|  |
| --- |
|  |
| Red VUCE |
| Documento de Arquitectura |
|  |
|  |
| **04/11/2013** |

|  |
| --- |
|  |

Contenido

[1. Introducción 5](#_Toc373951027)

[1.1 Objetivos 5](#_Toc373951028)

[1.2 Audiencia 6](#_Toc373951029)

[2. Arquitectura SOA 7](#_Toc373951030)

[3. Marco Conceptual 9](#_Toc373951031)

[3.1 Interacción Punto a Punto 9](#_Toc373951032)

[3.2 Modelo de Interoperabilidad VUCE 10](#_Toc373951033)

[3.3 Diagrama de Contexto Red VUCE 12](#_Toc373951034)

[4. Arquitectura general 13](#_Toc373951035)

[4.1 Enterprise Service Bus 13](#_Toc373951036)

[4.2 Spring-MVC y HTML5 14](#_Toc373951037)

[4.3 Web Services 14](#_Toc373951038)

[4.4 Servicios EJB 14](#_Toc373951039)

[4.5 Data Base 14](#_Toc373951040)

[5. Arquitectura detalle 15](#_Toc373951041)

[5.1 Enterprise Service Bus 15](#_Toc373951042)

[5.2 Tecnología JEE 16](#_Toc373951043)

[5.2.1 DAO 17](#_Toc373951044)

[5.2.2 Transaccionalidad 17](#_Toc373951045)

[5.2.3 Lógica de negocio 18](#_Toc373951046)

[5.2.4 Persistencia en base de datos 19](#_Toc373951047)

[5.2.5 Integración 19](#_Toc373951048)

[5.2.6 Contrato/Implementación 19](#_Toc373951049)

[5.2.7 Capa WEB 20](#_Toc373951050)

[5.3 Gestión de errores 21](#_Toc373951051)

[5.4 Base de datos 21](#_Toc373951052)

[5.4.1 Llaves primarias 21](#_Toc373951053)

[6. Web Services 22](#_Toc373951054)

[7. Logging 23](#_Toc373951055)

[8. Patrones de diseño 23](#_Toc373951056)

[8.1 Interfaz-implementación 23](#_Toc373951057)

[8.2 Session façade 23](#_Toc373951058)

[8.3 Business delegate 24](#_Toc373951059)

[8.4 Service locator 24](#_Toc373951060)

[8.5 Factory 24](#_Toc373951061)

[8.6 Data access object 24](#_Toc373951062)

[8.7 Value object y data transfer object 24](#_Toc373951063)

[8.8 Front controller 24](#_Toc373951064)

[9. Vista Lógica 25](#_Toc373951065)

[1.1. Diagrama 25](#_Toc373951066)

[10. Vista de Despliegue 28](#_Toc373951067)

[1.2. Diagrama 28](#_Toc373951068)

[11. Consideraciones Generales 29](#_Toc373951069)

[1.3. Herramientas 29](#_Toc373951070)

[1.4. Desarrollo 29](#_Toc373951071)

[1.4.1. Acceso a datos 29](#_Toc373951072)

[1.4.2. Lógica Empresarial 30](#_Toc373951073)

[1.4.1. Presentación 30](#_Toc373951074)

[1.5. Pruebas 31](#_Toc373951075)

[1.1. Herramienta de Desarrollo 31](#_Toc373951076)

[1.2. Seguridad 32](#_Toc373951077)

[1.3. Administración de funcionamiento 33](#_Toc373951078)

[1.3.1. Hardware 33](#_Toc373951079)

[1.3.1. Software 33](#_Toc373951080)

[1.4. Comunicación 34](#_Toc373951081)

**Ilustraciones**

[Ilustración 1: Diagrama de interacción Punto a Punto 7](#_Toc373935524)

[Ilustración 2, Modelo de Interoperabilidad VUCE 8](#_Toc373935525)

[Ilustración 3, Diagrama de Contexto Red VUCE 10](#_Toc373935526)

[Ilustración 4: Diagrama de Arquitectura General. 11](#_Toc373935527)

[Ilustración 5: Diagrama de Arquitectura Física. 11](#_Toc373935528)

[Ilustración 6: Diagrama de Arquitectura de Detalle. 13](#_Toc373935529)

[Ilustración 7: Diagrama de componentes JEE 14](#_Toc373935530)

[Ilustración 8: Diagrama de componentes EJB 16](#_Toc373935531)

[Ilustración 9: Diagrama de componentes de lógica de negocio 16](#_Toc373935532)

[Ilustración 10: Diagrama de entidades 17](#_Toc373935533)

[Ilustración 11: Diagrama de Componentes Vista Lógica 23](#_Toc373935534)

[Ilustración 12: Diagrama de Vista Lógica Detallada. 24](#_Toc373935535)

[Ilustración 13: Diagrama de Vista Despliegue 26](#_Toc373935536)

**Tablas**

[Tabla 1, Descripción de interacciones de la Red VUCE 10](#_Toc373935599)

[Tabla 2, Tabla de Componentes JEE. 15](#_Toc373935600)

[Tabla 3, Descripción de componentes del Vista Lógica 25](#_Toc373935601)

[Tabla 4, Descripción de Elementos de Vista Lógica Detallada. 26](#_Toc373935602)

[Tabla 5, Descripción de los Nodos de Despliegue 27](#_Toc373935603)

[Tabla 6, Herramientas a utilizar en el Sistema 28](#_Toc373935604)

[Tabla 7, Herramientas/Componentes para el Acceso a Datos 28](#_Toc373935605)

[Tabla 8, Herramientas/Componentes para la Lógica Empresarial 29](#_Toc373935606)

