tucanoo

TUCANOO SOLUTIONS · CASE STUDY

Modernising Mapolitical

A 14-year framework leap, a spatial database rebuild, and passwordless login.

HEAVIEST SPATIAL QUERY

20s → 0.1s

~200× FASTER

At a glance

- Up to **~200× faster** on the platform's heaviest spatial query — an aggregation page that previously took around 20 seconds now returns in about 0.1 seconds.
- **58–87% faster** across everyday boundary lookups after the move to PostgreSQL/PostGIS.
- Grails 1.3.7 upgraded to Grails 6.2 — a jump across roughly 14 years of framework, language and library versions, onto a fully supported, security-patched stack running on Java 17.
- A whole category of standing security risk removed, and modern authentication added: passwordless passkeys (WebAuthn) and multi-factor authentication.
- Delivered in phases for a long-standing UK business serving utilities and government.

The client

Commercial Evaluations Ltd is a UK company that has been running for over twenty years. Its product, Mapolitical, is a political-mapping and geospatial platform used by major utilities, government agencies and government departments to understand the political and administrative geography behind their data — matching locations to constituencies, councils, wards and other boundaries.

It is a business-critical product with a demanding, security-conscious customer base, and a specialist tool built up carefully over many years — which is exactly why the technology underneath it needed staged modernisation rather than a rewrite.

The challenge

Mapolitical ran on Grails 1.3.7, a framework version released in 2011. Over the years the platform had become difficult and risky to maintain:

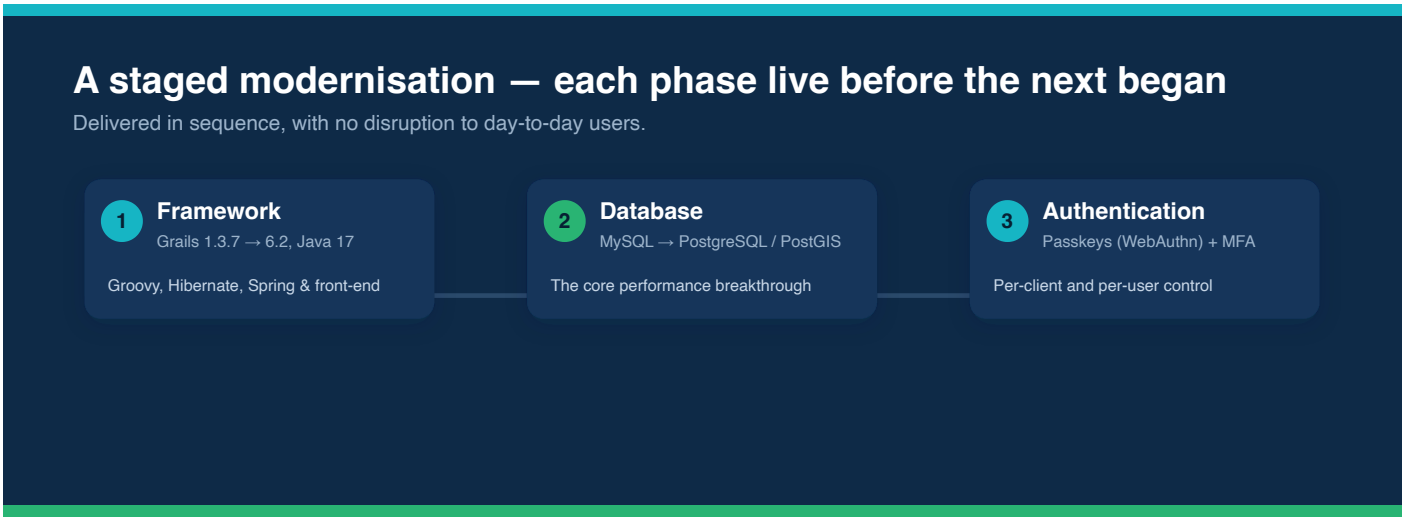
- The stack was end-of-life. Grails 1.3.7, and the Groovy, Hibernate and Spring libraries around it, had been unsupported for well over a decade. No security patches were reaching the application, so any newly discovered vulnerability in that stack would remain permanently unpatched.
- Spatial performance was a real bottleneck. The platform's core job is spatial queries, and on MySQL the heaviest of them were painfully slow. The worst affected were large, complex boundaries — detailed Scottish councils, constituencies and wards built from many thousands of coordinate points. Running one of these could take many minutes, and sometimes never finished at all: a user would set the operation going, leave it, and come back later hoping it had completed. A commonly used aggregation page took around 20 seconds to load.
- Modern features were hard to add. An ageing framework made it difficult to introduce the kind of security and authentication features the client's customers were beginning to expect.

