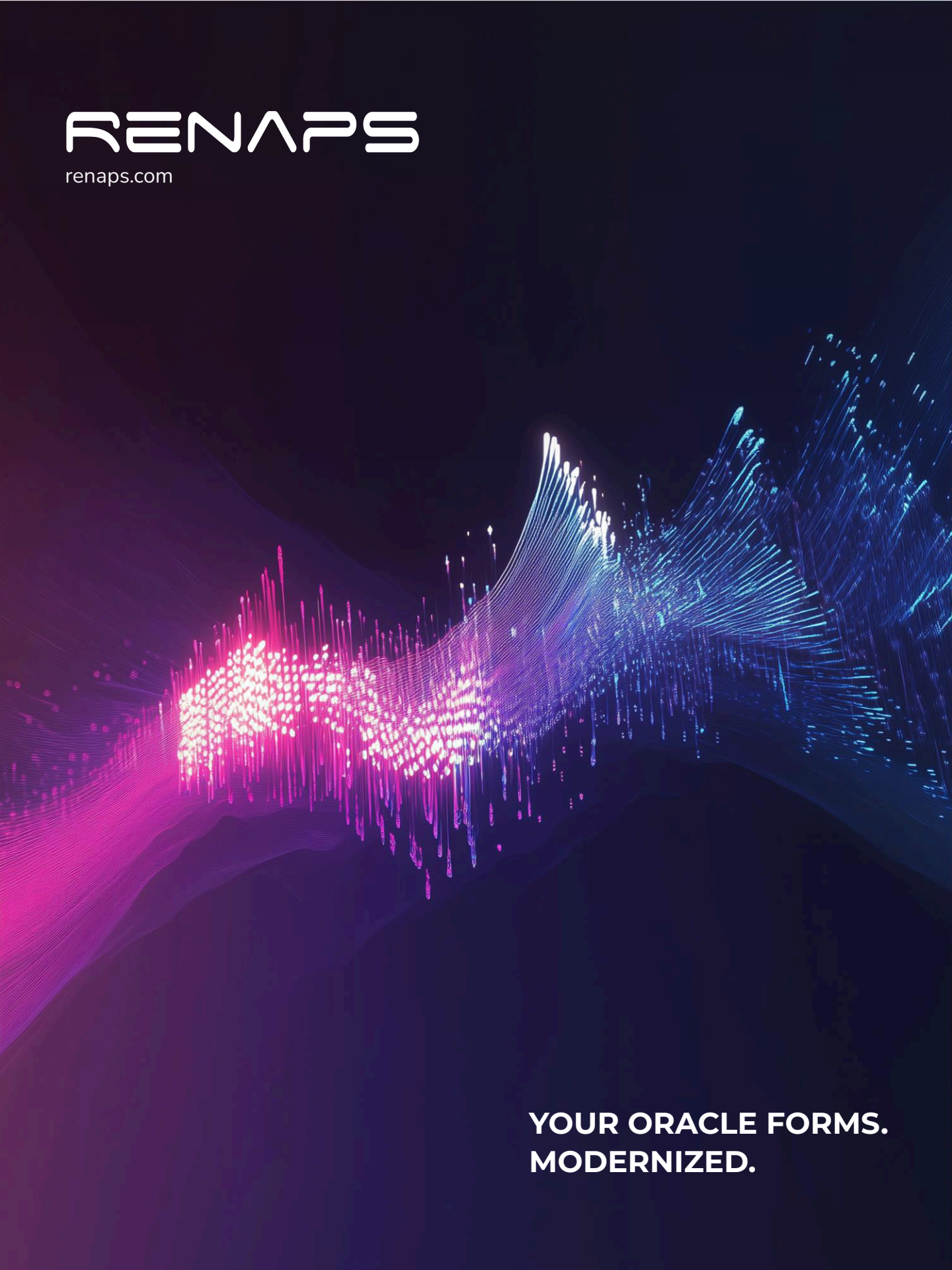
RENAPS

renaps.com



YOUR ORACLE FORMS.
MODERNIZED.

From Oracle Forms to AI Agents. Migrate the Rules, Not the Screens.

Modernizing Oracle Forms with ORMIT™-OpenJava:

Why a migration must keep your business rules executable, in an open architecture you own, so that the web application you deploy today is the one an AI agent can run safely tomorrow, without a second migration.

October 2026 · RENAPS

By

RENAPS

USA & International HQ: 2875 NE 191 Street, Suite 500, Aventura, Florida, 33180, USA
Canadian HQ: 101 Marcel-Laurin, Suite 300, Montréal, Québec H4N 2M3, Canada
1-888-98-RENAPS (1-888-987-3627) · info@renaps.com

Table of Contents

1. EXECUTIVE SUMMARY 05

2. ORACLE FORMS AT THE END OF THE ROAD 06

Four decades of Oracle Forms 06
What Forms 14c changes, and what it does not 06
The pain points driving modernization 07
What the analysts measure 08
The cost of waiting 08

3. 2026: THE YEAR THE MARKET REPRICED SOFTWARE 09

The SaaSpocalypse, and the rebound 09
Forrester and Gartner put dates on it 09
Gartner's trends for 2027: from adoption to governance 10
Even the vendors say so 12
Why screens are no longer the whole destination 12

4. TOMORROW'S AI IS A BUREAUCRAT 13

An agent, not a tool 13
The bureaucracies that run on Oracle Forms 13
No rules, no agent 14

5. WHAT YOUR FORMS CONTAIN 15

Where the rules live 15
Why Oracle Forms is so "sticky" 15

6. THE PATHS THAT LOSE THE RULES 16

The rewrite trap 16
Why a bolted-on assistant is not enough 16
Why Oracle APEX is a dead end for complex Forms applications 17
A packaged product: the hidden cost of starting over 18

7. MIGRATE THE RULES, NOT THE SCREENS 19

What ORMIT™-OpenJava is 19
Start with a complete inventory: ORMIT™-Analyzer 20
OFCS: rules that run without a screen 20
What you get today, and what it enables tomorrow 22
What your users keep 22
What stays in the database stays in the database 22
What this makes possible 23

Table of Contents

8.	**THE ARCHITECTURE: FLEXIBLE, OPEN, YOURS**	**24**
	Two backends, one result	24
	The two options, side by side	25
	React or Angular	25
	Oracle Forms to the modern stack: technological equivalents	26
	What it looks like: Oracle Forms next to ORMIT™-OpenJava	27
	Cloud-ready by default	30
	Security, privacy and DevOps compatibility	31
	Performance by design	31
	From stateful to stateless: scaling made simple	31
9.	**BEYOND FORMS: REPORTS, DATABASE, EBS AND ISVS**	**32**
	Oracle Reports with ORMIT™-Jasper	32
	The database: Oracle to PostgreSQL with ORMIT™-Cortex	33
	Oracle EBS customizations and ISV products	34
10.	**AN APPLICATION ANY AGENT CAN OPERATE**	**35**
	Two tiers	35
	Three ways to run a process, one source of rules	36
	A worked example	36
	What the rules cannot do	37
	What your people do instead	37
	Safe by construction	38
	The context layer Gartner asks for, and where a migrated form already holds it	40
11.	**PROOF, AND HOW WE COMPARE**	**41**
	Proven equivalence	41
	Customer spotlight: a large manufacturer	41
	Customer spotlight: a utility	41
	Migration automation metrics	41
	How ORMIT™-OpenJava compares with the other migration tools	43
	Summary: choose modernization, not just migration	44
12.	**THE FIRST STEP**	**45**
	Your team	45
	Not just the what, but the how	45
	Hypercare and post-go-live support	46
	The financial case for migration over maintenance	46
	A proof of concept	46
	From the migration to the first agent: seven steps	47
	What RENAPS brings	48
	SOURCES	**49**

Executive Summary

01

For twenty years, the question for Oracle Forms owners was *which web stack* to migrate to. In 2026 the question changed. Oracle Forms 12c leaves Premier Support in December 2026, the developers who wrote the forms are retiring, and the software market, the analysts and the largest vendors now agree on where enterprise applications are going: the work moves to AI agents, and the screen becomes the exception.

That is good news for organizations that run on Oracle Forms, because of what their forms contain. Behind every screen sit hundreds of triggers, validations and procedures: the rules that decide what a valid transaction is, who may change what, and when a posting is refused. Those rules are exactly what an AI agent needs in order to be trusted in a transactional system, and exactly what a rewrite, by hand, with AI or into a packaged product, loses along the way.

ORMIT™-OpenJava by RENAPS migrates the rules with the screens. It is a highly automated platform that turns Oracle Forms applications into modern web applications: React or Angular in the browser, Java + Node.js or Node.js alone on the server, the form's logic in TypeScript, and your database packages where they are. At its heart, OFCS, the Oracle Forms Compatibility Service, keeps every trigger, validation and procedure as executable code, and that code already runs without a screen. The migration you deploy today is therefore the one an agent can be added to. First comes an application any agent can operate, your own, a platform vendor's or RENAPS's, under an identity of its own and the same rules an employee goes through. Then, process by process, an agent takes over the work, and your people move from data entry to supervision.

This white paper shows

- Why modernization can no longer wait: Oracle's support dates, a shrinking talent pool, the cost of the status quo, and the figures from Gartner, IDC, McKinsey and Ensono.
- Why 2026 ended the idea that a modern web application is the final destination, and why it remains the necessary first step.
- What tomorrow's enterprise AI looks like: an agent inside the bureaucracies that run on Oracle Forms today, and why the rules in your forms are the control most agentic AI projects lack.
- How ORMIT™-OpenJava migrates those rules: the analysis, OFCS, two backends, React or Angular, a cloud-ready, secure and fast architecture, with Oracle Reports and the database covered too.
- How the migrated application becomes operable by any agent, safely, and how a process is handed to one with your people approving its work.
- The proof: tested equivalence, customer results, the comparison with the other migration tools, and the business case.
- A first step small enough to decide this quarter.

Oracle Forms at the End of the Road

02

Four decades of Oracle Forms

Oracle Forms has evolved over more than four decades, from character-based screens to web deployment. Despite continuous updates, its architecture is still anchored in the client-server model of the 1990s: stateful, monolithic, tied to Oracle WebLogic. That is what makes it hard to meet today's cloud, mobile and integration requirements, and what a modernization removes.

Oracle Forms version history

- **1979 IAF (Interactive Application Facility)**
  Initial release. Character-based, with IAG as compiler and IAP as runtime interpreter.
- **1982 FastForms**
  Introduced with Oracle Database version 4; adds form generation and editing tools.
- **1985 SQL*Forms 2.0**
  Character-based; introduces the screen painter and a primitive trigger language.
- **1986 SQL*Forms 2.3**
  Enhanced procedural capabilities; the foundation of Oracle Financials.
- **1988 Oracle Forms 3.0**
  First to support PL/SQL; improved IDE; character mode with GUI window support.
- **1992 Oracle Forms 4.0**
  First true GUI version; new IDE with object-specific editors.
- **1993 Oracle Forms 4.5**
  Major IDE overhaul: object navigator and property sheets, aligned with Visual Basic.
- **1995 Oracle Forms 5.0**
  Accompanies Oracle Database version 7; enhanced GUI features.
- **1998 Oracle Forms 6.0 / 6i**
  Introduces Forms Server for web deployment; keeps client-server support.
- **2001 Oracle Forms 9i**
  Web deployment only; the client-server runtime is removed; deeper Java integration.
- **2003 Oracle Forms 10g**
  Forms 9.0.4 rebranded, aligned with Oracle Database 10g; improved web deployment.
- **2009 Oracle Forms 11g**
  External events, JavaScript integration and performance monitoring tools.
- **2015 Oracle Forms 12c**
  Java Web Start support, enhanced REST integration, improved UI components.
- **2024 Oracle Forms 14.1.2**
  Latest release: a facelift, and native REST data source support.

What Forms 14c changes, and what it does not

Forms 14c brings modest improvements. The notable one is native REST data sources: a form can now expose or consume an API without custom code, which helps interoperability and falls far short of a modernization. The rest is browser compatibility and small adjustments to the UI. The core stays monolithic, stateful and tied to WebLogic, with no mobile design, no microservices and no place in a DevOps pipeline. Forms 14c extends the life of a legacy system. It does not remove the limits that keep an organization from innovating, scaling and hiring.

The pain points driving modernization

In every industry, IT leaders are under pressure to modernize aging Oracle Forms systems. The platforms have delivered decades of reliable service, and they now stand in the way of what the rest of the organization is doing. Four pain points come up every time.

A shrinking talent pool

The expertise to maintain Oracle Forms is disappearing. As the article *The Quest for the Last Oracle Forms Developer* shows, most Forms developers are nearing retirement, and fewer than one organization in ten reports an active developer under 45. Schools no longer teach Forms, and younger developers choose the technologies in demand. The risk is continuity: support, maintenance and institutional knowledge.

Outdated architecture, limited innovation

Forms is tightly coupled to the Oracle Database and to proprietary middleware. native REST data sources arrived only in 14c and remain limited; there is no native support for microservices, mobile or responsive interfaces. Today's IT runs on APIs, microservices and DevOps pipelines, and connecting Forms to that world takes workarounds that lengthen every delivery and add technical debt.

