed; inference validation remains open	Partial

4 Results

4.1 Formalized Security Model and DevSecOps Framework

The first result was the formalization of the security and risk-management model, thereby fulfilling the first specific objective of the proposal. Critical assets, CIA-oriented considerations, and major threats affecting authentication, authorization, availability, privacy, and traceability were identified. The second result was the definition of a DevSecOps framework articulated with OWASP ASVS categories, which established an explicit relationship among risks, controls, and expected evidence.

Tables 2 and 3 summarize the prioritized risks, applied controls, and available evidence, preserving traceability between modeling and validation.

At the current stage, ASVS should be interpreted as an implementation matrix rather than as a closed compliance percentage. Table 4 therefore sepa-

rates what is already implemented from what remains partial or pending, thereby avoiding an overstatement of achieved maturity.

Table 4. Summarized ASVS matrix for the ICC-Topo prototype

ASVS	Control	Implementation snapshot	Status
V1 Architecture	Separation of domains and defense in depth	Gateway, auth, core, analytics, and persistence are decoupled	Implemented
V2 Authentication	JWT lifecycle, lockout, MFA	JWT with `jti`, temporary lockout, and TOTP-based MFA	Implemented
V3 Cryptography	Protection of tokens and secrets	JWT signing active; vault-based secret management pending	Partial
V4 Access control	Least-privilege authorization	RBAC by role, option, and action; local checks in core	Partial
V7 Error handling and logging	Error handling and traceability	Generic responses active; centralized traceability incomplete	Partial
V8 Data protection	Minimization and restricted outputs	Aggregated results only; formal anonymization still pending	Partial
V10 API security	Traffic abuse control	Rate limiting, restricted CORS, and payload limits	Implemented

The detailed evidence behind this matrix is provided in the risk tables and in the results on MFA, blacklist validation, rate limiting, and restricted analytical inputs.

4.2 Implemented Architecture and Functional Validation

The implemented architecture consisted of six main components: Angular frontend, gateway, authentication service, academic core, analytical engine, and PostgreSQL persistence. This separation of responsibilities materialized the third specific objective and aligned with the proposed distributed design. For the analytical path, the consolidated route was `front-icc -> spring-gateway -> spring-core -> fastapi-icc`, while authentication and permission management remained centralized in `spring-auth`. PostgreSQL supported catalogs and analytical snapshots. Table 5 links the implemented layers with their main trust boundaries and associated controls.

Functional validation showed correct operation in three main ICC-domain flows: demo analysis, persisted analysis with `mallaId`, and analysis history

Table 5. Components, trust boundaries, and associated controls

Layer or boundary	Main components	Flow type	Associated controls
Public edge	Browser and `front-icc`	North-South	Initial authentication, authenticated session, and no direct access to the analytical engine
Service perimeter	`spring-gateway`	North-South	JWT validation, secure headers, rate limiting, and route control
Identity and authorization	`spring-auth`	East-West internal	Token issuance and revocation, TOTP-based MFA, and RBAC
Academic business logic	`spring-core`	East-West internal	Additional local JWT validation and fine-grained authorization
Internal analytics	`fastapi-icc`	East-West internal	Restricted CORS, generic errors, and anti-DoS payload limits
Persistence	PostgreSQL	Internal	Support for catalogs and snapshots; advanced privacy safeguards still require formal validation

retrieval. In addition, five administrative CRUD modules were validated: Users, Menu Options, Organizations, Programs, and Curricular Meshes. The frontend supported Excel upload, template download, pre-validation of files, indicator visualization, the `iccBySemester` chart, structural graph rendering, and tabular views, confirming that the platform already operates as a curricular analytics prototype.

4.3 Observable Security Results

At the current stage, four security controls were functionally and end-to-end verified: JWT blacklist, rate limiting, granular authorization based on roles, options, and actions, and administrator multi-factor authentication. The architecture also incorporates token revocation, double JWT validation, anti-DoS limits, restricted CORS, generic responses, and temporary lockout after failed attempts.

