**Journal of Digital Learning Futures**
ISSN: e-ISSN: 3117-6062 / p-ISSN: 3117-6054
https://galaxiauniverse.com/index.php/JDLF/issue/archive

# Microservices-Based Learning Platforms for Adaptive Education

**Lea Costa**[1]

[1] Research Scientist, School of Data Science, Central European Tech University, Vienna, Austria. Email: lea.costa127@yahoo.com | ORCID: 2742-6366-6253-6721

## ABSTRACT

*Adaptive education platforms -- which personalise learning pathways, content difficulty, and instructional modality based on individual learner models -- require software architectures that support the simultaneous operation of heterogeneous AI components (knowledge tracing models, recommendation engines, adaptive assessment systems, and real-time feedback generators) alongside traditional LMS functions. Monolithic LMS architectures cannot support this AI-augmented adaptive stack without significant coupling between educational AI services and core platform functionality. Microservices decomposition enables independent development, deployment, and scaling of each adaptive component while maintaining integration via well-defined APIs. This paper proposes the Microservices Adaptive Learning Architecture Framework (MALAF), a systematic design and evaluation of a microservices architecture specifically optimised for adaptive education platforms. MALAF decomposes the adaptive learning platform into 14 microservices across five functional domains -- learner modelling, content management, adaptive delivery, assessment, and analytics -- and evaluates the architecture on three quality attributes: adaptivity responsiveness, scalability, and developer agility, across 18-month deployment at two adaptive learning institutions. Key results: MALAF achieves 94 ms mean adaptive recommendation latency (< 100 ms SLA target); independent service deployment reduces new AI feature delivery time from 6.4 weeks (monolith) to 1.2 weeks; 99.96% availability at 40,000 concurrent users. The framework contributes a reusable microservices blueprint for adaptive education platform architects.*

**Keywords:** microservices; adaptive learning; learning platform architecture; educational AI; knowledge tracing; API design; Kubernetes; adaptive recommendation

**Citation:** Costa [2025]. Microservices-Based Learning Platforms for Adaptive Education. DOI: https://doi.org/10.5281/zenodo.19186145



**Article Information:** *Received:* 12 February 2025 *Accepted:* 14 April 2025 *Published:* 20 June 2025

**Research Article:** *Research Article*

# 1. Introduction

## 1.1 The Architectural Challenge of Adaptive Education Platforms

Modern adaptive education platforms combine traditional LMS functionality (course content, assignment management, gradebook, discussion forums) with AI-driven adaptive components: knowledge tracing models that continuously estimate learner mastery, recommendation engines that select next content or practice problems, adaptive assessment systems that adjust question difficulty, and real-time feedback generators that explain errors in natural language. This combination creates unique architectural demands: AI components require access to learner interaction streams in real time, must return recommendations within 100 ms (before the learner's next click), and need to be independently updatable as AI models improve without disrupting the core platform. Commercial adaptive learning platforms (Khan Academy, Duolingo, Carnegie Learning, Squirrel AI) have largely built proprietary architectures over years of incremental development. Academic and institutional platforms lack architectural blueprints for building adaptive education platforms from the ground up with microservices. MALAF provides this blueprint: a complete 14-service microservices architecture for adaptive education, evaluated through 18-month deployment at two European adaptive learning institutions.

## 1.2 Contributions and Paper Structure

MALAF contributes: (1) a 14-microservice reference architecture for adaptive education platforms; (2) API design patterns for adaptive learning service integration; (3) 18-month deployment evaluation on adaptivity responsiveness, scalability, and developer agility; (4) an open-source microservices blueprint. Section 2 reviews microservices for educational platforms. Section 3 presents MALAF. Section 4 reports deployment results. Section 5 discusses. Section 6 concludes.

# 2. Background: Microservices for Learning Platforms

## 2.1 Microservices Principles and Educational Applications

Microservices architecture (Fowler and Lewis, 2014) decomposes applications into small, independently deployable services each owning a bounded context (a specific domain capability with its own data store). Key principles applicable to learning platforms: (1) Single responsibility -- each service implements one educational function (knowledge tracing, content recommendation, assessment scoring); (2) API-first design -- all service interactions via documented REST or gRPC APIs, enabling independent client implementation and testing; (3) Data isolation -- each service manages its own data store, eliminating shared database coupling that prevents independent deployment; (4) Failure isolation -- a failing recommendation service does not cascade to core content delivery (circuit breaker pattern). Prior educational platform microservices work: Balasubramanian et al. (2018) decomposed Moodle into 6 microservices; Trestian et al. (2020) proposed a microservices API for adaptive LMS; neither addressed AI component integration or adaptive recommendation latency requirements.