Rising cost of ownership

Licences for Oracle middleware, aging infrastructure and the specialized talent needed for support add up to a steep total cost of ownership, especially next to cloud-native alternatives.

Poor digital experience

Users expect responsive, intuitive interfaces. Forms applications rarely meet that expectation, which costs satisfaction, productivity and adoption, among digital-native employees and external users above all.



What the analysts measure

61%

Organizations that paused, delayed, scaled back or abandoned an IT modernization initiative in the past 24 months (Ensono, 2026)

20–40%

Share of development time that unmanaged technical debt can consume (IDC, 2025)

40–50%

Acceleration of modernization timelines with generative AI, and 40% less cost from technology debt (McKinsey, 2024)

48%

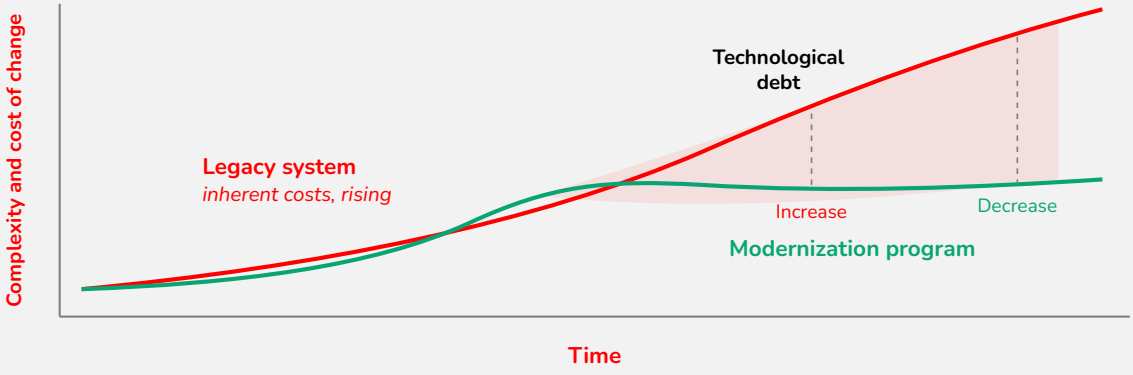
Enterprise digital initiatives that meet or exceed their business outcome targets (Gartner CIO Survey, 2024)

Read together, the findings say one thing: delaying modernization does not only cost more, it limits the organization's ability to innovate, compete and grow, and the gap between recognizing it and acting on it is where technical debt builds.

The cost of waiting

What was once a stable platform is now a roadblock. The cost of change on a legacy system rises every year, as the people who know it leave and the systems around it move on. A modernization program costs more at the start and less every year after. The gap between the two curves is the technological debt the organization chooses to carry. **Modernization is no longer optional.** This white paper is about doing it once: into an application that removes today's limits and is ready for what comes next. The next chapter is about what that is.

The rising costs of the status quo



2026: The Year the Market Repriced Software

03

The SaaSpocalypse, and the rebound

On January 12, 2026, Anthropic released Claude Cowork, an agent that does professional work with a user's files and business tools. On January 30 it added eleven open-source plugins, one per white-collar function: finance, legal review, sales, customer support and others. On February 5, OpenAI launched Frontier, a platform to build and run agents that operate other enterprise software.

Investors read the signal at once. Between January 12 and February 23, the S&P Software & Services index lost a quarter of its value. The session after the plugin release alone erased more than 285 billion US dollars of market capitalization. A Jefferies strategist gave the episode its name: the SaaSpocalypse.

−25%

S&P Software & Services index, January 12 to February 23, 2026

$285B

Market value erased in one session after the agent plugins shipped

+41%

Software stocks (the IGV fund) from their April 13 low to October 12, 2026, back at their highs for the year

The fear was specific. Enterprise software is sold per seat, and an agent that executes a workflow needs no seat. When the logic moves to the agent, the application behind it becomes a database with an interface nobody opens.

Then the market changed its mind, in a way that matters more than the scare. By October 2026 the software index had regained everything it lost: up 41% from its April low, at new highs for the year, flat over twelve months. The companies that led the recovery were not the ones that denied the shift. They were the ones that built agents into their core products. The verdict was not that software is dead. It was that software without agents gets repriced, and software that embeds them keeps its value.

Forrester and Gartner put dates on it

Forrester answered the selloff within the week, and did not dispute its direction. Whichever way a company deploys agents, Forrester wrote, "it'll erode the remit of your current enterprise SaaS applications", and seat pricing will give way to consumption and outcome pricing.

Gartner puts dates on it:

By the end of 2026, 40% of enterprise applications will embed task-specific AI agents, up from less than 5% in 2025.

By 2028, a third of user experiences will move from native applications to agent interfaces.

By 2030, 234 billion US dollars of enterprise application spending is at risk from agentic AI, and the user interface stops being a differentiator.

By 2028, more than half of enterprises will stop paying for AI that assists employees, in favour of platforms that deliver the outcome of a process. Vendors that only bolt AI onto existing applications face margin compression of up to 80% by 2030.

Gartner's trends for 2027: from adoption to governance

On October 20, 2026, Gartner published its Top 10 Strategic Technology Trends for 2027. Agents are no longer on the list as a trend. They are the platform the list assumes every organization now runs, and the question has become what governs them. In Gartner's words, the conversation in 2027 "shifts from adoption to governance": traditional IT relied on software scarcity, 2027 brings software abundance, and CIOs "must shift from managing technology assets to governing machine behavior".

+80%

Agentic AI accuracy for organizations that prioritize semantics in AI-ready data, by 2027

25%

Enterprise security breaches that will trace back to AI agent abuse, by 2028

80%

New applications that leading organizations will make disposable and easy to replace, by 2030

Three of the ten trends describe what an Oracle Forms migration has to preserve:

AI context platforms.

The knowledge an agent needs "to interpret inputs, make decisions and act in alignment with business objectives". Gartner gives the layer three components: semantics, which include the business rules and policies; operational state, the live data; and provenance, the record of decisions, actions and outcomes. Without it, organizations face "overprivileged agents, compliance failures, low trust and excessive token consumption".

AI agent identity.

Agents "expose massive vulnerabilities in legacy systems that can't deliver just-in-time access", and organizations that do not resolve it will see over half of their AI initiatives halt. Gartner's first instruction: an agent never shares a person's credentials. It gets an identity of its own, with policy-controlled access.

Disposable applications.

The first action Gartner lists is to "classify which core capabilities require permanent software and which can be served by temporary, on-demand solutions". The application portfolio becomes "a dynamic system that continuously renews itself as redundant assets are retired".

Read together, they say where the value of an Oracle Forms application sits. The screens are the replaceable part. The rules are the permanent core, and they are the one component of Gartner's context layer that no platform vendor can write for an organization, because they exist only in its forms. A migration that keeps them executable, under an identity the database controls, is the part of the 2027 agenda an Oracle Forms owner can deliver from what it already has. Chapter 10 shows the mapping in detail.



Even the vendors say so

In September 2026, Salesforce opened its Dreamforce conference by addressing the selloff head on. Its CEO, Marc Benioff, dismissed the talk of an end of software, then conceded the point that matters:

> ***"SaaS apocalypse was not about the end of software but it may be about the end of software that makes humans do all the work."***
>
> *Marc Benioff, CEO of Salesforce, Dreamforce keynote, September 2026*

He said it while launching an agent interface over Salesforce's own applications, built with Anthropic's Claude. The vendor that mocked the apocalypse answered it by embedding agents. Software that makes humans do all the work is a precise description of a data-entry screen. It is also a precise description of what a conventional Oracle Forms rewrite delivers: the same screen, in a browser, with the rules behind it lost on the way.

Why screens are no longer the whole destination

None of this means that screens disappear. High-volume keyboard entry stays faster on a screen, and most tasks will keep one for years. Nor does it mean waiting. Oracle's published support dates give Forms 12c until December 2026 on Premier Support, and Forms 14.1.2 until 2030: the migration has a date either way, and it still has to happen. It means that the migration has to be chosen for what it makes possible afterwards. A migration project runs one to three years; the figures above describe the same horizon. A migration that delivers screens and loses the rules ends where it started. A migration that delivers screens and keeps the rules executable is the platform the agents will run on, without a second migration.

The question for an Oracle Forms owner in 2026

Not only "which web framework?", but "when an agent runs this process on the migrated application, what stops it from doing the wrong thing?" The next chapter is about who that agent is. The chapter after it is about that question, and why your forms already answer it, provided the migration keeps the answer.

Tomorrow's AI Is a Bureaucrat

04

An agent, not a tool

The historian Yuval Noah Harari has spent two years correcting the picture most people hold of artificial intelligence. In his 2026 Tanner Lecture at Oxford, he put the distinction in one sentence:

> ***"It's not a tool in our hands. It is an agent with its own hands."***
>
> *Yuval Noah Harari, Tanner Lecture on Human Values, Oxford, 2026*

And he named where that agent will live. Not in the streets, and not in a robot body:

> ***"Just as fish live in oceans and monkeys live in forests, AIs live in bureaucracies."***
>
> ***"In the coming years, millions of AI bureaucrats will increasingly take over the world's bureaucracies. AI bankers will decide whether to give you a loan."***
>
> *Yuval Noah Harari, same lecture*

The bureaucracies that run on Oracle Forms

Harari says this as a warning. For an organization that runs on Oracle Forms, it is also a forecast, and it raises one question: who writes the rules the AI bureaucrat will follow?

A bureaucracy, in Harari's sense, is any system that lets strangers trust each other at scale: a bank, an insurer, a hospital's billing, a ministry's permits, a payroll. These systems are made of records, rules and approvals. For three decades, a large share of them have been built on Oracle Forms, precisely because Forms made rules and approvals easy to express: a trigger on every event, a validation on every field, a procedure behind every button.

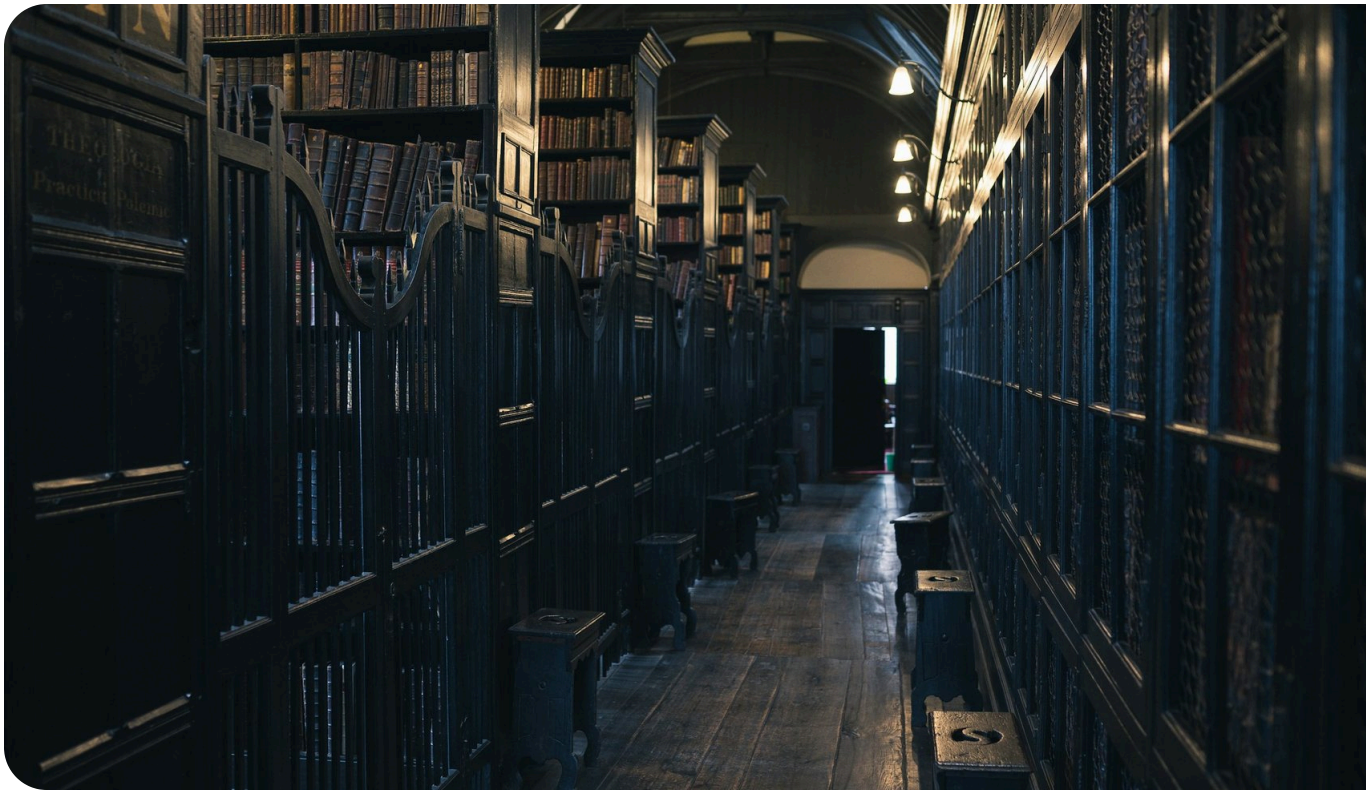
So the AI bureaucrat Harari describes will not arrive in a new system. It will arrive in yours. The loan it decides on is in a block of a form. The claim it settles, the invoice it posts, the leave it approves are records that today pass through a WHEN-VALIDATE-RECORD trigger before they reach the database.

