instinctools

Full Revamp of a Ticketing Platform for Officina

How rebuilding a flawed ticketing platform unlocked business agility, cut legal risk, and future-proofed accelerate software development by **40%**, reduce infrastructure and development costs by **30%**, and get a **production-ready solution**.

Business challenge

Every growing product carries the weight of early stage decisions. What accelerates launch in year one can quietly contain scalability by year two. Our client, Officina srl, an Italian startup, was no exception.

After launching as an offline event platform, the company decided to enter the regulated online ticketing market by obtaining SIAE certification, a mandatory requirement for selling tickets to music events online in Italy.

To move fast with this new project - **NOWR** - and reduce certification time, they made some tradeoffs that **shaped the platform's operational and technical constraints** and became hard to ignore.

Ticks-imposed limitations

Instead of pursuing direct SIAE regulatory approval of their own ticketing infrastructure, Officina chose to operate through Ticka, a licensed SIAE intermediary. While this decision reduced time-to-market, it also meant a few technical bottlenecks:

- **No smooth integration path.** Connecting the platform as a ticketing provider required a custom API wrapper around Ticka and manual copy-paste.
- **No built-in booking or refund logic.** Critical safeguards had to be implemented outside Ticka.

Rigid architecture and self-contradictory infrastructure

Although outsourcing software development has numerous benefits, you reap none of them if you partner with an unstable company. That was what initially happened to Officina. Their previous contractor walked away leaving:

- A monolithic, hard-to-scale architecture
- A mix of Google Cloud and AWS infrastructure
- Three separate backends
- Fragmented frontend drafts
- No CI/CD pipeline
- Multiple code repositories

Torn between operational constraints and a brittle technical foundation, the client needed a trusted tech partner to assess what could be preserved, what required rebuilding, and how to avoid Ticka-related limitations. instinctools' dedicated team proved to be up to the task.

Solution

After evaluating the development artifacts, it became clear that rewriting the platform from scratch would be faster, safer, and more future-proof than continuing to patch up and move the legacy core.

01 Replacing a rigid monolith with flexible and secure microservices

Once the decision to rebuild was made, the question was how to design a platform that would not collapse under its own growth again. That's why, we deliberately chose a modular, event-driven architecture structured around of the new NOWR core.

What it meant in practice:

- Clear function boundaries
- A dedicated database per service
- No shared internal logic
- Service-to-service communication via REST and message queues

The outcome was **strategic optionality. NOWR is no longer locked into architectural decisions made under time pressure years ago.** The new foundation supports independent scaling, isolated failure domains, and controlled upgrades, without destabilizing the entire system.

As Officina's business model depends on third-party providers, including SIAE intermediaries like Ticka and payment operators like Stripe, staying flexible is a tough call. We designed every integration with a clear boundary, so if one of the client's providers changes, the platform can adapt without undergoing another full rewrite.

Our delivery scope included:

- **Redis** for structure scalable backend services
- **No more repository** to maintain development speed without sacrificing modular boundaries
- **REST** for fast frontend iteration
- **Message queues** for asynchronous service coordination
- **Separate databases** per microservice to prevent tight coupling

02 Moving from two clouds to one

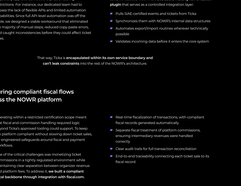
Ticket sales, especially for SIAE-regulated events, require strict ordering. Any inconsistency brings legal risk: duplicate seat assignments, chargebacks, or regulatory violations, and reversal fees. Officina's fragmented infrastructure simply wasn't built for that level of rigor.

With some components running on Google Cloud and others on AWS, it caused operational inconsistencies and increased scaling challenges. We chose to maintain the cloud split for now, but with a clear boundary: AWS would remain the primary cloud storage, we consolidated the rest of the stack on AWS to simplify operations and ensure correct processing order and resilience under load.

- **Amazon SQS/SNS** for queued workflows and event notifications
- **RDS** with Elastic Load Balancing (ELB) to seamlessly scale active performance
- **Asynchronous** processing for non-blocking operations

As part of the consolidation, we **migrated operational services and related datasets from Google Cloud to AWS S3.** This included backfilling application data, reconfiguring storage, and aligning ownership and access policies.

Operationally, shifting to a single cloud environment instead of juggling two actively changing integrations was a clear win. Fewer moving parts meant faster troubleshooting, simpler permissions management, and more predictable handling of rate limits and transient errors.


S3
Elasticsearch
Elasticsearch
Elasticsearch