[Tabla 9, Herramientas/Componentes para la Capa de Presentación 29](#_Toc373935607)

[Tabla 10, Herramientas/Componentes para el Proceso de Pruebas 30](#_Toc373935608)

[Tabla 11, Herramientas de Desarrollo 30](#_Toc373935609)

[Tabla 12, Herramientas y Componentes de Seguridad 31](#_Toc373935610)

[Tabla 13, Herramientas/Componentes para el Acceso a Datos 32](#_Toc373935611)

[Tabla 14, Herramientas/Requerimientos de Software Base 32](#_Toc373935612)

[Tabla 15, Componentes para la Comunicación 33](#_Toc373935613)

# Introducción

El presente documento corresponde a la propuesta de los detalles de la arquitectura de diseño técnico general VUCE, centrándose principalmente en los componentes que se utilizaran para el desarrollo de la solución.

Como parte de los objetivos principales de la Red VUCE, de promover el intercambio de información entre los países de la región para ejecución de procesos de comercio exterior, se incluyen tres consultorías Normativa, Armonización y Técnica que apoyen en el cumplimiento de este objetivo.

Para satisfacer el objetivo principal, el presente documento tiene como objetivo secundario, proponer una solución de arquitectura acorde a la necesidad de intercambio de información entre países de la red, para el proceso de comercio exterior.

La arquitectura contempla la utilización de los estándares orientados a servicios, representadas por SOA (Arquitectura Orientada a Servicios), marco de trabajo para apoyar a las organizaciones a integrar sus negocios mediante la armonización de datos y lógica de sus plataformas.

En el caso de la Red VUCE, esta arquitectura cumple con las necesidades de integración de varios países con definiciones y normativas independientes, pero con un proceso de comercio en común.

Para este proceso de comercio exterior la propuesta se basa en tecnología que opera bajo el marco Java Enterprise Edition, el cual cuenta con una gran oferta de herramientas y de utilidades, propietarias y opensource, para dar implementar una solución acorde a las necesidades de la Red.

## Objetivos

Como objetivos de la consultoría y del presente documento se consideran los siguientes:

* Proponer una arquitectura SOA.
* Utilización de protocolos estándares a cualquier plataforma tecnológica. (Definido en documento de protocolos)
* Proponer el uso de herramientas de clase mundial que permitan asegurar la interoperabilidad entre los miembros de la red.
* Definir una arquitectura lógica para el desarrollo de la solución.
* Proponer una arquitectura física para implementar esta solución.

## Audiencia

La audiencia objetivo del presente documento corresponde a:

* El equipo de arquitectura, que utiliza el documento para revisión de los artefactos o componentes construidos, velando por que se cumplan las definiciones establecidas en el presente documento.
* El equipo desarrollo, que encontrarán en este documento el detalle de cómo se debe construir los bloques básicos para la generación de gran parte de los elementos del sistema.
* El equipo mantenimiento, que encontrarán en este documento el lineamiento seguido en la construcción de los artefactos del sistema a ser mantenidos.

# Arquitectura SOA

Una arquitectura orientada a servicios (SOA), contempla un concepto de utilización de servicios para dar cumplimiento a las necesidades de negocio. Un software se diseña mediante una arquitectura SOA, debido a que en sus requerimientos se requiere un intercambio de información con diversas entidades.

En el caso de la Red VUCE, requiere un fuerte intercambio de información, entre los países miembros de la red, específicamente para solucionar las necesidades del proceso de comercio exterior.

Por este motivo se ha definido que cada país, deberá integrarse con la plataforma de interoperación de su país destino, mediante una conexión punto a punto.

SOA, posee varias ventajas que la hacen una excelente alternativa para sistemas que necesitan interoperación:

* Mejora en los tiempos de realización de cambios en procesos. Ayudando a mejorar la agilidad y flexibilidad de las organizaciones.
* Facilidad para evolucionar a modelos de negocios basados en interoperar con terceros como parte de una transacción.
* Facilidad para abordar modelos de negocios basados en colaboración con otras entidades organizacionales. Permitiendo alinear y acercar las áreas de tecnología y negocio.
* Capacidad de reemplazar elementos de la capa aplicativa SOA, sin entorpecer el proceso de negocio (caja negra). Aisla mejor a los sistemas frente a los cambios generados por otras partes de la organización.
* Permite la simplificación del desarrollo de soluciones mediante la utilización de estándares de la industria y capacidades comunes de industrialización.
* Facilidad para la integración independiente y personalización masiva de las tecnologías de la información.
* Facilidad para reutilizar servicios.
* Posibilidad de orquestación de servicios.

Inicialmente las organizaciones abordaban la implantación de SOA solamente como un elemento de integración, no utilizando sus principales ventajas. Actualmente, este escenario está cambiando, las herramientas e infraestructura SOA están alcanzando la madurez necesaria y cada vez más, las organizaciones entienden que el valor de SOA se encuentra en la orientación a procesos.

Las organizaciones deben ser capaces de crear nuevos productos y servicios para clientes cada vez más exigentes. El aumento de la colaboración con los clientes y proveedores, y la mayor capacidad para interpretar los datos que proporcionan a las organizaciones con los medios necesarios para interpretar los cambios del mercado de una forma más precisa y rápida. Lo importante es conseguir que sus procesos de negocio sean capaces de ser adaptados al mismo ritmo de las necesidades del mercado. Este dinamismo exige un nuevo conjunto de capacidades tecnológicas que permitan adaptar rápidamente los sistemas informáticos.