Within the MFA flow, it was verified that administrative login could return intermediate setup and verification states, that a valid TOTP code completed authentication, and that the `challengeToken` could not be used as a final token on protected endpoints. These observations show that the implemented controls were not only configured but also operational under the tested scenarios.

4.4 Integrated Visual Evidence

Figure 2 summarizes representative interface evidence supporting authentication, analysis execution, and persistence behavior; the remaining screenshots are treated as supplementary material.

4.5 Automated Evidence

In the analytical engine `fastapi-icc`, the current automated test suite reported 9 tests passed in 1.05 s. This result supports the basic consistency of both ICC computation and the FastAPI implementation. Additionally, the local Bandit execution reported no findings, indicating the absence of evident static-security issues in the current pass. Although this result does not replace a comprehensive security evaluation, it does provide favorable technical evidence for the prototypical stage.



Fig. 2. Representative interface evidence: administrator MFA verification, curricular-mesh analysis, and persisted execution history.

Table 6. Observed latency in main endpoints

Endpoint	Requests	Min (ms)	Avg. (ms)	Max (ms)	Observation
Login	5	55.81	207.04	290.54	The next attempt returned `429`, confirming rate limiting
Demo analysis	3	15.33	108.20	288.26	First request was higher because of warm-up cost
Persisted analysis	3	17.57	77.13	194.98	Initial request was higher, then stabilized
History detail	3	15.03	87.47	228.13	Query stabilized after first access

4.6 Batch Processing and Data Quality

Batch runs performed with the Python script registered 92 attempts, of which 52 returned HTTP 200, 32 corresponded to transient HTTP 401 responses, and 8 were associated with local data-quality errors. After consolidation by mesh, 57 meshes were considered, with 52 successful cases and 5 excluded because of input inconsistencies.

The excluded meshes presented two types of issues: courses with credits lower than or equal to zero and duplicated course identifiers. This finding was not interpreted as a system failure but as evidence that input-validation mechanisms prevented the processing of defective curricular structures and protected traceability of the analysis.

4.7 Exploratory Performance Evidence

Table 6 summarizes the observed latency in the main system endpoints.

The earliest requests showed higher latency because of initialization and cache warm-up. Later runs declined toward lower and more consistent values, a behavior coherent with distributed architectures involving component startup, security validation, and persistence access.

Table 7. Controlled concurrent-load observations in the local environment

Scenario	Concurrent users	Avg. (ms)	p95 (ms)	Error rate	Req/s
Demo analysis	5	17.02	27.69	0.00	293.80
Demo analysis	10	31.88	49.80	0.00	313.20
Demo analysis	20	59.00	78.08	0.00	338.30
Persisted analysis	5	19.87	29.97	0.00	251.55
Persisted analysis	10	33.65	50.58	0.00	297.35
Persisted analysis	20	65.53	86.77	0.00	305.70
History detail	5	6.14	7.52	0.00	813.50
History detail	10	7.78	9.75	0.00	1284.15
History detail	20	10.08	12.50	0.00	1982.75

To strengthen the validation, an additional controlled concurrent-load campaign was executed in the local environment using a valid curricular mesh with 40 courses and 35 prerequisites. Three representative scenarios were evaluated: demo analysis, persisted analysis, and history-detail retrieval. For each scenario, the system was exercised with 5, 10, and 20 concurrent virtual users during a fixed 20-second window. The observed metrics included average latency, 95th-percentile latency, error rate, and throughput in requests per second.

Because the gateway-level rate-limiting policy is intentionally restrictive for abuse prevention, the concurrent campaign was configured to characterize business-flow behavior without letting that protective threshold dominate

the measurements. Accordingly, the measurements should be interpreted as exploratory but reproducible evidence of prototype behavior under controlled local concurrency, not as a full-scale production benchmark.

Table 7 summarizes the observed results.

An additional gateway-facing check on demo analysis showed stable responses under a short sequence of 10 consecutive requests, whereas a subsequent 20-request burst activated the configured rate-limiting threshold and returned HTTP `429`. These responses were therefore interpreted as expected abuse-prevention behavior rather than as service degradation.

