CASE STUDY: How Oregon PERS Implemented Employee Pension Stability Account (EPSA)

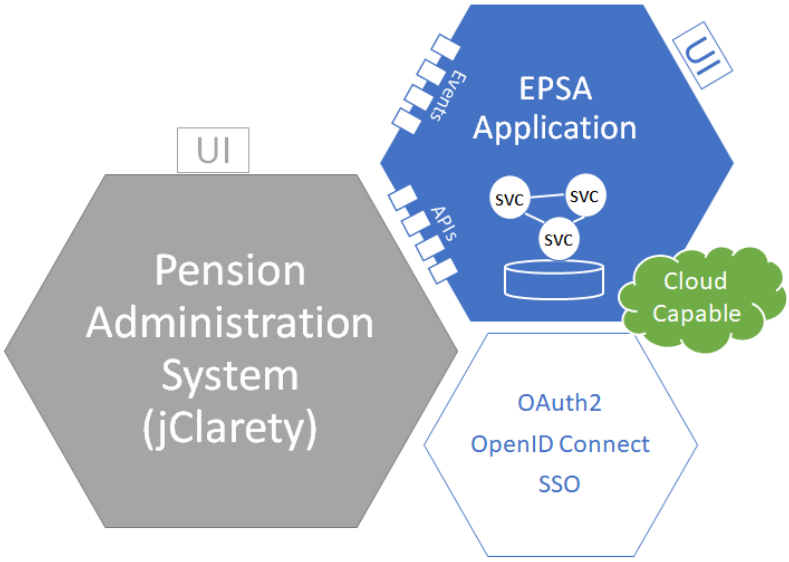
When the State Senate Bill 1049 passed, Oregon Public Employees' Retirement System (PERS) was mandated to redirect a portion of member contributions to new Employee Pension Stability Account (EPSA) to stabilize the PERS funding for future pension benefit of PERS retirees.

The deadline to implement PERS systems for SB1049 was very short--there was no time to re-architect, internal resources were stretched, and the current PAS (Pension Administration System) is a monolithic application with complex functionality developed around 18 years ago.

The Solution

Provaliant's implementation strategy is simple:

- **Vision**: Establish the Solution Architecture which has the flexibility to meet current demands and the capability to scale/adapt/integrate with forward looking technology
- **Key for Success**: Build a viable partial version of a software solution, then expand through successive iterations
- **Flexibility**: Responding to change is more important than conforming
- **Focus**: Strip away non-value-adding activities
- **Priority**: If a feature will not be used immediately, spend effort elsewhere
- **Deliver**: Agility in delivering functionality to on-board users to test features they can use



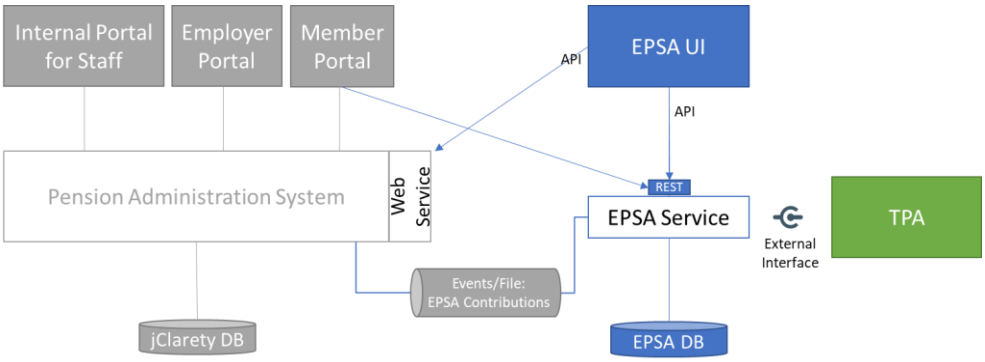
Based on this architecture depicted in the diagram above, the EPSA system is a separate well encapsulated software component, that is well-recognized by business users for all EPSA related business functions. It uses Spring Boot, an open-source Java-based framework to create RESTful APIs which are capable of being deployed anywhere (on-prem/cloud/hybrid), or switch deployment targets anytime. It is lightweight and capable of being deployed with or without middleware with no compromise in scalability or availability.