La arquitectura SOA contempla los procesos de negocio y las tecnologías de la información. Además, el alto nivel de desacoplamiento e interoperabilidad que proporciona permite un alto grado de reutilización (interno y externo) y de parametrización. Todo ello redunda en una mayor facilidad y flexibilidad para adaptar y mejorar los procesos de las organizaciones según los cambios de prioridad del negocio.

El desacoplamiento de una plataforma es relevante para garantizar su crecimiento, por esta razón la arquitectura SOA desacopla los tres componentes de una aplicación: presentación, orquestación de procesos y lógica de negocio, a la vez se estandariza la comunicación entre cada una de las capas. Además, las organizaciones se pueden focalizar en los componentes de mayor valor como los procesos y externalizar si es necesario el resto de componentes.

El ordenamiento en módulos más pequeños o servicios, reduce notablemente el impacto de los cambios. A través de la creación de un modelo flexible que pueda reconfigurarse en función de las necesidades del negocio, la arquitectura SOA reutiliza, los distintos sistemas tecnológicos actuales, encapsulándolos en servicios que pueden ser utilizados por diferentes aplicaciones y procesos.

# Marco Conceptual

Con el objetivo de entender las necesidades de la Red VUCE, se define el modelo VUCE, que se componen de un conjunto de macroprocesos considerados como transversales en los procesos de comercio exterior de los países de la región, los cuales se toman como base para definir los puntos de interoperabilidad que son la base para el intercambio electrónico de información entre los países miembros.

Los macroprocesos definidos son los siguientes:

* Proceso de exportación, importación.
* Proceso de ingreso y revisión zona primaria o zona aduanera.

Cada proceso define interacciones entre los países que interactúan en un proceso de exportación e importación, origen y destino respectivamente.

En la siguiente ilustración se pueden visualizar los macroprocesos mencionados y su interacción.

## Interacción Punto a Punto

La interacción en entre los países miembros de la red VUCE, se implementara en un sistema punto a punto. Esto implica que cada país deberá establecer un canal de comunicación con los países de la red o con el país que deba iniciar un proceso de comercio exterior.

Para estandarizar la forma en que los países se comunican, se propone la utilización de mecanismos y protocolos estándar, para que el proceso de integración sea fluido y amigable.