The concurrent-load results show a consistent increase in latency as concurrency rises, which is expected in the analyzed flows, while the error rate remained at zero in all tested cases. Demo and persisted analysis exhibited higher response times than history-detail retrieval because they involve analytical execution and additional backend coordination. In contrast, the history-detail scenario showed the highest throughput and the lowest latency range, indicating that read-oriented retrieval remained comparatively lightweight in the tested environment.

Resource observations during the campaign showed that the academic core concentrated the highest operational load among the application services, whereas the analytical backend and auxiliary security services remained comparatively stable during the measured window. These findings do not yet constitute a full resource-consumption model, but they do provide initial evidence that the prototype remained operational under controlled concurrent access without functional failure in the tested scenarios.

4.8 Quantitative Synthesis

In quantitative terms, the validated prototype integrated six main components, three end-to-end ICC flows, five administrative CRUD modules, and four security controls with direct observable evidence. Batch execution considered 57 meshes, of which 52 were processed successfully and 5 were excluded because of data-quality issues. The processed set comprised 2423 courses and 1353 prerequisites, with averages of 42.51 courses and 23.74 prerequisites per mesh. In addition, the `fastapi-icc` test suite reported 9 passing tests in 1.05 s, while the enforced hardening thresholds remained fixed at 5 failed attempts before lockout, 15 min of temporary lockout, 5 login requests per 60 s, 1 MB per analytical request, 200 courses per mesh, and 800 prerequisites per mesh.

5 Discussion

The results show that the proposal moved from conceptual design to a distributed prototype with direct observable evidence. This is consistent with literature arguing that microservice security should be implemented and tested rather than stated only at the principle level [4,8,15]. The architecture aligns with prior work on gateway-mediated protection, JWT-based authentication,

RBAC, and measurable DevSecOps and ASVS practices [1,7,9,18,22,27,34–36]. Its distinguishing factor is contextual: the protected assets are curricular structures, trajectories, and institutional indicators rather than generic enterprise records [23,29,30,37].

The empirical evidence supports the applied relevance of the artifact. Processing 52 of 57 curricular meshes shows that the system goes beyond a toy scenario, while the exclusion of 5 inconsistent meshes suggests that input-validation mechanisms protected analytical integrity instead of forcing invalid computations. MFA, blacklist validation, RBAC checks, and rate limiting provide a coherent set of controls, although broader validation is still needed.

From an applied university perspective, these results also have governance implications. Separating trust boundaries between external access, identity management, academic business logic, and analytical execution reduces direct exposure of the analytics engine and improves control over curricular indicators that may inform accreditation, redesign, and institutional decision-making. In that sense, the contribution is not limited to API hardening; it also supports a more controlled handling of academic data and institutional metrics in distributed higher-education environments.

Performance evidence should still be interpreted as preliminary only. The measured latencies, the concurrent-load campaign, and the controlled rate-limiting activation provide stronger operational observations under local conditions, but they do not amount to a production-capacity benchmark or a full multi-environment resource characterization.

Table 8. Conceptual comparison between a simpler baseline and the ICC-Topo prototype

Dimension	Simpler baseline	ICC-Topo prototype	Evidence status
Analytical service exposure	Frontend may consume analytical endpoints directly	Access mediated through `spring-gateway` and `spring-core`	Implemented
Authentication and business logic	Coupled or partially shared backend responsibilities	Identity handled in `spring-auth`; business logic isolated in `spring-core`	Implemented
Token validation	Single validation point or gateway-only enforcement	Gateway validation plus additional local validation in `spring-core`	Implemented
Authorization granularity	Coarse endpoint protection	RBAC by role, option, and action	Partial
Abuse resistance	No explicit throttling or payload restrictions	IP-based rate limiting and analytical payload limits	Implemented
Administrative hardening	Password-only administration	Temporary lockout and TOTP-based MFA for administrators	Implemented
Privacy safeguards	Limited separation between query and export concerns	Aggregated outputs and restricted analytical exposure; formal anonymization pending	Partial
Validation maturity	Functionality may be demonstrated without explicit security evidence	Functional, security, and exploratory performance evidence over the prototype	Partial

5.1 Comparison with a Conceptual Baseline

Although no A/B experiment or monolithic benchmark was performed, a conceptual comparison can still be made with a simpler baseline in which the frontend consumes the analytical engine directly or in which authentication, authorization, and business logic remain coupled. Relative to that baseline, ICC-Topo introduces gateway mediation, separation between identity and business logic, additional internal validation in `spring-core`, and reduced direct exposure of the analytical service.

