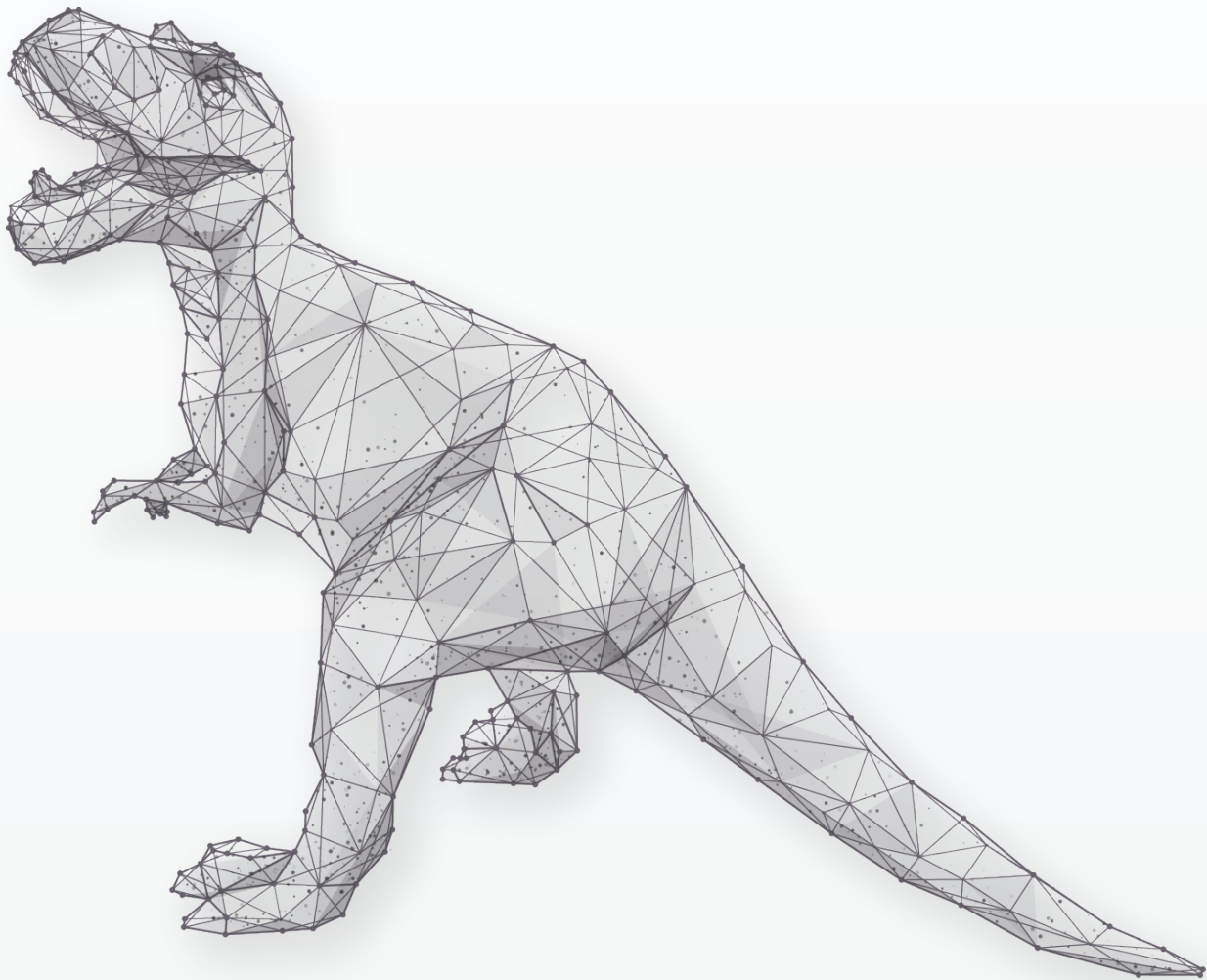


# Oracle Forms to Microservices

Evolutionary Machine to Transform your Legacy



—ONATE

# ABSTRACT

Large enterprises often run business-critical back-office applications on Oracle Forms, particularly for data entry and data retrieval. Oracle Forms as a technology has been around for more than 30 years, now facing EOL owing to some serious limitations on maintenance, cost, and security. Today, many enterprises are migrating their legacy systems – including applications built using Oracle Forms and Reports – to a modern architecture on cloud to gain higher scalability, agility and cost benefits.

This paper examines the challenges of running Oracle Forms, the need for modernization along with the different strategies to modernization and their pros and cons. This paper will be useful to IT and business stakeholders in organizations looking to transform their legacy applications built on Oracle Forms and Reports.

# Table of Contents

<b>Abstract</b>	.....	<b>02</b>
<b>Introduction</b>	.....	<b>04</b>
<b>Time for a Change</b>	.....	<b>05</b>
<b>What are your Options?</b>	.....	<b>06</b>
<b>How do we solve it?</b>	.....	<b>08</b>
<b>Success Story</b>	.....	<b>13</b>
<b>Conclusion</b>	.....	<b>14</b>

# INTRODUCTION

Oracle Forms is a legacy product from the Oracle camp, built on Java Applets. For more than three decades, enterprises have been using Oracle Forms to build UI applications for data-entry. Oracle Forms has been an early RAD (Rapid Application Development) tool for Oracle Databases.