Why now

The trigger was as much commercial as technical. With a customer base of utilities, government agencies and departments, Mapolitical operates in a sector where expectations around supplier security keep rising. Staying on a 2011 framework with unpatchable dependencies sat awkwardly against that direction of travel — modernisation was about keeping a business-critical product dependable and secure for a demanding customer base.

The approach

Tucanoo delivered the work in three deliberate phases, each fully working before the next began, and always preserving existing data and day-to-day behaviour for end users.



Phase 1 — Framework modernisation. We migrated the application from Grails 1.3.7 to Grails 6.2, which also meant uplifting Groovy, Hibernate and Spring, moving the build to Gradle, and modernising the entire front end. The application now runs on Java 17 on a supported, security-patched platform. URLs, navigation and look-and-feel were kept the same, so users saw continuity rather than disruption.

Phase 2 — Database and spatial engine. We migrated from MySQL to PostgreSQL with PostGIS, the spatial database used across serious mapping applications. Geometry tables were moved with OGR, the remaining data with PGLoader, spatial indexes were rebuilt, and queries were re-tuned for PostGIS. This is where the platform's core performance problem was solved.

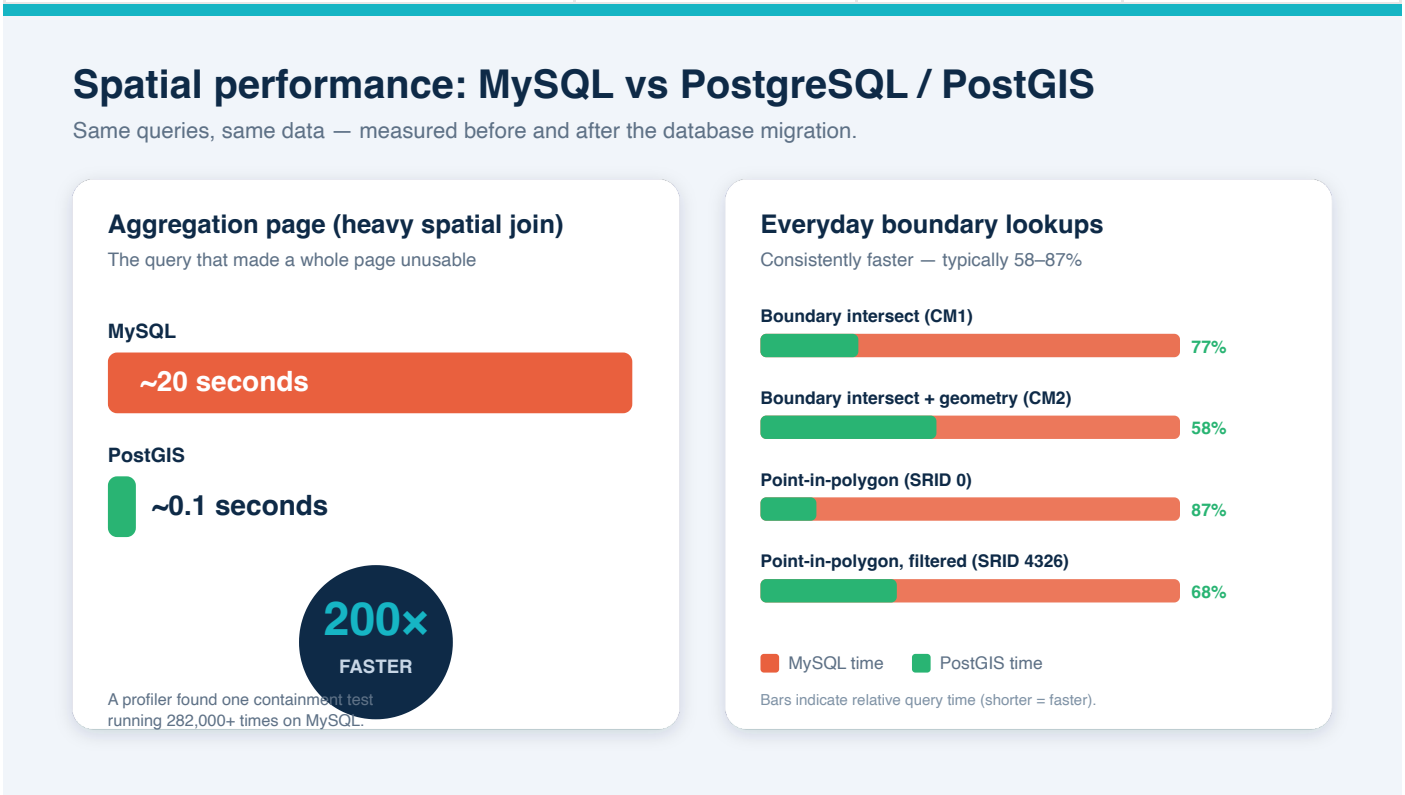
Phase 3 — Modern authentication. We added passwordless passkey login built on the WebAuthn standard, and multi-factor authentication by SMS or email — both configurable per client and per user, so security-conscious customers can enforce them while others continue unchanged.