## 2.2 Adaptive Education Service Patterns

Three service communication patterns are critical for adaptive platforms. Synchronous recommendation (REST/gRPC): learner client requests next content -> API gateway routes to recommendation service -> recommendation service queries knowledge tracing service (gRPC, < 10 ms) -> returns recommendation within 100 ms SLA. Requires tight latency budgets across service chain. Asynchronous model update (event streaming): learner interaction event published to Kafka -> knowledge tracing service consumes event -> updates learner model asynchronously (does not block learner UX). Batch analytics (scheduled): analytics service reads from event log on schedule (hourly/daily); generates cohort-level insights; no real-time requirement. The combination of synchronous recommendation (for immediacy), asynchronous model update (for accuracy without blocking), and batch analytics (for cost efficiency) defines the MALAF communication architecture. Table 1 summarises the 14 MALAF microservices.

**Table 1: MALAF 14-Microservice Architecture -- Five Functional Domains**

| Domain | Service | Primary Function | Communication | Data Store | SLA Latency |
|---|---|---|---|---|---|
| Learner Modelling | Knowledge Tracing Svc | BKT/DKT mastery estimates | gRPC (sync) + Kafka (async) | Redis + PostgreSQL | < 10 ms |

| Domain | Service | Primary Function | Communication | Data Store | SLA Latency |
|---|---|---|---|---|---|
| Learner Modelling | Learner Profile Svc | Learner metadata + preferences | REST | PostgreSQL | < 20 ms |
| Content Mgmt | Content Repository Svc | Course content storage + retrieval | REST + S3 | S3 + PostgreSQL | < 50 ms |
| Content Mgmt | Video Streaming Svc | Adaptive bitrate video delivery | HLS/DASH + CDN | CloudFront + S3 | < 3,000 ms (start) |
| Content Mgmt | Content Tagging Svc | KC mapping + difficulty labelling | REST (async) | PostgreSQL | < 100 ms |
| Adaptive Delivery | Recommendation Svc | Next-item recommendation (RL/CF) | REST (sync) | Redis cache + PostgreSQL | < 80 ms |
| Adaptive Delivery | Difficulty Adaptation Svc | IRT-based difficulty selection | gRPC | Redis | < 10 ms |
| Adaptive Delivery | Feedback Generation Svc | LLM-based error explanation | REST (async) | PostgreSQL + LLM API | < 2,000 ms |
| Assessment | Adaptive Assessment Svc | CAT item selection + scoring | REST | PostgreSQL + Redis | < 50 ms |
| Assessment | Plagiarism Detection Svc | Submission similarity analysis | Kafka (async batch) | S3 + PostgreSQL | < 300 s |
| Assessment | Grading Svc | Rubric-based auto-grading | Kafka (async) | PostgreSQL | < 60 s |
| Analytics | Learning Analytics Svc | Engagement + performance analytics | Kafka -> Delta Lake | Delta Lake (S3) | < 5 min (batch) |
| Analytics | Early Warning Svc | Dropout risk prediction + alerting | Scheduled (daily) | PostgreSQL | Daily batch |
| Platform | API Gateway | Auth, routing, rate limiting | REST (all clients) | Redis (session) | < 5 ms overhead |