Primarily for its ease of use, Oracle Forms gained a lot of popularity and still enjoys a large customer base. Few more reasons that support the loyalty of customers are:

- It was stable and actively supported by Oracle until late 2019
- It built robust and reliable applications
- It was extremely efficient for heavy duty data-entry operations
- Expert users were very productive on Oracle Forms applications

Companies using Oracle Forms and Reports have made huge investments in development and talent retention over the years. While these investments have satisfied their critical business needs, companies are facing the dilemma of continuing with these near extinction technologies.

In today's fast moving market and increasing customer demands for better user experience, Oracle Forms fall short of providing the necessary solutions to meet these demands. The new age user expects cool dashboards, mobile access, and integration with modern web technologies.



***If you can get today's work done today, but you do it in such a way that you can't possibly get tomorrow's work done tomorrow, then you lose.***

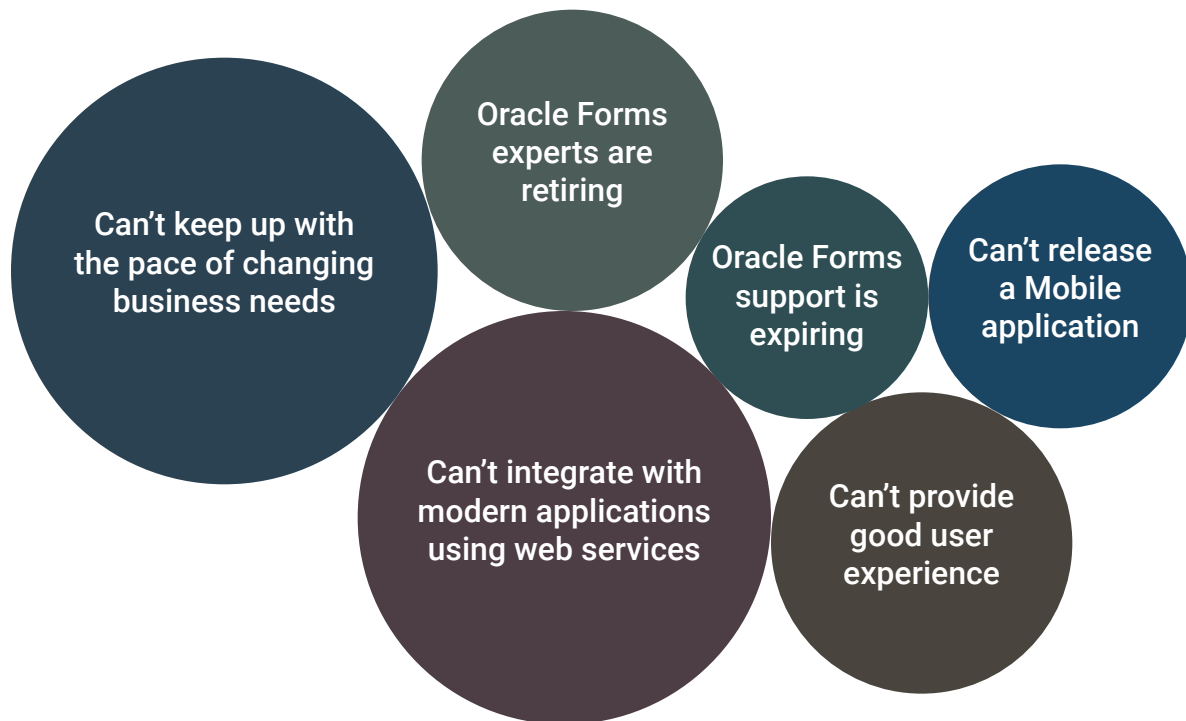
Martin Fowler

# TIME FOR A CHANGE

If you are running Oracle Forms, chances are you are pondering over these questions:

- Why do I need to change anything at all?
- When is a good time for a change?
- Should I take a piecemeal approach or a big bang?
- Or do I just follow what other leaders in my industry are doing?

Let's first deal with the 'why' and the 'when.' If you are experiencing one or more of the following symptoms, your 'why' is answered and your 'when' is **NOW**.



Last but not the least, security vulnerability is a serious risk to business and federal regulations. Legacy systems such as Oracle Forms and Reports are extremely vulnerable and to avoid future problems, they must be modernized to more secured solutions.

# WHAT ARE YOUR OPTIONS?

Let's examine the full spectrum of industry approaches when it comes to modernizing a legacy like Oracle Forms. And then let's weigh them against a few sound criteria that are critical to the future of your business.

- 1 **Reface** – Get a new front-end user interface that's web-enabled and cloud-ready, still working with Oracle Forms in the background.
- 2 **Replace** – Replace your entire Oracle Forms legacy application with COTS (Commercial off the shelf) third-party software.
- 3 **Rewrite** – Rewrite your legacy applications from scratch in a new microservices architecture, or using a low-code application development like Oracle APEX.
- 4 **Convert (Manual)** – Manually convert your legacy Oracle Forms into modern microservices architecture.
- 5 **Convert (Automation)** – Automatically convert your legacy Oracle Forms into modern microservices architecture.

