Insurance software development: build around the book system you already run

By Andy Slack | November 18, 2026 | AI Development

Insurance software development from one real build: what we integrated with, the first MVP, what to buy instead, and who holds the system after.

You run an insurer, an MGA, a broker or an insurtech, and you are deciding whether to commission software. Most pages on insurance software development are vendor lists or module catalogues. This one is a single build: an insurance client we worked for, which we don't name. It covers what the client already ran, what we built first, what we would tell you to buy instead, and who holds the system afterwards. Insurance is one of the sectors we have worked in, not a specialism we claim. What we say about the client comes from our project records (Andy Slack, confirmed 18 November 2026).

What we built for an insurance client

For an insurance client, we built a real-time asset tracking portal and a data analytics portal that process over 1GB of flight data a day. The same client also runs a no-code client risk portal and an AI-driven risk management assistant. All four are listed with our client results (/) (read 18 November 2026).

Build around the book system you already run

The client already had its own insurance book system. We didn't replace it. We integrated with it, so the client could track real asset values and insured values in real time (Andy Slack, confirmed 18 November 2026).

So the first question on an insurance build is not which modules to buy. It is what already holds your book, and what the new software has to read from it. If that system works, the new software can sit beside it and add what it can't do.

If the system you depend on was built by someone else and you want us to take it over, we assess it before we commit. What we check is in technical due diligence for software (/articles/technical-due-diligence-for-software).

The first release proved one thing

The first thing we built was an MVP to prove we could track flights in real time, using demo insurance data (Andy Slack, confirmed 18 November 2026). For an insurer, that shape is worth copying: prove the live part on demo data, so no customer data is needed to find out whether it works.

We scope any first release the same way: one usable path, agreed at onboarding as the first milestone. How we decide what it leaves out is in MVP development services (/articles/mvp-development-services).

What to buy, and what to build

Buy when you have no in-house technical people and an off-the-shelf platform is easier to adopt: we have advised a client against custom software for that reason. Build when the system is the product, or has to carry large volumes of live data, as the flight-data portals do (Andy Slack, confirmed 18 November 2026). The rest of that rule, for tools your team already works in, is in what to automate first, and when to build (/articles/business-process-automation-what-to-automate-first). This page doesn't repeat it.

It doesn't have to be one or the other. Our insurance client runs both: a no-code client risk portal beside the portals we wrote in code (Andy Slack, confirmed 18 November 2026).

We wrote the insurance platform in code, on NestJS and Vue with MySQL, because it carries large volumes of live data. The tooling listed for that client is AWS, Elestio, MySQL, NestJS and Vue. We choose tools for each project around what the client already runs (Andy Slack, confirmed 18 November 2026).

Regulation: who owns what

We don't take on specialist regulatory responsibility. It sits outside the fractional CTO fee unless it is scoped and priced separately, so you keep your own compliance owner. What we do: your code, accounts and data stay in your name, the AI engineering team works with agreed workflows and access controls, and we ask you not to send passwords, keys or customer data before the first call (Andy Slack, confirmed 18 November 2026).

The rules come from your regulators, not from a vendor's summary of them, ours included. The Financial Conduct Authority (https://www.fca.org.uk/) regulates financial services firms and markets in the UK and sets the standards firms must meet. The ICO (https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/) publishes UK GDPR guidance and resources for organisations. Read them with your compliance owner before you brief any partner.

Who holds the system after launch

Your code, accounts, documentation and engineering environment stay in your name from the start, so if you leave you take all of it with you. Specs and tests sit in your own repository. For an insurer, that means you hold the repository, hosting and data accounts behind the system, not us (Andy Slack, confirmed 18 November 2026). That is how the service works now. We don't describe here what was handed over on the past insurance build.

If someone else holds those keys today, start with who owns my app code, and who holds the keys (/articles/who-owns-my-app-code).

What to have ready before a first call

Have a few sentences on the product and who it is for, and its stage: prototype, MVP or existing product. Know who leads the technical side today and what already exists. For an insurer, that means the systems you run, the data feeds they take in and who holds their accounts. Check whether the published price (/) fits. Don't send passwords, keys or customer data. Access and inputs come after the scope is agreed, and onboarding targets two to four weeks from when we receive them (Andy Slack, confirmed 18 November 2026). How spend is approved after that is in software development costs out of control (/articles/software-development-costs-out-of-control).

If you are deciding whether to build insurance software, we lead the technical decisions and an AI engineering team delivers them. Book a founder fit call (/#lead), or read more in our articles (/articles).