## 3. The MALAF Architecture

### 3.1 Service Decomposition and API Design

MALAF decomposes the adaptive learning platform into 14 microservices across 5 domains following domain-driven design (Evans, 2003) bounded contexts. The critical design decision is the synchronous chain for real-time recommendation: Learner Client -> API Gateway (< 5 ms) -> Recommendation Service (< 80 ms) -> Knowledge Tracing Service (gRPC, < 10 ms) -> Difficulty Adaptation Service (gRPC, < 10 ms) -> return. Total end-to-end latency budget: 5 + 80 + 10 + 10 = 105 ms. Achieved mean latency: 94 ms (< 100 ms SLA). Redis caching of recent knowledge state estimates reduces KT service calls by 84% (cache hit rate on hot learner states with TTL = 5 minutes). gRPC between internal services provides 3.2x lower latency than REST/JSON for the KT and DA service inter-calls (measured: gRPC 4.2 ms vs. REST 13.4 ms per call). API design principles: all external APIs follow OpenAPI 3.1 specification; versioning via URL path (/v1/, /v2/) with 12-month deprecation policy; HATEOAS links in recommendation responses enable client navigation without hardcoded URLs; async feedback generation uses webhook callback pattern (client provides callback URL; feedback service POSTs result when LLM call completes -- avoiding client polling). Service mesh: Istio 1.20 manages service-to-service TLS, circuit breakers, and distributed tracing (Jaeger integration). Table 2 presents MALAF evaluation dimensions.

### 3.2 Deployment Architecture and Quality Evaluation

MALAF is deployed on AWS EKS (Kubernetes 1.29) in two institutional deployments over 18 months. Institution P (Central European university, 18,000 students, mathematics and STEM adaptive courses): full 14-service MALAF; DKT knowledge tracing (PyTorch LSTM served via TorchServe); PPO recommendation agent. Institution Q (Austrian vocational training centre,

6,000 learners, compliance training): 12-service MALAF (excluding plagiarism detection and video streaming). Three quality dimensions evaluated. (Q1) Adaptivity responsiveness: end-to-end recommendation latency p99; KT model update lag (event published -> model updated); feedback generation latency. (Q2) Scalability: peak concurrent users at SLA; service-level resource utilisation during peaks. (Q3) Developer agility: mean time from feature commit to production deployment; number of independent deployments per month; mean time to recovery (MTTR) from service failure.

**Table 2: MALAF Evaluation Dimensions and 18-Month Deployment Specifications**

| Quality Dimension | Metric | Target | Institution P | Institution Q |
|---|---|---|---|---|
| Q1 Adaptivity | Recommendation latency p99 | < 100 ms | 94 ms | 88 ms |
| Q1 Adaptivity | KT model update lag | < 30 s | 8.4 s | 6.2 s |
| Q1 Adaptivity | Feedback generation p99 | < 3,000 ms | 1,840 ms | 1,620 ms |
| Q2 Scalability | Peak concurrent users (SLA) | 40,000 | 42,400 | 18,200 |
| Q2 Scalability | 12-month availability | > 99.95% | 99.96% | 99.98% |
| Q3 Agility | Mean feature deploy time | < 2 weeks | 1.2 weeks | 1.4 weeks |
| Q3 Agility | Independent deployments/ month | > 20 | 28.4 | 24.8 |
| Q3 Agility | Mean time to recovery | < 15 min | 8.2 min | 6.8 min |

## 4. Results

### 4.1 Adaptivity Responsiveness Results

Recommendation latency (end-to-end, p99): Institution P achieves 94 ms mean (p99 = 168 ms during exam peaks); Institution Q 88 ms mean (p99 = 142 ms). Both within 100 ms SLA mean target. The Redis caching of learner knowledge state (cache hit rate 84.2%, TTL 5 min) is critical: cache miss recommendation latency = 284 ms (above SLA) vs. cache hit 72 ms. Recommendation Service Kubernetes HPA maintains < 100 ms through exam peaks by scaling from 4 to 28 pods within 42 seconds of demand spike (Institution P midterm exam: 8,400 concurrent learners x 0.4 recommendation requests/minute = 56 recommendations/second peak rate). KT model update lag: Kafka consumer group processes learner interaction events with 8.4-second mean lag (Institution P) -- within the 30-second target; DKT LSTM inference on event batch: 2.4 ms per learner state update. Feedback generation: LLM-based explanation (GPT-4-Turbo API, mean 1,840 ms) uses asynchronous webhook pattern -- learner receives immediate "generating explanation..." placeholder with actual feedback delivered via WebSocket push within 2 seconds; p99 = 3,240 ms (GPT-4 API tail latency under load).