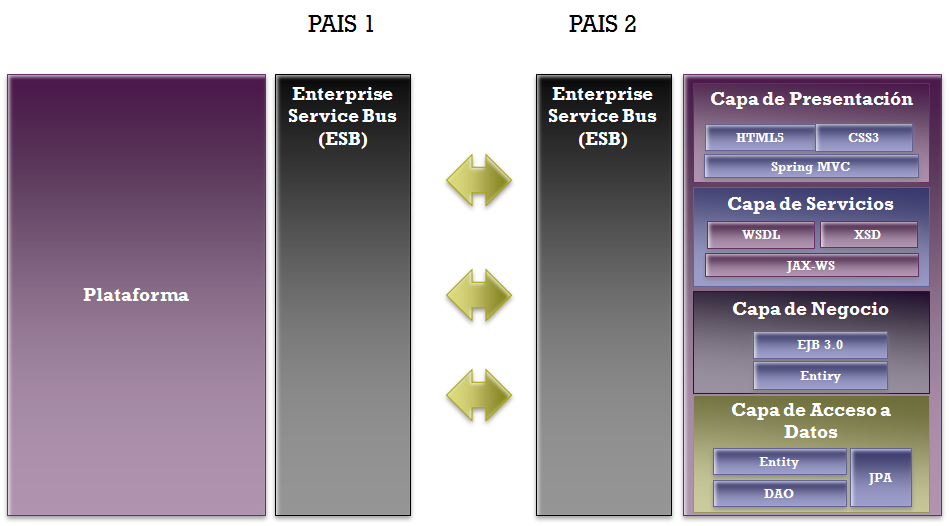


Ilustración : Diagrama de interacción Punto a Punto

## Modelo de Interoperabilidad VUCE

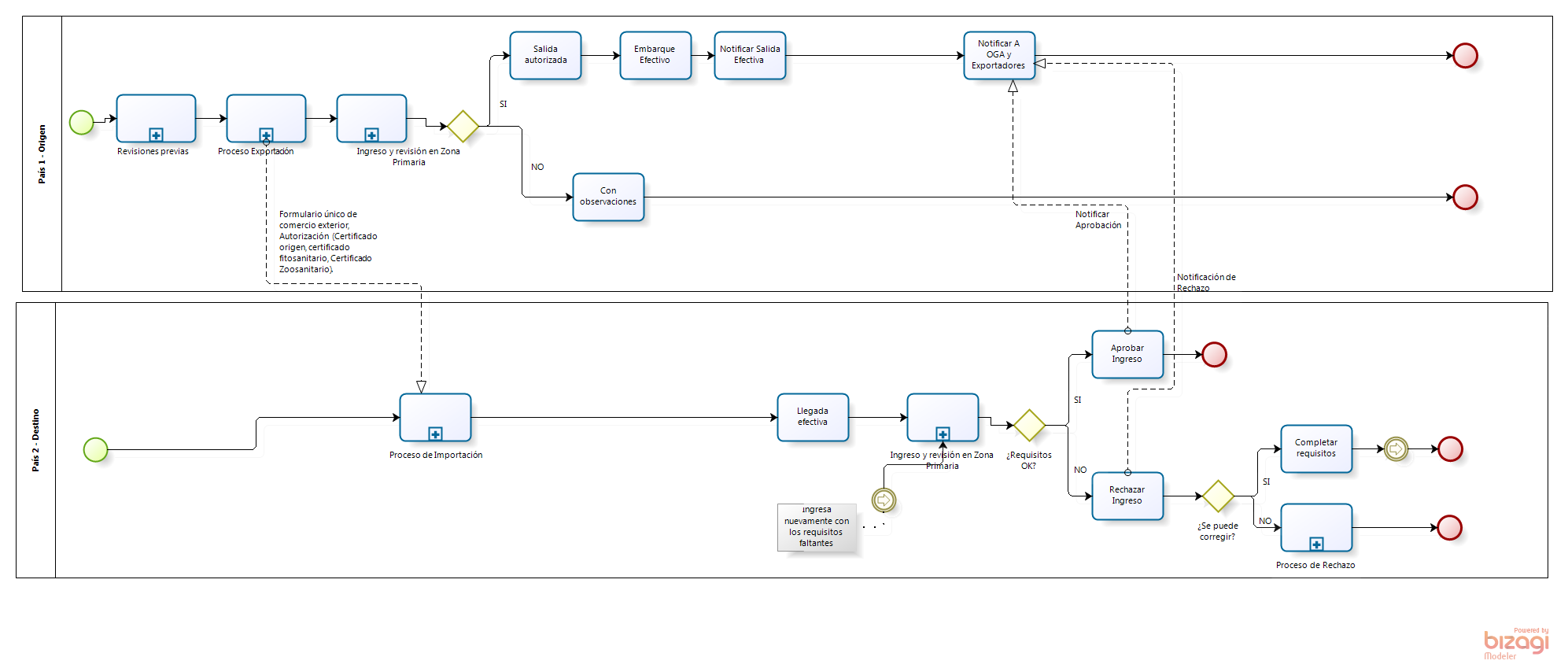


Ilustración , Modelo de Interoperabilidad VUCE

Los puntos de interacción mencionados en el diagrama de contexto son dos, inicio de proceso, aprobación y rechazo:

| **Interoperabilidad** | **Desde** | **Hacia** | **Información** |
| --- | --- | --- | --- |
| Notificación inicio de proceso | Proceso de exportación | Proceso de importación | Formulario único de comercio exterior documentación requerida. |
| Notificación aprobación de ingreso y distribución | Proceso Ingreso y revisión zona primaria | Proceso exportación | Identificador del formulario único con fecha, hora y folio de autorización. |
| Notificación rechazo con observaciones | Proceso Ingreso y revisión zona primaria | Proceso exportación | Identificador del formulario único con fecha, hora, folio de rechazo y listado de observaciones. |

Tabla , Descripción de interacciones de la Red VUCE

El segundo punto de notificación se divide en dos tipos de respuesta, la aprobación del proceso de exportación, según la información enviada y el rechazo con sus respectivas observaciones.

## Diagrama de Contexto Red VUCE

En el siguiente diagrama se presenta el modelo de contexto de las interacciones de los países de origen y destino de la Red VUCE.

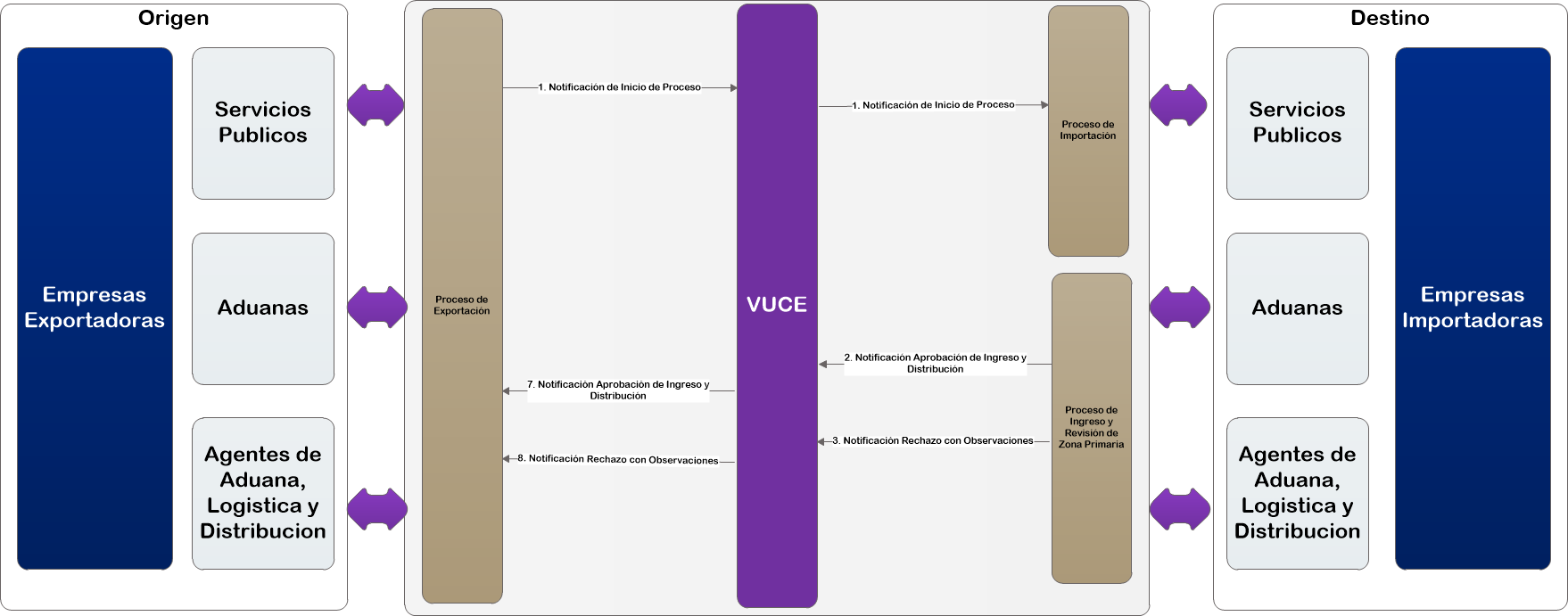


Ilustración , Diagrama de Contexto Red VUCE

# Arquitectura general

La arquitectura general propuesta para la plataforma de interoperación, de la Red VUCE, contempla los elementos expuestos en las siguientes figuras:

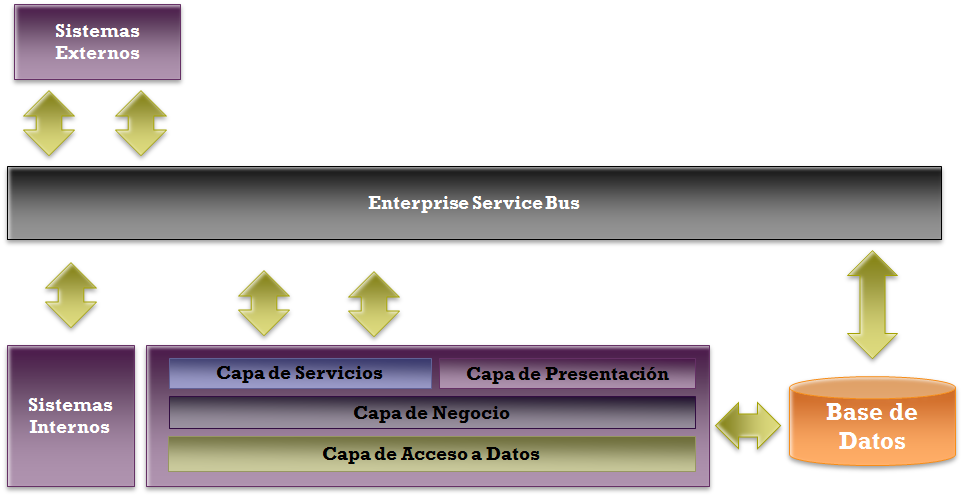


Ilustración : Diagrama de Arquitectura General.

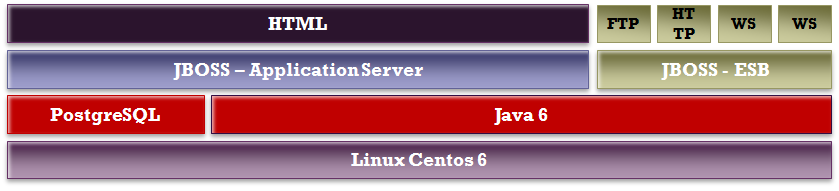


Ilustración : Diagrama de Arquitectura Física.

## Enterprise Service Bus

El bus de servicios ESB se utilizará para publicar los Web Services del sistema y procesos. Además le corresponderá realizar orquestaciones de servicios. En la capa ESB se realizarán las orquestaciones necesarias para la correcta validación de datos, previas al inicio de todo proceso o llamada a servicios.

## Spring-MVC y HTML5

La capa de presentación del sistema se implementará utilizando Spring MVC y HTML5. Esta capa se integrará con la capa de negocio vía servicios EJB y, en caso de ya existir publicados, se primará el uso de Web Services existentes.

## Web Services

La lógica de negocio Java se expondrá a la capa SOA mediante WebServices implementados con JAX-WS. Estos Web Services se publicarán en el ESB para su consumo y gobernanza. Se considera sólo exponer aquellos servicios requeridos y que se consideren necesarios, no se expondrán por defecto lógicas de negocio internas.

## Servicios EJB

La lógica de negocio transaccional se implementará utilizando una fachada EJB Session Stateless. Para el mapeo relacional a Java se utilizará JPA junto con Entity Beans.

## Data Base

La persistencia de datos se implementará en PostgreSQL.

# Arquitectura detalle

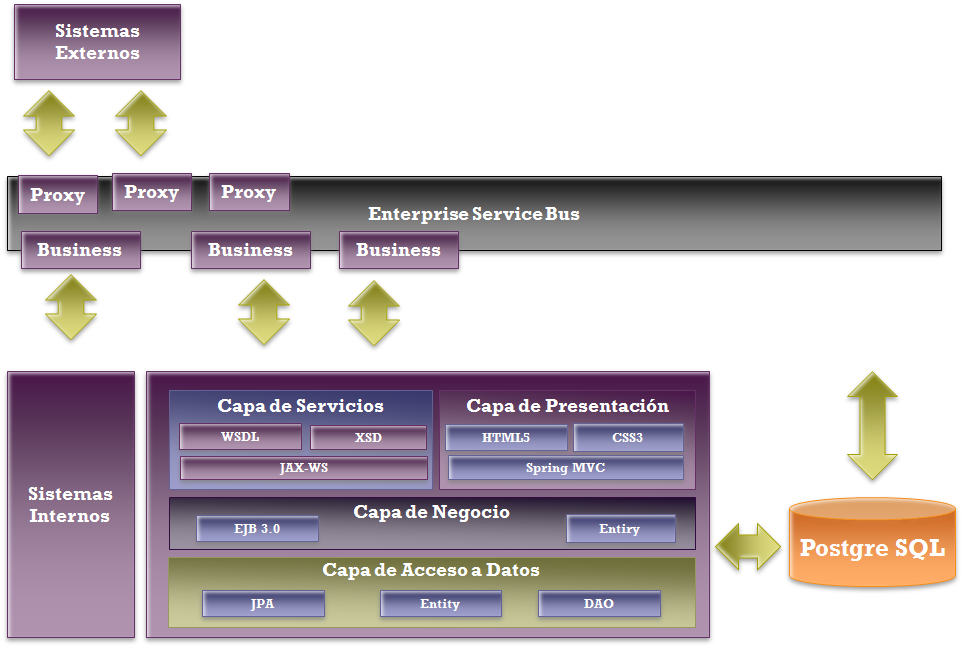


Ilustración : Diagrama de Arquitectura de Detalle.