What next? What are the key factors to consider when strategizing the modernization of Oracle Forms? Are you looking for a short-term band-aid solution or one that propels you forward without compromise? Which of these are your compelling business drivers?

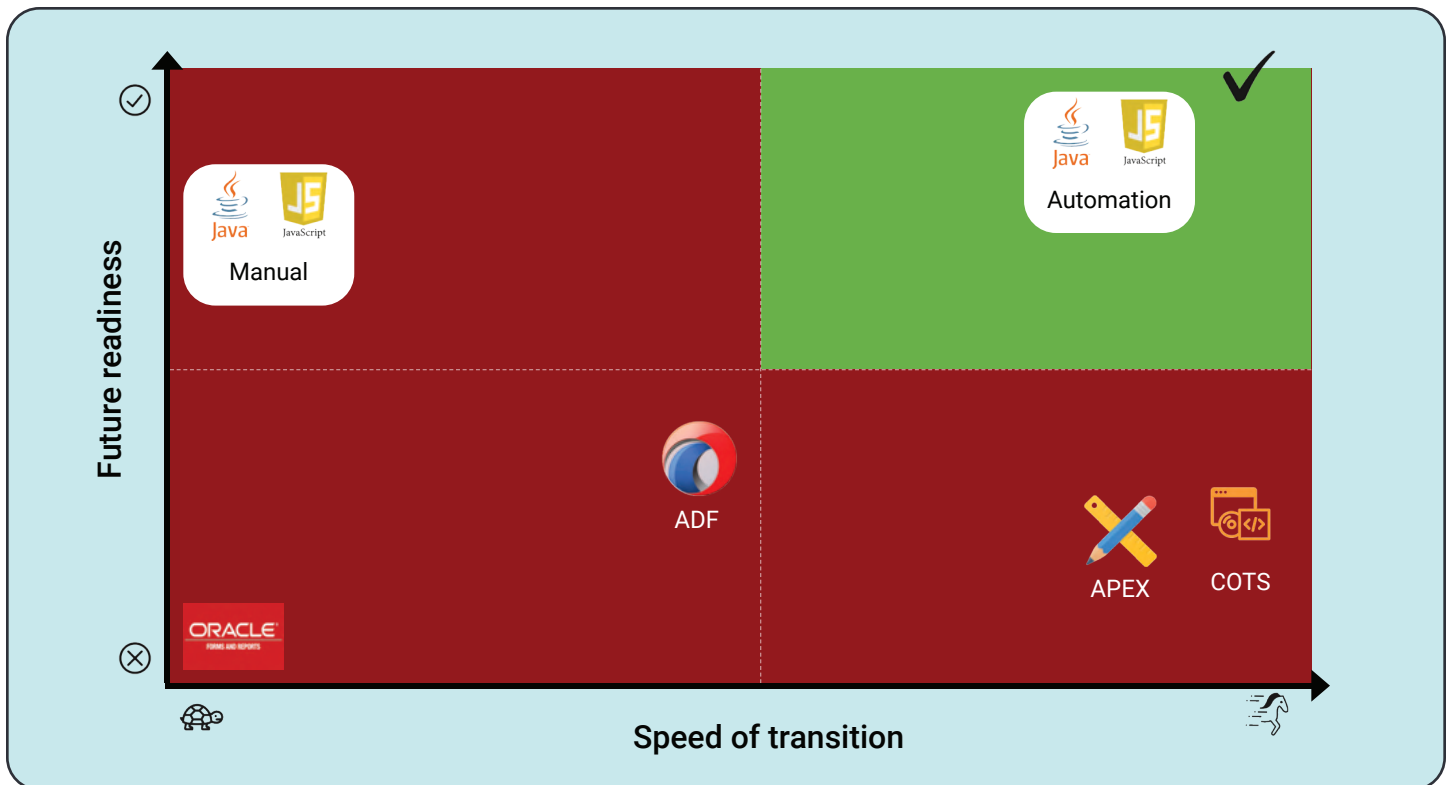
- Cost (modernization, licensing, maintenance)
- Time to market
- Quality of code (and architecture)
- Business parity
- Future-readiness (security, scalability, performance)

Sometimes organizations opt for a "band-aid" solution knowingly, as they need more time to strategize and plan out the more permanent approach. Although the first two methods—Reface and Replace—can get the company to cloud sooner, those methods retain some business challenges, which can limit the future growth. Let's weigh our options against the key business drivers.

	Reface	Replace	Rewrite	Convert (Manual)	Convert (Automation)
Cost effective	★ ★ ★ ★ ★	★ ★ ★ ★ ★	★ ★ ★ ★ ★	★ ★ ★ ★ ★	★ ★ ★ ★ ★
Time to market	★ ★ ★ ★ ★	★ ★ ★ ★ ★	★ ★ ★ ★ ★	★ ★ ★ ★ ★	★ ★ ★ ★ ★
Quality of code	★ ★ ★ ★ ★	★ ★ ★ ★ ★	★ ★ ★ ★ ★	★ ★ ★ ★ ★	★ ★ ★ ★ ★
Business Parity	★ ★ ★ ★ ★	★ ★ ★ ★ ★	★ ★ ★ ★ ★	★ ★ ★ ★ ★	★ ★ ★ ★ ★
Future-readiness	★ ★ ★ ★ ★	★ ★ ★ ★ ★	★ ★ ★ ★ ★	★ ★ ★ ★ ★	★ ★ ★ ★ ★

According to the comparison chart, the last approach scores the most in all departments. But conversion using automation and AI/ML is a delicate process and many organizations have failed, when opting for a complete machine automation without the right process and toolsets.

