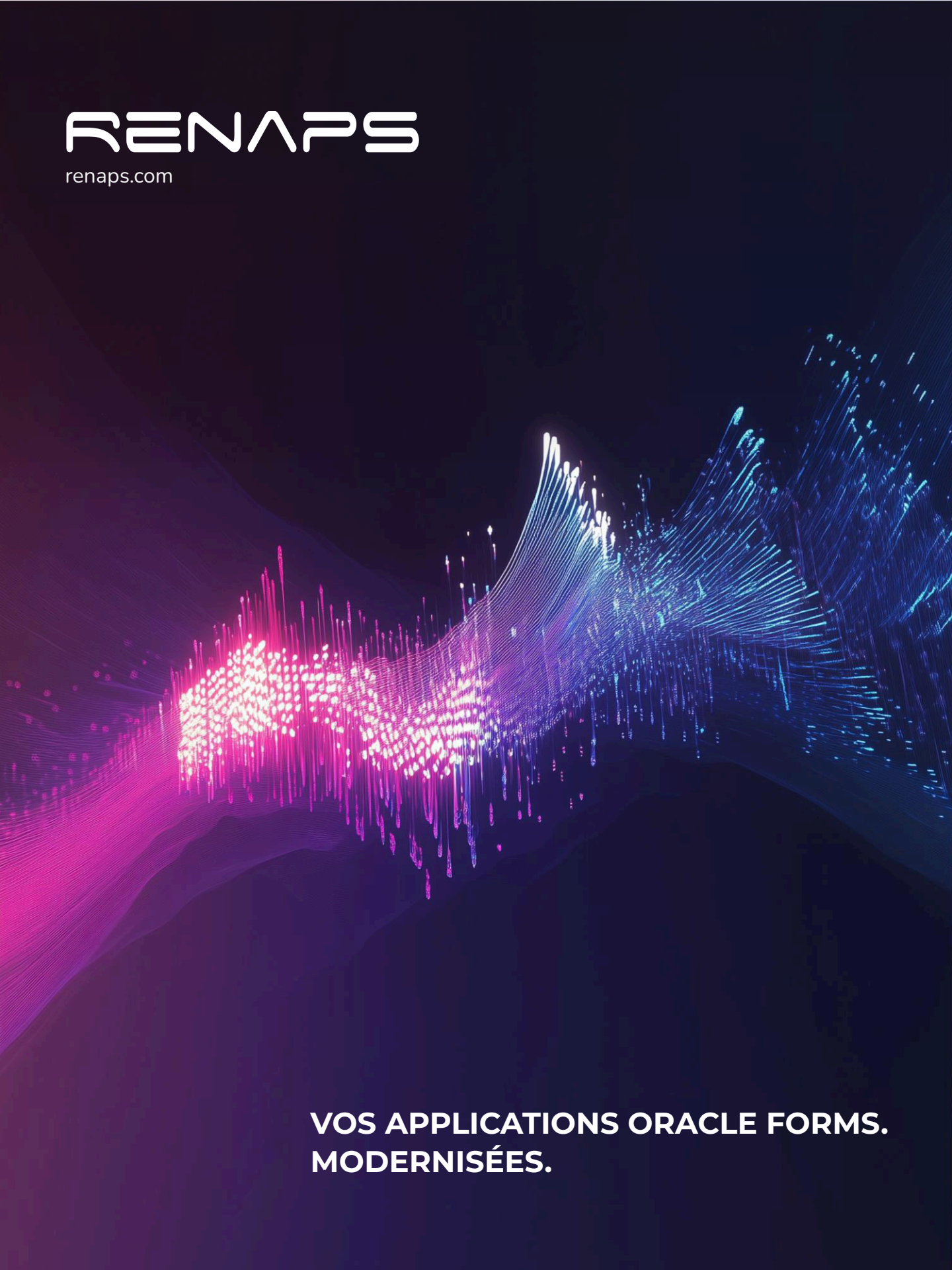
RENAPS

renaps.com



VOS APPLICATIONS ORACLE FORMS. MODERNISÉES.

D’Oracle Forms aux agents IA. Migrer les règles, pas les écrans.

Moderniser Oracle Forms avec ORMIT™-OpenJava :

Pourquoi une migration doit garder vos règles d’affaires exécutables, dans une architecture ouverte qui vous appartient, pour que l’application Web que vous déployez aujourd’hui soit celle qu’un agent IA pourra exploiter en toute sécurité demain, sans seconde migration.

Octobre 2026 · RENAPS

Par

RENAPS

Siège social, États-Unis et international : 2875 NE 191 Street, Suite 500, Aventura, Florida, 33180, USA
Siège social canadien : 101, Marcel-Laurin, bureau 300, Montréal (Québec) H4N 2M3, Canada
1-888-98-RENAPS (1-888-987-3627) · info@renaps.com

Table des matières

1.	**SYNTHÈSE**	**05**
2.	**ORACLE FORMS EN FIN DE PARCOURS**	**06**
	Quatre décennies d'Oracle Forms	06
	Ce que Forms 14c change, et ce qu'il ne change pas	06
	Les difficultés qui poussent à moderniser	07
	Ce que mesurent les analystes	08
	Le coût de l'attente	08
3.	**2026 : L'ANNÉE OÙ LE MARCHÉ A RÉÉVALUÉ LE LOGICIEL**	**09**
	La SaaSpocalypse, puis le rebond	09
	Forrester et Gartner y mettent des dates	09
	Les tendances de Gartner pour 2027 : de l'adoption à la gouvernance	10
	Même les éditeurs le disent	12
	Pourquoi les écrans ne sont plus toute la destination	12
4.	**L'IA DE DEMAIN EST UN BUREAUCRATE**	**13**
	Un agent, pas un outil	13
	Les bureaucraties qui fonctionnent sur Oracle Forms	13
	Pas de règles, pas d'agent	14
5.	**CE QUE CONTIENNENT VOS FORMULAIRES**	**15**
	Où vivent les règles	15
	Pourquoi Oracle Forms est si « collant »	15
6.	**LES VOIES QUI PERDENT LES RÈGLES**	**16**
	Le piège de la réécriture	16
	Pourquoi un assistant greffé ne suffit pas	16
	Pourquoi Oracle APEX est une impasse pour les applications Forms complexes	17
	Un progiciel : le coût caché de repartir à zéro	18
7.	**MIGRER LES RÈGLES, PAS LES ÉCRANS**	**19**
	Ce qu'est ORMIT™-OpenJava	19
	Commencer par un inventaire complet : ORMIT™-Analyzer	20
	OFCS : des règles qui s'exécutent sans écran	20
	Ce que vous obtenez aujourd'hui, et ce que cela permet demain	22
	Ce que vos utilisateurs conservent	22
	Ce qui reste dans la base de données y reste	22
	Ce que cela rend possible	23

Table des matières

8. L'ARCHITECTURE : SOUPLE, OUVERTE, À VOUS 24

Deux options côté serveur, un même résultat 24
Les deux options, côte à côte 25
React ou Angular 25
D'Oracle Forms à la pile moderne : les équivalences technologiques 26
À quoi cela ressemble : Oracle Forms à côté d'ORMIT™-OpenJava 27
Prêt pour l'infonuagique, par défaut 30
Sécurité, confidentialité et compatibilité DevOps 31
La performance dès la conception 31
Du mode avec état au mode sans état : une montée en charge simplifiée 31

9. AU-DELÀ DE FORMS : REPORTS, BASE DE DONNÉES, EBS ET ÉDITEURS 32

Oracle Reports avec ORMIT™-Jasper 32
La base de données : d'Oracle à PostgreSQL avec ORMIT™-Cortex 33
Personnalisations Oracle EBS et produits d'éditeurs de logiciels 34

10. UNE APPLICATION QUE TOUT AGENT PEUT EXPLOITER 35

Deux volets 35
Trois façons d'exécuter un processus, une seule source de règles 36
Un exemple concret 36
Ce que les règles ne peuvent pas faire 37
Ce que font vos équipes à la place 37
Sûr par conception 38
La couche de contexte que demande Gartner, et où un formulaire migré la contient déjà 40

11. LA PREUVE, ET LA COMPARAISON 41

Une équivalence prouvée 41
Cas client : un grand manufacturier 41
Cas client : une société de services publics 41
Indicateurs d'automatisation de la migration 41
Comment ORMIT™-OpenJava se compare aux autres outils de migration 43
En résumé : choisir la modernisation, pas seulement la migration 44

12. LA PREMIÈRE ÉTAPE 45

Votre équipe 45
Pas seulement le quoi, mais le comment 45
Accompagnement intensif et soutien après la mise en production 46
L'argument financier : migrer plutôt que maintenir 46
Une preuve de concept 46
De la migration au premier agent : sept étapes 47
Ce qu'apporte RENAPS 48

SOURCES 49

Synthèse

01

Pendant vingt ans, les organisations qui exploitent Oracle Forms se sont demandé vers *quelle pile Web* migrer. En 2026, la question a changé. Oracle Forms 12c quitte le Premier Support en décembre 2026, les développeurs qui ont écrit les formulaires partent à la retraite, et le marché du logiciel, les analystes et les plus grands éditeurs s'entendent désormais sur la direction que prennent les applications d'entreprise : le travail passe aux agents IA, et l'écran devient l'exception.

C'est une bonne nouvelle pour ces organisations, en raison de ce que contiennent leurs formulaires. Derrière chaque écran se trouvent des centaines de déclencheurs, de validations et de procédures : les règles qui déterminent ce qu'est une transaction valide, qui peut modifier quoi, et quand une écriture est refusée. Ces règles sont exactement ce dont un agent IA a besoin pour mériter la confiance dans un système transactionnel, et ce qu'une réécriture, manuelle, par IA ou vers un progiciel, perd en chemin.

ORMIT™-OpenJava de RENAPS migre les règles avec les écrans. Cette plateforme hautement automatisée transforme les applications Oracle Forms en applications Web modernes : React ou Angular dans le navigateur, Java + Node.js ou Node.js seul côté serveur, la logique des formulaires en TypeScript, et vos paquetages de base de données là où ils sont. En son cœur, OFCS (Oracle Forms Compatibility Service) conserve chaque déclencheur, validation et procédure sous forme de code exécutable, qui s'exécute déjà sans écran. La migration que vous déployez aujourd'hui est donc celle à laquelle on peut ajouter un agent. Vient d'abord une application que tout agent peut exploiter, le vôtre, celui d'un éditeur ou celui de RENAPS, sous une identité propre et devant les mêmes règles qu'un employé. Puis, processus par processus, un agent prend le travail en charge, et vos équipes passent de la saisie à la supervision.

Ce livre blanc montre

- Pourquoi la modernisation ne peut plus attendre : dates de support d'Oracle, talents qui se raréfient, coût du statu quo, chiffres de Gartner, d'IDC, de McKinsey et d'Ensono.
- Pourquoi 2026 a mis fin à l'idée que l'application Web est la destination finale, et pourquoi elle reste la première étape.
- À quoi ressemblera l'IA d'entreprise de demain, et pourquoi vos règles sont le contrôle qui manque à la plupart des projets d'IA agentique.
- Comment ORMIT™-OpenJava migre ces règles : l'analyse, OFCS, l'architecture, Oracle Reports et la base de données.
- Comment l'application migrée devient exploitable par tout agent, en toute sécurité, et comment lui confier un processus, vos équipes approuvant son travail.
- La preuve : équivalence testée, résultats clients, comparaison avec les autres outils, rentabilité.
- Une première étape assez petite pour être décidée ce trimestre.

Oracle Forms
en fin de parcours

02

Quatre décennies d'Oracle Forms

Oracle Forms a évolué pendant plus de quatre décennies, des écrans en mode caractère au déploiement Web. Malgré des mises à jour continues, son architecture reste ancrée dans le modèle client-serveur des années 1990 : avec état, monolithique, liée à Oracle WebLogic. C'est ce qui l'empêche de répondre aux exigences actuelles d'infonuagique, de mobilité et d'intégration, et c'est ce qu'une modernisation fait disparaître.

Historique des versions d'Oracle Forms

- **1979 IAF (Interactive Application Facility)**
  Première version. En mode caractère, avec IAG comme compilateur et IAP comme interpréteur d'exécution.
- **1982 FastForms**
  Lancé avec la version 4 d'Oracle Database; ajoute des outils de génération et d'édition de formulaires.
- **1985 SQL*Forms 2.0**
  En mode caractère; introduit l'éditeur d'écran et un langage de déclencheurs rudimentaire.
- **1986 SQL*Forms 2.3**
  Capacités procédurales accrues; la fondation d'Oracle Financials.
- **1988 Oracle Forms 3.0**
  Premier à prendre en charge PL/SQL; EDI amélioré; mode caractère avec fenêtres graphiques.
- **1992 Oracle Forms 4.0**
  Première vraie version graphique; nouvel EDI avec des éditeurs propres à chaque objet.
- **1993 Oracle Forms 4.5**
  Refonte majeure de l'EDI : navigateur d'objets et feuilles de propriétés, dans l'esprit de Visual Basic.
- **1995 Oracle Forms 5.0**
  Accompagne la version 7 d'Oracle Database; interface graphique enrichie.
- **1998 Oracle Forms 6.0 / 6i**
  Introduit Forms Server pour le déploiement Web; conserve le client-serveur.
- **2001 Oracle Forms 9i**
  Déploiement Web seulement; l'exécution client-serveur disparaît; intégration Java plus poussée.
- **2003 Oracle Forms 10g**
  Forms 9.0.4 renommé, aligné sur Oracle Database 10g; déploiement Web amélioré.
- **2009 Oracle Forms 11g**
  Événements externes, intégration JavaScript et outils de suivi des performances.
- **2015 Oracle Forms 12c**
  Prise en charge de Java Web Start, intégration REST améliorée, composants d'interface améliorés.
- **2024 Oracle Forms 14.1.2**
  Dernière version : une refonte visuelle, et la prise en charge native des sources de données REST.

Ce que Forms 14c change, et ce qu'il ne change pas

Forms 14c apporte des améliorations modestes. La plus notable est la prise en charge native des sources de données REST : un formulaire peut désormais exposer ou consommer une API sans code sur mesure, ce qui aide l'interopérabilité et reste loin d'une modernisation. Le reste tient aux navigateurs et à de petits ajustements de l'interface. Le noyau reste monolithique, avec état et lié à WebLogic, sans conception mobile, sans microservices et sans place dans un pipeline DevOps. Forms 14c prolonge la vie d'un système hérité. Il ne lève pas les limites qui freinent l'innovation, la croissance et le recrutement.

Les difficultés qui poussent à moderniser

Dans tous les secteurs, les responsables des TI subissent une pression pour moderniser des systèmes Oracle Forms vieillissants. Ces plateformes ont rendu des décennies de bons services, et elles freinent désormais tout ce que le reste de l'organisation entreprend. Quatre difficultés reviennent chaque fois.

Un bassin de talents qui se vide

L'expertise nécessaire pour maintenir Oracle Forms disparaît. Comme le montre l'article *The Quest for the Last Oracle Forms Developer*, la plupart des développeurs Forms approchent de la retraite, et moins d'une organisation sur dix dit compter un développeur actif de moins de 45 ans. Les écoles n'enseignent plus Forms, et les jeunes développeurs choisissent les technologies en demande. Le risque porte sur la continuité : le soutien, la maintenance et le savoir institutionnel.

Une architecture dépassée, une innovation limitée

Forms est étroitement lié à la base de données Oracle et à un intergiciel propriétaire. Les sources de données REST natives ne sont arrivées qu'avec la version 14c et restent limitées; rien n'est prévu pour les microservices ni pour les interfaces mobiles ou adaptatives. Les TI d'aujourd'hui reposent sur les API, les microservices et les pipelines DevOps, et relier Forms à ce monde exige des contournements qui allongent chaque livraison et alourdissent la dette technique.