## Enterprise Service Bus

En el ESB se publicarán los WebServices del sistema y los procesos de orquestación. Se definen las siguientes reglas:

* Como nomenclatura en ESB de WSDLs, se tomará el mismo nombre del nombre del WSDL que ya cuenta con el sufijo ‘WSDL’ (ej: ValidarEjercicioWSDL). Los WSDL deberán tener una ruta relativa a los XSDs que importe, la cual será ‘../xsd’). La importación del WSDL en ESB realiza la importación automática de sus XSD asociados.
* Como nomenclatura en ESB de componentes Business, se tomará el nombre del WebService o proceso (ej: ValidarEjercicio) agregándole el sufijo ‘Business’ (ej: ValidarEjercicioBusiness).
* Como nomenclatura en ESB de componentes proxy, se tomará el nombre del WebService o proceso (ej: ValidarEjercicio) Agregándole el sufijo ‘Proxy’.
* Para la publicación directa de Web Services se implementará el business y el proxy correspondiente sin lógica adicional.
* La capa ESB deberá agregará WS-Policies como autenticación, autorización o seguridad.

## Tecnología JEE

Para la implementación de la capa JEE se utilizan la siguiente especificación:

Los componentes base de la capa de lógica son los siguientes:

Ilustración : Diagrama de componentes JEE

Los componentes de la capa de lógica de negocio se separan en:

| **Clases** | **Descripción** |
| --- | --- |
| FooEJB | Interfaz de negocio de un servicio EJB, oculta la implementación del EJB ya sea vía interfaz local o remota. |
| FooEJBLocal | Interfaz local del EJB, se primará el acceso local a EJBs para mayor rendimiento. |
| FooEJBBean | Implementación concreta del EJB bajo el patrón Façade. Se prima el uso de EJB Session Stateless, agregando el control transaccional manejado por el contenedor. Se delega la implementación de lógica de negocio a clases Java Pojo (FooMgr). |
| FooMgr | Interfaz de la lógica de negocio. |
| FooMgrImpl | Implementación de la lógica de negocio. |
| ZooDAO | Interfaz de componente de acceso a datos, siguiendo el patrón DAO, implementa al menos métodos CRUD. |
| ZooJPADAO | Implementación JPA de acceso a datos. |
| Koo | Clase entidad persistente relacionada a una tabla de base de datos. |

Tabla , Tabla de Componentes JEE.

* EJB locales (EJB), para incorporar capacidad transaccional.
* Clases de lógica de negocio (Mgr), en donde se implementan las funcionalidades del sistema.
* Clases de acceso a datos e integración (DAO).

### DAO

Para el almacenamiento y obtención de datos se utiliza el mapeo de objetos relacionales Entities, para las búsquedas se realizan contra la Base de Datos.

Para ocultar esta implementación se utiliza el patrón delegating de manera de direccionar algunas acciones hacia la Base de Datos.

### Transaccionalidad

Para manejo de eventos transaccionales se utilizará EJB Session Stateless Local con transaccionalidad Required. A nivel de base de datos se utilizará el nivel de aislación transaccional Read Commited.

El nivel de transacción Read Commited permite un compromiso entre aislación transaccional y rendimiento, permitiendo desestimar acciones de SELECT como parte del rollback segment. Esto permite acelerar comandos de SELECT, que son normalmente los comandos más utilizados así como disminuir la carga de bloqueo sobre la base de datos.

El uso del patrón Façade permite tener una exposición de servicios junto con el rendimiento de un único punto de entrada transaccional, implementando internamente la lógica en clases POJO. La transaccionalidad Required permite indicar que el servicio se ejecutará en un contexto transaccional, ya sea iniciando una transacción o acoplándose a una transacción existente.

El contar con un único punto de inicio de transacciones permite optimizar la ejecución en la capa de lógica de negocio y a cada EJB invocado agrega una capa de componentes de control transaccional. En casos de componer un EJB de otros EJB, se agregan capas de control transaccional innecesarias.

El uso de transaccionalidad Required permite al EJB controlar el commit o rollback de las acciones realizadas contra la base de datos, lo que asegura la consistencia de la información en base de datos.

Se considera la utilización de EJBs locales debido a su mejor rendimiento en comparación con EJB remotos tradicionales ya que EJB remotos serializan la información de parámetros y retorno por sobre una capa de comunicación TCP/IP. Sin embargo, se debe notar que el Application Server, en particular, permite habilitar una optimización a nivel de EJB remotos de forma que actúen como EJB locales. Esta optimización de Application Server no es estándar JEE.

El uso de servicios EJB Session Stateless Local permite un mayor performance en la ejecución de servicios, delegando cualquier necesidad de manejo de información de estados a nivel de sesión Web para la interacción con usuario.

En el siguiente diagrama se muestran las clases EJB:



Ilustración : Diagrama de componentes EJB

### Lógica de negocio

La lógica de negocio del sistema será implementada en clases Java POJO, utilizando el patrón de contrato-implementación definiendo interfaces y desarrollando clases Java que las realicen.