Another dimension that's useful to plot is the spectrum of Oracle Forms modernization technologies against speed of transition and future-readiness.



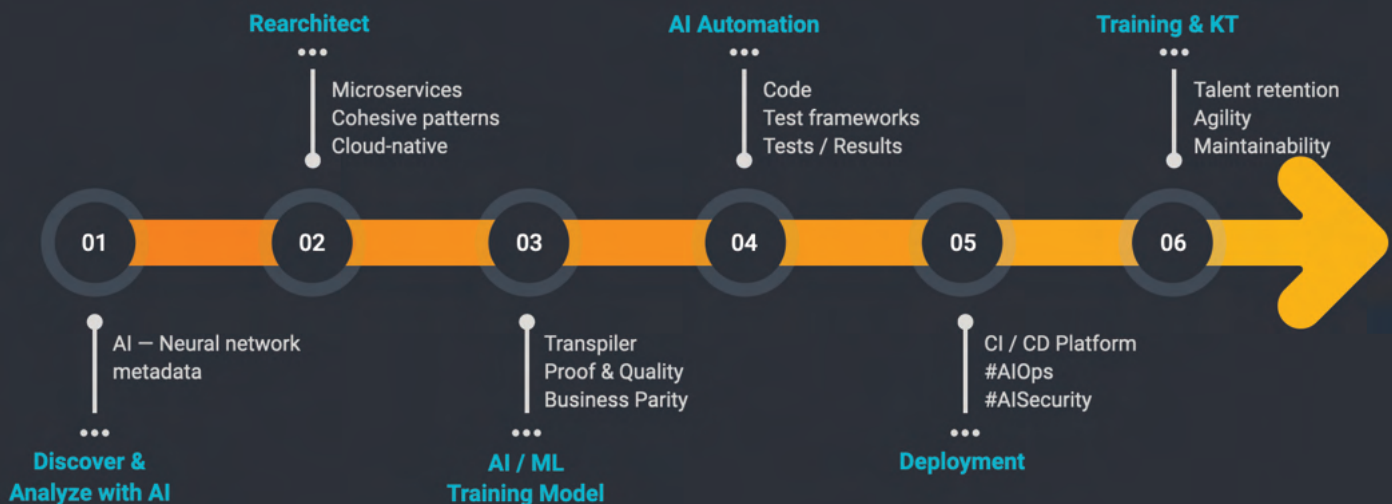
# HOW DO WE SOLVE IT?

We merge the human analysis with machine precision to yield an output that's standard, repeatable, and quick. With our approach, even detection of failure and recovery from it is quicker, as the patterns in failure recognized by our human experts are learnt and fixed in bulk via machine.

Our team is well-versed with many legacy code including Oracle Forms. Over the years, we've built a huge library that increases the accuracy of our machines every time we modernize. Diagnosis-which is the most important part-is now a shared between our experts and machine. Thus the hand-off from human to machine, and iterations that follow is quicker and with a higher degree of business integrity.

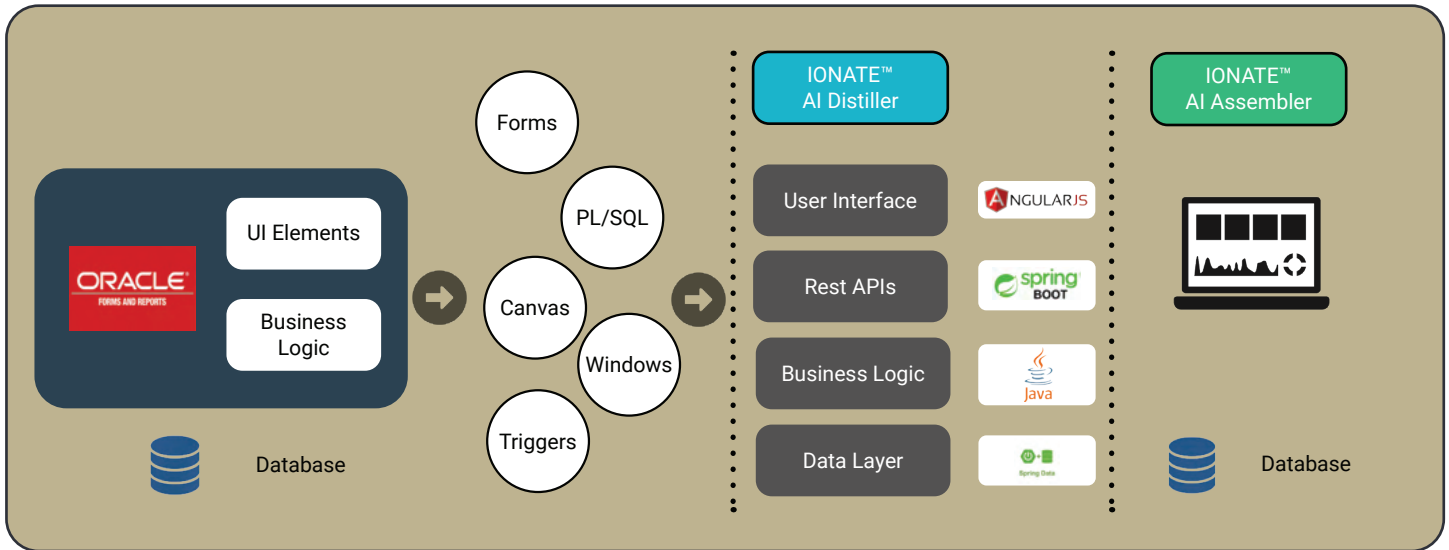
We really ensure 100% Business Parity, which means complete preservation of your business data, business logic, and business processes.