Un coût de possession en hausse

Licences de l'intergiciel Oracle, infrastructure vieillissante et talents spécialisés font grimper le coût total de possession, surtout face aux solutions infonuagiques.

Une expérience numérique décevante

Les utilisateurs attendent des interfaces adaptatives et intuitives. Les applications Forms y répondent rarement, au prix de la satisfaction, de la productivité et de l'adoption, surtout chez les employés natifs du numérique et les utilisateurs externes.

Ce que mesurent les analystes

61 %

Organisations qui ont suspendu, reporté, réduit ou abandonné un projet de modernisation des TI au cours des 24 derniers mois (Ensono, 2026)

20–40 %

Part du temps de développement que peut absorber une dette technique non maîtrisée (IDC, 2025)

40–50 %

Accélération des projets de modernisation grâce à l'IA générative, et 40 % de coûts en moins liés à la dette technologique (McKinsey, 2024)

48 %

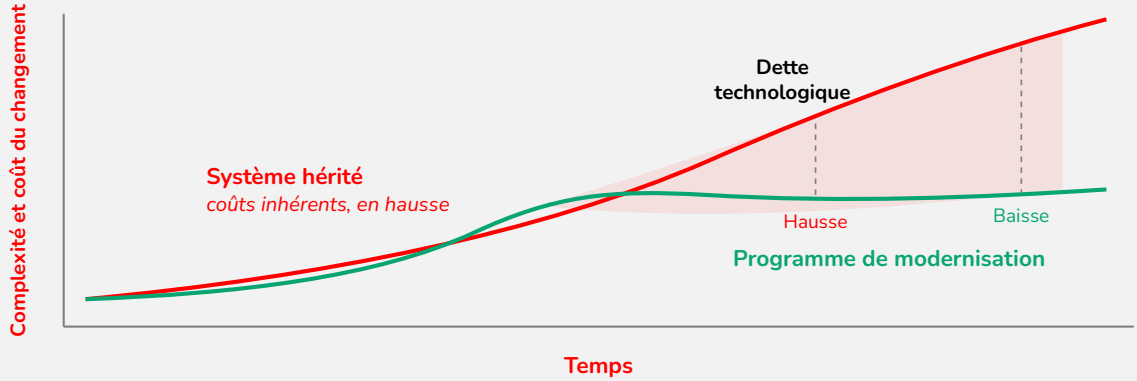
Initiatives numériques d'entreprise qui atteignent ou dépassent leurs objectifs d'affaires (sondage Gartner auprès des responsables des TI, 2024)

Lus ensemble, ces résultats disent une seule chose : retarder la modernisation ne coûte pas seulement plus cher, cela limite la capacité de l'organisation à innover, à soutenir la concurrence et à croître, et c'est dans l'écart entre le constat et l'action que la dette technique s'accumule.

Le coût de l'attente

Ce qui était autrefois une plateforme stable est devenu un obstacle. Le coût du changement sur un système hérité augmente chaque année, à mesure que ceux qui le connaissent partent et que les systèmes autour de lui évoluent. Un programme de modernisation coûte plus cher au départ, puis moins chaque année. L'écart entre les deux courbes est la dette technologique que l'organisation choisit de porter. **La modernisation n'est plus facultative.** Ce livre blanc montre comment la faire une seule fois : vers une application qui lève les limites d'aujourd'hui et qui est prête pour la suite, que décrit le prochain chapitre.

Le coût croissant du statu quo



2026 : l'année où le marché a réévalué le logiciel

03

La SaaSpocalypse, puis le rebond

Le 12 janvier 2026, Anthropic a lancé Claude Cowork, un agent qui accomplit du travail professionnel avec les fichiers et les outils d'affaires d'un utilisateur. Le 30 janvier, il y a ajouté onze modules d'extension à code source ouvert, un par fonction de bureau : finance, revue juridique, ventes, service à la clientèle et d'autres. Le 5 février, OpenAI a lancé Frontier, une plateforme pour bâtir et exécuter des agents qui font fonctionner d'autres logiciels d'entreprise.

Les investisseurs ont lu le signal sur-le-champ. Entre le 12 janvier et le 23 février, l'indice S&P Software & Services a perdu le quart de sa valeur. La séance qui a suivi la sortie des modules a effacé à elle seule plus de 285 milliards de dollars américains de capitalisation boursière. Un stratège de Jefferies a donné son nom à l'épisode : la SaaSpocalypse.

−25 %

Indice S&P Software & Services, du 12 janvier au 23 février 2026

285 G$

Valeur boursière effacée en une séance après la sortie des modules d'agents

+41 %

Titres du logiciel (IGV), du creux du 13 avril au 12 octobre 2026, à leurs sommets de l'année

La crainte était précise. Le logiciel d'entreprise se vend par utilisateur, et un agent qui exécute un flux de travail n'occupe aucune licence. Quand la logique passe à l'agent, l'application derrière lui devient une base de données dotée d'une interface que personne n'ouvre.

Puis le marché a changé d'avis, et c'est ce qui compte le plus. En octobre 2026, l'indice du logiciel avait regagné tout ce qu'il avait perdu : en hausse de 41 % depuis son creux d'avril, à de nouveaux sommets pour l'année, stable sur douze mois. La reprise n'a pas été menée par les entreprises qui niaient le changement, mais par celles qui avaient intégré des agents au cœur de leurs produits. Le verdict n'était pas que le logiciel est mort. C'était que le logiciel sans agents est réévalué à la baisse, et que le logiciel qui les intègre garde sa valeur.

Forrester et Gartner y mettent des dates

Forrester a répondu à la dégringolade dans la semaine, sans en contester la direction. Quelle que soit la manière de déployer des agents, écrivait Forrester, « cela érodera le périmètre de vos applications SaaS d'entreprise actuelles », et la tarification par utilisateur cédera la place à la consommation et au résultat.

Gartner y met des dates :

D'ici la fin de 2026, 40 % des applications d'entreprise intégreront des agents IA spécialisés par tâche, contre moins de 5 % en 2025.

D'ici 2028, le tiers des expériences utilisateur passeront des applications natives aux interfaces d'agents.

D'ici 2030, 234 milliards de dollars américains de dépenses en applications d'entreprise sont menacés par l'IA agentique, et l'interface utilisateur cesse d'être un facteur de différenciation.

D'ici 2028, plus de la moitié des entreprises cesseront de payer pour une IA qui assiste les employés, au profit de plateformes qui livrent le résultat d'un processus. Les éditeurs qui se contentent de greffer l'IA à des applications existantes s'exposent à une compression de leurs marges pouvant atteindre 80 % d'ici 2030.

Les tendances de Gartner pour 2027 : de l'adoption à la gouvernance

Le 20 octobre 2026, Gartner a publié ses 10 grandes tendances technologiques stratégiques pour 2027. Les agents n'y figurent plus comme une tendance. Ils sont la plateforme que la liste suppose déjà en place dans toute organisation, et la question est devenue de savoir ce qui les gouverne. Selon Gartner, la conversation en 2027 « passe de l'adoption à la gouvernance » : les TI traditionnelles reposaient sur la rareté du logiciel, 2027 en apporte l'abondance, et les responsables des TI « doivent passer de la gestion des actifs technologiques à la gouvernance du comportement des machines ».

+80 %

Précision de l'IA agentique pour les organisations qui priorisent la sémantique dans des données prêtes pour l'IA, d'ici 2027

25 %

Des atteintes à la sécurité en entreprise découleront d'un usage abusif d'agents IA, d'ici 2028

80 %

Des nouvelles applications seront rendues jetables et faciles à remplacer par les organisations de pointe, d'ici 2030

Trois des dix tendances décrivent ce qu'une migration d'Oracle Forms doit préserver :

Plateformes de contexte d'IA.

Les connaissances dont un agent a besoin « pour interpréter les intrants, prendre des décisions et agir en accord avec les objectifs d'affaires ». Gartner donne à cette couche trois composantes : la sémantique, qui inclut les règles d'affaires et les politiques; l'état opérationnel, les données en temps réel; et la provenance, la trace des décisions, des actions et des résultats. Sans elle, les organisations s'exposent à « des agents aux privilèges excessifs, des manquements à la conformité, une faible confiance et une consommation excessive de jetons ».

Identité des agents IA.

Les agents « exposent des vulnérabilités massives dans les systèmes hérités incapables d'accorder des accès juste-à-temps », et les organisations qui ne régleront pas la question verront plus de la moitié de leurs initiatives d'IA s'arrêter. Première consigne de Gartner : un agent ne partage jamais les identifiants d'une personne. Il reçoit une identité propre, avec des accès régis par des politiques.

Applications jetables.

La première action que cite Gartner consiste à « déterminer quelles capacités essentielles exigent un logiciel permanent et lesquelles peuvent être servies par des solutions temporaires, à la demande ». Le portefeuille d'applications devient « un système dynamique qui se renouvelle continuellement à mesure que les actifs redondants sont retirés ».

Lues ensemble, elles disent où se trouve la valeur d'une application Oracle Forms. Les écrans en sont la partie remplaçable. Les règles en sont le noyau permanent, et elles sont la seule composante de la couche de contexte de Gartner qu'aucun éditeur de plateforme ne peut écrire pour une organisation, parce qu'elles n'existent que dans ses formulaires. Une migration qui les garde exécutables, sous une identité que la base de données contrôle, est la part du programme 2027 qu'une organisation qui exploite Oracle Forms peut livrer à partir de ce qu'elle possède déjà. Le chapitre 10 détaille la correspondance.

Même les éditeurs le disent

En septembre 2026, Salesforce a ouvert sa conférence Dreamforce en abordant la dégringolade de front. Son PDG, Marc Benioff, a écarté les discours sur la fin du logiciel, puis a concédé le point qui compte :

> ***« L'apocalypse du SaaS ne portait pas sur la fin du logiciel, mais elle pourrait bien porter sur la fin du logiciel qui fait faire tout le travail aux humains. »***
>
> *Marc Benioff, PDG de Salesforce, discours d'ouverture de Dreamforce, septembre 2026*

Il l'a dit en lançant une interface d'agent pour les propres applications de Salesforce, bâtie avec Claude d'Anthropic. L'éditeur qui raillait l'apocalypse y a répondu en intégrant des agents. Un logiciel qui fait faire tout le travail aux humains, c'est la description exacte d'un écran de saisie. C'est aussi la description exacte de ce que livre une réécriture classique d'Oracle Forms : le même écran, dans un navigateur, sans les règles qui étaient derrière, perdues en chemin.

Pourquoi les écrans ne sont plus toute la destination

Rien de cela ne signifie que les écrans disparaissent. La saisie au clavier à haut volume reste plus rapide sur un écran, et la plupart des tâches en garderont un pendant des années. Cela ne signifie pas non plus qu'il faille attendre. Selon les dates de support publiées par Oracle, Forms 12c est couvert par le Premier Support jusqu'en décembre 2026, et Forms 14.1.2 jusqu'en 2030 : la migration a une échéance dans tous les cas, et elle doit avoir lieu. Cela signifie que la migration doit être choisie pour ce qu'elle rend possible ensuite. Un projet de migration dure d'un à trois ans; les chiffres ci-dessus décrivent le même horizon. Une migration qui livre des écrans et perd les règles finit là où elle a commencé. Une migration qui livre des écrans et garde les règles exécutables est la plateforme sur laquelle les agents fonctionneront, sans seconde migration.

La question pour une organisation qui exploite Oracle Forms en 2026

Non plus seulement « quelle technologie Web? », mais « quand un agent exécute ce processus sur l'application migrée, qu'est-ce qui l'empêche de mal faire? » Le prochain chapitre porte sur la nature de cet agent. Le suivant porte sur cette question, et sur la raison pour laquelle vos formulaires y répondent déjà, à condition que la migration garde la réponse.

L’IA de demain est un bureaucrate

04

Un agent, pas un outil

L’historien Yuval Noah Harari passe depuis deux ans à corriger l’image que la plupart des gens se font de l’intelligence artificielle. Dans sa Tanner Lecture de 2026 à Oxford, il a résumé la distinction en une phrase :

> ***« Ce n’est pas un outil entre nos mains. C’est un agent qui a ses propres mains. »***
>
> *Yuval Noah Harari, Tanner Lecture on Human Values, Oxford, 2026*

Et il a dit où cet agent vivra. Pas dans la rue, ni dans un corps de robot :

> ***« Comme les poissons vivent dans les océans et les singes dans les forêts, les IA vivent dans les bureaucraties. »***
>
> ***« Dans les années qui viennent, des millions de bureaucrates IA prendront de plus en plus le contrôle des bureaucraties du monde. Des banquiers IA décideront s’ils vous accordent un prêt. »***
>
> *Yuval Noah Harari, même conférence*

Les bureaucraties qui fonctionnent sur Oracle Forms

Harari le dit en guise d’avertissement. Pour une organisation qui fonctionne sur Oracle Forms, c’est aussi une prévision, et elle soulève une question : qui écrit les règles que suivra le bureaucrate IA?

Une bureaucratie, au sens de Harari, est tout système qui permet à des inconnus de se faire confiance à grande échelle : une banque, un assureur, la facturation d’un hôpital, les permis d’un ministère, une paie. Ces systèmes sont faits de dossiers, de règles et d’approbations. Depuis trois décennies, une grande partie d’entre eux sont bâtis sur Oracle Forms, précisément parce que Forms rendait les règles et les approbations faciles à exprimer : un déclencheur sur chaque événement, une validation sur chaque champ, une procédure derrière chaque bouton.

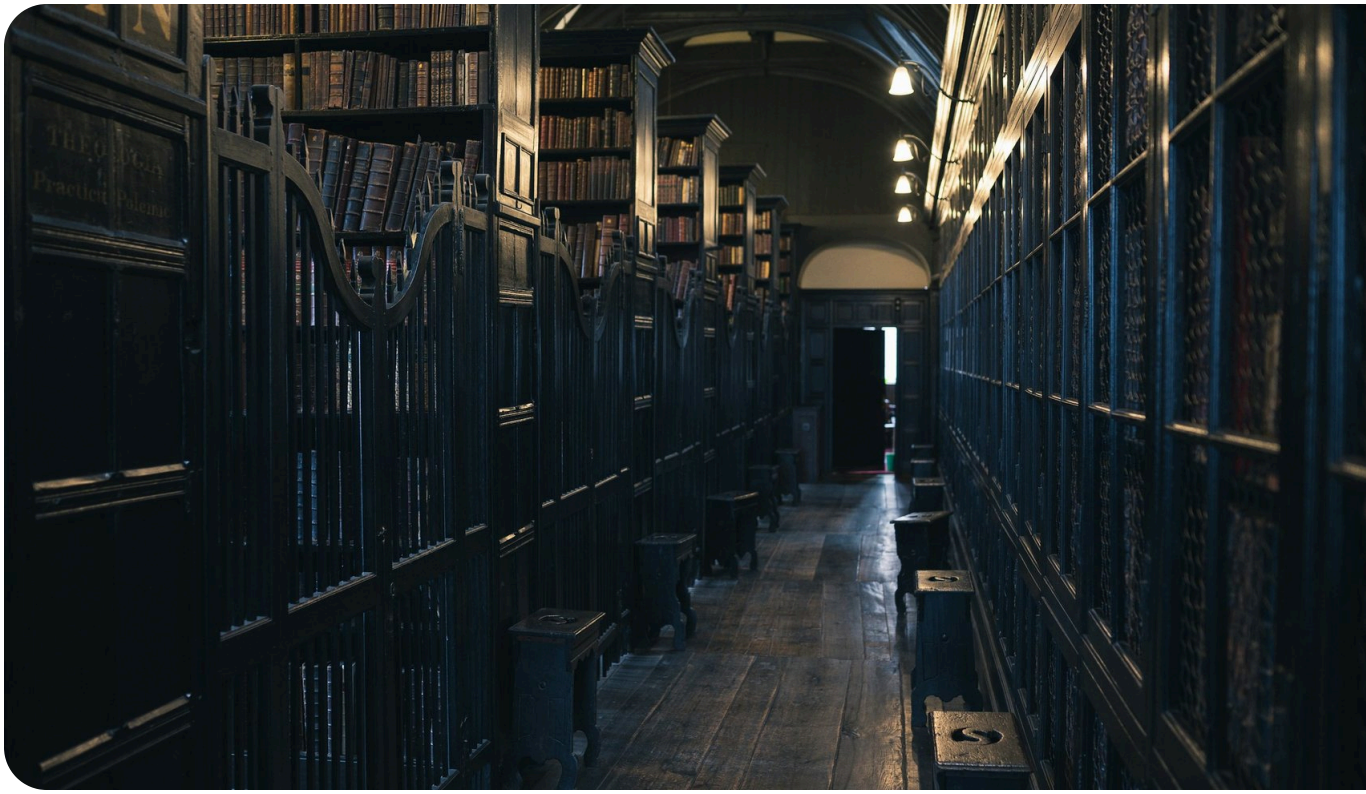
Le bureaucrate IA que décrit Harari n’arrivera donc pas dans un nouveau système. Il arrivera dans le vôtre. Le prêt sur lequel il statue se trouve dans un bloc d’un formulaire. La réclamation qu’il règle, la facture qu’il comptabilise, le congé qu’il approuve sont des enregistrements qui passent aujourd’hui par un déclencheur WHEN-VALIDATE-RECORD avant d’atteindre la base de données.