La implementación de la capa de lógica de negocio se implementa en clases ‘Mgr’, las cuales manipulan los objetos de dominio para realizar las funcionalidades del sistema y se diagraman a continuación:

Ilustración : Diagrama de componentes de lógica de negocio

### Persistencia en base de datos

Se utilizará el estándar JPA para el mapeo entre objetos Entities Java y base de datos POstgres, utilizando anotaciones Java 5. JPA es el estándar de mapeo relacional y permite reducir el esfuerzo y complejidad de desarrollo.

Las relaciones entre tablas de base de datos se implementarán en Java mediante relaciones entre objetos Entities, mediante referencias a clases padres y listas de clases hijas, como se muestra a continuación:



Ilustración : Diagrama de entidades

La implementación de acceso se realizará utilizando el patrón DAO en donde se definirán los métodos CRUD necesarios.

### Integración

A nivel de integración, se contempla contar con implementación de integración con Web Services, mediante el uso del patrón DAO (Data Access Object) para ocultar la implementación y los tipos de dato:

### Contrato/Implementación

Como patrón general de uso de componentes se recomienda el uso del patrón Contrato, en el cual los componentes son declarados en función de su interfaz.

Los componentes EJB, de lógica de negocio (Mgr) y de integración (DAO) serán referenciados vía su interfaz de negocio, escondiendo su implementación en componentes ServiceLocator y Factory.

Además, otros componentes como listas, sets y mapas serán referenciados vía sus interfaces (List, Set, Map).

El uso de interfaces reduce el impacto respecto del cambio de la implementación de los componentes, centralizándolo en sus puntos de instanciación.

Los componentes de entidades y estructuras de datos serán referenciados directamente a sus clases de implementación y no contarán con interfaces de negocio.

### Capa WEB

El framework de presentación funciona de la siguiente forma:

* Las páginas son representadas por archivos htmlx, donde se define la interfaz de usuario. Estas páginas son transformadas a HTML5 y javascript mediante Spring MVC.
* Las páginas htmlx se comunican con clases Controller, las cuales son JavaBeans encargadas de mapear las variables manejadas en la presentación, además son las clases que implementan las acciones del usuario.
* Posteriormente, los Controller interactúan con otras clases, las cuales se encargan de comunicar la presentación con la lógica de negocio, manejada por EJBs. Esto se conoce como patrón de diseño Facade.
* Finalmente, para no mezclar la lógica de negocio, con la lógica de presentación se utilizan las clases Helper, las cuales se encargan de transformar las clases de negocio en clases de presentación. Estas últimas siguen el patrón Value Object.

## Gestión de errores

Para la gestión de errores en la implementación de lógica de negocio de Java, se utilizarán excepciones de negocio particulares para cada error catalogado dentro del sistema. El uso de excepciones permite el escalamiento de errores para ser gestionados en el componente que corresponda, eliminando mecanismos tradicionales de control de errores por código, el cual es propenso a errores de tipeo y baja mantenibilidad.

## Base de datos

La persistencia de información se realizará en base de datos PostgreSQL utilizando un modelo de datos relacional.

Las consideraciones generales son:

* Uso de llaves primarias simples y secuenciales, las que no tienen relación con información de negocio.
* Uso de llaves foráneas simples.

### Llaves primarias

El mapeo de identificaciones de objetos Entity JPA se realiza mediante objetos Long y EntidadId:

* Long se utiliza para objetos que trascienden los ejercicios (personas, bienes).
* EntidadId se utiliza para objetos enmarcados en un ejercicio (Actividad Presupuestaria). Esta clase contiene un id secuencial (Long) y el ejercicio (Integer).

# Web Services

El uso de Web Services, basados en estándares de interoperación, potencia la integración de sistemas, en particular de diversas tecnologías, Sin embargo, se debe tener en cuenta que esta tecnología no soporta control transaccional desde el cliente consumidor ni maneja transaccionalidad distribuida, requiriéndose mecanismos de compensación de ser necesarios.

Para la especificación de Web Services, se diseñarán los XSD de dominio que describen las entidades de negocio que formarán parte de los parámetros o resultado de los Web Services.

Como regla para XSDs, se definirán tipos de datos complejos (complexType). Estos tipos tendrán el sufijo ‘Type’.

Adicionalmente, la propagación de errores se implementará mediante la generación de eventos SOAP-Fault, que contendrán un tipo de dato conteniendo código y descripción del problema encontrado, según el siguiente diagrama:

Las implementaciones de Web Services se realizarán en Java en el módulo Web, se implementarán vía JAX-WS a partir de contratos WSDLs y XSDs.

Los servicios Web Service serán publicados en el ESB y las características de seguridad, firma o encriptación de datos serán agregadas por Enterprose Web Service Manager para publicar Web Services al exterior de cada miembro de la Red VUCE.

La implementación JAX-WS generará tanto los objetos de dominio JAXB basados en los XSD como las implementaciones Web Service basados en los WSDL. Estos objetos de dominio formarán parte del contrato de los EJB que implementen los Web Services.

Los componentes Web Service realizarán invocaciones a:

* Componentes EJB transaccionales.
* Componentes EJB de configuración.

# Logging

Se utilizará manejo de trazas de ejecución mediante el framework Log4j de Apache sobre la aplicación. Mediante este componente se logra generar información de traza a niveles de DEBUG, WARNING, INFO, ERROR y FATAL.

Log4j permite controlar la habilitación de los distintos niveles de traza mediante su archivo de configuración, en donde además se especifican niveles de traza por componentes de software y los esquemas de almacenado de trazas.

Como esquema de almacenamiento de trazas en entornos de test y producción, se utilizará el esquema de Appender que permitirá seleccionar el formato de información siendo almacenada y el formado de archivos siendo utilizado. Se recomienda que cada nodo WebLogic genere trazas de logging en su filesystem local para evitar fallas por indisponibilidad de recursos de red. Con Log4j se soportará el esquema de generar archivos diarios de logging.

