ANTIDEFAULT

A legacy macOS app that had to become a web app before a merger deadline

Case study — CAI International

Florian Ionescu

August 17, 2026

Client	CAI International, mid-merger
Engaged via	Toptal
Industry	Container leasing and logistics
Stack	Legacy macOS desktop app → React, PHP REST API, Sybase
My role	Senior developer added to a team already in motion, for speed under a hard deadline
Engagement	2022 – 2023

CAI International sells, leases, and invests in shipping containers, and the whole operation ran on a legacy macOS desktop app. A merger was closing in, and they brought me in through Toptal in 2022 to help rebuild the app for the web before the two sides came together. The date was the whole problem. On merger day, everyone would be trained on whatever software existed: the people who'd worked there for years, and the wave of new colleagues arriving with the merger. Miss it, and the company trains its entire workforce on an app it has already decided to kill, then trains them all again once the replacement lands.

1 The team was already halfway there

I wasn't the first one on this. A team lead and another developer were already deep in it, and they'd done the hard early part: stood up a React and PHP app and migrated several of the old screens across. The foundation was solid, and I wasn't there to second-guess it. What they needed from me was speed: someone senior who could take screens off the pile and clear them fast enough to make the date.

2 Buying a MacBook so I could start the same day

The old app only ran on macOS, and I didn't own a Mac. The day I officially got hired, I went out and bought a MacBook off a psychology student in town. It spared my client a scramble: no macOS virtual machine to set up for me, and no waiting for it to work. I had a machine, and I was working on it that same day.

3 Reverse-engineering the old app, screen by screen

Getting it running was the simple part. Understanding it was not. It was written in a language I'd never seen before and have never seen since, old enough that it followed none of the conventions you'd normally lean on to guess what a screen was doing. So I didn't guess. I read it a screen at a time and worked out exactly what each one did and why. I've spent much of my career inside cryptic, undocumented systems, and this was squarely that.

4 Migrating screens against the clock

The loop itself was unremarkable, which under a deadline is exactly what you want. The team lead pointed me at a screen, I pulled it apart to see what it did, and I rebuilt it in the new app: the interface in React and JavaScript, and PHP REST endpoints behind it wired into the Sybase database underneath. Then the next screen, and the next. A few dozen in all.

Alongside the building, I met with the stakeholders regularly, keeping the product spec and their expectations from drifting apart over a long build. We finished ahead of the merger. When the two companies combined, their people were trained on the new app instead of the dead one.

5 Staying on after the deadline

I didn't hand it off at launch. I stayed with CAI for several months after, taking feature requests and fixing issues as they came up. A migration's rough edges only show once real use leans on it, and I was still there for them.

6 If this sounds like your codebase

If you've got a legacy app that has to become a modern web app on a deadline you can't move, and a team already in motion that just needs more speed, that's my favourite kind of problem. Message me and tell me what you're dealing with.