

Built against a mock. Proven against the national system.

A live eligibility check against SII, Italy's national energy-market data system, built for an energy retailer's back office. **An operator types a supply-point code and learns, in well under a second, whether that point can be taken on.** It was the first REST integration with a system that otherwise speaks SOAP and bulk files, and it was finished months before the credentials to call it arrived.

MEDIAN RESPONSE	TEST SUITE	AUTHENTICATION	WAITING ON ACCESS
369 ms 49 real calls, test environment	~300 tests 99% coverage when written	2 layers mutual TLS and a signed JWT	4 months with the build already done
ROLE	Senior backend contractor on an ongoing engagement: the API client, authentication, mock server, test suite, audit logging and operator screen.		
CORE STACK	PHP 8 · Laravel · Filament · Oracle · Docker · REST over mutual TLS with RS256 JWT · Python, Streamlit and SQLite FTS5 for the documentation tool		
NOTE	Client withheld. The code is merged and the production certificate has been issued. Every figure here comes from the national system's test environment or from the test suite; no production figures are claimed.		

WHAT IT DOES

Every electricity supply point in Italy has a code, and the national system knows its state. CheckPOD is the service that answers a narrow question about one of them: is this point active, can it be taken on by the requester, would this be a first activation, and can the activation go through the national process or must it go to the distributor directly.

In the back office, that became one screen. The operator enters the code, optionally with the requested power and voltage, and gets a plain result card: accepted or rejected with the official reason, and a badge for each answer. The same check is exposed as an internal API, behind an origin allowlist and a hashed key, for other systems to call. Every request leaves one row in an audit table: who asked, from where, what was sent, what came back and how long it took, with the bearer token masked.

THE CALL PATH

Operator screen or internal API → input validation → request built to the national schema → signed JWT, valid ten seconds, minted fresh for each attempt → mutual TLS to the national gateway → response validated and mapped to typed objects → result card, with the request and response written to the audit log.

Transient failures (timeouts, gateway errors, dropped connections) are retried up to three times with exponential backoff and jitter. Rejections by the national system are not retried: they are answers, and the operator sees the official code and reason.

THREE DECISIONS THAT MATTERED

01 Building the national system before having access to it

Access to a regulated system arrives on its own schedule: an activation request, a client certificate, registration on the gateway, an allowlisted address. Waiting for all of it would have meant months with nothing built. So I wrote a mock of the national service first, running as a container next to the application, where the last four digits of the supply-point code choose the outcome: success, each official rejection code, and a deliberately flaky range that fails twice before succeeding, to exercise the retry path. The client, the retry logic, the operator screen and about three hundred tests were finished against it.

The cost is that a mock only knows what its author knew. It was wrong in five places, and each surfaced only against the real system.

02 Proving whose problem it was

The first real calls were rejected with an authentication error saying no public key was registered for the token. The tempting response is to keep adjusting your own side. I wrote a test that sent ten variants of the token header instead: different key identifiers, with and without the embedded certificate, a minimal header. All ten failed identically. That was the evidence: nothing on our side changed the outcome, so the certificate had not been registered on theirs. The reproducer went to the operator's service desk and stays in the repository.

It did not make the wait shorter. The integration sat blocked for about four months. It did mean those months were not spent rewriting working code.

03 A synchronous call, with the retries inside it

The alternative was a queued job with a status the screen polls. The service is a synchronous lookup and the real median turned out to be 369 milliseconds, so the operator gets the answer on the same screen, with no job to track. The price is the worst case: if the national system hangs, three attempts and their backoff can keep the operator waiting about a minute and a half. For a lookup that normally returns before the operator has moved the mouse, that trade was worth making.

WHAT THE REAL SYSTEM TAUGHT THAT THE SPECIFICATION DIDN'T

WHAT WE ASSUMED	WHAT THE REAL SYSTEM DOES
The token is addressed to the URL being called	The audience must be the canonical production service address, even when calling the test environment.
The service URL is the endpoint	The transport path needs an extra segment and the operation name on the end. Miss either and the gateway answers "undefined operation".
Any unique request ID will do	The ID is capped at fifteen characters. Ours was longer.
Supply-point codes follow the textbook pattern	Real codes carry alphanumeric distributor parts and a trailing check letter. The validator had to be loosened.

"Can it be activated through the national process?" is yes or no

In 38 of 39 accepted answers it was a third value: not determinable. The mock had only ever said yes.

One more, before any of those: the client certificate was first extracted from its bundle with the authority's certificate mixed in. The gateway's reply was a generic "request does not conform to the interoperability profile", fourteen times, with no hint about which part. Checking that the key and certificate actually matched found it.

In the 47 responses recorded over the first three days of real calls: 39 accepted, of which 16 could be taken on and 23 could not; six "supply point does not exist"; and two rejections caused by a mistyped tax number on our side, which is why that value now lives in configuration.

A LOCAL LIBRARY OF THE RULEBOOK

CheckPOD is one piece of wider, ongoing SII work on the same platform: SOAP process flows and bulk measurement imports already in production, and gas activation flows being moved off a legacy system and manual spreadsheets. All of it is governed by the national system's published documents, and there are a lot of them, in several versions, some only drafts under consultation.

So I built a small tool for it. A crawler mirrors the public document library locally: 886 documents, about 800 MB. An indexer reads each one and extracts what you actually search by: process codes, regulator resolution numbers, versions, and whether it is a draft. That goes into a SQLite full-text index, with a Streamlit interface and a command line on top. Asking for one process code returns the six documents that define it and shows which are superseded or still in consultation.

It indexes titles, codes and metadata, not the full text of every page, and it was built during the wider SII work, after the CheckPOD client was written. Its job is the question that comes first on every new flow: which documents govern this, and which version is in force.

WHAT THIS MEANS IF YOU'RE HIRING

Integrations with regulated systems are slow for reasons no engineer controls. What an engineer does control is whether the work is finished and tested when access finally arrives, and whether there is evidence on the table when something fails on the other side.

The same applies to any authority, bank or partner API: build against a faithful mock, expect the real system to correct you, and keep a record of every call.

Richard Trujillo • Desemax. Senior backend and AI-integration engineer. Eleven years of backend engineering, independent since 2015. Based in Milan, working across Europe.

richard@desemax.it · richardtrujillo.desemax.it