The results

Spatial performance

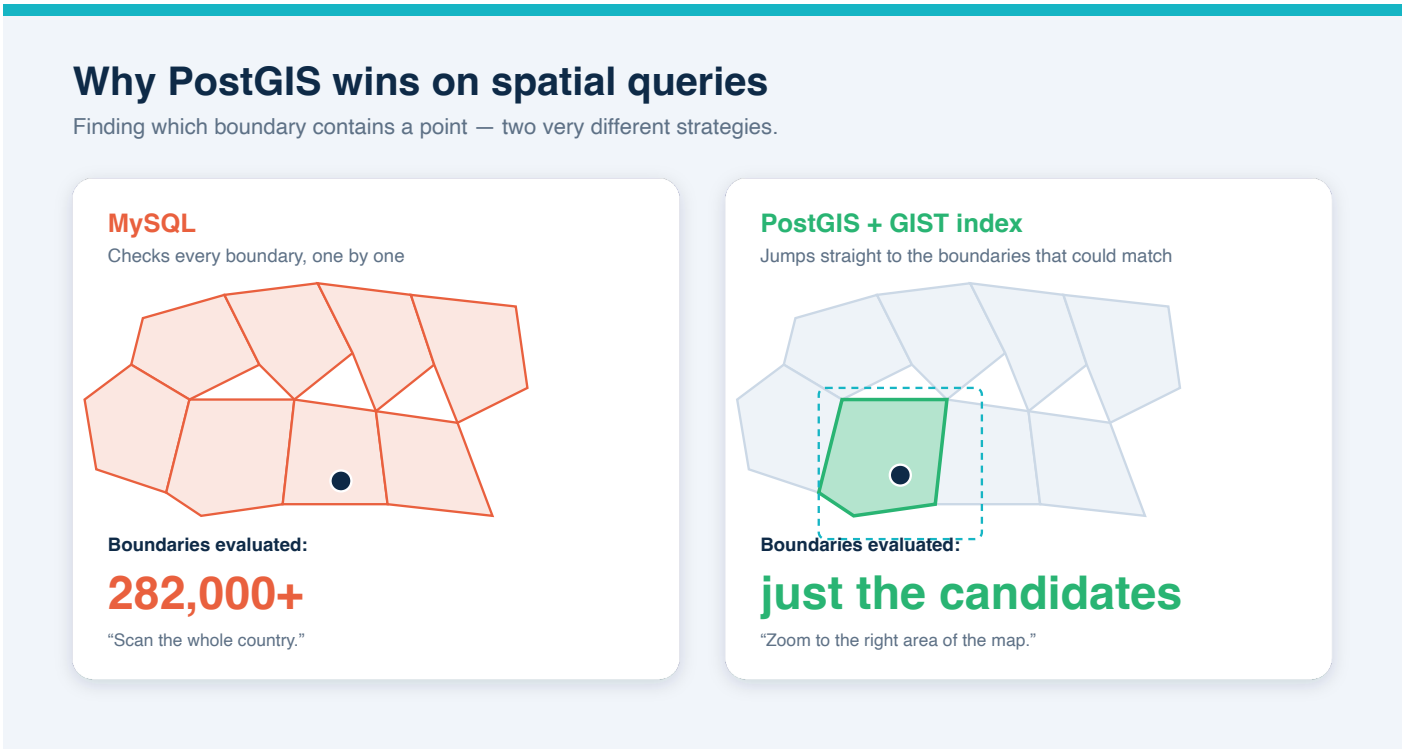
Moving to PostGIS delivered consistent gains across everyday spatial queries and a dramatic improvement on the worst-case query that had been hurting the product most.