Pas de règles, pas d’agent

Gartner a aussi mesuré comment ces projets échouent. Plus de 40 % des projets d’IA agentique seront annulés d’ici la fin de 2027, pour trois raisons : des coûts qui s’envolent, une valeur d’affaires floue et des contrôles des risques insuffisants. En octobre 2026, ses tendances pour 2027 ont nommé le contrôle manquant : une couche de contexte dont la sémantique est faite des règles et des politiques de l’organisation, sans laquelle les agents ont trop de privilèges et n’inspirent pas confiance. Seulement 22 % des entreprises tirent aujourd’hui une valeur importante de l’IA générative, alors que 42 % prévoient de déployer des agents d’ici la fin de 2026. Ce contrôle est celui qu’un système transactionnel ne peut pas contourner. Aucune organisation ne laissera un agent approuver des paiements, régler des réclamations ou passer des écritures de journal sans garde-fous qu’elle peut auditer.

La conclusion

Un bureaucrate IA n’est accepté que s’il obéit aux règles. Ces règles doivent être du code, appliqué avant la base de données, et le même pour l’agent que pour l’employé. Les organisations qui exploitent Oracle Forms les ont déjà. La seule question est de savoir si la migration les garde.



Ce que contiennent vos formulaires

05

Où vivent les règles

Demandez à une équipe Oracle Forms où se trouvent les règles d'affaires : elle vous montrera les paquetages de la base de données. La moitié de la réponse est là. L'autre moitié est dans les formulaires : des déclencheurs WHEN-VALIDATE-ITEM qui refusent une date hors de la période ouverte, des déclencheurs PRE-INSERT qui dérivent une valeur que l'utilisateur ne saisit jamais, des déclencheurs KEY-COMMIT qui appellent un paquetage dans un ordre précis, des propriétés qui rendent un élément non modifiable une fois l'enregistrement comptabilisé. Une grande application en contient des milliers, écrits sur vingt ans par des gens aujourd'hui à la retraite. Personne n'en a la liste complète.

Pourquoi Oracle Forms est si « collant »

Cette richesse explique pourquoi Forms est toujours là. Son manuel de référence décrit un environnement événementiel où la logique côté client et côté serveur se mêlent librement, souvent selon des modèles hybrides :

15
types d'éléments

92
types de déclencheurs, un par événement

282
fonctions intégrées, appelables par tout déclencheur

476
propriétés, sur chaque objet

Cette expressivité, que nous appelons la *diversité du code*, a permis à Forms de gérer des applications avec état, proches du poste de travail, comme les formulaires à blocs multiples, que les plateformes Web déclaratives peinent à reproduire. C'est aussi ce qui rend le départ difficile. Migrer, ce n'est pas seulement changer de plateforme : logique, interactions et architecture doivent toutes être transposées, chaque déclencheur s'exécutant là où Oracle Forms l'aurait déclenché. La cible doit être aussi ouverte et aussi expressive que la source. TypeScript avec React ou Angular l'est. Une plateforme déclarative ne l'est pas.

La question à laquelle une migration doit répondre

Non pas « quels écrans? », mais « quelles règles, et où s'exécutent-elles ensuite? ». Les voies du prochain chapitre échouent toutes sur sa seconde moitié. Le chapitre 7 montre comment ORMIT™-OpenJava y répond, en commençant par l'inventaire que personne ne possède.

Les voies qui perdent les règles

06

Quatre voies s'offrent à une organisation qui exploite Oracle Forms, en dehors d'une migration qui garde les règles : une réécriture, manuelle ou par IA; un assistant à côté des anciens écrans; Oracle APEX; un progiciel. Chacune a ses mérites. Chacune perd les règles.

Le piège de la réécriture

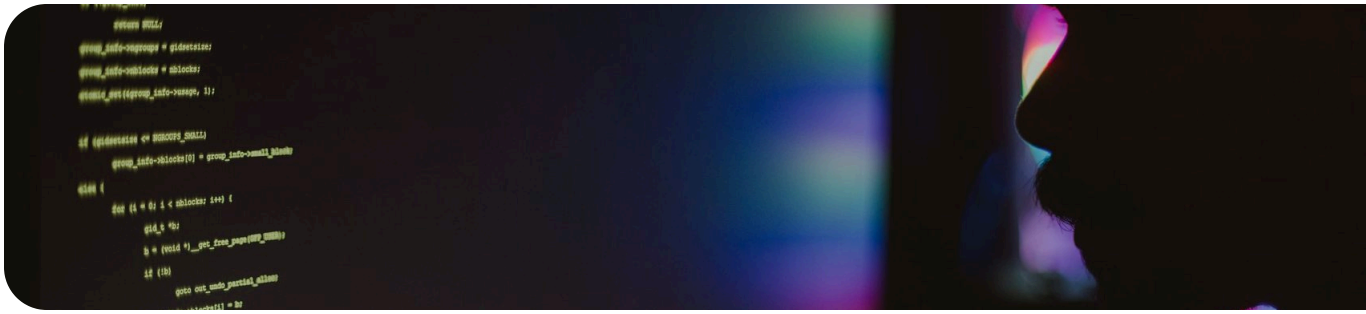
Toute réécriture, manuelle ou par une IA qui génère du code, part d'une spécification. La spécification est rédigée à partir des écrans et des flux de travail évidents. Les règles sont découvertes plus tard, un incident à la fois : une écriture acceptée dans une période fermée, un rabais appliqué sans l'approbation qu'exigeait l'ancien formulaire, un enregistrement sauvegardé avec un champ que l'ancien déclencheur aurait dérivé.

Une règle perdue, et ce qu'elle coûte avec un agent

Un formulaire porte un déclencheur de validation qui refuse une date de comptabilisation hors de la période comptable ouverte. La réécriture reproduit l'écran, la liste de valeurs et la sauvegarde, mais pas le déclencheur, parce que personne ne l'a demandé. Avec un opérateur humain, l'erreur apparaît à la fin du mois et elle est corrigée. Avec un agent qui passe des centaines d'écritures par jour, la même lacune produit des centaines d'écritures dans une période fermée avant que quiconque y jette un œil. La règle dont l'agent avait le plus besoin est celle que la réécriture a laissé tomber.

Pourquoi un assistant greffé ne suffit pas

L'autre raccourci consiste à garder les écrans et à ajouter un assistant IA à côté. Il aide l'opérateur à saisir plus vite. Il ne change pas qui fait le travail, et c'est la stratégie dont Gartner prévoit qu'elle comprimera les marges des éditeurs jusqu'à 80 % d'ici 2030, parce que l'acheteur cesse de payer pour de l'assistance et commence à payer pour des résultats.



Pourquoi Oracle APEX est une impasse pour les applications Forms complexes

APEX semble la voie naturelle, à faible code, pour une organisation Oracle. Sous le capot, il présente des limites qui le rendent mal adapté aux applications Oracle Forms complexes, pour trois raisons.

Incompatibilité fonctionnelle

APEX est une plateforme déclarative à l'extensibilité limitée. Il lui manque la profondeur procédurale et le modèle événementiel nécessaires pour reproduire des applications avec état fondées sur les déclencheurs : les fonctions intégrées, l'intégration OLE et la plupart des déclencheurs d'événements ne se transposent tout simplement pas. Migrer vers APEX signifie une refonte complète de l'interface, des déclencheurs réécrits à la main, des compromis fonctionnels et une refonte de la logique d'affaires, soit exactement le risque qu'une modernisation est censée écarter. APEX est une plateforme déclarative à l'extensibilité limitée. Il lui manque la profondeur procédurale et le modèle événementiel nécessaires pour reproduire les applications Oracle Forms complexes, avec état et fondées sur les déclencheurs.

La réponse d'Oracle : APEX 26.1

Oracle n'est pas resté immobile. APEX 26.1, sorti en mai 2026, livre des agents IA dont les outils sont écrits en SQL ou en PL/SQL, un choix de modèles et une étape d'approbation pour les actions sensibles. Oracle publie aussi un assistant IA qui transforme les modules Forms en artefacts de modernisation APEX : une spécification fonctionnelle, une matrice de traçabilité, un rapport. C'est une option sérieuse, et elle a la même faiblesse que toute réécriture. Les modules Forms sont analysés pour en tirer une spécification, et l'application APEX, agents compris, est bâtie à neuf à partir de celle-ci. Les déclencheurs qui encodaient vingt ans de règles deviennent de la documentation, et les outils de l'agent sont du code nouveau. La question à poser à toute voie de migration n'est pas de savoir si elle offre des agents, mais si l'agent fonctionne sous les règles que vous avez déjà.

Dépendance stratégique

APEX ne fonctionne que sur Oracle Database. Le choisir vous maintient dans les licences et l'infrastructure d'Oracle, bloque toute migration future de la base de données et limite les économies que permettraient des solutions à code source ouvert.

APEX

Dépendance envers le fournisseur

- Lié à Oracle Database
- Options de migration de la base de données limitées
- Coût total de possession plus élevé

React / Angular

Aucune dépendance

- Une voie vers PostgreSQL avec ORMIT™-Cortex
- La latitude pour réduire les coûts
- Le choix du déploiement

Un progiciel : le coût caché de repartir à zéro

Pour une organisation qui remplace Oracle Forms ou Reports, un progiciel commercial (COTS) semble attrayant : des fonctions toutes faites, le soutien de l'éditeur. Derrière cette promesse se cache un parcours fait de risques, de perturbations et de coûts cachés.

Des opérations perturbées

Les processus familiers cèdent la place à un modèle défini par l'éditeur, et la continuité des activités quotidiennes en fait les frais.

Des coûts de mise en œuvre bien plus élevés

Beaucoup de services-conseils, de longs délais, et des licences par utilisateur ou par module.

Une logique d'affaires perdue

Une application Forms contient généralement des centaines de milliers de lignes de logique soigneusement optimisée. La réimplanter dans un progiciel est lent, sujet aux erreurs et plein de lacunes fonctionnelles.

Un risque d'échec plus élevé

Bien des projets de progiciel manquent leurs objectifs, faute d'adéquation avec les activités ou parce que leur complexité a été sous-estimée.

Une gestion du changement lourde

Des équipes entières à former de nouveau, des processus à redéfinir, l'adhésion des utilisateurs à reconstruire de zéro.

Des investissements passés perdus

Repartir à zéro, c'est jeter des années de flux de travail, de validations et de connaissances des utilisateurs intégrées au système existant.

Une voie plus judicieuse : tirer parti plutôt que remplacer

Plutôt que de tout remplacer, une migration directe avec ORMIT™-OpenJava et ORMIT™-Jasper vous permet de préserver la logique d'affaires qui a fait ses preuves, de garder les repères de vos utilisateurs et la continuité des opérations, de moderniser l'architecture sans réinventer la roue, et de réduire le risque et le coût total de la transformation.

Le prochain chapitre décrit la seule voie qui garde les règles.

Migrer les règles, pas les écrans

07

ORMIT™-OpenJava ne réinterprète pas un formulaire. Il le migre : chaque déclencheur, validation et procédure devient du TypeScript qu'OFCS, le service de compatibilité Oracle Forms, exécute au moment même où Oracle Forms l'exécutait. Ce chapitre décrit la plateforme, l'analyse par laquelle commence chaque projet, et l'environnement d'exécution qui rend une application migrée prête pour les agents.

Ce qu'est ORMIT™-OpenJava

ORMIT™-OpenJava est une plateforme de modernisation, pas seulement un outil de conversion. Ses moteurs d'automatisation analysent, convertissent et optimisent le code Oracle Forms en applications TypeScript, en faisant correspondre chaque partie d'un système Forms, déclencheurs, fonctions intégrées, blocs de données et bibliothèques PLL, à des composants : Redux Toolkit ou NgRx Signals pour l'état du formulaire, une API REST, React ou Angular, et Node.js. Elle préserve fidèlement la logique et la livre avec peu de travail manuel. Elle fournit aussi l'environnement opérationnel : points d'intégration, normes de sécurité, générateurs de documentation et outils pour l'après-mise en production. Sa couche serveur, Java + Node.js ou Node.js seul, et son interface, React ou Angular, s'adaptent aux forces de votre équipe.

Ce que vous obtenez

- **Une automatisation prête à l'emploi :** des centaines de gabarits, de modèles et de convertisseurs préconçus pour les composants, les déclencheurs, les bibliothèques et les fonctions intégrées de Forms. Des délais plus courts, des coûts plus bas.
- **Une logique d'affaires préservée :** déclencheurs, validations et logique centrale migrés en TypeScript et exécutés par OFCS, dans le navigateur, puis de nouveau sur le serveur avant que chaque sauvegarde n'atteigne la base de données.
- **Des composants standards :** MUI pour React, Angular Material pour Angular; rien de sur mesure à maintenir.
- **Natif de l'infonuagique et portable :** prêt pour AWS, Azure, Google Cloud, Oracle Cloud ou une grappe privée.
- **Un coût total de possession plus bas :** ni WebLogic ni environnement d'exécution Forms, une structure de coûts à code source ouvert, un soutien simplifié.
- **Une interface moderne et adaptative :** sur téléphone, tablette et ordinateur, sans travail supplémentaire.

Commencer par un inventaire complet : ORMIT™-Analyzer

Une migration réussie commence par la connaissance, pas par des suppositions. Chaque projet RENAPS commence par une analyse complète du système avec ORMIT™-Analyzer, notre outil d'analyse et d'évaluation. Son rapport vous permet de :

- Comprendre la structure et la portée exactes de tous les objets Forms et Reports (FMB, MMB, OLB, PLL, RDF).
- Répertorier toutes les fonctions intégrées, les déclencheurs, les instructions et les niveaux de complexité, module par module.
- Évaluer la qualité du code source, la répartition de la logique d'affaires et la duplication du code.
- Repérer les fonctionnalités obsolètes, le code mort et les occasions d'optimisation.
- Établir l'ordre de migration idéal à partir des dépendances entre les modules.
- Obtenir une estimation détaillée du temps, du coût et de l'effort d'équipe de chaque phase du projet.