*Since software doesn't age well, Ionate has created a repeatable proprietary process that delivers the goods in a short time-in most cases, under 6 months!  
(Global Partner)*





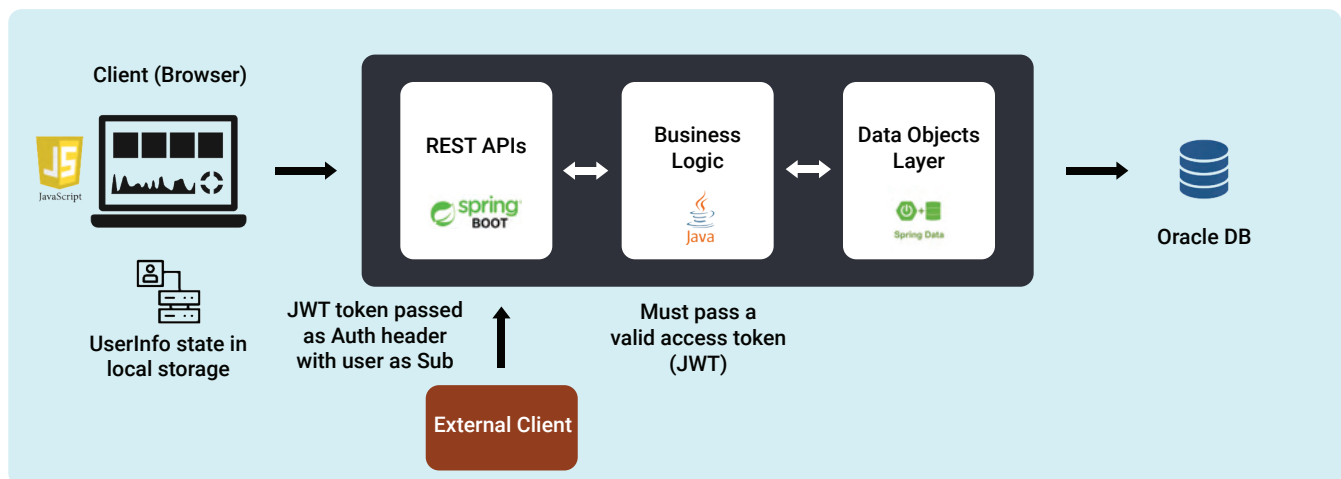
# BUSINESS LOGIC TRANSFORMATION



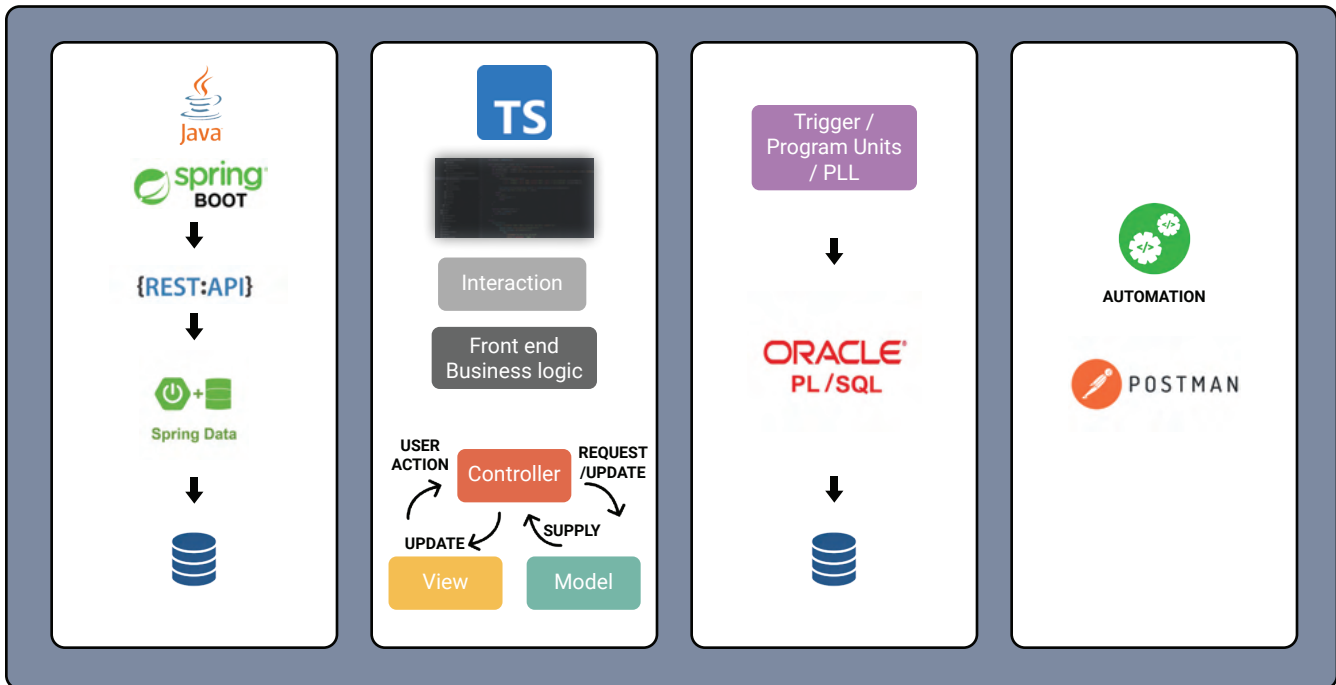
What you see on the left is a breakdown of the high-level components - **Forms, PL/SQL, Canvas, Triggers, and Windows** - inside of an Oracle Forms legacy system. Each of these components need to be transformed into their equivalent cloud-native components.