No rules, no agent

Gartner also measured how these projects fail. More than 40% of agentic AI projects will be cancelled by the end of 2027, for three reasons: escalating costs, unclear business value, and inadequate risk controls. In October 2026, its trends for 2027 named the missing control: a context layer whose semantics are the organization's rules and policies, without which agents are overprivileged and untrusted. Only 22% of enterprises capture significant value from generative AI today, while 42% plan to deploy agents by the end of 2026. That control is the one a transactional system cannot work around. No organization will let an agent approve payments, settle claims or post journal entries without guardrails it can audit.

The conclusion

An AI bureaucrat is accepted only if it obeys the rules. Those rules have to be code, enforced before the database, and the same for the agent as for the employee. Oracle Forms owners already have them. The only question is whether the migration keeps them.



What Your Forms Contain

05

Where the rules live

Ask an Oracle Forms team where the business rules are and they will point at the database packages. Half of the answer is there. The other half is in the forms: WHEN-VALIDATE-ITEM triggers that refuse a date outside the open period, PRE-INSERT triggers that derive a value the user never types, KEY-COMMIT triggers that call a package in a precise order, properties that make an item read-only once a record is posted. A large application holds thousands of these, written over twenty years by people who have since retired. Nobody has a complete list.

Why Oracle Forms is so “sticky”

That richness is why Forms is still here. Its reference manual describes an event-driven environment where client-side and server-side logic mix freely, often in hybrid patterns:

15
item types

92
trigger types, one on every event

282
built-ins, callable from any trigger

476
properties, on every object

This expressiveness, what we call *code diversity*, is what let Forms handle stateful, desktop-like applications such as multi-block forms, which purely declarative web platforms struggle to reproduce. It is also what makes leaving hard. Migrating away is more than replatforming: the logic, the user interactions and the architecture all have to be carried across, with each trigger fired at the point where Oracle Forms would have fired it. The target has to be as open and as expressive as the source. TypeScript with React or Angular is. A declarative platform is not.

The question a migration has to answer

Not “which screens?”, but “which rules, and where do they run afterwards?”. Every path in the next chapter fails on the second half of that question. Chapter 7 shows how ORMIT™-OpenJava answers it, starting with the complete inventory nobody has.

The Paths That Lose the Rules

06

Four paths are open to an Oracle Forms owner besides a migration that keeps the rules: a rewrite, by hand or with AI; an assistant beside the old screens; Oracle APEX; a packaged product. Each has merits. Each loses the rules.

The rewrite trap

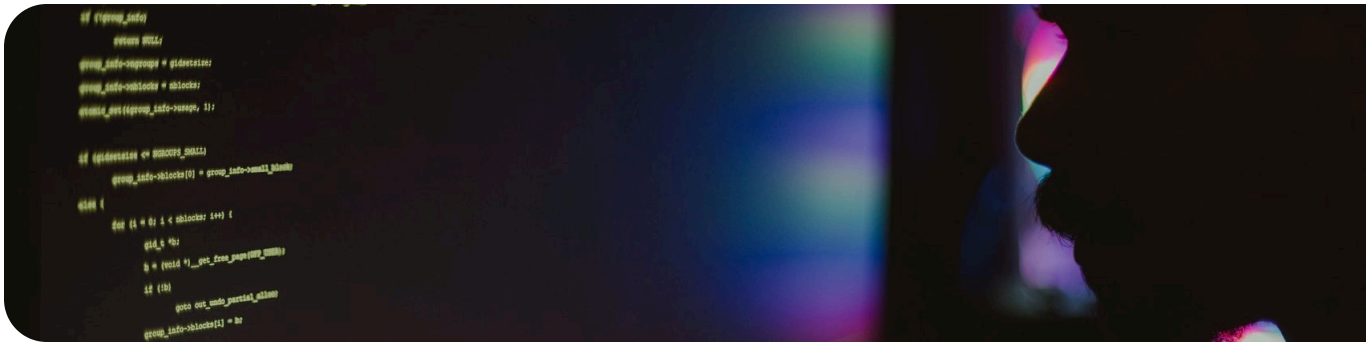
Every rewrite, whether by hand or with a code-generating AI, starts from a specification. The specification is written from the screens and the obvious workflows. The rules are discovered later, one incident at a time: a posting accepted into a closed period, a discount applied without the approval the old form demanded, a record saved with a field the old trigger would have derived.

One lost rule, and what it costs with an agent

A form carries a validation trigger that refuses a posting date outside the accounting period currently open. The rewrite reproduces the screen, the lookup and the save, but not the trigger, because nobody asked for it. With a human operator, the error surfaces at month end and gets corrected. With an agent executing hundreds of postings a day, the same gap produces hundreds of entries in a closed period before anyone looks. The rule the agent needed most is the one the rewrite dropped.

Why a bolted-on assistant is not enough

The other shortcut is to keep the screens and add an AI assistant beside them. It helps the operator type faster. It does not change who does the work, and it is the strategy Gartner expects to compress vendor margins by up to 80% by 2030, because the buyer stops paying for assistance and starts paying for outcomes.



Why Oracle APEX is a dead end for complex Forms applications

APEX looks like the natural low-code path for an Oracle shop. Under the hood, it has limits that make it a poor fit for complex Oracle Forms applications, for three reasons.

Functional mismatch

APEX is a declarative platform with limited extensibility. It lacks the procedural depth and the event model needed to reproduce trigger-based, stateful applications: built-ins, OLE integration and most event triggers simply do not translate. Migrating to APEX means a full UI redesign, triggers rewritten by hand, functional compromises and a redesign of the business logic, which is exactly the risk a modernization is supposed to remove. APEX is a declarative platform with limited extensibility. It lacks the procedural depth and event model needed to replicate Oracle Forms' complex, trigger-based, stateful applications.

Oracle's answer: APEX 26.1

Oracle has not stood still. APEX 26.1, released in May 2026, ships AI agents with tools written in SQL or PL/SQL, a choice of models and an approval step for sensitive actions. Oracle also publishes an AI assistant that turns Forms modules into APEX modernization artifacts: a functional specification, a traceability matrix, a report. It is a serious option, and it has the same weakness as every rewrite. The Forms modules are analyzed into a specification, and the APEX application, agents included, is built anew from it. The triggers that encoded twenty years of rules become documentation, and the agent's tools are new code. The question to ask any migration path is not whether it offers agents, but whether the agent runs under the rules you already have.

Strategic lock-in

APEX only runs on Oracle Database. Choosing it keeps you on Oracle licensing and infrastructure, blocks any future database migration, and limits what open-source alternatives could save.

APEX

Vendor lock-in

- Tied to Oracle Database
- Limited database migration options
- Higher total cost of ownership

React / Angular

No lock-in

- A path to PostgreSQL with ORMIT™-Cortex
- Flexibility to reduce costs
- Choice of deployment

A packaged product: the hidden cost of starting over

For an organization replacing Oracle Forms or Reports, a commercial off-the-shelf (COTS) product looks attractive: pre-built features, vendor support. Beneath that promise lies a path of risk, disruption and hidden costs.



Disruption to operations

Familiar processes are abandoned for a vendor-defined model, and daily business continuity pays for it.



Loss of business-specific logic

A Forms application typically holds hundreds of thousands of lines of carefully optimized logic. Re-implementing it in a package is slow, error-prone and full of functional gaps.



Change management overhead

Entire teams retrained, processes redefined, user acceptance rebuilt from scratch.



Much higher implementation costs

Extensive consulting, long timelines, and per-user or per-module licensing.



Higher risk of failure

Many COTS projects miss their targets through poor fit with the business or underestimated complexity.



No leverage of past investments

Starting over throws away years of workflows, validations and user knowledge embedded in the existing system.

A smarter path: leverage, don't replace

Instead of replacing everything, a direct migration with ORMIT™-OpenJava and ORMIT™-Jasper lets you preserve the business logic you have proven, keep your users' familiarity and operational continuity, modernize the architecture without reinventing the wheel, and lower the risk and the total cost of the transformation.

The next chapter describes the one path that keeps the rules.

Migrate the Rules, Not the Screens

07

ORMIT™-OpenJava does not reinterpret a form. It migrates it: every trigger, validation and procedure becomes TypeScript that OFCS, the Oracle Forms Compatibility Service, fires at the moment Oracle Forms fired it. This chapter describes the platform, the analysis every project starts with, and the runtime that makes a migrated application ready for agents.

What ORMIT™-OpenJava is

ORMIT™-OpenJava is a modernization platform, not only a conversion tool. Its automation engines analyze, convert and optimize Oracle Forms code into TypeScript applications, mapping every part of a Forms system, triggers, built-ins, data blocks and PLL libraries, onto components: Redux Toolkit or NgRx Signals for the form's state, a REST API, React or Angular, and Node.js. It preserves the logic faithfully and delivers it with little manual work. It ships with the operational environment as well: integration hooks, security standards, documentation generators and post-go-live tooling. Its backend, Java + Node.js or Node.js only, and its front end, React or Angular, adapt to your team's strengths.

What you get

- **Automation out of the box:** hundreds of prebuilt templates, patterns and converters for Forms components, triggers, libraries and built-ins. Shorter timelines, lower cost.
- **Business logic preserved:** triggers, validations and core logic migrated into TypeScript and run by OFCS, in the browser and again on the server before every save reaches the database.
- **Standard components:** MUI for React, Angular Material for Angular; nothing custom to maintain.
- **Cloud-native and portable:** ready for AWS, Azure, Google Cloud, Oracle Cloud or a private cluster.
- **Lower total cost of ownership:** no WebLogic, no Forms runtime, an open-source cost structure, simpler support.
- **A modern, responsive UI:** on phones, tablets and desktops, with no additional work.

Start with a complete inventory: ORMIT™-Analyzer

A successful migration begins with knowledge, not guesswork. Every RENAPS project starts with a full system analysis by ORMIT™-Analyzer, our scanning and assessment tool. Its report lets you:

- Understand the exact structure and scope of all Forms and Reports objects (FMB, MMB, OLB, PLL, RDF).
- Identify all built-ins, triggers, instructions and complexity levels, per module.
- Evaluate source code quality, business logic distribution and code duplication.
- Pinpoint obsolete features, dead code and opportunities for optimization.
- Establish the ideal migration order from the dependencies between modules.
- Get a detailed estimate of the time, cost and team effort of every project phase.

The result is a roadmap with no surprises: every step scoped, calculated and backed by automation. Nobody had the complete list of rules. Now you do.

OFCS: rules that run without a screen

What OFCS is

OFCS, the Oracle Forms Compatibility Service, is the runtime at the heart of ORMIT™-OpenJava. It executes a migrated form's rules, every trigger, validation and procedure, as TypeScript, at the moment Oracle Forms executed them: in the browser as the user works, and in a Node.js server with no screen attached. Everything this white paper describes runs on it.