The beauty of this is, requirement for additional infrastructure is very low to none--it can coexist side-by-side with existing infrastructure environments, deployed to serverless platforms, containers, VMs or to any deployment choices. With the DevOps process fully integrated with Code Quality tools, Static Application Security Testing (SAST) tool, automated unit tests (for all components) and with Continuous Integration/Continuous Delivery (CI/CD), the codebase is always high-quality, up to date with security patches, ready for deployment to a test environment.

For business users, the user interface combines capabilities from the PAS and EPSA systems, providing a seamless experience, under-guarded with a robust security based on OAuth 2, OpenID Connect and Single Sign-on. In addition, being an independent web component, EPSA UI takes advantage of extensive modern browser standards features for rich user experience.

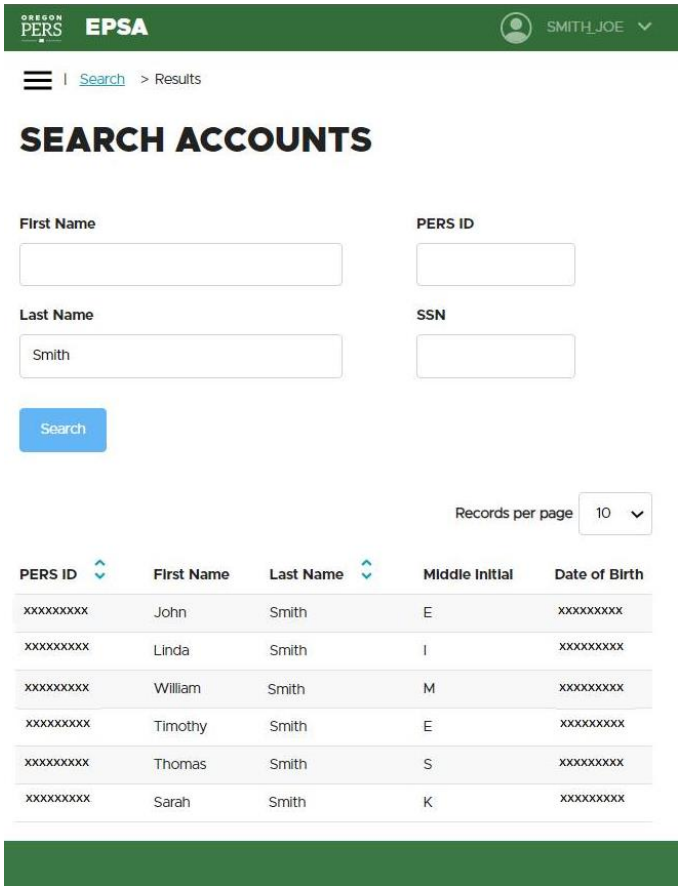
Technical Detail

EPSA Service layer is built on Open API Specifications, providing programming language-agnostic interfaces, allowing API's auto discoverable interfaces to be configurable, eliminating the need to understand the source code and dependency on the documentation. It supports a broad spectrum of standard interfaces, such as secure files, to events, and non-blocking streams.



REST APIs inherent statelessness, coupled with stateless user interface, provides the ability to scale up and down based on demand. In subscription modelled platforms, this could mean efficient use of resources with no downside to business.

A business user focused UI designed with HTML5 and CSS, with responsive-design approach, means it works on all modern browsers and devices. A smart use of tabs, optimal use of screen real estate, and consistency with placement of controls, provides feedback to users for their actions.



The Benefits

1. **Minimum to no changes to existing PAS** system to implement Senate Bill (new EPSA system).
2. **Short development lifecycle** with incremental releases.
3. **Cost Savings**: Easy to build and maintain in-house, saving costs. Easy to upgrade the framework as new versions are released.
4. **Builds Business Capability**: EPSA is aligned to business capabilities, allowing the technical team to focus on a specific service versus the entire pension system. It is easier to customize to the business needs because working on individual modules allows teams to focus on business capabilities rather than technologies.
5. **Secure**: Strict adherence to NIST & State Cyber Security framework is employed throughout the software lifecycle to reduce the security risks. Static Application Security Testing and Dynamic Application Security Testing are incorporated in the DevOps process, and automated with CI/CD pipelines. Penetration testing is recommended for major releases.
6. **Integration**: Enterprise Application Integration (EAI) Capabilities can be implemented through prebuilt connectors, B2B/EDI, Message Queue, APIs or traditional SFTP. The Spring framework supports capabilities without having to build to interfaces with any third party.
7. **Modern Architecture**: Composable APIs that work with traditional systems and modern applications.