Résultat : une feuille de route sans surprise, chaque étape cadrée, chiffrée et appuyée par l'automatisation. Personne n'avait la liste complète des règles. Maintenant, vous l'avez.

OFCS : des règles qui s'exécutent sans écran

Ce qu'est OFCS

OFCS (Oracle Forms Compatibility Service), le service de compatibilité Oracle Forms, est l'environnement d'exécution au cœur d'ORMIT™-OpenJava. Il exécute les règles d'un formulaire migré, chaque déclencheur, validation et procédure, en TypeScript, au moment où Oracle Forms les exécutait : dans le navigateur pendant que l'utilisateur travaille, et dans un serveur Node.js sans écran. Tout ce que décrit ce livre blanc repose sur lui.

94

Types de déclencheurs Oracle Forms reproduits par OFCS, au moment où Forms les déclenche

~250

Fonctions intégrées d'Oracle Forms et fonctions SQL offertes au code migré

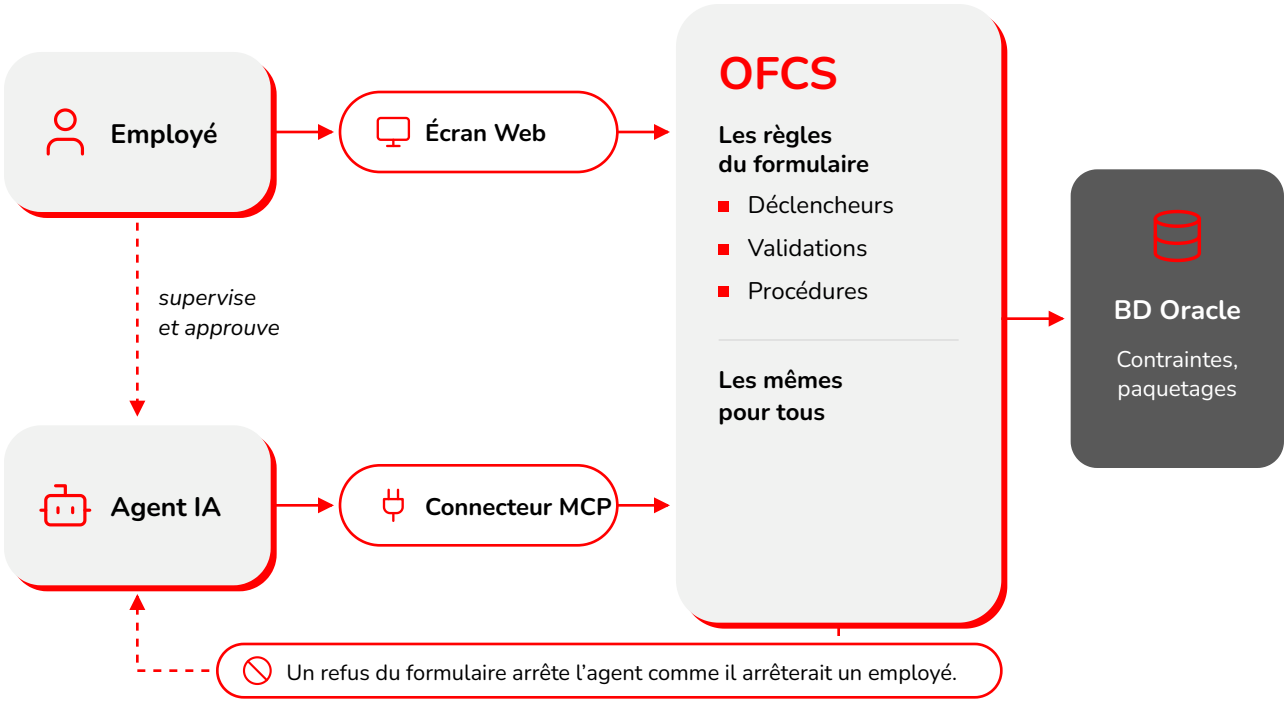
2×

Chaque sauvegarde est validée deux fois : dans le navigateur, puis sur le serveur avant la base de données

La seconde validation est le détail qui compte pour les agents. Depuis 2026, chaque application migrée est livrée avec un serveur de validation : un processus Node.js qui exécute le même OFCS, sans écran. Avant qu'une insertion ou une mise à jour n'atteigne Oracle, ce serveur charge le formulaire, rejoue ses déclencheurs de validation et de commit sur l'enregistrement, et refuse la sauvegarde si le formulaire l'aurait refusée. Contourner l'écran ne permet pas de contourner les règles.

Du point de vue du formulaire, un agent IA est un appelant de plus, sans écran. Par le connecteur (le Model Context Protocol, MCP), il lira les blocs et les éléments du formulaire, saisira des valeurs dans l'ordre de navigation pour que la validation et les effets de chaque élément s'exécutent, appuiera sur ses boutons, puis validera la transaction. Chacune de ces actions passera par OFCS. Quand un déclencheur lève une erreur, l'agent recevra le message du formulaire lui-même et s'arrêtera, comme un employé s'arrêterait devant le même message.

L'agent et l'employé passent par les mêmes règles.



Écran ou connecteur : les mêmes règles. Dans cette conception, l'agent n'a aucun chemin à lui vers la base de données : il passe par le connecteur, sous une identité propre, puis par les mêmes règles que l'écran, avant que quoi que ce soit n'atteigne la base de données.

Ce que vous obtenez aujourd'hui, et ce que cela permet demain

Aujourd'hui, ORMIT™-OpenJava livre une application React ou Angular que vos utilisateurs utilisent comme ils utilisaient les formulaires, avec les mêmes touches, listes de valeurs, validations et messages. Cette application est le produit. Ce qu'elle permet ensuite, c'est une application que n'importe quel agent peut exploiter, puis un agent qui prend un processus en charge : le chapitre 10 décrit ces deux volets. Les clients qui migrent maintenant n'achètent pas une solution provisoire. Ils achètent le seul type de migration auquel on peut ajouter un agent sans tout recommencer.

Ce que vos utilisateurs conservent

Une force de l'approche tient à ce qu'elle conserve pour les gens qui utilisent les formulaires chaque jour. Raccourcis clavier (F1 à F12), tabulation entre les champs, listes de valeurs, validations et messages, interactions entre formulaires : tout est transposé, de sorte que l'application migrée s'utilise comme les formulaires. Elle est aussi adaptative par défaut, utilisable sur un ordinateur, une tablette ou un téléphone, iPhone et iPad compris, sans retouche. La gestion du changement est plus légère, et le déploiement plus fluide.

Ce qui reste dans la base de données y reste

Migrer les règles ne veut pas dire les déplacer. La logique des paquetages, les contraintes et les privilèges restent dans Oracle, intacts, et chaque écriture s'exécute dans la base de données sous l'identité Oracle de son appelant : celle de l'employé depuis un écran aujourd'hui, et celle de l'agent une fois le connecteur en place. Les rôles et les privilèges qu'applique déjà la base de données continuent de fonctionner. Les règles du formulaire, qu'Oracle Forms exécutait dans son propre environnement d'exécution, s'exécutent maintenant dans OFCS. Rien n'est relégué à une consigne donnée à l'IA.

Ce que cela rend possible

Pour la migration

- Le comportement du formulaire d'origine est reproduit, bogues compris, et prouvé par des tests de bout en bout rédigés à partir du PL/SQL d'origine.
- Les mêmes règles servent une interface React ou Angular, et la même suite de tests s'exécute sur les deux.
- L'application fonctionne sur la base de données Oracle existante du client.

Pour l'agent

- Les métadonnées du formulaire décrivent chaque bloc, élément, liste de valeurs et bouton : ce dont un agent a besoin pour agir.
- Le serveur de validation répond déjà aux appelants sans écran.
- Les tests de bout en bout qui prouvent la migration deviennent le moyen d'évaluer l'agent.

La différence, en une ligne

Tout le monde peut placer une IA devant un système transactionnel. L'y placer en toute sécurité exige des règles qui sont du code, appliquées avant la base de données, et les mêmes pour l'agent que pour l'employé. OFCS, au cœur d'ORMIT™-OpenJava, garde les règles que vous avez déjà exactement sous cette forme, et les exécute pour l'agent comme il les exécute pour l'employé.

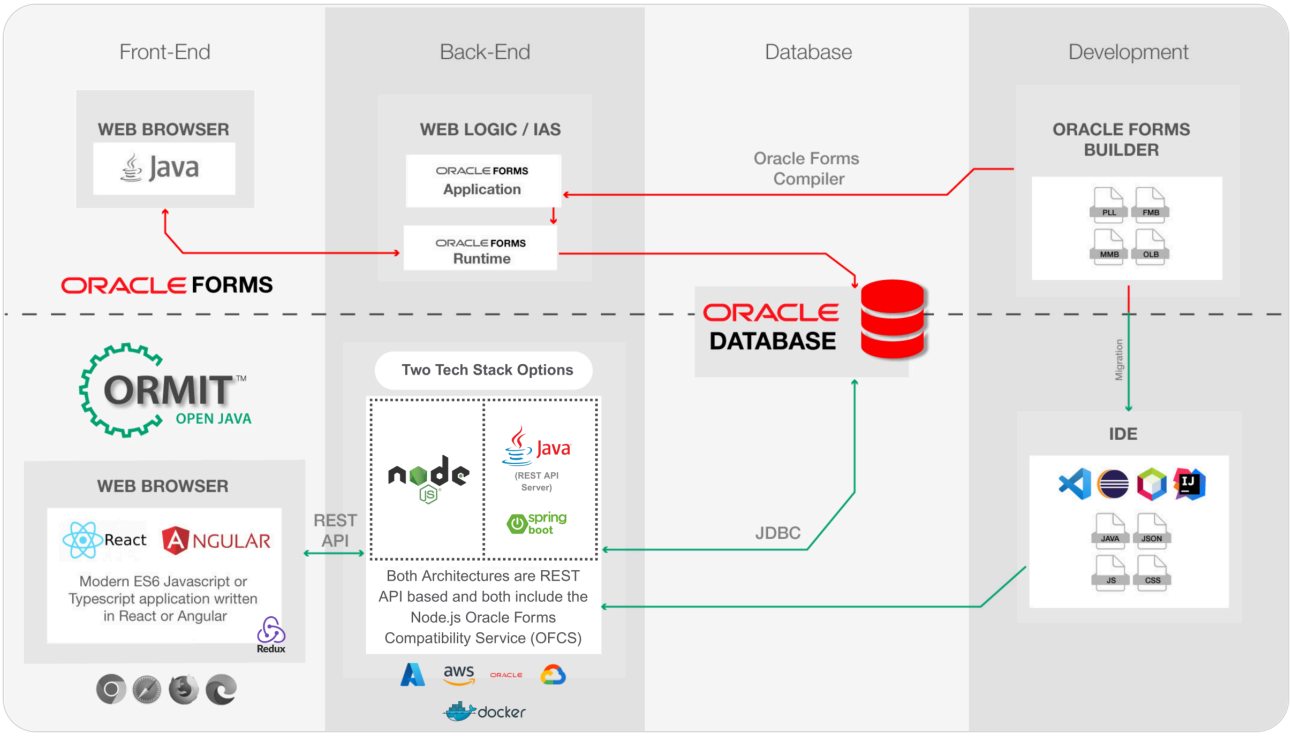
L’architecture : souple, ouverte, à vous

08

Deux options côté serveur, un même résultat

ORMIT™-OpenJava s’adapte à la pile technologique que vous avez et à celle vers laquelle vous allez. Certaines équipes exploitent une infrastructure Java, connaissent Spring Boot et gèrent des serveurs d’applications comme Tomcat ou JBoss. D’autres ont misé sur JavaScript et veulent une seule pile TypeScript. Les deux sont servies :

- **Java + Node.js (par défaut) :** une API REST Spring Boot, plus un serveur de validation Node.js où OFCS relance la validation du formulaire avant chaque insertion et chaque mise à jour. Pour les équipes qui ont une expertise Java et un serveur d’applications en place.
- **Node.js seul :** une couche serveur légère, TypeScript de bout en bout, pour les organisations qui veulent rester à 100 % sans Java.



D’Oracle Forms à ORMIT™-OpenJava. Au-dessus de la ligne, la pile Oracle Forms : applet Java, WebLogic, environnement d’exécution et compilateur Forms, Forms Builder. En dessous, la pile migrée : React ou Angular dans le navigateur, une API REST en Node.js ou en Java avec Spring Boot, toutes deux avec OFCS sous Node.js, la même base de données Oracle, et n’importe quel EDI moderne. Les deux livrent la même application : moderne, évolutive, pilotée par API.

Les deux options, côte à côte

Composant	Java + Node.js (par défaut)	Node.js seul
Cadre d'interface	Angular ou React	Angular ou React
Gestion de l'état	Redux Toolkit; NgRx Signals pour Angular	Redux Toolkit; NgRx Signals pour Angular
Langage côté serveur	Java (API), TypeScript (OFCS)	TypeScript
Serveur d'API REST	Java 21 (Spring Boot 4) + serveur de validation Node.js	100 % Node.js
Conteneur	Docker (avec Tomcat intégré)	Docker ou Node natif
Serveur d'applications	Tomcat, JBoss ou équivalent	Node.js
Compatibilité Oracle Forms (OFCS)	Navigateur + serveur de validation Node.js	Navigateur + serveur Node.js

React ou Angular

Les deux interfaces sont prises en charge et maintenues par RENAPS. Le choix suit les préférences de votre équipe et vos normes : la légèreté et la souplesse de React, ou la structure tout-en-un d'Angular. Interface, règles d'affaires, validations et logique Oracle Forms sont traitées de façon identique, et les mêmes tests automatisés de bout en bout s'exécutent sur les deux.

Aspect	React	Angular
Architecture	Une bibliothèque centrée sur l'interface	Un cadre complet, tout compris
Courbe d'apprentissage	Plus douce; souple, avec des bibliothèques tierces	Plus raide; routage, injection de dépendances et formulaires intégrés
Souplesse	Très personnalisable	Structure imposée, pour les grandes applications
Communauté et soutien	Très répandu, vaste écosystème	Solide soutien en entreprise, secteur public compris
Intégration	Facile à intégrer aux applications existantes	Pour les nouvelles applications à structure stricte

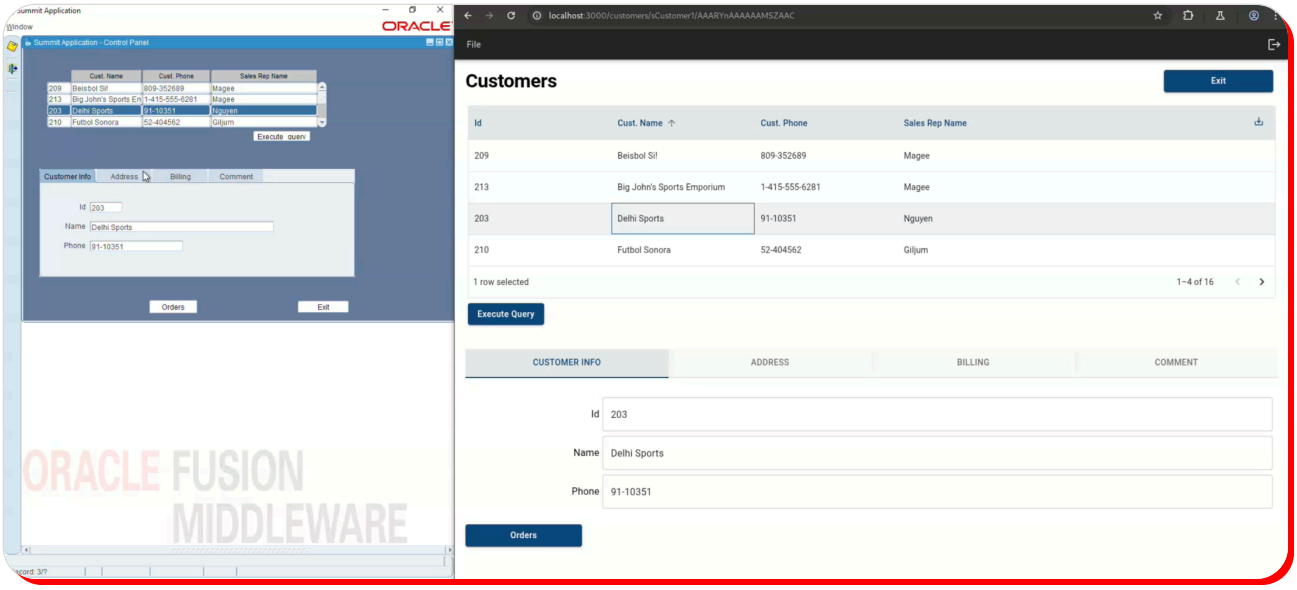
D’Oracle Forms à la pile moderne : les équivalences technologiques

Le tableau montre comment les composants Oracle Forms correspondent à leurs équivalents dans ORMIT™-OpenJava, avec une interface React ou Angular : l’étendue des fonctionnalités conservées, et les outils qui reproduisent chaque comportement.