94

Oracle Forms trigger types reproduced by OFCS, fired at the point where Forms fires them

~250

Oracle Forms built-ins and SQL functions available to migrated code

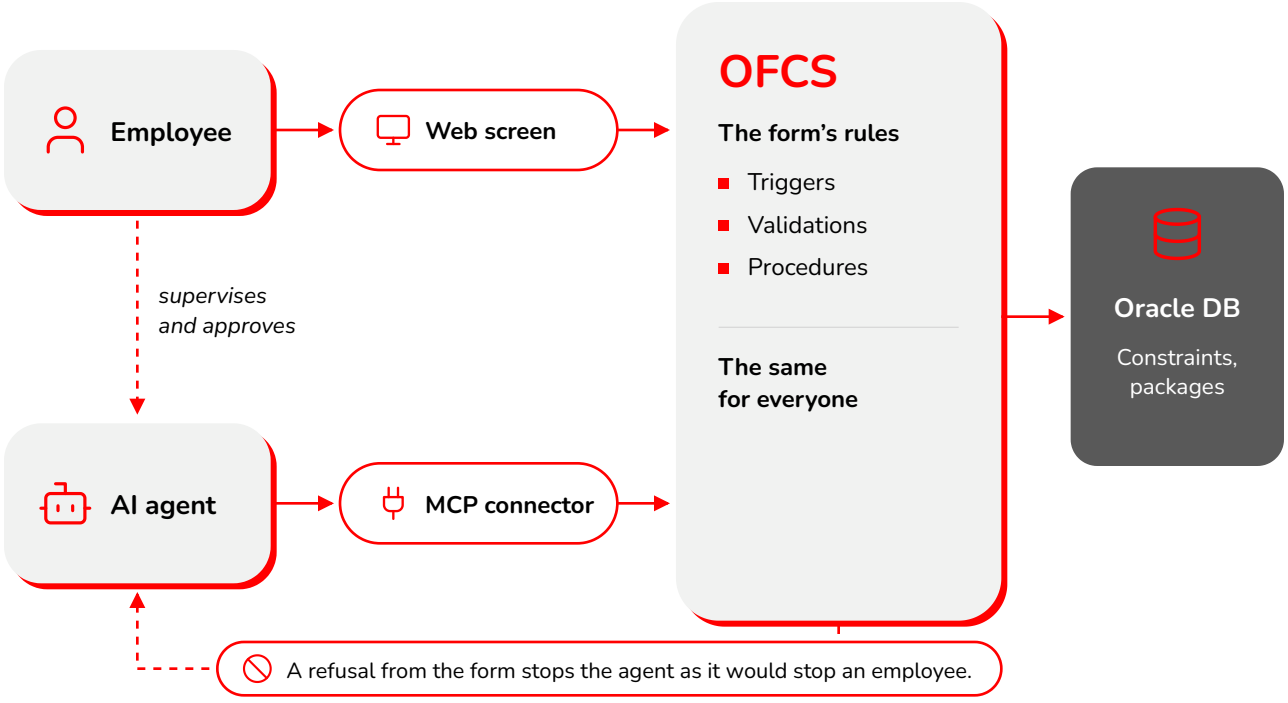
2×

Every save is validated twice: in the browser as the user works, and again on the server before the database

The second validation is the detail that matters for agents. Since 2026, every migrated application ships with a validation server: a Node.js process running the same OFCS, with no screen attached. Before any insert or update reaches Oracle, this server loads the form, replays its validation and commit triggers on the record, and refuses the save if the form would have refused it. Going around the screen does not go around the rules.

An AI agent is, from the form's point of view, another caller without a screen. Through the connector (the Model Context Protocol, MCP), it will read the form's blocks and items, set values in the form's navigation order so that each item's validation and side effects run, press the form's buttons, and commit. Every one of those actions will go through OFCS. When a trigger raises an error, the agent will receive the form's own message and stop, as an employee would stop at the same message.

The agent and the employee go through the same rules.



Screen or connector: the same rules. In this design, the agent has no path of its own to the database: it goes through the connector, under an identity of its own, then through the same rules as the screen, before anything reaches the database.

What you get today, and what it enables tomorrow

Today, ORMIT™-OpenJava delivers a React or Angular application that your users operate as they operated the forms, with the same keys, lists of values, validations and messages. That application is the product. What it enables next is an application any agent can operate, then an agent that takes over a process: chapter 10 describes both tiers. Clients who migrate now are not buying a stopgap. They are buying the only kind of migration an agent can be added to without starting over.

What your users keep

One strength of the approach is what it keeps for the people who use the forms every day. Keyboard shortcuts (F1 to F12), tabbing between fields, lists of values, validations and messages, multi-form interactions: all of it is carried across, so the migrated application is operated as the forms were. It is also responsive by default, usable on a desktop, a tablet or a phone, iPhones and iPads included, without rework. Change management is lighter, and the rollout smoother.

What stays in the database stays in the database

Migrating the rules does not mean moving them. Package logic, constraints and grants remain in Oracle, untouched, and every write runs in the database under its caller's own Oracle identity: the employee's from a screen today, and the agent's own once the connector is in place. The roles and grants the database already enforces keep working. The form's rules, which Oracle Forms ran in its own runtime, now run in OFCS. Nothing is demoted to a prompt.



What this makes possible

For the migration

- The behaviour of the original form is reproduced, bugs included, and proven by end-to-end tests written from the original PL/SQL.
- The same rules serve a React or an Angular front end, and the same test suite runs on both.
- The application runs on the client's existing Oracle database.

For the agent

- The form's metadata describes every block, item, list of values and button: what an agent needs in order to act.
- The validation server already answers callers without a screen.
- The end-to-end tests that prove the migration become the way to grade the agent.

The difference, in one line

Everyone can put an AI in front of a transactional system. Putting one there safely takes rules that are code, enforced before the database, and the same for the agent as for the employee. OFCS, at the heart of ORMIT™-OpenJava, keeps the rules you already have in exactly that form, and runs them for the agent as it runs them for the employee.



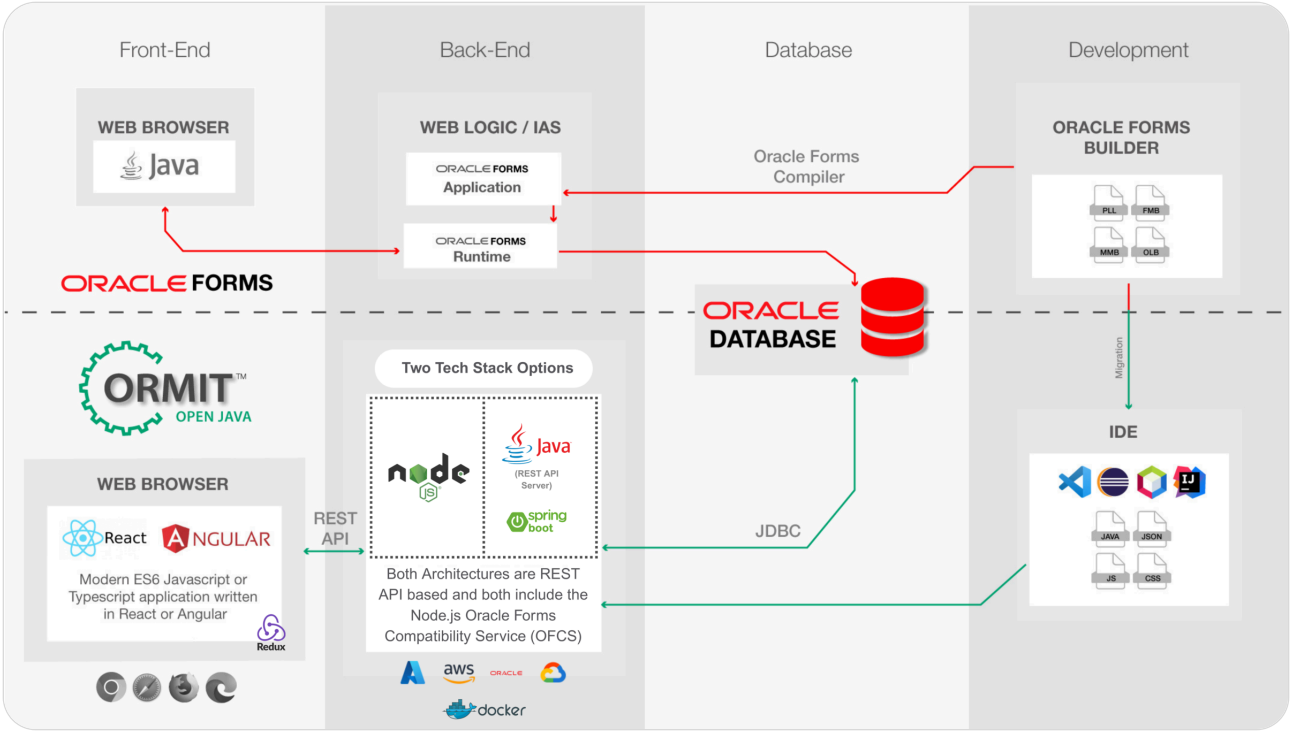
The Architecture: Flexible, Open, Yours

08

Two backends, one result

ORMIT™-OpenJava fits the stack you have and the one you are going to. Some teams run Java infrastructure, know Spring Boot and operate application servers such as Tomcat or JBoss. Others are invested in JavaScript and want a single TypeScript stack. Both are served:

- **Java + Node.js (default):** a Spring Boot REST API, plus a Node.js validation server where OFCS re-runs the form's validation before every insert and update. For teams with Java expertise and an application server in place.
- **Node.js only:** a lightweight, full-stack TypeScript backend for organizations that want to stay 100% Java-free.



From Oracle Forms to ORMIT™-OpenJava. Above the line, the Oracle Forms stack: Java applet, WebLogic, Forms runtime and compiler, Forms Builder. Below it, the migrated stack: React or Angular in the browser, a REST API in Node.js or in Java with Spring Boot, both with the Node.js OFCS, the same Oracle Database, and any modern IDE. Both deliver the same application: modern, scalable, API-driven.

The two options, side by side

Component	Java + Node.js (default)	Node.js only
UI framework	Angular or React	Angular or React
State management	Redux Toolkit; NgRx Signals for Angular	Redux Toolkit; NgRx Signals for Angular
Backend language	Java (API), TypeScript (OFCS)	TypeScript
REST API server	Java 21 (Spring Boot 4) + Node.js validation server	100% Node.js
Container	Docker (with embedded Tomcat)	Docker or Node-native
Application server	Tomcat, JBoss or equivalent	Node.js
Oracle Forms compatibility (OFCS)	Browser + Node.js validation server	Browser + Node.js backend

React or Angular



Both front ends are supported and maintained by RENAPS. The choice follows your team's preference and your standards: the lightweight, customizable nature of React, or the all-in-one structure of Angular. UI behaviour, business rules, validations and Oracle Forms logic are handled identically under both, and the same automated end-to-end tests run against the two front ends.

Aspect	React	Angular
Architecture	A library focused on UI rendering	A full framework, batteries included
Learning curve	Lower; flexible, needs third-party libraries	Steeper; routing, dependency injection and forms built in
Flexibility	Highly customizable	Opinionated structure, suited to large applications
Community and support	Widely adopted, large ecosystem	Strong enterprise support, government included
Integration	Easy to integrate with existing applications	Well suited to greenfield applications with a strict structure

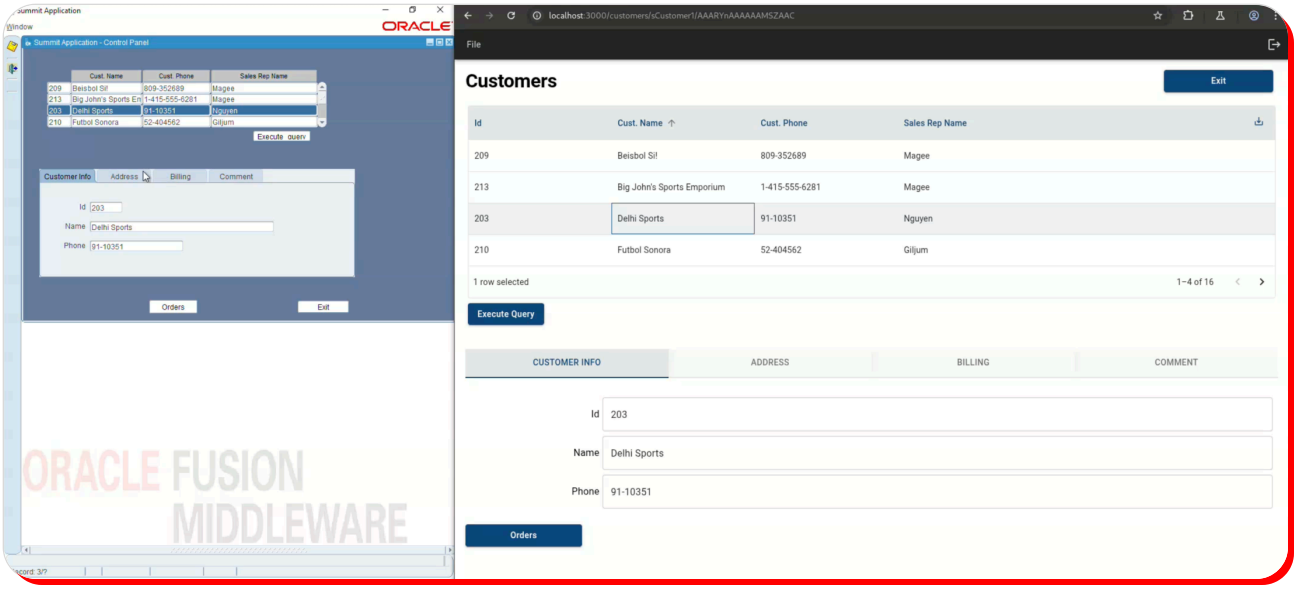
Oracle Forms to the modern stack: technological equivalents

The table shows how Oracle Forms components map onto their counterparts in ORMIT™-OpenJava, for a React or an Angular front end: the depth of functionality retained, and the tools that reproduce each behaviour.