**IONATE™ DISTILLER** will convert the high-level components into their corresponding layers. Forms and PL/SQL get converted to Data Layer (Hibernate), the business logic gets converted to Java, the triggers get translated into Rest APIs (Spring Boot), and finally the Windows and Canvas get converted into UI (Angular, React). Finally, **IONATE™ ASSEMBLER** will put everything together as the modern cloud-native application.

The final architecture looks like this:



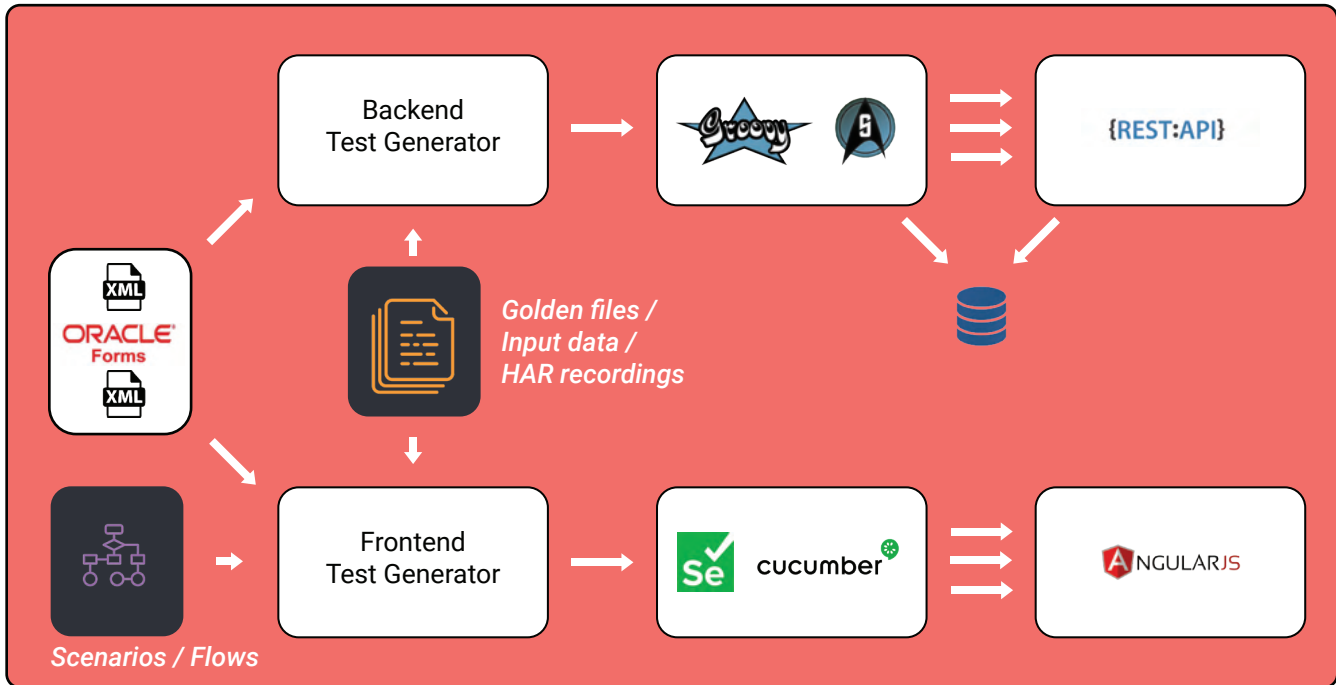
# TRANSPILATION - GENERATED CODE



Transpilation (auto code conversion) happens in four distinct areas.

