

Rebuild against the live system, not the source.

The risk in rewriting a legacy system is everything it does that nobody wrote down. My method: **drive the old and new systems side by side, screen by screen, and treat every divergence as a numbered defect** until none are left. Developed rehabilitating a decade-old multi-tenant compliance portal onto Laravel — procedural PHP and classic ASP, no framework, no tests, no original developer to ask.

SCREENS AUDITED	DIVERGENCES	VERIFIED LIVE	REGRESSIONS
43 twice, from two roles	146 each numbered and tracked	140 6 blocked on client access	0 at final verification

FOUR PRINCIPLES

01 The source is not the specification

A first pass compared the rebuild against the legacy code. Re-run against the live system, it contradicted four of its own conclusions and found four things the source couldn't show. With no tests and no documentation, production behaviour is the only spec.

02 Side by side, chunked, numbered

Ten screens per pass, both systems driven live, as administrator and as a lower-privileged user. Every divergence is a numbered item; the database is restored after each chunk; nothing advances until the chunk is fixed and re-verified.

03 A gap is not a feature request

A gap is a divergence from what the old system does. Anything it never did goes on a separate list. Without that edge, the register quietly becomes a backlog and "done" stops meaning anything.

04 Parity is the default, not an absolute

The legacy system displayed passwords in cleartext, and the client asked for that in the rebuild. It was refused. Reproducing a critical vulnerability isn't parity — every exception is deliberate and written down.

Walking every screen with an engineer's eye also surfaced nine security findings in the legacy system, three of them critical — a security review delivered as a by-product of the rebuild, before cutover, with nothing written to or tested against production.

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

Full method, with the audit steps and security findings: richardtrujillo.desemax.it/case-studies