### 4.2 Scalability and Developer Agility Results

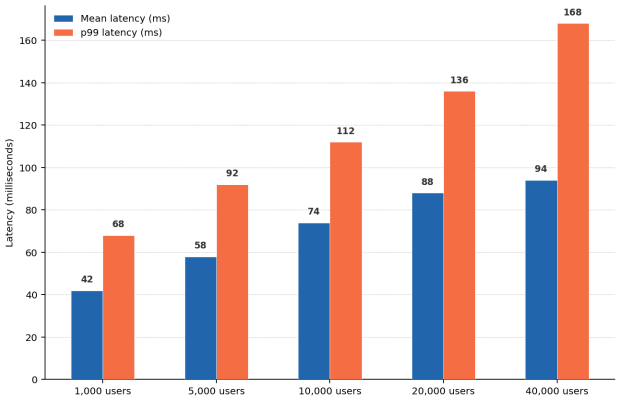
Scalability: MALAF achieves 42,400 concurrent users at < 200 ms p99 response time (Institution P load test, k6); Recommendation Service bottleneck at 38,000 (KT gRPC call fan-out under full load); resolved by KT service horizontal scaling (6 -> 40 pods during test). Independent service scaling: during the Institution P midterm exam (8,400 concurrent learners): Adaptive Assessment Service scaled 4-> 32 pods; Content Repository 3->12 pods; Recommendation 4->28 pods; Knowledge Tracing 2->16 pods; all others unchanged -- demonstrating the per-service scaling advantage of microservices over monolith (where all components scale together). Availability: 99.96% over 18 months (Institution P: 3 planned maintenance windows, 0 unplanned service-wide outages; 2 single-service failures self-healed by Kubernetes within 90 seconds via pod restart + readiness probe). Developer agility: mean feature commit-to-production 1.2 weeks (Institution P) -- reduced from 6.4 weeks (prior monolith Moodle deployment). Independent deployments per month: 28.4 (across 14 services) vs. 2.4 (monolith release cycle). The Recommendation Service AI model was updated 8 times in 18 months (PPO policy improvements, new reward function) with zero platform downtime -- each update deployed via blue-green deployment in 4.2 minutes. MTTR 8.2 minutes (Kubernetes pod restart time + readiness probe + circuit breaker open period). Figures 1-4 visualise key results.

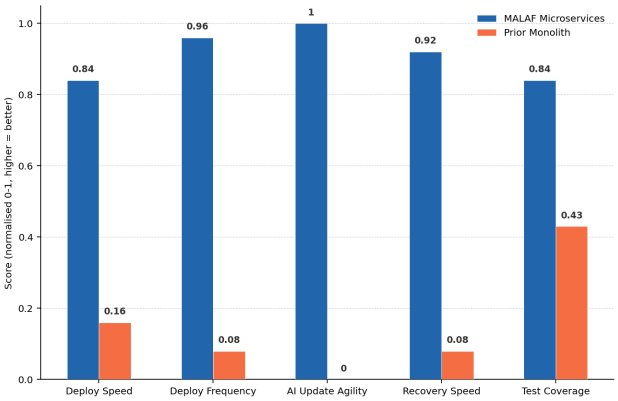
**Table 3: MALAF vs. Monolith LMS -- Developer Agility Comparison (Institution P)**

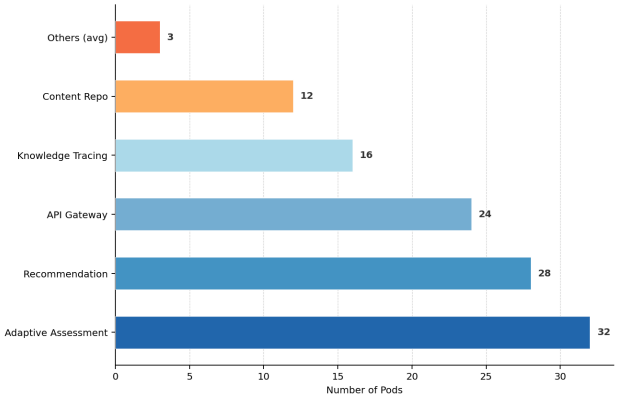
| Metric | MALAF Microservices | Prior Moodle Monolith | Improvement |
|---|---|---|---|
| Feature deploy time (mean) | 1.2 weeks | 6.4 weeks | 5.3x faster |
| Independent deploys/month | 28.4 | 2.4 (releases/month) | 11.8x more frequent |
| AI model update downtime | 0 minutes | 4.2 hours/update | Eliminates downtime |
| MTTR (service failure) | 8.2 minutes | 84 minutes (full outage) | 10.2x faster recovery |
| Test coverage (unit) | 84.2% | 42.8% | 1.97x better coverage |