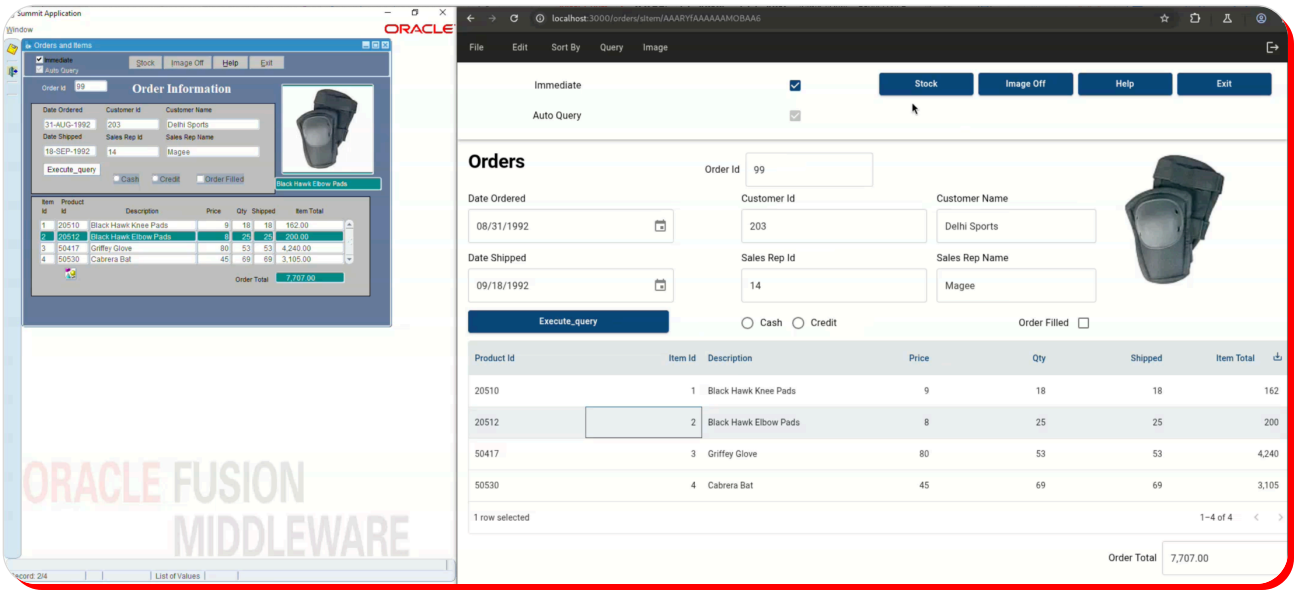
Oracle Forms	ORMIT™-OpenJava, React	ORMIT™-OpenJava, Angular
Positionnement absolu (X-Y)	Positionnement CSS Grid et Flex	Positionnement CSS Grid et Flex
Style des champs	CSS + thèmes Material UI (MUI)	CSS + thèmes Angular Material
Canevas	Composants React	Composants Angular
Section de canevas	Zone React	Composant de zone Angular
Élément	État du formulaire (Redux Toolkit)	État du formulaire (NgRx Signals)
Type de champ d’élément	Composants MUI React	Composants Angular Material
Propriétés	État du formulaire (Redux Toolkit)	État du formulaire (NgRx Signals)
Menu	Menus MUI	Menus Angular Material
PL/SQL	TypeScript	TypeScript
Contrôle d’accès par la base de données	Jeton signé; chaque écriture s’exécute sous le compte Oracle de l’utilisateur	Jeton signé; chaque écriture s’exécute sous le compte Oracle de l’utilisateur
Table de la base de données	État du formulaire (Redux Toolkit)	État du formulaire (NgRx Signals)
Fonctions intégrées (built-ins)	Fonctions intégrées d’OFCS	Fonctions intégrées d’OFCS
Bibliothèque PLL	Fonctions TypeScript partagées ou propres au formulaire	Fonctions TypeScript partagées ou propres au formulaire
Paramètres du formulaire	État du formulaire (Redux Toolkit)	État du formulaire (NgRx Signals)
Fenêtre	Composant React qui reproduit une page ou une boîte de dialogue	Composant Angular qui reproduit une page ou une boîte de dialogue
Bloc multienregistrement	MUI X Data Grid Pro	AG Grid Community
Bloc à enregistrement unique	Modèle + état + tranche de données	Modèle + état + tranche de données
Alertes	Composant React + tranche de données pour les propriétés de l’alerte	Composant Angular + tranche de données pour les propriétés de l’alerte
Déclencheurs	Fonction TypeScript, déclenchée par OFCS	Fonction TypeScript, déclenchée par OFCS
Validation à la sauvegarde	Revérifiée par OFCS sur le serveur avant la sauvegarde	Revérifiée par OFCS sur le serveur avant la sauvegarde

À quoi cela ressemble : Oracle Forms à côté d'ORMIT™-OpenJava

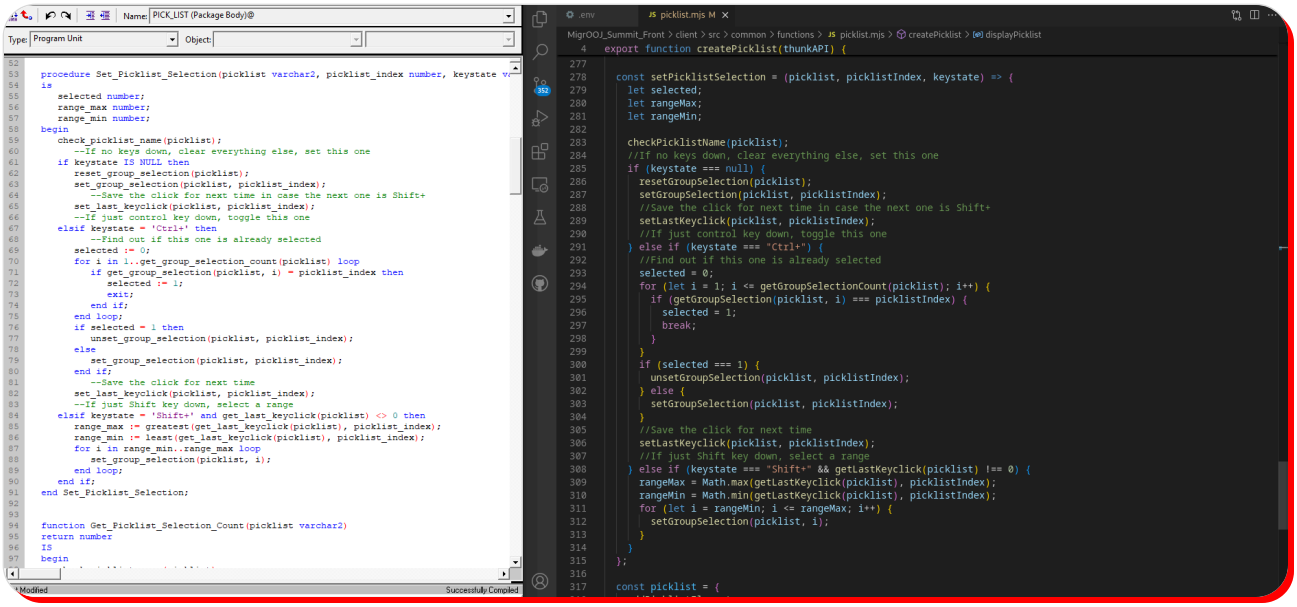
Les pages suivantes montrent de vrais exemples de transformation d'artefacts Oracle Forms, tirés de l'application de démonstration Summit livrée avec Oracle Forms, migrée vers React.



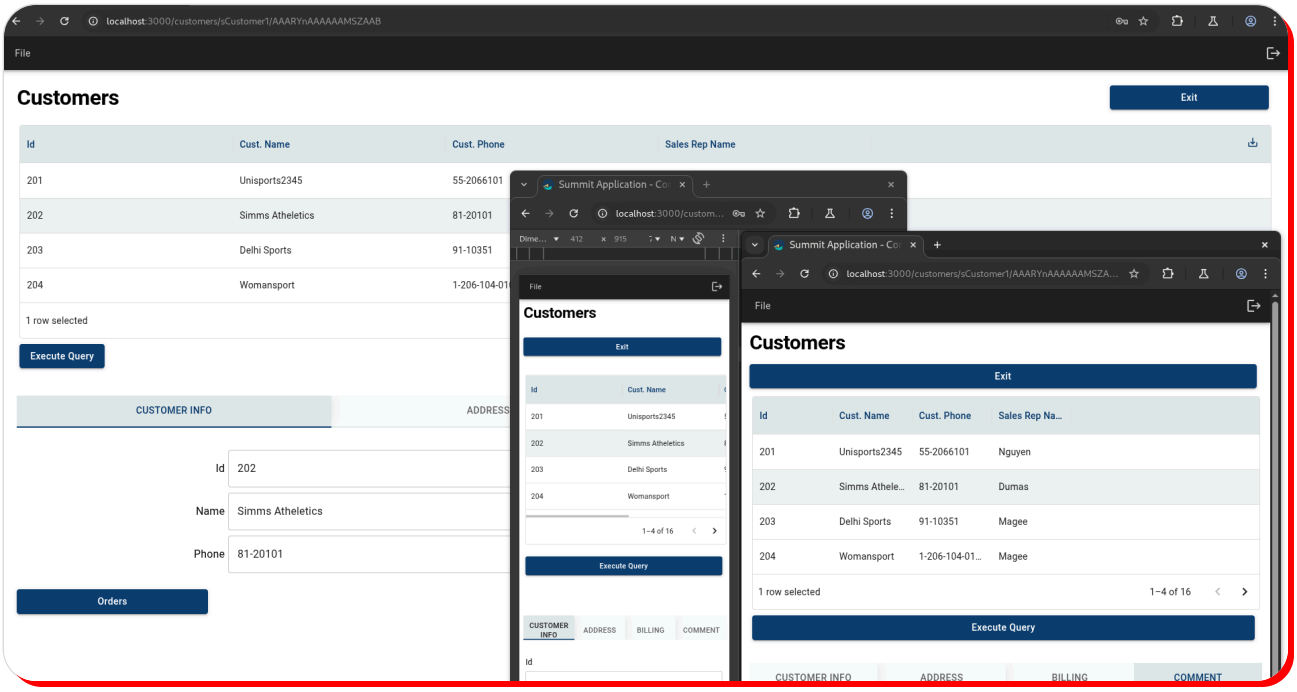
Côte à côte. Le formulaire Customers dans Oracle Forms, et le même formulaire migré : même bloc, mêmes champs, mêmes touches, sous forme de page adaptative dans un navigateur, avec le canevas à onglets sous la grille.



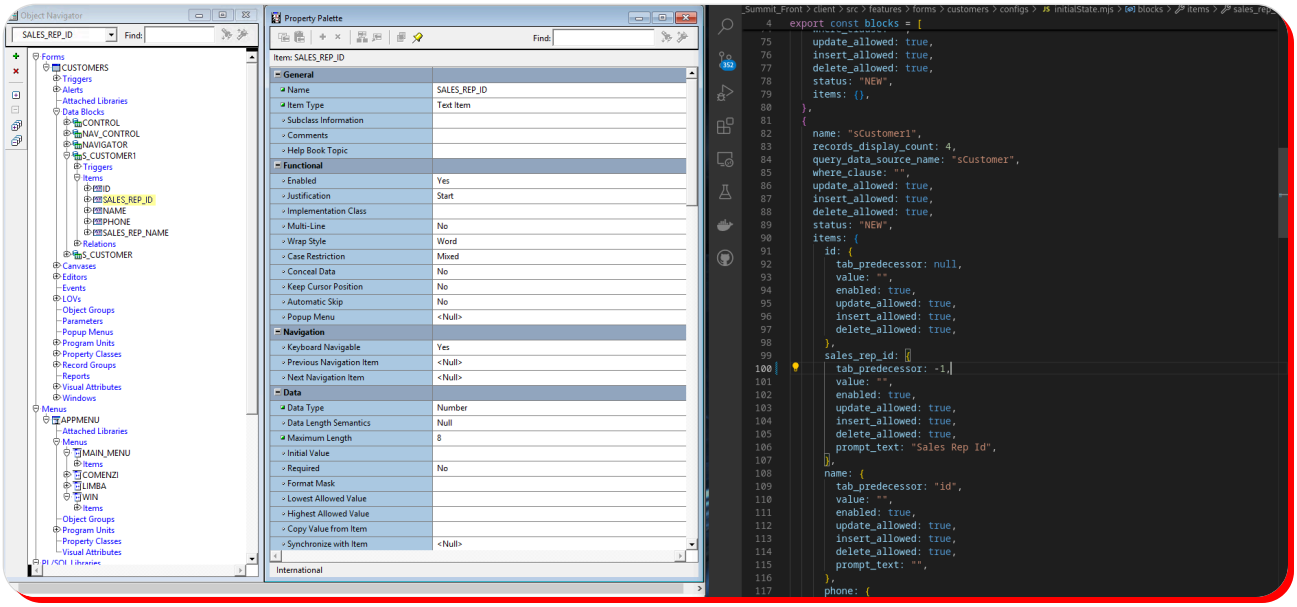
Le formulaire Orders. Un bloc maître, une grille de détail et un élément image, migrés avec leurs listes de valeurs, leurs cases à cocher et leurs validations.