Table 8 shows that the improvement demonstrated at this stage is mainly architectural and related to controllability, reduced direct exposure, and observability of security controls rather than to a universal quantified reduction of risk.

5.2 Privacy, Data Minimization, and Remaining Risks

The privacy dimension must be interpreted more cautiously than the operational security dimension. The system works with aggregated results, does not expose individual student queries, and limits export to the aggregated plane, which constitutes an initial data-minimization strategy aligned with the developed LINDDUN model. However, risks of linkability, disclosure, and inference cannot be considered closed solely because of functional restriction. Formal validation of k-anonymity, measurement of re-identification risk, and verifiable suppression or anonymization techniques remain pending tasks.

5.3 Threats to Validity

Internal validity is conditioned by the fact that most of the evidence comes from functional testing, operational control validation, and exploratory performance measurements in a local environment. External validity is moderate because the results rely on 57 curricular meshes and on a specific academic context. Construct validity also requires caution because the observed security reflects activation of selected controls, not comprehensive certification against the full ASVS catalog or all LINDDUN scenarios. Additional limitations include warm-up effects, transient `401` responses during batch runs, bounded concurrent-load scenarios, and the lack of empirical privacy-risk quantification.

5.4 Limitations and Future Work

The study still has clear limits. Although the current version includes a controlled concurrent-load campaign with latency, percentile, throughput, and initial resource observations, the evidence comes from a local environment and should not be interpreted as a production-capacity benchmark. Pending elements therefore include multi-environment performance replication, deeper per-service resource profiling, an evidence-based ASVS compliance percentage, formal re-identification and k-anonymity testing, CI/CD security automation, and reproducible SAST, SCA, and DAST reports.

6 Conclusions

The study made it possible to develop a secure microservices architecture for ICC-Topo, aligned with the academic proposal and grounded in an explicit sequence of risk modeling, control selection, prototype implementation, and initial validation. The integration of frontend, gateway, authentication, academic business logic, analytical engine, and persistence indicates that the project already has a technological base for graph-based curricular analytics. In that sense, the work fulfills the general objective and the first three specific objectives through explicit risk modeling, DevSecOps and OWASP framing, and implemented architectural separation with verifiable controls.

The main empirical contribution lies in connecting prioritized risks, implemented controls, and verifiable evidence across three ICC flows, five administrative CRUD modules, MFA, JWT blacklist, rate limiting, batch processing, automated testing of the analytical engine, and controlled concurrent-load observations. This evidence also supports fulfillment of the fourth specific objective at the level of functional, security, and exploratory performance validation. Nevertheless, the scope must remain realistic: ASVS coverage, formal privacy validation, and advanced security automation still require consolidation. Consequently, the present contribution should be understood as an implemented prototype with a clear methodological path toward a more complete validation phase. The study also demonstrates that security-aware curricular analytics can move beyond conceptual discussion toward an operational distributed prototype with integrated cybersecurity controls for higher-education environments.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Abdiukov, T.: Automated security testing in devsecops pipelines: integrating AI-based vulnerability discovery and compliance validation. World J. Adv. Res. Rev. **22**(01), 2083–2093 (2024). https://doi.org/10.30574/wjarr.2024.22.1.1083
2. Abiona, O.O., Oladapo, O.J., Modupe, O.T., Oyeniran, O.C., Adewusi, A.O., Komolafe, A.M.: The emergence and importance of devsecops: Integrating and reviewing security practices within the devops pipeline. World J. Adv. Eng. Technol. Sci. **11**(2), 127–133 (2024). https://doi.org/10.30574/wjaets.2024.11.2.0093
3. Alboqmi, R., Gamble, R.F.: Enhancing microservice security through vulnerability-driven trust in the service mesh architecture. Sensors **25**(3), 914 (2025). https://doi.org/10.3390/s25030914
4. Aldea, C.L., Bocu, R.: Authentication challenges and solutions in microservice architectures. Appl. Sci. **15**(22), 12088 (2025). https://doi.org/10.3390/app152212088
5. Araújo, L., Marinho, E.: Desafios da autenticação e autorização na comunicação entre serviços em arquiteturas de microsserviços. CESAR School (). Tech. rep
6. Avuthu, Y.R.: Microservices security threat modelling in devops pipelines. J. Math. Comput. Appl. (2023). https://doi.org/10.47363/JMCA/2023(2)E138