Query	Before (MySQL)	After (PostGIS)	Improvement
Everyday boundary lookups	0.04–1.25s	0.01–0.17s	~58–87% faster
Aggregation page (heavy spatial join)	~20 seconds	~0.1 seconds	~200× faster



For the operations that had caused the most pain, the change was transformational. The large, intricate Scottish boundaries — the ones a user would previously start, walk away from, and return to in the hope they had finished — now resolve in around a second. The commonly used aggregation page dropped from about 20 seconds to roughly a tenth of a second, an improvement of around 200×.

The reason lies in how the two databases handle spatial joins. On the aggregation page, MySQL re-checked a boundary containment test hundreds of thousands of times, which accounted for most of the delay. PostGIS, using its GIST spatial indexing, only tests the boundaries that could actually match — the difference between scanning the whole country and zooming straight to the right area on the map. Operations that had previously been unusable now complete in seconds or less.



Security

- The platform moved off an unsupported, unpatchable framework onto a current, actively maintained stack — removing an entire category of standing risk.
- Password storage was strengthened from unsalted SHA-256 hashes to BCrypt, with a transparent upgrade path so existing users were never forced to reset their passwords.
- The modernisation put the platform on a footing to meet the rising security expectations of its government and utility customers.

Modern authentication

- Passwordless passkey (WebAuthn) login, built on a standards-based library rather than home-grown cryptography.
- Multi-factor authentication by SMS or email, with per-client and per-user control so it can be rolled out gradually or enforced organisation-wide.