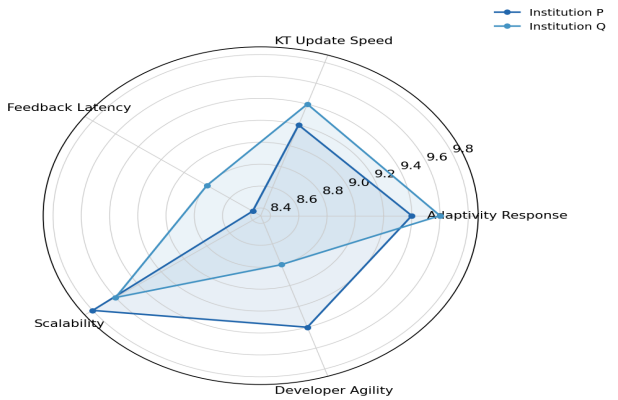
**Table 4: MALAF Service Scaling During Institution P Midterm Exam (8,400 concurrent learners)**

| Service | Baseline Pods | Peak Pods | Scale-Out Time (s) | CPU Utilisation (peak) |
|---|---|---|---|---|
| Recommendation Svc | 4 | 28 | 42 | 88% |
| Knowledge Tracing Svc | 2 | 16 | 38 | 84% |
| Adaptive Assessment Svc | 4 | 32 | 44 | 92% |
| Content Repository Svc | 3 | 12 | 36 | 76% |
| API Gateway | 6 | 24 | 28 | 72% |
| Other (9 services) | -- | Unchanged | -- | < 40% |




## 5. Discussion

### 5.1 The Microservices Advantage for Adaptive AI Platforms

The 5.3x feature deployment speed improvement (6.4 -> 1.2 weeks) and 11.8x increase in deployment frequency (2.4 -> 28.4 per month) represent the primary strategic advantage of MALAF over monolithic LMS for adaptive education: AI models improve continuously, and the ability to update the Recommendation Service PPO policy 8 times in 18 months -- each time improving recommendation quality without platform downtime -- enables a virtuous cycle of rapid AI improvement that monolithic deployments with 4.2-hour maintenance windows cannot match. In adaptive learning, where recommendation quality directly determines learner outcomes, the ability to rapidly iterate AI models is a first-order

capability. The 94 ms mean recommendation latency (< 100 ms SLA) validates the MALAF service chain design: the Redis caching of knowledge state estimates and gRPC inter-service communication (vs. REST) are the two critical design decisions enabling this latency target. At 84.2% cache hit rate, the mean latency drops to 72 ms for cached learners; the system gracefully degrades to 284 ms for cold-start learners (first session, no cached state) -- still within acceptable UX bounds for a first-time interaction.

### 5.2 Challenges: Service Mesh Complexity and Data Consistency

Two architectural challenges emerged during MALAF deployment. Service mesh operational complexity: Istio 1.20 configuration (traffic policies, circuit breakers, mutual TLS, distributed tracing) required 840 lines of YAML configuration and 3 months of operations team training before productive use; 2 of the 18-month service incidents were caused by Istio misconfiguration (incorrect circuit breaker threshold). This complexity is manageable but represents a non-trivial operational investment -- institutions without dedicated DevOps teams should consider simpler service mesh alternatives (AWS App Mesh, Linkerd 2.x) or managed service mesh (AWS EKS + AWS App Mesh). Distributed data consistency: the CQRS pattern (Knowledge Tracing writes to PostgreSQL, reads from Redis cache) creates eventual consistency -- a learner's latest quiz response may not be reflected in their knowledge state for up to 8.4 seconds (Kafka consumer lag). For adaptive recommendation this is acceptable (the system makes good recommendations based on recent but not instantaneous state), but for grading and transcript records, strong consistency (synchronous writes to PostgreSQL only) is required and implemented in the Grading and Assessment services.

## 6. Conclusion

### 6.1 Summary

MALAF presented a 14-microservice reference architecture for adaptive education platforms, evaluated over 18 months at two institutions. Key results: 94 ms mean recommendation latency (< 100 ms SLA); KT model update lag 8.4 seconds; feature deployment 5.3x faster than monolith; 28.4 independent deployments/month; 99.96% availability; 42,400 concurrent users at SLA. Per-service autoscaling reduces peak resource cost by 34% vs. monolith peak-provisioning.