- **Backend** - The Backend business logic inside Oracle Forms is converted to Java Spring Boot, which connects to REST APIs, in turn connecting to Spring Data, and then the Database
- **Frontend** - The UI interactions and Frontend business logic inside Oracle Forms is converted to TypeScript / Angular components, which uses the MVC pattern to call the REST APIs. View created in TypeScript invokes the Controller based on user action, which calls the REST APIs and updates the Model, which in result updates the View.
- **Triggers, Program Units, PLLs** - Parts of these are split between Backend (Java) and Frontend (TypeScript) components. If these involve Database interactions, they are converted into PL/SQL packets, and pushed to the database.
- **Test Automation** - While the application components are created, Backend (API) tests and Frontend (UI) tests are automatically created for the entire application.

# TEST AUTOMATION



As mentioned in the previous section, the transpilation generates the test code for Backend and Frontend. Let's look into the strategies we use for test automation.

## Oracle Forms SQL / Backend - Springboot + REST APIs

Since the volume of APIs is usually enormous, a structured strategy will not work. We employ a mix of chaotic strategies that converge meaningfully to auto-generate *Groovy* / *Spock* tests.

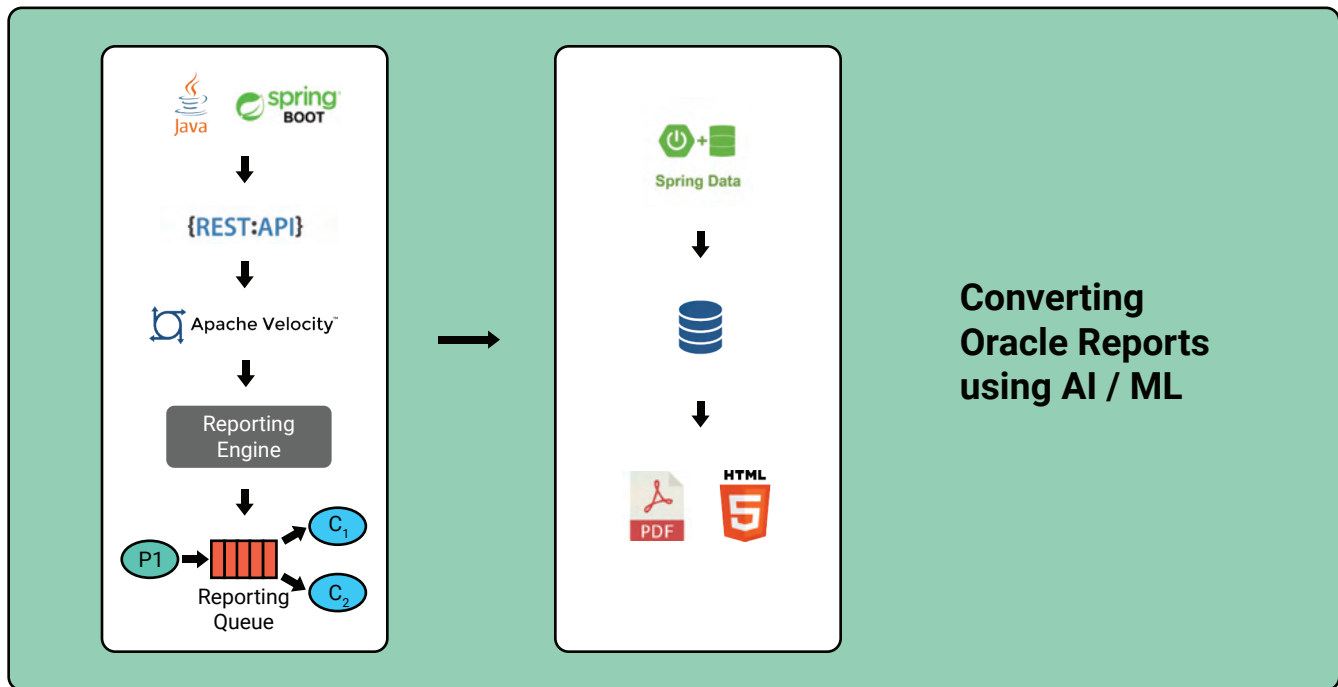
Breaking it down:

- Database Block REST APIs - *Figure out actual inputs based on Table/View*
- Procedure / Function REST APIs - *Use file-based inputs (Golden files)*
- Pure SQL Statement REST APIs - *Randomize the inputs*
- RecordGroup - *Figure out actual inputs based on Table/View*
- Procedure Datablocks - *Use file-based inputs (Golden files)*

## Oracle Forms UI - Angular + TypeScript

For commonly identified Form scenarios, we have ready Gherkin tests. For special case scenarios and data-dependent use cases, we use file-based inputs as in Golden files or HAR recordings, to auto-generate the *Selenium* / *Gherkin* / *Cucumber* tests.

## HOW WE CONVERT ORACLE REPORTS



Oracle Reports is a unique system where each of the reports have unique co-ordinates that map the elements to the screen. Since the mapping is static, it is challenging to create JasperReports or similar.

We solve it as follows:

- For each report, we create a velocity template (PDF, HTML) that has unique items, which calls a Reporting engine to render it using the backend data
- The positioning of elements inside the template is derived from our proprietary **machine learning algorithm**, that analyzes the Oracle Forms and figures out the layout
- We also employ a Reporting queue, as the data generation and retrieval can often take a long time.