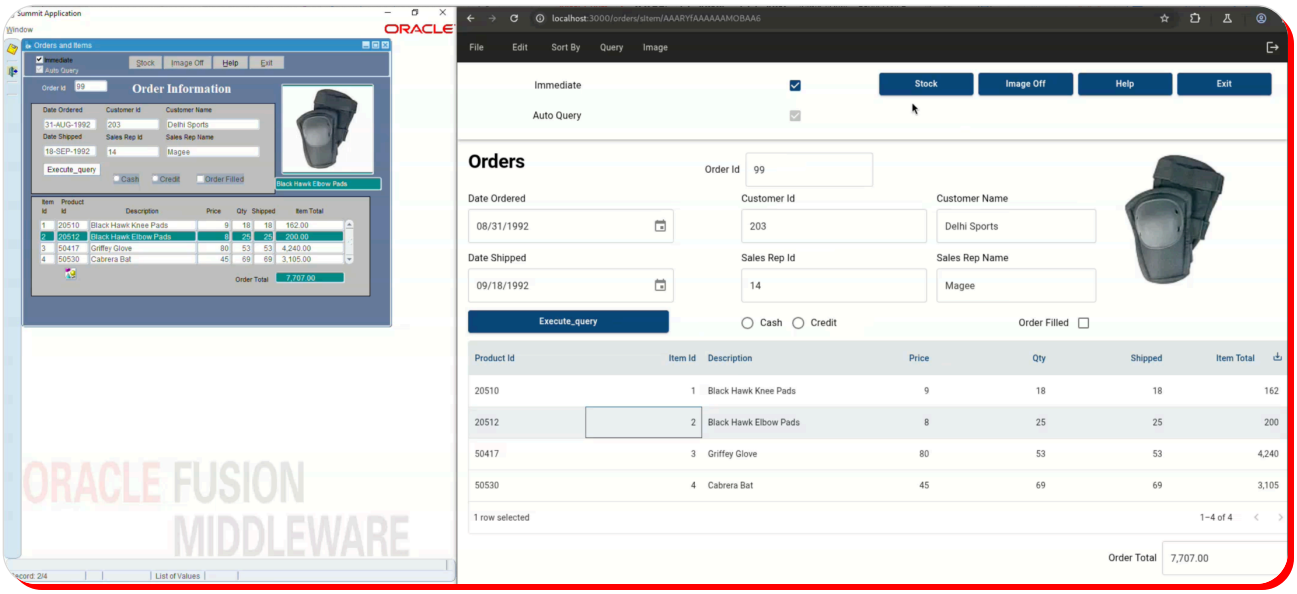
Oracle Forms	ORMIT™-OpenJava, React	ORMIT™-OpenJava, Angular
Absolute positioning (X-Y)	CSS Grid and Flex positioning	CSS Grid and Flex positioning
Field styling	CSS + Material UI (MUI) themes	CSS + Angular Material themes
Canvas	React components	Angular components
Canvas section	React zone	Angular zone component
Item	Form store (Redux Toolkit)	Form store (NgRx Signals)
Item field type	MUI React components	Angular Material components
Properties	Form store (Redux Toolkit)	Form store (NgRx Signals)
Menu	MUI menus	Angular Material menus
PL/SQL	TypeScript	TypeScript
Database-based access control	Signed token; each write runs under the user's own Oracle account	Signed token; each write runs under the user's own Oracle account
Database table	Form store (Redux Toolkit)	Form store (NgRx Signals)
Built-ins	OFCS built-in functions	OFCS built-in functions
PLL library	Shared or form-level TypeScript functions	Shared or form-level TypeScript functions
Form parameters	Form store (Redux Toolkit)	Form store (NgRx Signals)
Window	React component emulating a page or dialog	Angular component emulating a page or dialog
Multi-record block	MUI X Data Grid Pro	AG Grid Community
Single-record block	Model + store + data slice	Model + store + data slice
Alerts	React component + data slice for the alert's properties	Angular component + data slice for the alert's properties
Triggers	TypeScript function, fired by OFCS	TypeScript function, fired by OFCS
Validation on save	Re-checked by OFCS on the server before saving	Re-checked by OFCS on the server before saving

What it looks like: Oracle Forms next to ORMIT™-OpenJava

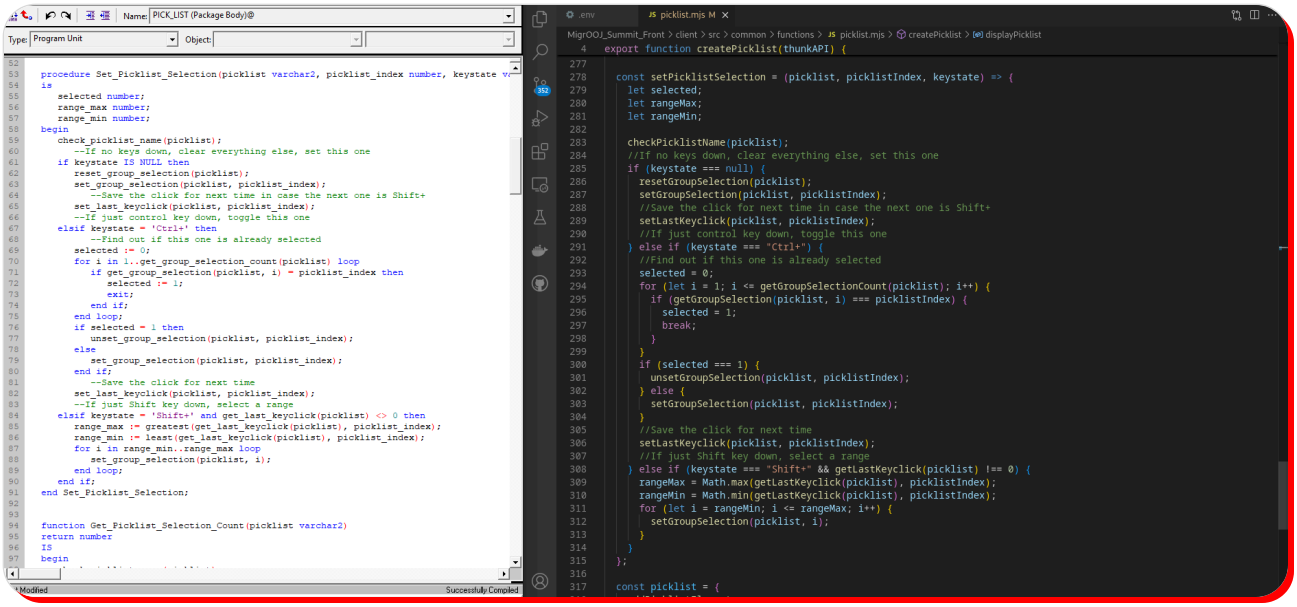
The following pages show real examples of how Oracle Forms artifacts are transformed, taken from the Summit demo application that ships with Oracle Forms, migrated to React.



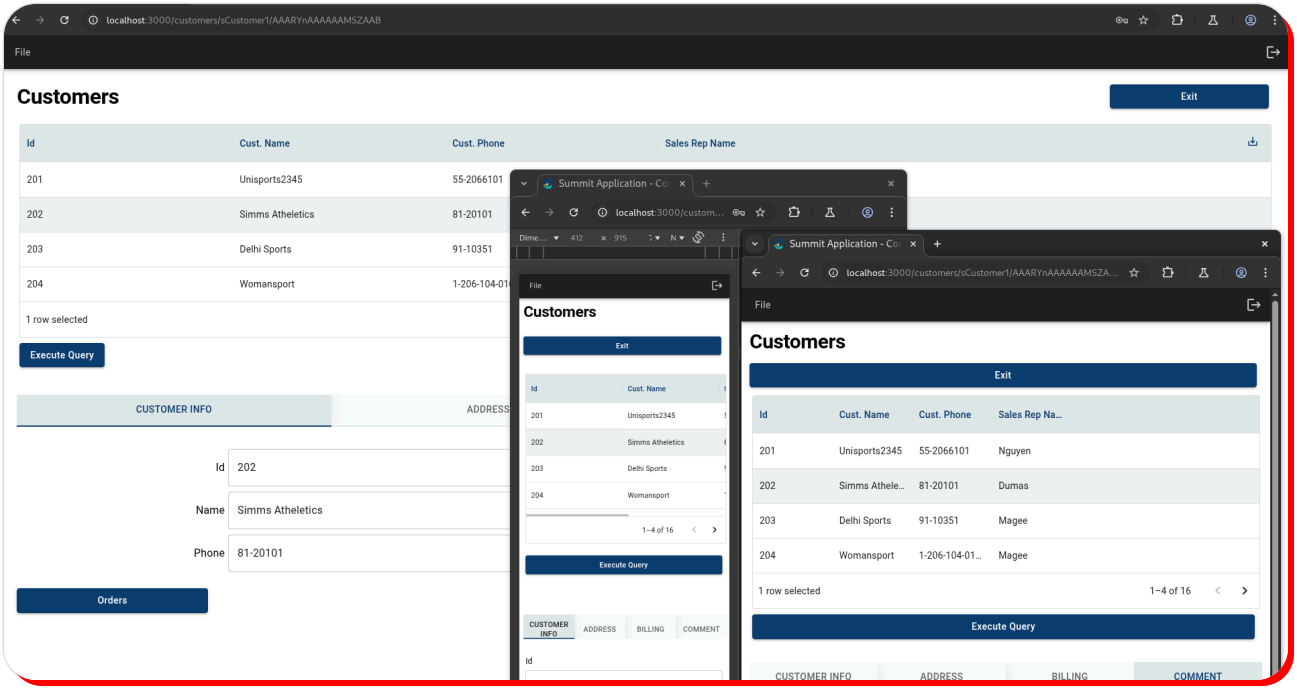
Side by side. The Customers form in Oracle Forms, and the same form migrated: same block, same fields, same keys, as a responsive page in a browser, with the tabbed canvas below the grid.



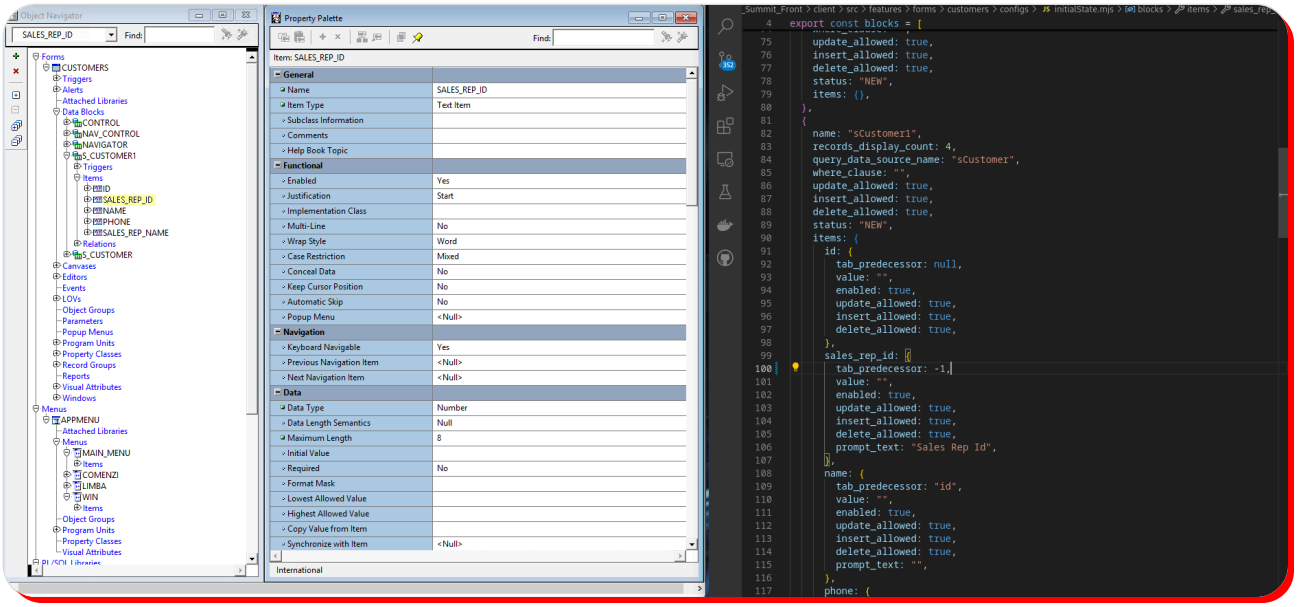
The Orders form. A master block, a detail grid and an image item, migrated with its lists of values, its check boxes and its validations.



