Created integration tests that automatically validates changes to Salesforce API contract

As our system grew, one of the biggest sources of friction wasn't writing SOQL. It was trusting it.

We define contracts in code — objects, fields, queries — but those contracts depend on external Salesforce state:
- fields existing in the org
- objects deployed in the right environment
- permissions correctly configured

Failures didn't show up early. They showed up late:
- during integration testing
- in staging
- or in production

That created a constant tax:
- engineers manually checking fields in Salesforce
- debugging permission issues after deployment
- writing one-off integration tests per object or query

Velocity slowed down and confidence dropped.

Action

I built a system that makes Salesforce contract validation automatic, without requiring engineers to write integration tests.

Classpath scanning as the source of truth

We scan the codebase for all models annotated with `@SalesforceObject` .

That means the code itself defines what we expect from Salesforce. No test registration. No duplication.

Live schema validation using describe

For every discovered object, we call the Salesforce describe API and compare:
- expected fields from the Kotlin model
- actual fields returned by Salesforce

We validate:
- field existence
- object existence
- updateable fields for write models
- permissions correctness

Dynamic test generation

Instead of writing test cases, we generate them.

- one test per Salesforce object
- zero boilerplate in each service

When you add a new object or field, it is automatically validated. No new test required.

Result

We removed an entire category of work.

- Engineers no longer write integration tests for Salesforce objects
- Schema mismatches are caught immediately in CI
- Permission issues are caught before deployment
- Onboarding got easier because there is nothing new to learn about testing

Salesforce went from something you “hope is correct” to something that is continuously verified.

Why this matters

Before:
- correctness depended on the engineer remembering to test
- every team solved integration testing differently
- bugs showed up late

After:
- correctness is enforced automatically
- testing is standardized across teams
- failures show up immediately

The real win

The biggest win was not better tests.

The biggest win was removing a repeated decision:
“did I validate this correctly?”

That decision is gone.

Just like the SOQL DSL standardized how queries are written, this system standardized how external contracts are trusted.

Learning

- If engineers repeatedly validate the same external system, it is a platform problem
- The best productivity gains come from removing work, not optimizing it
- Zero-cost adoption is what makes systems stick
- Integration testing should be:
- automatic
- complete
- enforced at the boundary