To summarize, each report gets queued to the Reporting engine; the reporting engine will get the data, and then create the final HTML and send it back to the UI web socket.

# SUCCESS STORY

The third largest bank in Colombia wanted to improve its overall enterprise efficiency. To continue staying competitive, the bank needed to prioritize IT transformation in key projects such as their leasing application. Built and powered by over **1600 Oracle Forms and 500 Reports**, the system posed some serious business risks.

## PROFILE

Founded in 1972, third largest bank in Columbia with more than 12,000 employees and \$3.05 billion in revenue.

## CHALLENGE

Needed to discontinue legacy and its inefficiencies, and move to micro services, while retaining 100% business parity.




## SOLUTION

Accelerate transformation of Oracle Forms and Reports with IONATE™ APPDATE, into micro services using AI / ML.

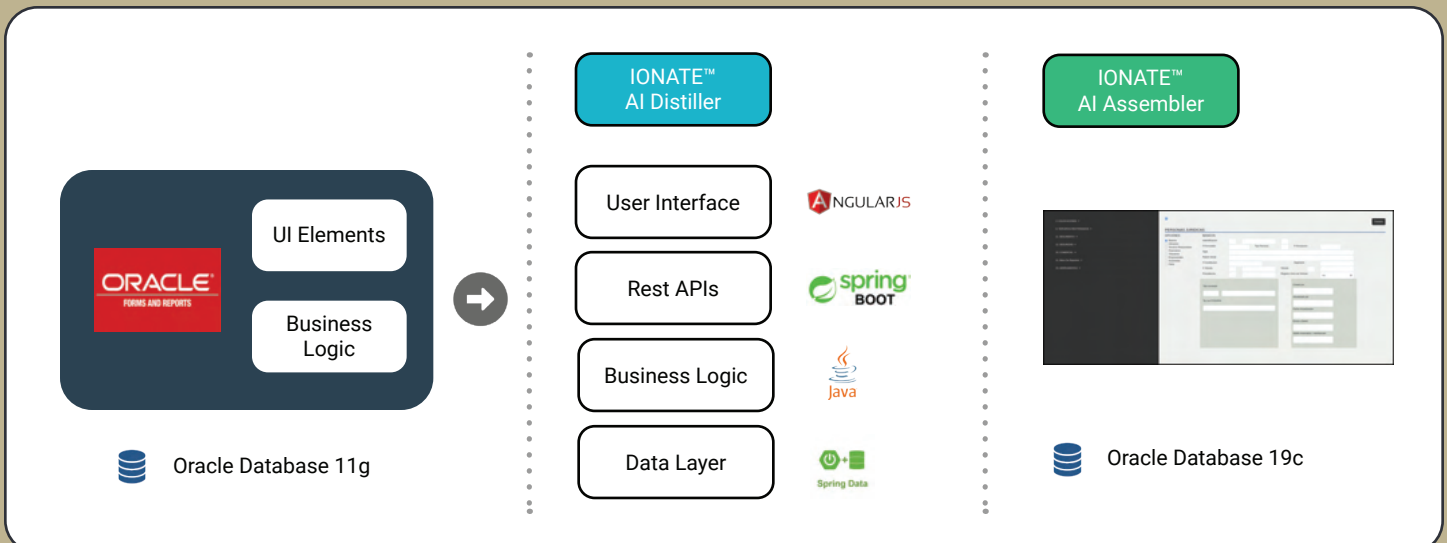
## HIGHLIGHTS

According to SCC generated report for a sub-project, it would have taken 31 engineers a period of 36 months to hand-code what IONATE™ APPDATE did in under 6 months.

## BENEFITS

-  Cost savings
-  Future readiness
-  Faster time to market

**Legacy to modern stack conversion is orchestrated by Ionate's proprietary technologies.**



# CONCLUSION

Oracle Forms are called 'legacy' for a good reason - they stood the test of time. But as almost everything comes with an expiry date, these systems running on overdrive - need an overhaul, a metamorphosis, a new life. Something that could be the next legacy and stand the test of time; not something that is temporary and will phase out in a short period of time.

Your business needs and legacy challenges will dictate the strategy and path to modernization. The shortest path to cloud may seem attractive and almost a commonsensical way forward. But the approaches - **Reface** and **Replace** - promising the shortest path bring a serious threat to the future readiness of the solution, to adapt to your business requirements. **Rewrite** and **Convert (Manual)** approaches can bring future readiness, but it comes with a compromise of longer time to market and high production cost.

In our experience, the hybrid approach of human analysis and training of the machine combined with **Convert (Automation)**, satisfies most criteria that defines a successful, true modernization.







## About Ionate

Global HQ | San Francisco  
[www.ionate.io](http://www.ionate.io) • [sales@ionate.io](mailto:sales@ionate.io)

Ionate provides an endgame (turnkey solution) for Digital Transformation with its proprietary AI / ML platform and products. Enterprises can not only modernize their legacy systems, but also operate on high- performing infrastructure and scale their business, with security, predictive forecasting, and data protection.

Ionate is a registered trademark of Ionate, Inc. All other trademarks or service marks are the property of their respective holders and are hereby acknowledged.

©2021 Ionate, Inc. All rights reserved.