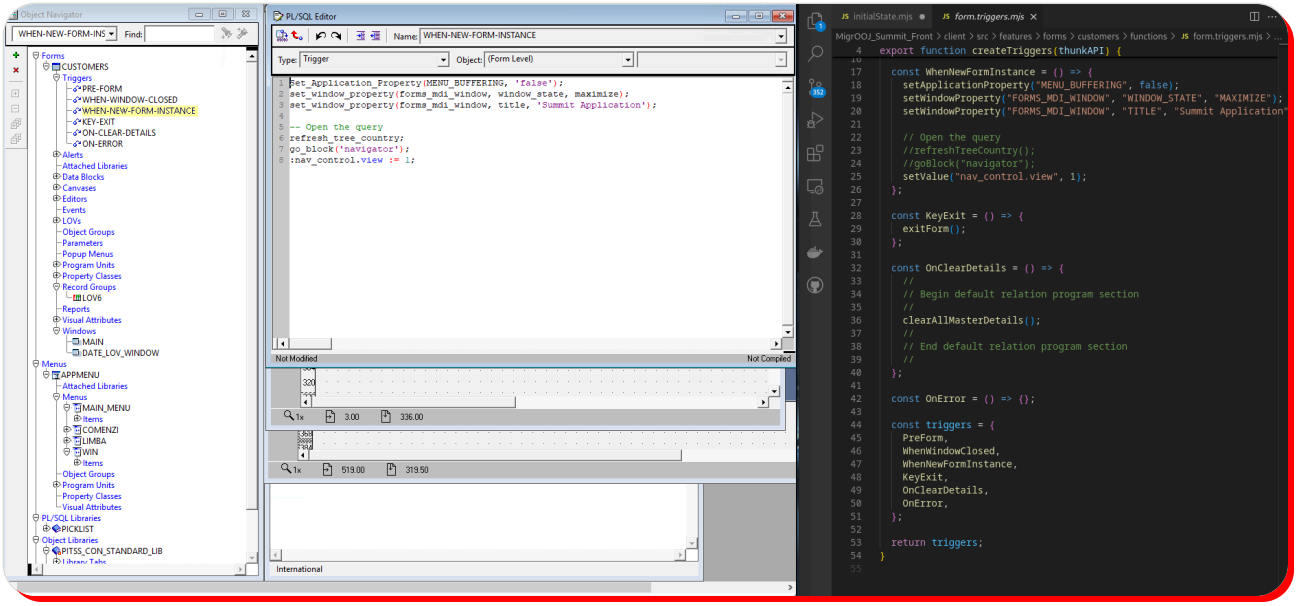
Program unit to TypeScript. The PL/SQL procedure Set_Picklist_Selection in Forms Builder, and its migrated version: same name, same logic, same order of operations. The built-ins become OFCS calls, and the comments travel with the code.



Responsive by default. The migrated Customers form on a desktop, a tablet and a phone, with no additional work.



Blocks and items. The CUSTOMERS block and the properties of one of its items in Forms Builder, and the block in the migrated form's initial state, where every item keeps its properties: insert and update allowed, prompt, navigation.



Triggers. WHEN-NEW-FORM-INSTANCE in the PL/SQL editor, and the migrated form's triggers file, where OFCS fires each TypeScript function at the point where Oracle Forms fired the trigger.

Cloud-ready by default

Once migrated, nothing depends on the Oracle Forms runtime, on WebLogic or on a proprietary IDE. The stack is open, built on mainstream frameworks and languages: TypeScript, React or Angular, Spring Boot and Node.js. Your organization gets:

Full control over the source code | **No vendor lock-in** | **Any modern IDE or toolchain**

DevOps pipelines, CI/CD and cloud-native infrastructure

When comparing modernization options, watch for the solutions that introduce a new dependency: a proprietary runtime, an obscure framework, a vendor-controlled library that ties you to one supplier for the long term. With ORMIT™-OpenJava, your team is autonomous in maintaining and evolving the application.

The applications are lightweight and cloud-agnostic, ready for Oracle Cloud, AWS, Microsoft Azure, Google Cloud or a private Kubernetes cluster, and designed to run in containerized, scalable environments:

- Fully compatible with Docker, Kubernetes and orchestration platforms.
- A REST-native architecture, suited to microservices and distributed systems.
- Backend services hosted in any cloud or hybrid model.
- Elastic scaling, monitoring, and integration with infrastructure-as-code pipelines.

By default, you adopt the cloud strategy you prefer, without reengineering.



Security, privacy and DevOps compatibility

The migrated application ships with role-based access control, secure session handling, and strong encryption for data at rest and in transit, for the standards of data protection, compliance and governance that enterprises and government clients demand. Identity management can stay at the Oracle Database level or integrate with enterprise identity providers such as LDAP, OAuth or SAML-based SSO.

AI assists the migration inside a closed loop, the pattern Gartner calls an AI coding agent harness. Every change passes deterministic checks, an audit against the original Oracle source, and end-to-end tests proven able to fail, before any human review.

The resulting stack is REST-native, with secure, efficient API connectivity to external systems, cloud services and microservices. Because the code is open and maintainable, the system fits into DevOps pipelines. The platform itself is covered by thousands of automated tests, including backend tests against a real Oracle database and one end-to-end suite that runs on both front ends, on every change.

Performance by design

ORMIT™-OpenJava does not only modernize a legacy system; it makes it faster. Each layer is built for responsiveness:

- **Form logic loaded only when the form opens**
- **Records fetched page by page, as in Oracle Forms**
- **The REST API keeps no server session**
- **Each migrated form timed against the original**

From stateful to stateless: scaling made simple

Oracle Forms relies on a stateful, session-based architecture: each user session holds a dedicated operating-system process and a persistent database connection, which limits scalability, consumes server resources and complicates failover and load balancing. ORMIT™-OpenJava replaces it with a stateless REST architecture. The form's state and trigger engine run in the browser, the API keeps no server session, and the backend uses connection pooling: cloud-ready, horizontally scalable, resilient to server interruptions. Lower resource usage, better concurrency, and simpler deployment in clustered or containerized environments. The migrated applications match, and often exceed, the speed and responsiveness of the original Oracle Forms environment.

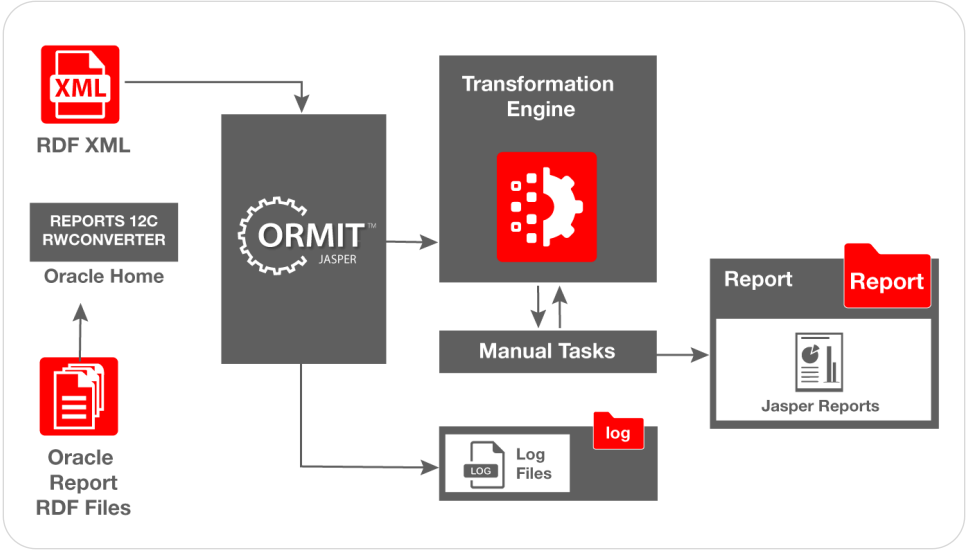
Beyond Forms: Reports, Database, EBS and ISVs

09

Oracle Reports with ORMIT™-Jasper

The architecture of the previous chapter covers the forms. Most Oracle Forms applications also rely on Oracle Reports for operational and regulatory reporting. With Oracle Reports at end of life, the reports need a migration path of their own. ORMIT™-Jasper converts Oracle Reports (RDF) into modern, maintainable JasperReports, on an open-source foundation, and the result is fully supported within the ORMIT™-OpenJava architecture, so Forms and Reports modernize in parallel.

- Full RDF to JasperReports transformation, including layout, parameters and business logic.
- Embedding through the REST API, for direct report execution inside the modernized application.
- Barcodes, multilingual output and scheduling options.
- Output fidelity and functional equivalence with the legacy reports.
- Jasper Community Edition as the foundation, with optional commercial upgrades.



How a report is migrated. The RDF files are exported to XML with Oracle's converter, the transformation engine produces the JasperReports, and the manual tasks that remain are listed in the logs.

The database: Oracle to PostgreSQL with ORMIT™-Cortex



Migrating the forms to React or Angular does not only preserve the logic and the architecture. It opens the door to database independence. With our partner Splendid Data, RENAPS offers ORMIT™-Cortex, a migration platform built to transform Oracle databases into PostgreSQL with precision, automation and minimal disruption.

Automatic PL/SQL to PL/pgSQL conversion

Unlike traditional open-source tools, ORMIT™-Cortex converts complex PL/SQL packages, procedures and triggers into native PL/pgSQL, preserving the logic without heavy manual rewriting.

Comprehensive schema migration

Tables, indexes, views, sequences and user-defined types are analyzed, mapped and restructured into their PostgreSQL equivalents.

Post-migration optimization

Performance tuning and best-practice adjustments, so that the PostgreSQL environment performs at or above Oracle standards.

A licence-free future

An enterprise-grade open-source database, and the end of Oracle licensing costs.

Critically, ORMIT™-Cortex targets pure PostgreSQL, with no hidden proprietary hooks, extensions or platform dependencies. Unlike alternatives such as EnterpriseDB or AWS Aurora, which introduce platform-specific lock-in, it preserves PostgreSQL's full portability: you deploy on Oracle Cloud, AWS, Azure or on premises, and move between clouds, without constraints or reengineering.

For organizations seeking full independence from Oracle technologies, ORMIT™-Cortex completes the journey: Forms, Reports and database modernization in one structured roadmap. Once the forms run in TypeScript, the database becomes a separate decision.

Oracle EBS customizations and ISV products

Many organizations run standalone Oracle Forms applications. A significant number also use custom Oracle Forms screens developed for Oracle E-Business Suite (EBS). These custom modules hold critical workflows, and they remain tied to legacy Forms technology. ORMIT™-OpenJava decouples and modernizes them into web interfaces, preserving the business logic and minimizing disruption to the integrated ERP processes.

Independent software vendors (ISVs) with a Forms-based product deployed at many customers face a larger version of the same problem: modernizing without rewriting their whole product line. ORMIT™-OpenJava lets an ISV migrate once, modernize the architecture, and redeploy to every client with zero vendor lock-in and an improved UI. The result: lower support costs, faster feature development, and a cloud-enabled offering.



An Application
Any Agent Can Operate

10

Whoever owns the forms, a standalone application, EBS customizations or an ISV product, the migrated application is the same, and so is what comes next. A migration to agents does not replace a screen with a chat window. It changes what the organization buys, in two tiers: first an application that any agent can operate under its rules, then processes executed by an agent, with its people in charge of the outcome.

Two tiers

Tier	What the organization gets	Whose agent	How it is proven
1. Agent-ready, to be delivered with the migration, or added to it	An open connector that exposes the migrated form to an agent, its blocks, items, lists of values and buttons, in the form's own order: the Model Context Protocol today, and the same operations as a command line or a plain API when the client's platform prefers it. Every call goes through OFCS and the form's rules. The agent will act under an identity of its own, with the database grants its process needs, and every action will be recorded with the rule it went through.	Any: the organization's own, a platform vendor's, Oracle's, RENAPS's.	Tested, like the rest of the migration: the refusal scene, the identity, the record. No model has to be right.
2. Operated processes, one process at a time	An agent takes over the work of a process under supervision: it proposes, an employee approves, the rules apply. The organization chooses the agent and the model, and sets the pace.	RENAPS's agent, or the one chosen in tier 1.	Measured: precision on ambiguous cases, cost and time per execution, the share of cases that no longer need a person.

Tier one builds directly on the migration. Tier two is what it makes possible, and the rest of this chapter describes it.

Three ways to run a process, one source of rules

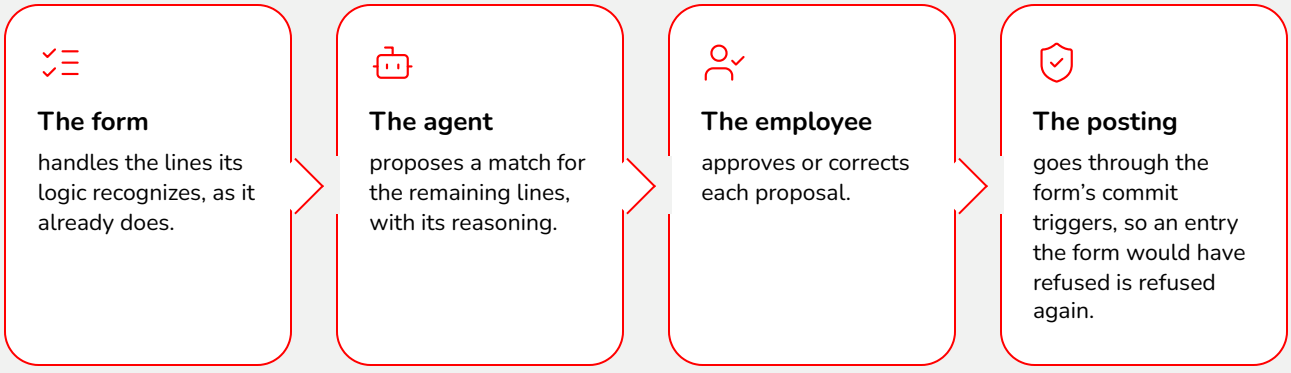
Who does the work	For which cases	Who approves
A program	Paths that require no judgement: the lines a rule can match on its own, the postings whose every field is known.	Nobody, as today. The rules apply.
An AI agent	Ambiguous cases: a payment that almost matches, a claim with a missing document, a request that needs a decision within the rules.	An employee, who sees what the agent proposes and why.
A person, on a screen	Where a screen still serves: high-volume keyboard entry, and the exceptions the agent cannot settle.	As today.