In the client's words

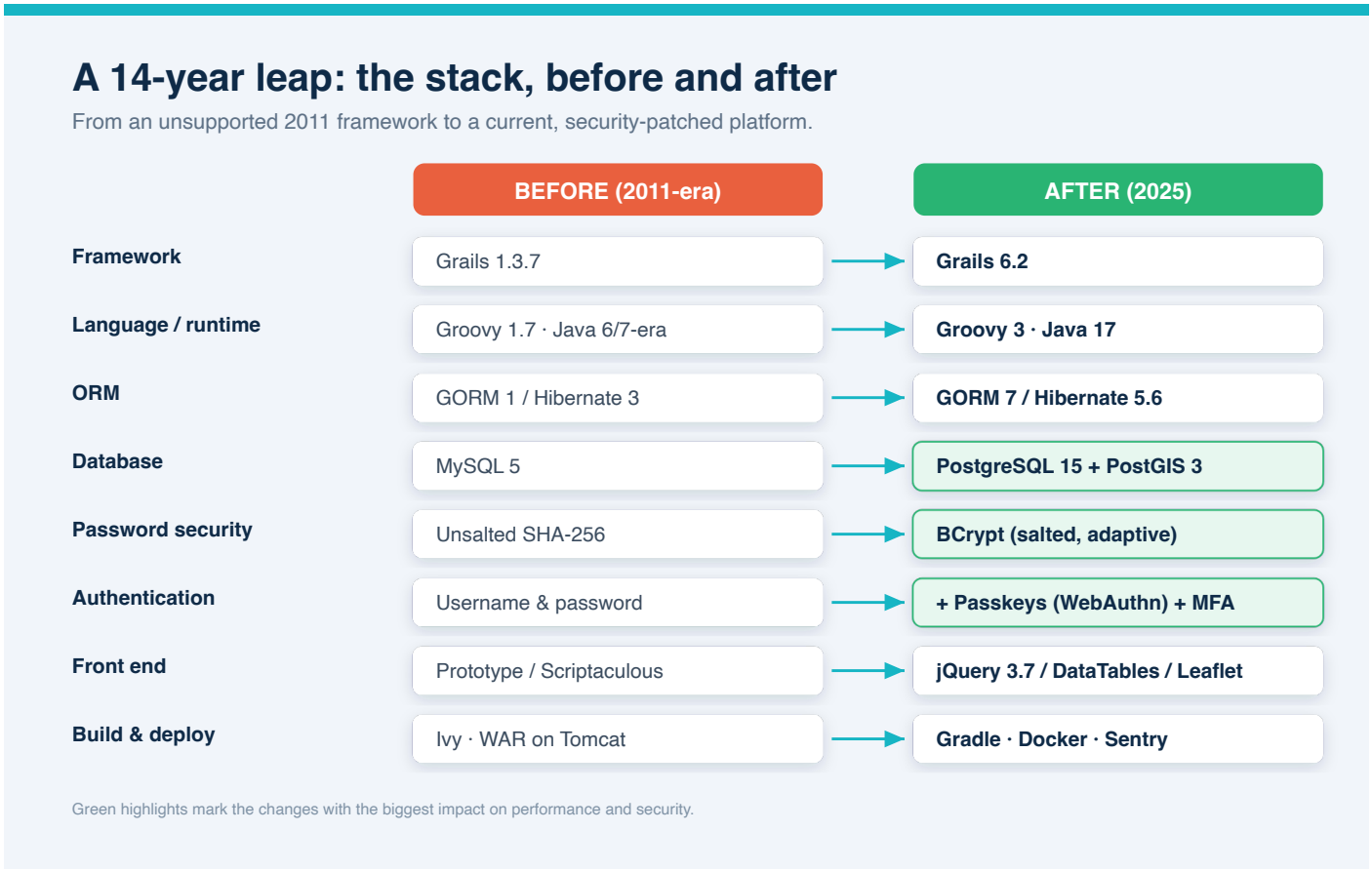
"We worked with Dave Brown and his team on a four month project to upgrade our 20 year old Grails application to the latest version. Dave demonstrated considerable expertise and knowledge throughout the project and met all project milestones on time. We have been very impressed with the quality of his

work and the upgraded application is live and being used by a range of clients. We would be pleased to recommend his services... this is the start of a long term working relationship."

— Daryl Hickman, Director, Mapolitical

Technical summary

Stack: before and after



Component	Before (2011-era)	After (2025)
Framework	Grails 1.3.7	Grails 6.2
Language / runtime	Groovy 1.7 · Java 6/7-era	Groovy 3 · Java 17
ORM	GORM 1 / Hibernate 3	GORM 7 / Hibernate 5.6
Database	MySQL 5	PostgreSQL 15 + PostGIS 3
Connection pool	C3P0	HikariCP
Security	Spring Security Core 1.2.7.2 · SHA-256	Spring Security 6 · BCrypt · WebAuthn · MFA
Front end	Prototype / Scriptaculous	jQuery 3.7 / DataTables / Leaflet

Component	Before (2011-era)	After (2025)
Build & deploy	Ivy · WAR on Tomcat	Gradle · Sentry monitoring

Geometry tables were migrated with OGR and the remaining data with PGLoader, with primary keys verified and indexes regenerated from the application's domain model. Spatial queries were rewritten for PostGIS, including resolving a coordinate-system (SRID) mismatch carried over from the original database. Authentication uses a dual-mode encoder that verifies legacy password hashes and transparently re-encodes them to BCrypt on next login, so no user is locked out during the transition.

About Tucanoo Solutions

Tucanoo Solutions specialises in staged modernisation of long-lived, business-critical Grails and geospatial applications — bringing them onto a supported, secure and performant stack without disrupting the people who rely on them every day.

Related projects