7. Bautista Ramos, R.C., Yoo, S.G.: Cybersecurity in devops environments: a systematic literature review. IEEE Access (2025). https://doi.org/10.1109/ACCESS.2025.3582892
8. Berardi, D., Giallorenzo, S., Mauro, J., Melis, A., Montesi, F., Prandini, M.: Microservice security: a systematic literature review. PeerJ Comput. Sci. **8**, e779 (2022). https://doi.org/10.7717/peerj-cs.779
9. Brezavšček, A., Baggia, A.: Recent trends in information and cyber security maturity assessment: a systematic literature review. Systems **13**(1), 52 (2025). https://doi.org/10.3390/systems13010052
10. Chandramouli, R.: Implementation of devsecops for a microservices-based application with service mesh. Tech. Rep. NIST Special Publication 800–204C (Draft), National Institute of Standards and Technology (2021). https://doi.org/10.6028/NIST.SP.800-204C
11. Fu, M., Pasuksmit, J., Tantithamthavorn, C.: AI for devsecops: a landscape and future opportunities. ACM Trans. Softw. Eng. Methodol. **34**(4), 115 (2025). https://doi.org/10.1145/3712190
12. Garzón, B., Perri, V., Qarkaxhija, L., Scholtes, I., Tomasik, M.J.: Exploring the structure of the school curriculum with graph neural networks. J. Comput. Soc. Sci. **8**, 99 (2025). https://doi.org/10.1007/s42001-025-00420-9
13. Gbenle, T.P., Abayomi, A.A., Uzoka, A.C., Ogeawuchi, J.C., Adanigbo, O.S., Odofin, O.T.: Applying oauth2 and JWT protocols in securing distributed API gateways: best practices and case review. Int. J. Multidis. Res. Growth Eval. **3**(5), 628–634 (2022). https://doi.org/10.54660/.IJMRGE.2022.3.5.628-634
14. Gupta, A.: An integrated framework for devsecops adoption. Int. J. Comput. Trends Technol. **70**(6), 19–23 (2022). https://doi.org/10.14445/22312803/IJCTT-V70I6P102
15. Haindl, P., Kochberger, P., Sveggen, M.: A systematic literature review of inter-service security threats and mitigation strategies in microservice architectures. IEEE Access **11**, 1–35 (2024). https://doi.org/10.1109/ACCESS.2024.3406500
16. Heinrich, M., Gerhart, N.: Privacy education effectiveness: does it matter? J. Inf. Syst. Educ. **34**(1), 49–69 (2023)
17. Kothapalli, M.: Securing microservices architecture: best practices and challenges. J. Sci. Eng. Res. **8**(10), 187–192 (2021). https://doi.org/10.5281/zenodo.12772079
18. Kurniawan, M.C., Gamayanto, I., Sanyoto, G.K., Widjajanto, B.: Security evaluation of keycloak-based role-based access control in microservice architectures using the owasp ASVS framework. J. Appl. Inf. Comput. (JAIC) **9**(6), 3964–3973 (2025)
19. Li, H., Li, J., Wang, Y., Zhou, C., Yin, M.: Leaving the business security burden to lisea: a low-intervention security embedding architecture for business APIs. Appl. Sci. **13**(21), 11784 (2023). https://doi.org/10.3390/app132111784
20. Li, Y., Liang, Y., Yang, R., Qiu, J., Zhang, C., Zhang, X.: Coursekg: an educational knowledge graph based on course information for precision teaching. Appl. Sci. **14**(7), 2710 (2024). https://doi.org/10.3390/app14072710
21. Mushtaq, S., Mohsin, M., Mushtaq, M.M.: A systematic literature review on the implementation and challenges of zero trust architecture across domains. Sensors **25**(19), 6118 (2025). https://doi.org/10.3390/s25196118
22. Nadipalli, R.: Security challenges in API gateway design for cloud-based applications. Int. J. Sci. Res. (IJSR) **12**(9), 2244–2248 (2023). https://doi.org/10.21275/SR23914095135
23. Paz, H.R.: The topology of hardship: empirical curriculum graphs and structural bottlenecks in engineering degrees. Preprint (2025). https://doi.org/10.48550/arXiv.2512.05561