All three go through the same OFCS rules. Moving a case from one column to the next changes who acts, never what is allowed.

A worked example

Take the processing of bank statements, a form found in most finance departments. Today an employee opens the statement, matches each line to an open item, and posts. The migrated application keeps the form's rules: which lines may be matched automatically, which accounts a posting may touch, what makes a reconciliation complete.

With the agent in place, the form's own matching logic handles the lines it recognizes, as it already does. The agent proposes a match for the remaining lines, with its reasoning, and the employee approves or corrects each proposal. The posting goes through the form's commit triggers, so an entry the form would have refused is refused again. The employee's day moves from typing every line to settling the exceptions.

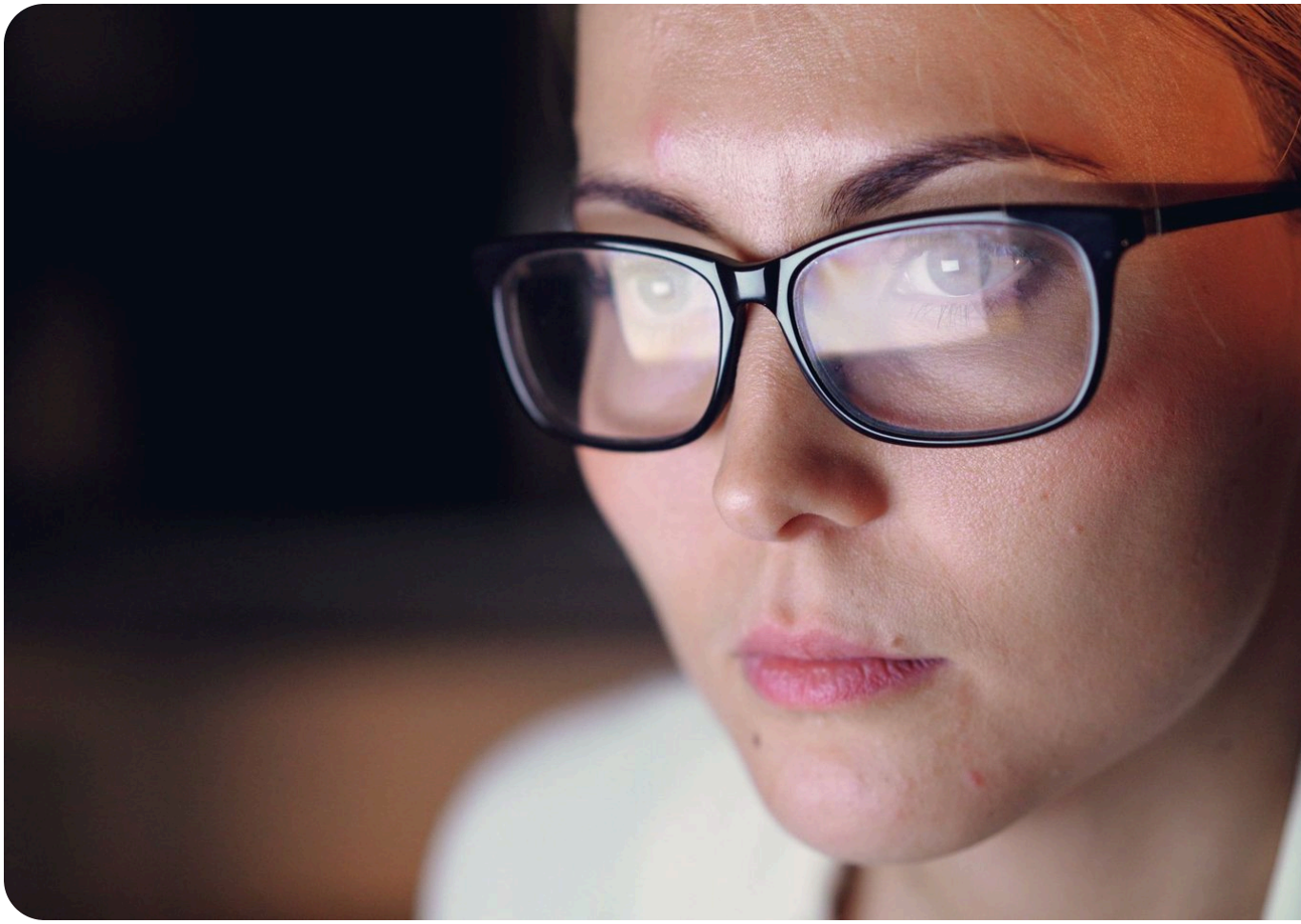


What the rules cannot do

The rules say what the form refuses. They do not say which match is the right one when several are valid, and that is exactly the work in the agent's column. This is why approval is not optional, and why the measure of an agent is its precision on ambiguous cases, not its pass rate on the easy ones. The rules make the agent safe to try. The measurements decide whether it is worth keeping.

What your people do instead

Nobody is asked to trust the agent. They are asked to supervise it, with the same rules they know, and to approve its work before it counts. The institutional knowledge in the team becomes the set of exceptions the agent learns to raise, and the measure of the process becomes the share of cases that no longer need a person.



Safe by construction

The controls that a transactional system requires are not added to the agent afterwards. They come from how the migration is built.

Identity and accountability

By design, the agent has an identity of its own: its own database account, with only the grants its process needs, and never an employee's credentials. Gartner's 2027 trends make this the first rule of AI agent identity; Oracle's roles and grants already enforce it. The agent prepares, an employee approves: the separation of duties the auditors expect, kept as it is. Every action will be logged with the rule it went through, the agent that proposed it and the person who approved it.

Progressive trust

An agent starts in observation: it proposes, and its proposals are compared with the employees' own work. Then each action needs approval. Then only the exceptions do. The organization decides the pace, process by process.

Your data, your region, your choice of model

The model runs where the organization chooses. Through Amazon Bedrock or Google Cloud Vertex AI, prompts and results stay in the client's cloud account and region. On Microsoft Foundry, Claude is available in the United States, with Europe announced; Canada is not yet covered there. Nothing is sent to a consumer AI service. And nothing depends on one model: the migrated rules are the client's code, the connector uses an open protocol, MCP, and the model can be replaced, including by an open model run on the client's own infrastructure. Those exit options are what Gartner's SovereignTech trend asks every organization outside the United States to secure by 2030.

A measured cost

Unlike a screen, each action an agent takes has a cost. RENAPS quotes it only once measured on a real form, per execution, and the client decides whether it pays the AI on its own cloud account or through the subscription.

The refusal test

The decisive test of a safe migration is what happens when the agent tries something the form forbids. The answer has to be the same as for a person: the form refuses, the agent receives the form's message, reports it, and does not work around it.

What this white paper does not claim

Today, the migration, OFCS and its validation server exist and ship with every migration. The connector and the agent offer described in this chapter and in chapter 12 are designed on that foundation, and will be proven on a real form before they are offered. No capability described here depends on the model being right. The first tier, the connector, depends on no model at all, and will be tested like the rest of the migration. The model proposes; the rules decide; a person approves. The rules are necessary, not sufficient: they stop what the form forbids, not a plausible choice the form allows. That is what the approval step and the measured precision are for. Where a rule does not exist in the form, the agent has no more authority than a new employee would, and the process stays with a person until the organization writes the rule.



The context layer Gartner asks for, and where a migrated form already holds it

Gartner's AI context platform, trend 4 for 2027, "unifies scattered corporate knowledge into a machine-readable truth layer to ground autonomous digital workers". Its three components map onto what ORMIT™-OpenJava preserves, and onto what the agent offer adds.

Gartner's component	What it holds, in Gartner's terms	In a migrated application
Semantics	Ontology, metrics, rules, ownership, policies	The rules: every trigger, validation and procedure of the form, as TypeScript that OFCS executes. Not a description of the rules. The rules.
Operational state	Entities, environment conditions, right-time data	The form's blocks, items and lists of values, and the live records in the client's Oracle database, read through the queries the form itself runs.
Provenance	Tracking of data, processes, decisions, actions, outcomes	The action log the agent-ready tier will add: for each action, the rule it went through, what the agent proposed, and the person who approved it.

Gartner's guidance for building that layer is to "separate passive business rules from active, machine-generated decision traces". That separation is the central figure of this white paper. The rules are code that does not change when the agent runs; the agent's reasoning is a trace kept beside them, and it never becomes a rule.

What a migrated form does not provide is the rest of the layer: the ontology Gartner expects, in standards such as RDF and OWL, and the knowledge an organization keeps outside its forms. OFCS feeds a context platform; it does not replace one. It supplies the component that exists only in your forms today, the rules, in the one form an agent can be held to: executable.

Gartner's first trend for 2027, large quantitative models, settles one more question clients ask. It starts from the observation that language models "fail at mathematical reasoning because they guess probabilistically". In a migrated application the agent never computes an amount. The form's procedures compute it, as the Oracle Forms runtime did, and the agent reads the result.

Proof, and How We Compare

11

Proven equivalence

The chapters so far describe the approach. This one is about the evidence. Every migrated form is checked against the original. End-to-end tests are written from the original PL/SQL and proven able to fail. The same suite runs on React and on Angular. The backend is tested against a real Oracle database, every change is audited against the Oracle source, and each form is timed against the original.

Customer spotlight: a large manufacturer

A North American manufacturing leader modernized over 100 Oracle Forms modules with ORMIT™-OpenJava, to a React + Node.js architecture. The project spanned six months and covered mission-critical production and inventory applications. The migration kept the business logic and the users' workflows, and the client's own Forms team, trained during the project, now maintains the application. The company is free of its WebLogic and Forms runtime licences.

Customer spotlight: a utility

A major North American utility entrusted RENAPS with over 500 Oracle Forms modules, more than 750,000 lines of business-critical code. With ORMIT™-Analyzer and ORMIT™-OpenJava, the project reached over 89% automation, preserved the business logic, kept the end users' workflows, and deployed quickly to a fully responsive web environment. The utility modernized its application architecture and reduced its operational and licensing costs significantly.

Migration automation metrics

RENAPS has analyzed hundreds of Oracle Forms and Reports applications across industries, geographies and use cases. From that dataset, one metric stands out: **ORMIT™-OpenJava automates, on average, 88% of the whole migration effort.** Nearly nine transformation steps in ten, logic conversion, UI component mapping, built-in replacement and code structure generation, are handled automatically. The remaining 12% is typically customer-specific business rules, legacy customizations, or modernization choices that need a person's validation. The 99% in the comparison table on page 44 is a different measure: the share of code converted automatically; 88% is the share of the whole project effort, testing included.

This degree of automation translates into:

No conversion drift

A converter applies the same rule to the ten-thousandth trigger as to the first.

Shorter delivery timelines

The automated share is done in hours; the project's time goes to the 12% that needs judgement.

Predictable costs and scope

What the analysis measured is what the migration delivers.

Reliable estimation from day one

The ORMIT™-Analyzer report prices every phase before the project starts.



How ORMIT™-OpenJava compares with the other migration tools

Modernizing Oracle Forms takes more than converting code: it takes a shift toward open, maintainable technologies, and with many tools on the market, it is hard to tell a superficial migration from a real modernization. This section compares the most recognized Oracle Forms migration tools on their technical approach, their architecture and their long-term viability.

Migration paths and target stacks

The tools differ first in what they generate. Re_Forms21 and Morphis-Tech produce Java applications; PITSS.CON migrates only to Oracle APEX; VisionX brings a low-code Java platform; Ispirer generates traditional Java with high automation. ORMIT™-OpenJava targets a fully modern, open-source stack: a TypeScript front end in React or Angular, with Node.js or Spring Boot behind it, which supports REST APIs, stateless deployment, containers and cloud-native scaling.

Vendor lock-in and proprietary code

Lock-in is one of the most critical, and most underestimated, factors. Re_Forms21 and VisionX require proprietary runtimes or frameworks, a long-term dependency on the vendor's ecosystem. Morphis-Tech relies on helper frameworks of its own. Ispirer produces standard Java, but fragmented, which makes it hard to maintain. ORMIT™-OpenJava avoids vendor lock-in entirely: clean, modular code on widely adopted open-source technologies, with no proprietary library, framework or black box after the migration. The customer owns the application.

UI/UX outcomes

Some tools aim at a pixel-perfect replica of the Oracle Forms interface (Re_Forms21), which yields an outdated, unintuitive experience. Others (Morphis, VisionX) deliver modern web components at the price of user retraining. ORMIT™-OpenJava keeps the key user workflows and delivers a fully responsive, web-native UI: modern, intuitive, on mobile devices out of the box, in any modern browser.