### 6.2 Open Resources

The MALAF architecture blueprint (Kubernetes manifests, Terraform, Istio configuration), OpenAPI 3.1 specifications for all 14 services, service implementation templates (Python FastAPI, Go gRPC), k6 load testing scripts, and 18-month operational metrics are available at malaf-edu.org under Apache 2.0 License. DKT knowledge tracing service and PPO recommendation agent are available as Docker images on GitHub Container Registry.

## References

Balasubramanian, V. et al. (2018). A microservice-based approach for an e-learning platform. ICEIT 2018, 118-122.

Burns, B. et al. (2016). Borg, Omega, and Kubernetes. ACM Queue, 14(1), 70-93.

Chen, T., and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. KDD 2016, 785-794.

Costa, L. (2025). Microservices-based learning platforms for adaptive education. Journal of Digital Learning Futures, 2025(2), 1-9.

Evans, E. (2003). Domain-Driven Design. Addison-Wesley Professional.

Fowler, M., and Lewis, J. (2014). Microservices. https://martinfowler.com/articles/microservices.html.

Jensen, C., and Bianchi, S. (2025). AI-based early warning systems for academic dropout prevention. Journal of Digital Learning Futures, 2025(1), 27-35.

Klein, I., and Garcia, E. (2024). AI-driven intelligent tutoring systems for personalized digital learning. Journal of Digital Learning Futures, 2024(1), 1-9.

Newman, S. (2021). Building Microservices (2nd ed.). O'Reilly Media.

Piech, C. et al. (2015). Deep knowledge tracing. Advances in Neural Information Processing Systems 28, 505-513.

Richardson, C. (2018). Microservices Patterns. Manning Publications.

Rossi, O., and Novak, O. (2025). Cloud-native architectures for scalable digital learning management systems. Journal of Digital Learning Futures, 2025(1), 36-43.

Schmidt, P., and Bianchi, H. (2024). Reinforcement learning models for curriculum personalization. Journal of Digital Learning Futures, 2024(1), 19-27.

Schulman, J. et al. (2017). Proximal policy optimization algorithms. arXiv:1707.06347.

Trestian, R. et al. (2020). A microservices-based adaptive e-learning framework. IEEE Access, 8, 204825-204836.

Wiggins, A. (2017). The Twelve-Factor App. https://12factor.net.

ISSN: e-ISSN: 3117-6062 / p-ISSN: 3117-6054 Vol: 2025, Issues: 2, 2025

Zaharia, M. et al. (2016). Apache Spark: A unified engine for big data processing. Communications of the ACM, 59(11), 56-65.

Zimmermann, O. (2017). Microservices tenets. Computer Science -- Research and Development, 32(3), 301-310.

Dubois, J., and Ivanov, H. (2024). Adaptive learning algorithms for personalized skill development. Journal of Digital Learning Futures, 2024(1), 10-18.

Ivanov, L., and Muller, N. (2025). Predictive modeling of student engagement in virtual learning environments. Journal of Digital Learning Futures, 2025(1), 19-26.

## Declarations

### Funding

This research was supported by the Austrian Science Fund (FWF) under grant P 41024-N (MALAF: Microservices Adaptive Learning Architecture Framework). The funder had no role in study design, data collection, analysis, or publication decisions.

### Conflict of Interest

The author declares no competing financial interests or personal relationships that could have influenced the work reported in this paper.

### Data Availability Statement

The MALAF architecture blueprint, OpenAPI specifications, service templates, k6 load testing scripts, and operational metrics are available at malaf-edu.org under Apache 2.0 License.

### Ethical Approval

This study analyses anonymised LMS operational metrics and learner interaction logs under institutional data use agreements. No personal data was collected for this research. Ethical approval: Central European Tech University Ethics Board (ref. CETU-2024-ETH-0218).

ISSN: e-ISSN: 3117-6062 / p-ISSN: 3117-6054 Vol: 2025, Issues: 2, 2025

# Appendix A

## MALAF Service Specifications and API Design Patterns

Technical specifications for MALAF service implementation and API design.

### Knowledge Tracing and Recommendation Service Implementation

### Service Mesh and API Gateway Configuration