24. Radway, S., Quintanilla, K., Ludden, C., Votipka, D.: An investigation of us universities' implementation of ferpa student directory policies and student privacy preferences. In: Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24) (2024). https://doi.org/10.1145/3613904.3642066
25. Sanfilippo, M.R., Apthorpe, N., Brehm, K., Shvartzshnaider, Y.: Privacy governance not included: analysis of third parties in learning management systems. Inf. Learn. Sci. **124**(9/10), 326–348 (2023). https://doi.org/10.1108/ILS-04-2023-0033
26. Schellpfeffer, S.E.: An introduction to curricular analytics: research and applications for curriculum assessment and review. Inter. J. Int. Assess. Learn. **6**(2), 128–138 (2025)
27. Shivam, Gupta, M.: Secure API gateway with rate limiting and JWT authentication. Int. J. Res. Appl. Sci. Eng. Technol. (IJRASET) (2025). https://doi.org/10.22214/ijraset.2025.68953
28. Slim, A., et al.: Enhancing academic pathways: a data-driven approach to reducing curriculum complexity and improving graduation rates in higher education. In: 2024 ASEE Annual Conference & Exposition (2024). https://doi.org/10.18260/1-2-47298
29. Srivastava, N., Nawaz, S., Tsai, Y.S., Gašević, D.: Curriculum analytics of course choices: Links with academic performance. J. Learn. Anal. **11**(1), 116–131 (2024). https://doi.org/10.18608/jla.2024.8095
30. Stavrinides, P., Zuev, K.M.: Course-prerequisite networks for analyzing and understanding academic curricula. Appl. Netw. Sci. **8**(1), 19 (2023). https://doi.org/10.1007/s41109-023-00543-w
31. Stelea, G.A., Robu, D., Sandu, F.: Accessilearnai: an accessibility-first, ai-powered e-learning platform for inclusive education. Educ. Sci. **15**(9), 1125 (2025). https://doi.org/10.3390/educsci15091125
32. Thiyagarajan, G., Bist, V., Nayak, P.: Strengthening grpc security in microservices: a proxy-based approach for MTLS, JWT, and RBAC enforcement. Int. J. Comput. Appl. **187**(28) (2025)
33. Valenti, M.T.: Secure devsecops integration for retail cloud resilience: strategies, challenges, and theoretical perspectives. Int. J. Adv. Sci. Res. **6**(02), 01–10 (2026)
34. Venčkauskas, A., Kukta, D., Grigaliūnas, Š, Brūzgienė, R.: Enhancing microservices security with token-based access control method. Sensors **23**(6), 3363 (2023). https://doi.org/10.3390/s23063363
35. Wen, S.F., Katt, B.: A quantitative security evaluation and analysis model for web applications based on OWASP application security verification standard. Comput. Secur. **135**, 103532 (2023). https://doi.org/10.1016/j.cose.2023.103532
36. Xu, R., Jin, W., Kim, D.: Microservice security agent based on API gateway in edge computing. Sensors **19**(22), 4905 (2019). https://doi.org/10.3390/s19224905
37. Yang, B., Gharebhaygloo, M., Rondi, H.R., Banu, S.Z., Huang, X., Ercal, G.: Analysis of student progression through curricular networks: a case study in an illinois public institution. Electronics **14**, 3016 (2025). https://doi.org/10.3390/electronics14153016