Business logic and maintainability

Every vendor claims to preserve the business logic; the extent and the quality vary. PITSS.CON externalizes the logic into the database (PL/SQL), which works and limits the architecture. Re_Forms21 and Morphis parse the code deeply to convert PL/SQL into Java, and the result can be tightly coupled to their framework. ORMIT™-OpenJava preserves nearly all of the Oracle Forms logic as clear, maintainable TypeScript, without sacrificing architectural best practices. Developers work on it with standard tools (VS Code, IntelliJ, Eclipse) and adopt CI/CD pipelines and modern testing frameworks with ease.

Summary: choose modernization, not just migration

Tool	Target stack	Lock-in	UX approach	Technology generation	Logic conversion	Ease of maintenance
VisionX	Java (JVx/Vaadin) + VisionX tooling	High	Proprietary low-code, applet UI	2010s	Limited; no PL/SQL mapping	Low; fragmented, tool-dependent
Re_Forms21	Java + proprietary runtime	High	Like-for-like Forms	2000s–2010s	~99%, proprietary APIs	Low; tied to the framework
Morphis-Tech	Java or .NET (MVC) + libraries	High	Modern web UX	2010s	~80–90%, optimized	Average; standard technology
PITSS.CON	Oracle APEX	High (Oracle)	Redesigned APEX UI	2000s–2010s	70%, in the Oracle DB	High (in APEX)
Ispirer	Java (traditional)	Medium–high	No UI migration; backend only	2010s	Partial; code only, no UI	Low–medium; developers refactor
ORMIT™-OpenJava	React/Angular + Node.js/Spring Boot	None	Modern, responsive (TypeScript)	2020s+	~99%, clean TypeScript	High; open and modular

The "technology generation" column. 2020s+: modern, cloud-native, microservices-compatible stacks (React, Angular, Node.js, Spring Boot). 2010s: still relevant, built on earlier web paradigms (Vaadin, standard Java MVC, basic automation). 2000s–2010s: older tools or runtimes with a less modern architecture (Forms look-alikes, tight vendor lock-in, APEX-like ecosystems).

Many tools automate the migration of Oracle Forms. Of these, only ORMIT™-OpenJava targets an open stack without vendor lock-in, and migrates the rules in the one form an agent can be held to.

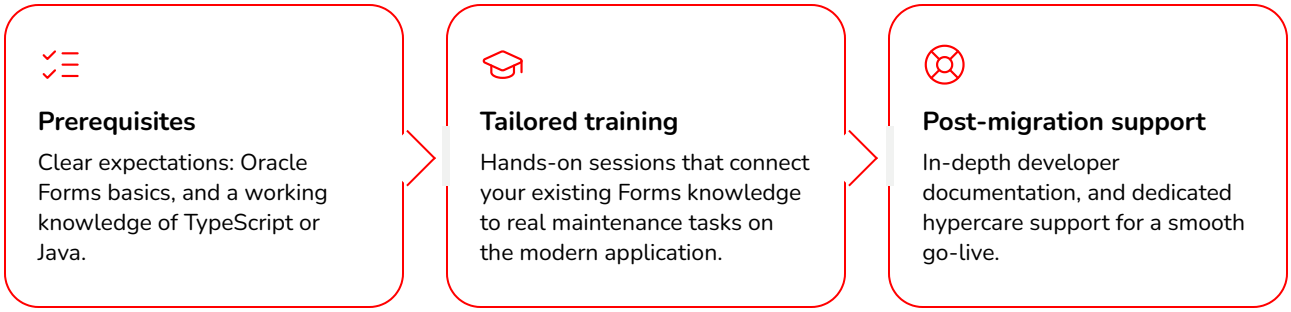
The First Step

12

A migration is a project, not a leap. This chapter covers who does it with you, what it costs against the status quo, and the seven steps from the migration to the first agent, with numbers at the end.

Your team

The success of a modernization hinges on your own team maintaining the new system for the long term. Every ORMIT™-OpenJava project therefore includes an enablement plan that turns your Oracle Forms team into confident maintainers of the new application. We work directly with your developers to bridge the gaps:



Not just the what, but the how

Tools alone do not deliver a transformation; dedicated teams do. The ORMIT™ Squad Model builds agile, multidisciplinary squads fully dedicated to your project, from kickoff to go-live: a tech lead as architecture anchor, front-end and backend developers, QA analysts for automated and regression testing, a product owner who aligns priorities with your business goals, and a scrum master or agile coach who drives velocity and clears blockers. For complex environments, a solution architect joins the squad. The squads work in sprints.

Hypercare and post-go-live support

Every engagement includes a dedicated hypercare phase. For the first three months after go-live, the RENAPS experts who delivered your modernization are on standby with a 15-minute response time SLA, to assist your team, resolve issues and keep the system stable. Your team does not only understand how the system works; it is equipped to evolve it with confidence.

The financial case for migration over maintenance

Chapter 2 put figures on the status quo. In RENAPS's analysis, an organization that keeps a legacy Forms environment pays more every year, for three reasons: Oracle licensing for the Forms runtime and WebLogic, above all when environments scale; the shrinking pool of Forms developers, which drives support rates up; and the custom workarounds it takes to connect Forms to modern tools and APIs. Migrating with ORMIT™-OpenJava reduces the total cost of ownership by removing the Oracle middleware, moving to open standards, and aligning the architecture with mainstream developer talent. You get a modernized stack with no runtime fees or proprietary constraints, easier hiring, onboarding and long-term maintenance, and better development velocity. For most clients, the long-term savings, with the gains in performance, UX and maintainability, make the migration a financial win as well as a strategic necessity.

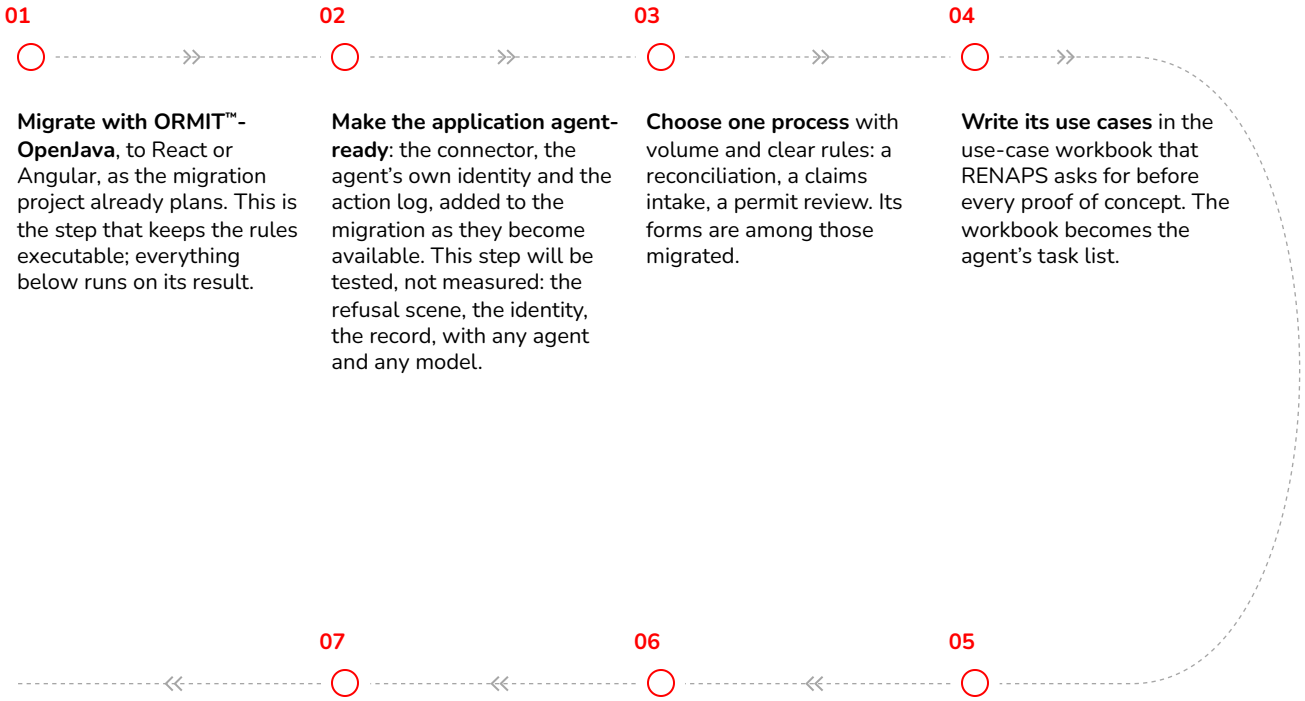
A proof of concept

RENAPS supports proof-of-concept initiatives with either architecture option. Both benefit from the same automation framework, expert onboarding, and a decade of experience in Oracle Forms modernization. The seven steps on the next page start there.



From the migration to the first agent: seven steps

The first step is the migration itself, then the connector, then one process, not a program. It reuses what every RENAPS migration already produces, so it starts from the material the client has, and it ends with numbers.



What RENAPS brings

- ORMIT™-OpenJava and OFCS, which keep the rules executable and already serve callers without a screen, and, in preparation, the connector that will make the migrated application operable by any agent.
- ORMIT™-Analyzer for the inventory, ORMIT™-Jasper for the reports, ORMIT™-Cortex for the database, and a decade of Oracle Forms modernization behind them.
- A migration pipeline built as an AI coding agent harness, with the checks, audits and tests of chapters 8 and 11: nothing reaches a reviewer before the gates pass.
- A dedicated squad, an enablement plan for your team, hypercare after go-live, and full-stack support for the migrated application, before and after the agent.

Ready to see it on your own forms?

Book a live demo, or a proof of concept on one of your processes, with either architecture. Bring the process and its forms, and we will show you how its rules become the guardrails of an agent, and what the first measured step looks like for your organization.

www.renaps.com · info@renaps.com · 1-888-987-3627



Sources

All figures and quotations were checked at the source listed. Translations, where any, are RENAPS's.

1. Oracle, Lifetime Support Policy for Oracle Fusion Middleware: support dates of Forms 12c and 14.1.2. oracle.com
2. Patrick Hamou, The Quest for the Last Oracle Forms Developer. medium.com
3. Ensono, September 2026. Survey of 500 IT leaders: 78% see growing value in legacy systems as AI reshapes modernization. ensono.com
4. IDC, December 2025. Turning Technical Debt into an AI Enabler. idc.com
5. McKinsey & Company, December 2024. AI for IT modernization: Faster, cheaper, better. mckinsey.com
6. Gartner, October 22, 2024. Only 48% of digital initiatives meet or exceed their business outcome targets. gartner.com
7. The Batch, DeepLearning.AI, February 27, 2026. Claude Cowork plugins trigger a SaaS stock selloff. deeplearning.ai
8. Fortune, February 5, 2026. OpenAI launches Frontier, an AI agent platform for enterprises. fortune.com
9. Forrester, February 11, 2026. SaaS as we know it is dead: how to survive the SaaS-pocalypse. forrester.com
10. Gartner, August 26, 2025. 40% of enterprise apps will feature task-specific AI agents by 2026, up from less than 5% in 2025. gartner.com
11. Gartner, July 1, 2026. US$234 billion in enterprise application software spend is at risk from agentic AI. gartner.com
12. Gartner, April 2, 2026. Most enterprises will abandon assistive AI for outcome-focused workflow by 2028. gartner.com
13. Gartner, June 25, 2025. Over 40% of agentic AI projects will be canceled by end of 2027. gartner.com
14. Gartner, October 20, 2026. Top Strategic Technology Trends for 2027, press release and eBook (Gene Alvarez). gartner.com; gartner.com/en/articles/top-technology-trends-2027
15. Marc Benioff, Dreamforce 2026 keynote, September 2026. Transcript. singjupost.com
16. Yuval Noah Harari, Tanner Lecture on Human Values, Oxford, 2026: "AI has hacked the code of human civilization". Transcript. singjupost.com
17. CNBC, September 3, 2026. Software stocks up nearly 40% since the "SaaSpocalypse" bottom. cnbc.com; 24/7 Wall St., October 13, 2026, IGV up 41% from April 13 to October 12. 247wallst.com
18. Oracle, May 2026. Announcing Oracle APEX 26.1. blogs.oracle.com; October 1, 2026, the Modernization Forms to APEX Skill. blogs.oracle.com
19. Anthropic, regional compliance: where Claude stores and processes data by cloud provider. claude.com
20. Oracle, Forms Developer 6i reference manual: the item types, trigger types, built-ins and properties counted in chapter 5.

Oracle, Java, Oracle Forms, Oracle Reports, Oracle E-Business Suite and APEX are registered trademarks of Oracle and/or its affiliates. Salesforce and Dreamforce are trademarks of Salesforce, Inc. Claude is a trademark of Anthropic, PBC. Gartner is a registered trademark of Gartner, Inc. Amazon Bedrock, AWS, Google Cloud Vertex AI, Microsoft Azure and Microsoft Foundry are trademarks of their respective owners. ORMIT™ is a trademark of RENAPS. All other names are trademarks of their respective owners.



Contact Us Today.

www.renaps.com

info@renaps.com

1-888-987-3627