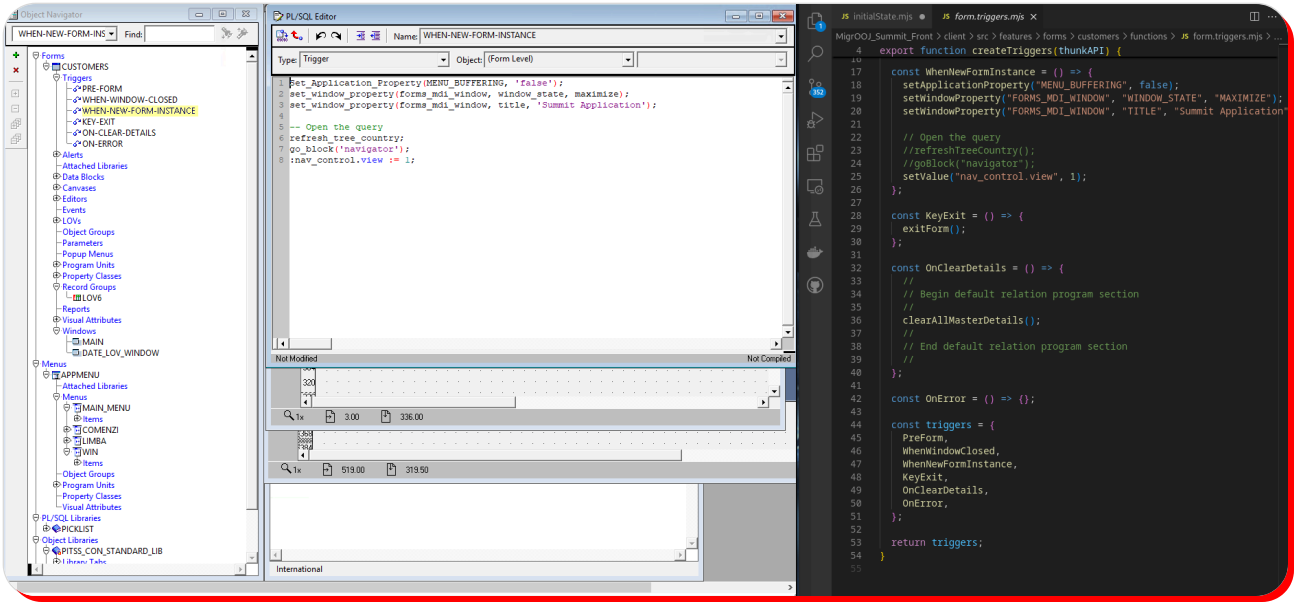
De l'unité de programme au TypeScript. La procédure PL/SQL Set_Picklist_Selection dans Forms Builder, et sa version migrée : même nom, même logique, même ordre d'opérations. Les fonctions intégrées deviennent des appels à OFCS, et les commentaires suivent le code.



Adaptatif par défaut. Le formulaire Customers migré sur un ordinateur, une tablette et un téléphone, sans travail supplémentaire.



Blocs et éléments. Le bloc CUSTOMERS et les propriétés de l'un de ses éléments dans Forms Builder, et le bloc dans l'état initial du formulaire migré, où chaque élément garde ses propriétés : insertion et mise à jour permises, libellé, navigation.



Déclencheurs. WHEN-NEW-FORM-INSTANCE dans l'éditeur PL/SQL, et le fichier des déclencheurs du formulaire migré, où OFCS exécute chaque fonction TypeScript au point où Oracle Forms exécutait le déclencheur.

Prêt pour l’infonuagique, par défaut

Une fois la migration faite, plus rien ne dépend de l’environnement d’exécution Oracle Forms, de WebLogic ni d’un EDI propriétaire. La pile est ouverte, bâtie sur des cadres et des langages courants : TypeScript, React ou Angular, Spring Boot et Node.js. Votre organisation obtient :

Le plein contrôle du code source

Aucune dépendance envers un fournisseur

N’importe quel EDI ou chaîne d’outils moderne

Pipelines DevOps, CI/CD et infrastructure native de l’infonuagique

Quand vous comparez des options de modernisation, méfiez-vous des solutions qui créent une nouvelle dépendance : un environnement d’exécution propriétaire, un cadre obscur, une bibliothèque contrôlée par un éditeur qui vous lie à un seul fournisseur à long terme. Avec ORMIT™-OpenJava, votre équipe est autonome pour maintenir et faire évoluer l’application.

Les applications sont légères et indépendantes du fournisseur infonuagique, prêtes pour Oracle Cloud, AWS, Microsoft Azure, Google Cloud ou une grappe Kubernetes privée, et conçues pour fonctionner dans des environnements conteneurisés et évolutifs :

- Entièrement compatibles avec Docker, Kubernetes et les plateformes d’orchestration.
- Une architecture native REST, adaptée aux microservices et aux systèmes distribués.
- Des services côté serveur hébergés dans n’importe quel nuage ou modèle hybride.
- Mise à l’échelle élastique, surveillance et intégration aux pipelines d’infrastructure en tant que code.

Par défaut, vous adoptez la stratégie infonuagique de votre choix, sans refonte.

Sécurité, confidentialité et compatibilité DevOps

L'application migrée est livrée avec un contrôle d'accès par rôles, une gestion sécurisée des sessions et un chiffrement fort des données au repos et en transit, selon les exigences de protection des données, de conformité et de gouvernance des entreprises et des organismes publics. La gestion des identités peut rester au niveau d'Oracle Database ou s'intégrer aux fournisseurs d'identité de l'entreprise, comme LDAP, OAuth ou une authentification unique SAML.

L'IA assiste la migration dans une boucle fermée, ce que Gartner appelle un harnais d'agent de codage IA. Chaque changement passe des vérifications déterministes, un audit par rapport à la source Oracle d'origine et des tests de bout en bout dont on a prouvé qu'ils peuvent échouer, avant toute revue humaine.

La pile obtenue est native REST, avec une connectivité API sécurisée et efficace vers les systèmes externes, les services infonuagiques et les microservices. Parce que le code est ouvert et maintenable, le système s'intègre aux pipelines DevOps. La plateforme elle-même est couverte par des milliers de tests automatisés, dont des tests côté serveur sur une vraie base de données Oracle et une suite de bout en bout qui s'exécute sur les deux interfaces, à chaque changement.

La performance dès la conception

ORMIT™-OpenJava ne se contente pas de moderniser un système hérité : il le rend plus rapide. Chaque couche est conçue pour la réactivité :

- **La logique d'un formulaire chargée seulement à son ouverture**
- **Les enregistrements lus page par page, comme dans Oracle Forms**
- **L'API REST ne garde aucune session serveur**
- **Chaque formulaire migré chronométré par rapport à l'original**

Du mode avec état au mode sans état : une montée en charge simplifiée

Oracle Forms repose sur une architecture avec état, fondée sur les sessions : chaque session utilisateur occupe un processus dédié du système d'exploitation et une connexion permanente à la base de données, ce qui limite la montée en charge, consomme des ressources serveur et complique le basculement et la répartition de charge. ORMIT™-OpenJava la remplace par une architecture REST sans état. L'état du formulaire et le moteur de déclencheurs s'exécutent dans le navigateur, l'API ne garde aucune session serveur, et la couche serveur utilise un regroupement de connexions : prête pour l'infonuagique, extensible horizontalement, résistante aux interruptions de serveur. Moins de ressources, une meilleure gestion des accès concurrents et un déploiement plus simple en grappe ou en conteneurs. Les applications migrées égalent, et souvent dépassent, la vitesse et la réactivité de l'environnement Oracle Forms d'origine.

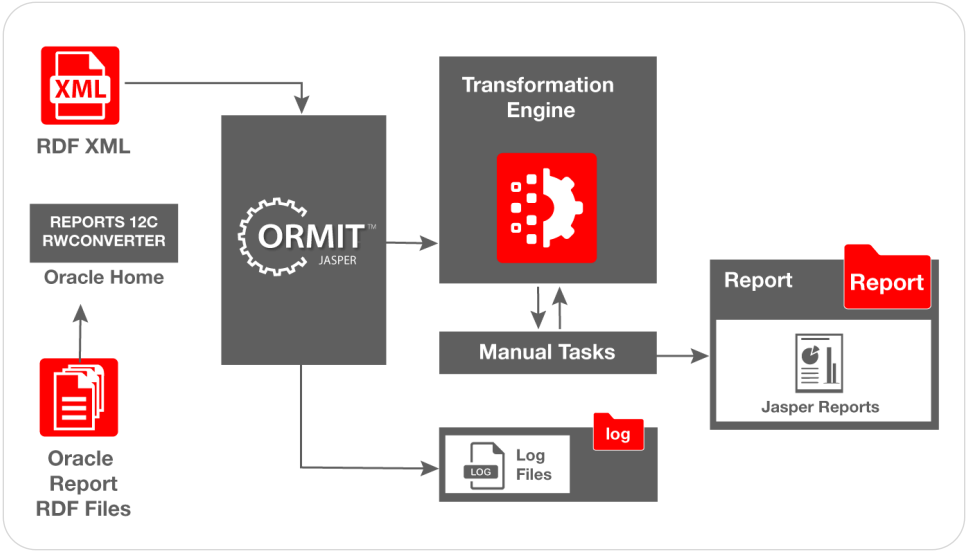
Au-delà de Forms : Reports, base de données, EBS et éditeurs

09

Oracle Reports avec ORMIT™-Jasper

L'architecture du chapitre précédent couvre les formulaires. La plupart des applications Oracle Forms s'appuient aussi sur Oracle Reports pour les rapports opérationnels et réglementaires. Oracle Reports arrivant en fin de vie, les rapports ont besoin de leur propre voie de migration. ORMIT™-Jasper convertit les rapports Oracle Reports (RDF) en JasperReports modernes et maintenables, sur une base à code source ouvert, et le résultat est pleinement pris en charge dans l'architecture d'ORMIT™-OpenJava, si bien que Forms et Reports se modernisent en parallèle.

- Transformation complète de RDF vers JasperReports, mise en page, paramètres et logique d'affaires compris.
- Intégration par l'API REST, pour exécuter les rapports directement dans l'application modernisée.
- Codes à barres, sorties multilingues et options de planification.
- Fidélité des sorties et équivalence fonctionnelle avec les rapports d'origine.
- Jasper Community Edition comme base, avec des mises à niveau commerciales en option.



Comment un rapport est migré. Les fichiers RDF sont exportés en XML avec le convertisseur d'Oracle, le moteur de transformation produit les JasperReports, et les tâches manuelles qui restent sont consignées dans les journaux.

La base de données : d'Oracle à PostgreSQL avec ORMIT™-Cortex

Migrer les formulaires vers React ou Angular ne préserve pas seulement la logique et l'architecture. Cela ouvre la porte à l'indépendance envers la base de données. Avec son partenaire Splendid Data, RENAPS offre ORMIT™-Cortex, une plateforme de migration conçue pour transformer les bases de données Oracle en PostgreSQL avec précision, automatisation et un minimum de perturbations.

Conversion automatique de PL/SQL vers PL/pgSQL

Contrairement aux outils traditionnels à code source ouvert, ORMIT™-Cortex convertit les paquetages, procédures et déclencheurs PL/SQL complexes en PL/pgSQL natif, en préservant la logique sans lourde réécriture manuelle.

Migration complète du schéma

Tables, index, vues, séquences et types définis par l'utilisateur sont analysés, mis en correspondance et restructurés en leurs équivalents PostgreSQL.

Optimisation après la migration

Réglage des performances et ajustements selon les bonnes pratiques, pour que l'environnement PostgreSQL atteigne ou dépasse les standards d'Oracle.

Un avenir sans licence

Une base de données à code source ouvert de niveau entreprise, et la fin des coûts de licence Oracle.

Point essentiel, ORMIT™-Cortex cible PostgreSQL pur, sans points d'ancrage propriétaires cachés, sans extensions ni dépendances de plateforme. Contrairement à des solutions comme EnterpriseDB ou AWS Aurora, qui créent une dépendance propre à leur plateforme, il préserve la pleine portabilité de PostgreSQL : vous déployez sur Oracle Cloud, AWS, Azure ou sur site, et passez d'un nuage à l'autre, sans contrainte ni refonte.

Pour les organisations qui visent une indépendance complète envers les technologies Oracle, ORMIT™-Cortex complète le parcours : la modernisation de Forms, de Reports et de la base de données dans une même feuille de route structurée. Une fois les formulaires en TypeScript, la base de données devient une décision distincte.

Personnalisations Oracle EBS et produits d’éditeurs de logiciels

Beaucoup d’organisations exploitent des applications Oracle Forms autonomes. Un nombre important utilise aussi des écrans Oracle Forms personnalisés, développés pour Oracle E-Business Suite (EBS). Ces modules sur mesure portent des flux de travail essentiels, et ils restent liés à une technologie Forms héritée. ORMIT™-OpenJava les découple et les modernise en interfaces Web, en préservant la logique d’affaires et en limitant les perturbations des processus du progiciel de gestion intégré (PGI).

Les éditeurs de logiciels indépendants (ISV) dont le produit basé sur Forms est déployé chez de nombreux clients font face à une version plus vaste du même problème : moderniser sans réécrire toute leur gamme de produits. ORMIT™-OpenJava permet à un éditeur de migrer une seule fois, de moderniser son architecture et de redéployer chez chacun de ses clients, sans aucune dépendance envers un fournisseur et avec une interface améliorée. Résultat : des coûts de soutien plus bas, des fonctions développées plus vite et une offre prête pour l’infonuagique.

Une application
que tout agent peut exploiter

10

Qu'on exploite une application autonome, des personnalisations EBS ou le produit d'un éditeur, l'application migrée est la même, et la suite aussi. Une migration vers les agents ne remplace pas un écran par une fenêtre de conversation. Elle change ce que l'organisation achète, en deux volets : d'abord une application que n'importe quel agent peut exploiter sous ses règles, puis des processus exécutés par un agent, ses équipes demeurant responsables du résultat.

Deux volets

Volet	Ce que l'organisation obtient	Quel agent	Comment c'est prouvé
1. Prêt pour les agents, à livrer avec la migration ou à y ajouter	Un connecteur ouvert qui expose le formulaire migré à un agent, ses blocs, éléments, listes de valeurs et boutons, dans l'ordre propre au formulaire : le Model Context Protocol aujourd'hui, et les mêmes opérations en ligne de commande ou par une simple API quand la plateforme du client le préfère. Chaque appel passe par OFCS et par les règles du formulaire. L'agent agira sous une identité propre, avec les privilèges de base de données dont son processus a besoin, et chaque action sera consignée avec la règle par laquelle elle est passée.	N'importe lequel : celui de l'organisation, d'un éditeur de plateforme, d'Oracle, de RENAPS.	Testé, comme le reste de la migration : la scène du refus, l'identité, le journal. Aucun modèle n'a besoin d'avoir raison.
2. Processus confiés à un agent, un processus à la fois	Un agent prend en charge le travail d'un processus sous supervision : il propose, un employé approuve, les règles s'appliquent. L'organisation choisit l'agent et le modèle, et fixe le rythme.	L'agent de RENAPS, ou celui choisi au volet 1.	Mesuré : précision sur les cas ambigus, coût et temps par exécution, part des cas qui n'ont plus besoin d'une personne.

Le premier volet s'appuie directement sur la migration. Le second est ce qu'il rend possible, et la suite de ce chapitre le décrit.

Trois façons d'exécuter un processus, une seule source de règles

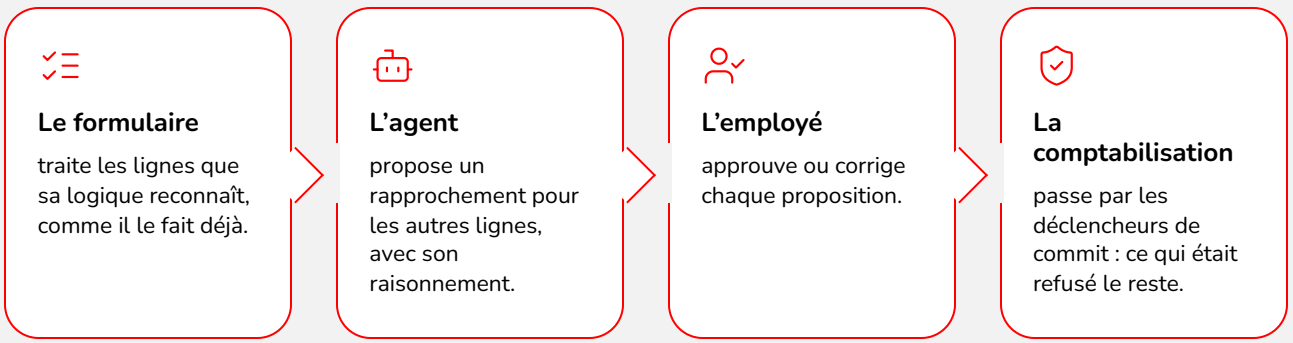
Qui fait le travail	Pour quels cas	Qui approuve
Un programme	Les chemins qui n'exigent aucun jugement : les lignes qu'une règle peut rapprocher seule, les écritures dont chaque champ est connu.	Personne, comme aujourd'hui. Les règles s'appliquent.
Un agent IA	Les cas ambigus : un paiement qui correspond presque, une réclamation à laquelle il manque un document, une demande qui exige une décision dans le cadre des règles.	Un employé, qui voit ce que l'agent propose et pourquoi.
Une personne, à l'écran	Là où un écran sert encore : la saisie au clavier à haut volume, et les exceptions que l'agent ne peut pas régler.	Comme aujourd'hui.

Les trois passent par les mêmes règles OFCS. Faire passer un cas d'une colonne à l'autre change qui agit, jamais ce qui est permis.

Un exemple concret

Prenons le traitement des relevés bancaires, présent dans la plupart des services financiers. Aujourd'hui, un employé ouvre le relevé, rapproche chaque ligne d'un poste ouvert, puis comptabilise. L'application migrée garde les règles du formulaire : quelles lignes peuvent être rapprochées automatiquement, quels comptes une écriture peut toucher, ce qui rend un rapprochement complet.

Avec l'agent en place, la logique de rapprochement du formulaire traite les lignes qu'elle reconnaît, comme elle le fait déjà. L'agent propose un rapprochement pour les autres lignes, avec son raisonnement, et l'employé approuve ou corrige chaque proposition. La comptabilisation passe par les déclencheurs de commit du formulaire : ce que le formulaire refusait reste refusé. L'employé passe de la saisie de chaque ligne au règlement des exceptions.

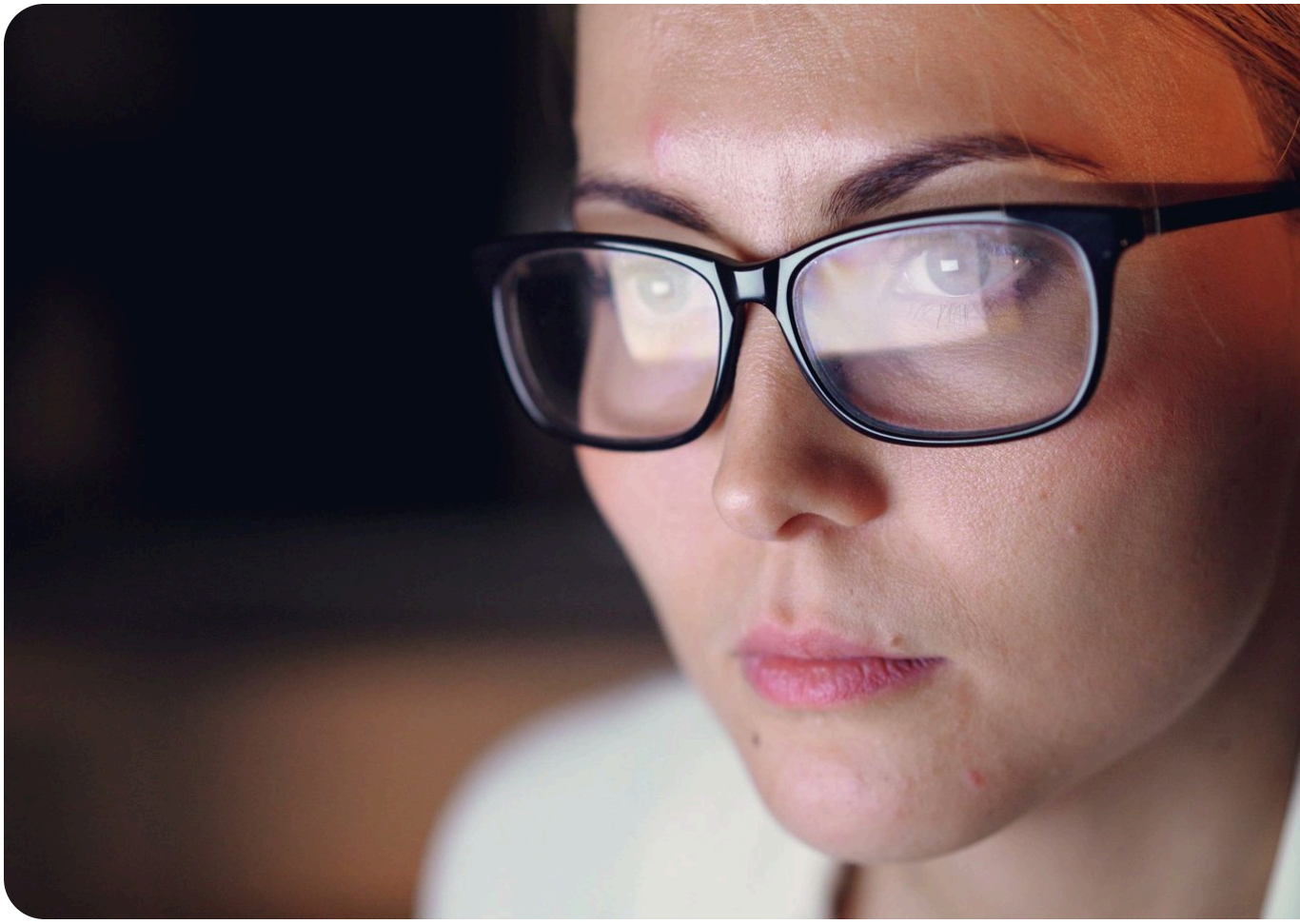


Ce que les règles ne peuvent pas faire

Les règles disent ce que le formulaire refuse. Elles ne disent pas quel rapprochement est le bon quand plusieurs sont valides, et c'est exactement le travail de la colonne de l'agent. Voilà pourquoi l'approbation n'est pas facultative, et pourquoi on mesure un agent à sa précision sur les cas ambigus, pas à son taux de réussite sur les cas faciles. Grâce aux règles, on peut essayer l'agent sans risque. Les mesures décident s'il vaut la peine de le garder.

Ce que font vos équipes à la place

Personne n'a à faire confiance à l'agent. On demande aux gens de le superviser, avec les mêmes règles qu'ils connaissent, et d'approuver son travail avant qu'il compte. Le savoir institutionnel de l'équipe devient l'ensemble des exceptions que l'agent apprend à signaler, et la mesure du processus devient la part des cas qui n'ont plus besoin d'une personne.



Sûr par conception

Les contrôles qu'exige un système transactionnel ne sont pas ajoutés à l'agent après coup. Ils découlent de la façon dont la migration est bâtie.

Identité et responsabilité

Par conception, l'agent a une identité propre : son propre compte de base de données, avec seulement les privilèges dont son processus a besoin, et jamais les identifiants d'un employé. Les tendances 2027 de Gartner en font la première règle de l'identité des agents; les rôles et privilèges d'Oracle l'appliquent déjà. L'agent prépare, un employé approuve : la séparation des tâches qu'attendent les auditeurs, conservée telle quelle. Chaque action sera journalisée avec la règle par laquelle elle est passée, l'agent qui l'a proposée et la personne qui l'a approuvée.

Une confiance progressive

Un agent commence en observation : il propose, et ses propositions sont comparées au travail des employés. Puis chaque action exige une approbation. Puis seules les exceptions en exigent une. L'organisation fixe le rythme, processus par processus.

Vos données, votre région, votre choix de modèle

Le modèle s'exécute là où l'organisation le choisit. Avec Amazon Bedrock ou Google Cloud Vertex AI, requêtes et résultats restent dans le compte infonuagique et la région du client. Sur Microsoft Foundry, Claude est offert aux États-Unis, l'Europe étant annoncée; le Canada n'y est pas encore couvert. Rien n'est envoyé à un service d'IA grand public. Et rien ne dépend d'un modèle : les règles migrées sont le code du client, le connecteur utilise un protocole ouvert, MCP, et le modèle peut être remplacé, y compris par un modèle ouvert exécuté sur l'infrastructure du client. Ces options de sortie sont ce que la tendance SovereignTech de Gartner demande aux organisations hors des États-Unis de s'assurer d'ici 2030.

Un coût mesuré

Contrairement à un écran, chaque action d'un agent a un coût. RENAPS ne le chiffre qu'une fois mesuré sur un vrai formulaire, par exécution, et le client décide s'il paie l'IA sur son compte infonuagique ou par l'abonnement.

Le test du refus

Le test décisif d'une migration sûre, c'est ce qui arrive quand l'agent tente ce que le formulaire interdit. La réponse doit être la même que pour une personne : le formulaire refuse, l'agent reçoit le message du formulaire, le signale, et ne le contourne pas.

Ce que ce livre blanc n’affirme pas

Aujourd’hui, la migration, OFCS et son serveur de validation existent et sont livrés avec chaque migration. Le connecteur et l’offre d’agents décrits dans ce chapitre et au chapitre 12 sont conçus sur cette base, et seront prouvés sur un vrai formulaire avant d’être offerts. Aucune capacité décrite ici ne dépend de la justesse du modèle. Le premier volet, le connecteur, ne dépend d’aucun modèle, et sera testé comme le reste de la migration. Le modèle propose; les règles décident; une personne approuve. Les règles sont nécessaires, mais pas suffisantes : elles arrêtent ce que le formulaire interdit, pas un choix plausible que le formulaire permet. C’est à cela que servent l’étape d’approbation et la précision mesurée. Là où une règle n’existe pas dans le formulaire, l’agent n’a pas plus d’autorité qu’un nouvel employé, et le processus reste confié à une personne jusqu’à ce que l’organisation écrive la règle.

La couche de contexte que demande Gartner, et où un formulaire migré la contient déjà

La plateforme de contexte d'IA de Gartner, quatrième tendance pour 2027, « unifie les connaissances dispersées de l'entreprise en une couche de vérité lisible par machine pour ancrer les travailleurs numériques autonomes ». Ses trois composantes correspondent à ce que préserve ORMIT™-OpenJava, et à ce qu'ajoute l'offre d'agents.

Composante selon Gartner	Ce qu'elle contient, selon Gartner	Dans une application migrée
Sémantique	Ontologie, indicateurs, règles, responsabilités, politiques	Les règles : chaque déclencheur, validation et procédure du formulaire, en TypeScript qu'exécute OFCS. Pas une description des règles. Les règles.
État opérationnel	Entités, conditions de l'environnement, données au bon moment	Les blocs, éléments et listes de valeurs du formulaire, et les enregistrements en temps réel de la base de données Oracle du client, lus par les requêtes qu'exécute le formulaire lui-même.
Provenance	Traçabilité des données, des processus, des décisions, des actions et des résultats	Le journal des actions qu'ajoutera le volet « prêt pour les agents » : pour chaque action, la règle par laquelle elle est passée, ce que l'agent a proposé et la personne qui l'a approuvée.

Pour bâtir cette couche, Gartner recommande de « séparer les règles d'affaires passives des traces de décision actives, générées par les machines ». Cette séparation est la figure centrale de ce livre blanc. Les règles sont du code qui ne change pas quand l'agent s'exécute; le raisonnement de l'agent est une trace conservée à côté, et il ne devient jamais une règle.

Ce qu'un formulaire migré ne fournit pas, c'est le reste de la couche : l'ontologie qu'attend Gartner, dans des normes comme RDF et OWL, et les connaissances qu'une organisation conserve hors de ses formulaires. OFCS alimente une plateforme de contexte; il ne la remplace pas. Il fournit la composante qui n'existe aujourd'hui que dans vos formulaires, les règles, sous la seule forme à laquelle on peut astreindre un agent : exécutable.

La première tendance de Gartner pour 2027, les grands modèles quantitatifs, règle une autre question que posent les clients. Elle part du constat que les modèles de langage « échouent au raisonnement mathématique parce qu'ils devinent de manière probabiliste ». Dans une application migrée, l'agent ne calcule jamais un montant. Les procédures du formulaire le calculent, comme le faisait l'environnement d'exécution d'Oracle Forms, et l'agent lit le résultat.

La preuve, et la comparaison

11

Une équivalence prouvée

Les chapitres précédents décrivent l'approche. Celui-ci porte sur les preuves. Chaque formulaire migré est vérifié par rapport à l'original. Les tests de bout en bout sont rédigés à partir du PL/SQL d'origine, et l'on prouve qu'ils peuvent échouer. La même suite s'exécute sur React et sur Angular. La couche serveur est testée sur une vraie base de données Oracle, chaque changement est audité par rapport à la source Oracle, et chaque formulaire est chronométré par rapport à l'original.

Cas client : un grand manufacturier

Un chef de file nord-américain de la fabrication a modernisé plus de 100 modules Oracle Forms avec ORMIT™-OpenJava, vers une architecture React + Node.js. Le projet s'est étalé sur six mois et a couvert des applications de production et de gestion des stocks essentielles. La migration a conservé la logique d'affaires et les flux de travail des utilisateurs, et l'équipe Forms du client, formée pendant le projet, maintient maintenant l'application. L'entreprise est libérée de ses licences WebLogic et de l'environnement d'exécution Forms.

Cas client : une société de services publics

Une grande société de services publics nord-américaine a confié à RENAPS plus de 500 modules Oracle Forms, plus de 750 000 lignes de code essentiel aux activités. Avec ORMIT™-Analyzer et ORMIT™-OpenJava, le projet a atteint plus de 89 % d'automatisation, préservé la logique d'affaires, conservé les flux de travail des utilisateurs finaux et été déployé rapidement dans un environnement Web entièrement adaptatif. La société a modernisé l'architecture de ses applications et réduit sensiblement ses coûts d'exploitation et de licences.

Indicateurs d'automatisation de la migration

RENAPS a analysé des centaines d'applications Oracle Forms et Reports, dans tous les secteurs, toutes les régions et toutes sortes d'usages. De cet ensemble de données, un indicateur ressort : **ORMIT™-OpenJava automatise en moyenne 88 % de l'effort total de migration.** Près de neuf étapes de transformation sur dix, conversion de la logique, correspondance des composants d'interface, remplacement des fonctions intégrées et génération de la structure du code, sont traitées automatiquement. Les 12 % restants portent généralement sur des règles d'affaires propres au client, des personnalisations héritées ou des choix de modernisation qui demandent la validation d'une personne. Les 99 % du tableau comparatif de la page 44 sont une autre mesure : la part du code converti automatiquement; 88 % est la part de l'effort total du projet, tests compris.

Ce degré d'automatisation se traduit par :

Aucune dérive de conversion

Un convertisseur applique la même règle au dix-millième déclencheur qu'au premier.

Des délais de livraison plus courts

La part automatisée se fait en quelques heures; le temps du projet va aux 12 % qui demandent du jugement.

Des coûts et une portée prévisibles

Ce que l'analyse a mesuré, c'est ce que la migration livre.

Une estimation fiable dès le premier jour

Le rapport d'ORMIT™-Analyzer chiffre chaque phase avant le début du projet.

Comment ORMIT™-OpenJava se compare aux autres outils de migration

Moderniser Oracle Forms demande plus que de convertir du code : cela demande un virage vers des technologies ouvertes et maintenables, et avec tant d'outils sur le marché, il est difficile de distinguer une migration superficielle d'une vraie modernisation. Cette section compare les outils de migration Oracle Forms les plus reconnus selon leur approche technique, leur architecture et leur viabilité à long terme.

Voies de migration et piles cibles

Les outils se distinguent d'abord par ce qu'ils génèrent. Re_Forms21 et Morphis-Tech produisent des applications Java; PITSS.CON migre seulement vers Oracle APEX; VisionX propose une plateforme Java à faible code; Ispirer génère du Java traditionnel avec une forte automatisation. ORMIT™-OpenJava cible une pile entièrement moderne et à code source ouvert : une interface TypeScript en React ou Angular, avec Node.js ou Spring Boot derrière, qui prend en charge les API REST, le déploiement sans état, les conteneurs et la montée en charge native de l'infonuagique.

Dépendance envers le fournisseur et code propriétaire

La dépendance envers le fournisseur est l'un des facteurs les plus déterminants, et les plus sous-estimés. Re_Forms21 et VisionX exigent des environnements d'exécution ou des cadres propriétaires, une dépendance à long terme envers l'écosystème de l'éditeur. Morphis-Tech s'appuie sur ses propres cadres d'assistance. Ispirer produit du Java standard, mais fragmenté, ce qui le rend difficile à maintenir. ORMIT™-OpenJava évite entièrement la dépendance envers un fournisseur : du code propre et modulaire, sur des technologies à code source ouvert largement adoptées, sans bibliothèque, cadre ni boîte noire propriétaire après la migration. Le client est propriétaire de l'application.

Résultats UI/UX

Certains outils visent une réplique au pixel près de l'interface Oracle Forms (Re_Forms21), ce qui donne une expérience datée et peu intuitive. D'autres (Morphis, VisionX) livrent des composants Web modernes au prix d'une nouvelle formation des utilisateurs. ORMIT™-OpenJava conserve les principaux flux de travail des utilisateurs et livre une interface Web native, entièrement adaptative : moderne, intuitive, sur appareils mobiles d'emblée, dans tout navigateur moderne.

Logique d'affaires et maintenabilité

Tous les éditeurs disent préserver la logique d'affaires; l'étendue et la qualité varient. PITSS.CON externalise la logique dans la base de données (PL/SQL), ce qui fonctionne mais limite l'architecture. Re_Forms21 et Morphis analysent le code en profondeur pour convertir le PL/SQL en Java, et le résultat peut être étroitement lié à leur cadre. ORMIT™-OpenJava préserve la quasi-totalité de la logique Oracle Forms sous forme de TypeScript clair et maintenable, sans sacrifier les bonnes pratiques d'architecture. Les développeurs y travaillent avec des outils standards (VS Code, IntelliJ, Eclipse) et adoptent sans peine les pipelines CI/CD et les cadres de test modernes.

En résumé : choisir la modernisation, pas seulement la migration

Outil	Pile cible	Dépendance	Approche UX	Génération technologique	Conversion de la logique	Facilité de maintenance
VisionX	Java (JVx/Vaadin) + outillage VisionX	Élevée	Faible code propriétaire, interface en applet	Années 2010	Limitée; aucune correspondance PL/SQL	Faible; fragmentée, dépendante de l'outil
Re_Forms21	Java + environnement d'exécution propriétaire	Élevée	Forms à l'identique	Années 2000-2010	~99 %, API propriétaires	Faible; liée au cadre
Morphis-Tech	Java ou .NET (MVC) + bibliothèques	Élevée	UX Web moderne	Années 2010	~80–90 %, optimisée	Moyenne; technologie standard
PITSS.CON	Oracle APEX	Élevée (Oracle)	Interface APEX repensée	Années 2000-2010	70 %, dans la base Oracle	Élevée (dans APEX)
Ispirer	Java (traditionnel)	Moyenne à élevée	Aucune migration de l'interface; côté serveur seulement	Années 2010	Partielle; code seulement, sans interface	Faible à moyenne; les développeurs refactorisent
ORMIT™-OpenJava	React/Angular + Node.js/Spring Boot	Aucune	Moderne, adaptative (TypeScript)	Années 2020+	~99 %, TypeScript propre	Élevée; ouverte et modulaire

La colonne « génération technologique ». Années 2020+ : piles modernes, natives de l'infonuagique et compatibles avec les microservices (React, Angular, Node.js, Spring Boot). Années 2010 : encore pertinentes, bâties sur des paradigmes Web antérieurs (Vaadin, Java MVC standard, automatisation de base). Années 2000-2010 : outils ou environnements d'exécution plus anciens, à l'architecture moins moderne (imitations de Forms, forte dépendance envers le fournisseur, écosystèmes de type APEX).

Bien des outils automatisent la migration d'Oracle Forms. Parmi eux, seul ORMIT™-OpenJava cible une pile ouverte, sans dépendance envers un fournisseur, et migre les règles sous la seule forme à laquelle on peut astreindre un agent.

La première étape

12

Une migration est un projet, pas un saut dans le vide. Ce chapitre présente qui la mène avec vous, ce qu'elle coûte par rapport au statu quo, et les sept étapes qui vont de la migration au premier agent, avec des chiffres au bout.

Votre équipe

Le succès d'une modernisation dépend de la capacité de votre propre équipe à maintenir le nouveau système à long terme. Chaque projet ORMIT™-OpenJava comprend donc un plan de transfert de compétences qui fait de votre équipe Oracle Forms des responsables confiants de la nouvelle application. Nous travaillons directement avec vos développeurs pour combler les écarts :

Prérequis

Des attentes claires : les bases d'Oracle Forms, et une connaissance pratique de TypeScript ou de Java.

Formation sur mesure

Des séances pratiques qui relient votre connaissance de Forms à de vraies tâches de maintenance sur l'application moderne.

Soutien après la migration

Une documentation détaillée pour les développeurs, et un accompagnement intensif réservé à votre projet pour une mise en production en douceur.

Pas seulement le quoi, mais le comment

Les outils seuls ne livrent pas une transformation; ce sont des équipes attitrées qui le font. Le modèle d'escouades ORMIT™ forme des escouades agiles et multidisciplinaires entièrement consacrées à votre projet, du lancement à la mise en production : un chef technique comme ancrage de l'architecture, des développeurs d'interface et côté serveur, des analystes en assurance qualité pour les tests automatisés et de régression, un responsable de produit qui aligne les priorités sur vos objectifs d'affaires, et un maître de mêlée ou coach agile qui soutient le rythme et lève les obstacles. Dans les environnements complexes, un architecte de solutions se joint à l'escouade. Les escouades travaillent par sprints.

Accompagnement intensif et soutien après la mise en production

Chaque mandat comprend une phase d'accompagnement intensif réservée à votre projet. Pendant les trois premiers mois suivant la mise en production, les experts RENAPS qui ont mené votre modernisation restent disponibles, avec un délai de réponse garanti de 15 minutes, pour aider votre équipe, régler les problèmes et garder le système stable. Votre équipe ne se contente pas de comprendre le fonctionnement du système : elle est outillée pour le faire évoluer en toute confiance.

L'argument financier : migrer plutôt que maintenir

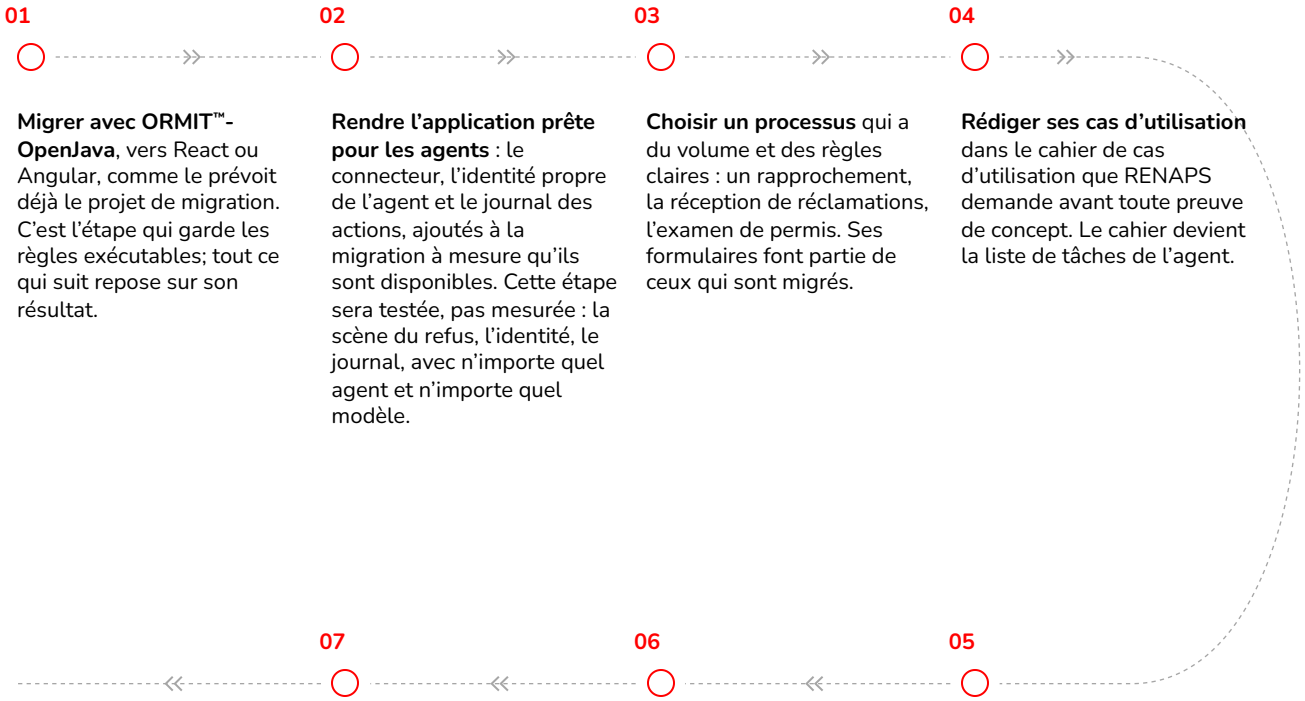
Le chapitre 2 a chiffré le statu quo. Selon l'analyse de RENAPS, une organisation qui conserve un environnement Forms hérité paie plus chaque année, pour trois raisons : les licences Oracle de l'environnement d'exécution Forms et de WebLogic, surtout quand les environnements grandissent; la rareté des développeurs Forms, qui fait grimper les tarifs de soutien; et les contournements sur mesure qu'il faut pour relier Forms aux outils et aux API modernes. Migrer avec ORMIT™-OpenJava réduit le coût total de possession en éliminant l'intergiciel Oracle, en passant à des standards ouverts et en alignant l'architecture sur les compétences courantes des développeurs. Vous obtenez une pile modernisée sans frais d'exécution ni contraintes propriétaires, un recrutement, une intégration et une maintenance à long terme plus faciles, et un développement plus rapide. Pour la plupart des clients, les économies à long terme, avec les gains en performance, en expérience utilisateur et en maintenabilité, font de la migration un gain financier autant qu'une nécessité stratégique.

Une preuve de concept

RENAPS accompagne les projets de preuve de concept avec l'une ou l'autre option d'architecture. Les deux profitent du même cadre d'automatisation, d'une intégration menée par des experts et d'une décennie d'expérience en modernisation d'Oracle Forms. Les sept étapes de la page suivante partent de là.

De la migration au premier agent : sept étapes

La première étape, c'est la migration elle-même, puis le connecteur, puis un processus, pas un programme. Elle réutilise ce que produit déjà chaque migration RENAPS, part donc du matériel dont dispose le client, et se termine par des chiffres.



Ce qu’apporte RENAPS

- ORMIT™-OpenJava et OFCS, qui gardent les règles exécutables et servent déjà des appelants sans écran, et, en préparation, le connecteur qui rendra l’application migrée exploitable par n’importe quel agent.
- ORMIT™-Analyzer pour l’inventaire, ORMIT™-Jasper pour les rapports, ORMIT™-Cortex pour la base de données, et une décennie de modernisation d’Oracle Forms derrière eux.
- Un pipeline de migration bâti comme un harnais d’agent de codage IA, avec les vérifications, les audits et les tests des chapitres 8 et 11 : rien n’arrive à un réviseur avant que les contrôles soient franchis.
- Une escouade attitrée, un plan de transfert de compétences pour votre équipe, un accompagnement intensif après la mise en production, et un soutien complet de l’application migrée, avant et après l’agent.

Prêt à le voir sur vos propres formulaires?

Réservez une démonstration en direct, ou une preuve de concept sur l’un de vos processus, avec l’une ou l’autre architecture. Apportez le processus et ses formulaires, et nous vous montrerons comment ses règles deviennent les garde-fous d’un agent, et à quoi ressemble la première étape mesurée pour votre organisation.

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

Sources

Chiffres et citations vérifiés à la source indiquée; citations traduites par RENAPS.

1. Oracle, Lifetime Support Policy for Oracle Fusion Middleware : dates de support de Forms 12c et 14.1.2. oracle.com
2. Patrick Hamou, The Quest for the Last Oracle Forms Developer. medium.com
3. Ensono, septembre 2026. Ensono research finds 78% of IT leaders see growing value in legacy systems as AI reshapes modernization. ensono.com
4. IDC, décembre 2025. Turning Technical Debt into an AI Enabler. idc.com
5. McKinsey & Company, décembre 2024. AI for IT modernization: Faster, cheaper, better. mckinsey.com
6. Gartner, 22 octobre 2024. Gartner survey reveals that only 48% of digital initiatives meet or exceed their business outcome targets. gartner.com
7. The Batch, DeepLearning.AI, 27 février 2026. Claude Cowork plugins trigger a SaaS stock selloff. deeplearning.ai
8. Fortune, 5 février 2026. OpenAI launches Frontier, an AI agent platform for enterprises. fortune.com
9. Forrester, 11 février 2026. SaaS as we know it is dead: how to survive the SaaS-pocalypse. forrester.com
10. Gartner, 26 août 2025. Gartner predicts 40% of enterprise apps will feature task-specific AI agents by 2026, up from less than 5% in 2025. gartner.com
11. Gartner, 1er juillet 2026. Gartner says US$234 billion in enterprise application software spend is at risk from agentic AI. gartner.com
12. Gartner, 2 avril 2026. Gartner expects most enterprises to abandon assistive AI for outcome-focused workflow by 2028. gartner.com
13. Gartner, 25 juin 2025. Gartner predicts over 40% of agentic AI projects will be canceled by end of 2027. gartner.com
14. Gartner, 20 octobre 2026. Top Strategic Technology Trends for 2027, communiqué de presse et livre numérique (Gene Alvarez). gartner.com; gartner.com/en/articles/top-technology-trends-2027
15. Marc Benioff, discours d'ouverture de Dreamforce 2026, septembre 2026. Transcription. singjupost.com
16. Yuval Noah Harari, Tanner Lecture on Human Values, Oxford, 2026 : « AI has hacked the code of human civilization ». Transcription. singjupost.com
17. CNBC, 3 septembre 2026. Software stocks are up nearly 40% since the "SaaSpocalypse" bottom. cnbc.com; 24/7 Wall St., 13 octobre 2026, l'IGV en hausse de 41 % du 13 avril au 12 octobre. 247wallst.com
18. Oracle, mai 2026. Announcing Oracle APEX 26.1. blogs.oracle.com; 1er octobre 2026, the Modernization Forms to APEX Skill. blogs.oracle.com
19. Anthropic, Regional compliance, lieux de traitement des données de Claude par fournisseur. claude.com
20. Oracle, manuel de référence de Forms Developer 6i : les décomptes du chapitre 5.

Oracle, Java, Oracle Forms, Oracle Reports, Oracle E-Business Suite et APEX sont des marques déposées d'Oracle ou de ses sociétés affiliées. Salesforce et Dreamforce sont des marques de Salesforce, Inc. Claude est une marque d'Anthropic, PBC. Gartner est une marque déposée de Gartner, Inc. ORMIT™ est une marque de commerce de RENAPS. Amazon Bedrock, AWS, Google Cloud Vertex AI, Microsoft Azure, Microsoft Foundry et les autres noms cités sont des marques de leurs propriétaires respectifs.

Contactez-nous dès aujourd'hui.

www.renaps.com

info@renaps.com

1-888-987-3627