Se considera el uso de traza FATAL para el logging de error de inicialización de la aplicación, como la falta de archivos de configuración o su contenido corrupto.

Se considera el uso de traza ERROR para depurar errores de sistema (no de negocio). La depuración ERROR se realizará a nivel de la capa de entrada a EJBs, a nivel de la capa de implementación de WebServices y a nivel de la capa Web de captura de datos.

La traza de DEBUG será utilizada para depuración de datos en el desarrollo del sistema, no será eliminada del código para evitar el cambio de línea base del código en promociones de entorno.

El log general de excepciones se realizará a nivel de EJBs mediante un interceptor.

# Patrones de diseño

A lo largo de la arquitectura se utilizan diversos patrones de diseño, los cuales se enumeran en este apartado.

## Interfaz-implementación

La implementación Java de código es ocultada mediante el uso de interfaces. Este patrón permite exponer contratos independientemente de la implementación, permitiendo reemplazar su implementación con un menor impacto.

## Session façade

Permite encapsular la lógica de negocio y la data de negocio, exponiendo una interfaz de forma de reducir la complejidad de los servicios y su lógica interna a los clientes que los consuman. Es utilizado para exponer la lógica de negocio vía servicios EJB Session Stateless.

## Business delegate

Permite ocultar los mecanismos de acceso a servicios en el cliente que los utilice. Este patrón es implementado mediante interfaces de negocio, la referencia al servicio se implementa mediante el patrón service locator.

## Service locator

Patrón que permite centralizar la complejidad y parametrización de la localización de servicios EJB.

## Factory

Patrón utilizado para centralización de la creación de componentes POJO, como clases de negocio (Mgr), y acceso a datos (DAO). Unido al patrón Interfaz-implementación, oculta y unifica la instanciación de componentes.

## Data access object

Patrón de centralización y normalización de acceso a datos, ya sea para acceso a base de datos o a sistemas externos. Para acceso a base de datos, implementa métodos CRUD, permitiendo la manipulación de entidades en base de datos.

## Value object y data transfer object

Este patrón permite estructurar la información de negocio y técnica a transferir entre componentes. Los objetos DTO corresponden a contenedores de información y no presentan lógica de negocio. Son utilizados, por ejemplo, para la representación JAXB de esquemas de datos.

## Front controller

Patrón utilizado en la capa de presentación con el propósito de coordinar los múltiples requerimientos de interacción usuaria. Este mecanismo permite gestionar centralizadamente las interacciones y las complejidades de protocolos de presentación, delegando al componente que corresponda la implementación del procesamiento del requerimiento. Este patrón es implementado por el framework ADF-Faces de Oracle.

# Vista Lógica

Esta vista describe la solución técnica para la implementación de la lógica de negocio de la plataforma de Integración de la Red VUCE.

## Diagrama



Ilustración : Diagrama de Componentes Vista Lógica

En la siguiente tabla se describen los componentes que conforman el diagrama de vista lógica.

| **Nombre** | **Descripción** |
| --- | --- |
| Web | Capa de implementación de captura de datos sobre tecnología Web HTML5 y Spring MVC. |
| Bus de Servicio | El bus de servicios es uno de los elementos del estándar SOA, que centraliza servicios en un único punto de acceso, además permite la orquestación de servicios y transformaciones de los datos de entrada con el fin de invocar a servicios que utilicen otras estructuras.  En el bus se publican los servicios para ser consumidos por los procesos de negocio, además de exponer los procesos de negocio, con el objetivo de interoperar con los sistemas de la Red. |
| Web Service | Capa de exposición de servicios en protocolo Web Service REST, SOAP, JMS. Este tipo de componentes no tendrá lógica de negocio, porque esta lógica estará contenida en los EJBs (señalados en el siguiente punto). Los servicios web construidos, se utilizarán para encapsular los EJBs, y exponer su lógica para que sea utilizada por sistemas, o cualquier aplicación que requiera hacer uso de estos. |
| EJB | Capa de implementación de lógica de negocio, provee además el manejo transaccional mediante las capacidades suministradas por el servidor de aplicaciones. |
| DAO | Capa de acceso a datos tanto de información transaccional como de configuración. |
| Terracota | Grilla de datos para el acceso acelerado a la información almacenada. |
| Base de Datos | Se debe contar con una base de datos relacional de alto rendimiento que permita la persistencia y consistencia de la información. |
| Procesos BPEL | Lenguaje para implementar procesos de negocios empresariales, que se utilizara para la orquestación de servicios en las funcionalidades de replicación. |
| Procesos BPM | Implementara los procesos de negocio definidos por el proyecto VUCE. |

Tabla , Descripción de componentes del Vista Lógica

En la siguiente imagen se detallan los elementos que contempla la arquitectura de la solución.



Ilustración : Diagrama de Vista Lógica Detallada.

| **Nombre** | **Descripción** |
| --- | --- |
| HTML 5 | Es la quinta versión del lenguaje HTML (HyperText Markup Language, versión 5) y como característica permite gestionar la visualización en distintos dispositivos. |
| CSS3 | Ultima especificación de hojas de estilos en cascada. |
| JavaScripts | Lenguaje de programación interpretado por los navegadores, utilizados para disponer de información y validaciones sin necesidad de sobrecargar el servidor. |
| Spring MVC | Framework Java que se utilizara para la gestión de la capa de presentación (Web) de la solución. |
| JQuery | Framework de presentación basado en JavaScripts que se utilizara para presentar la información en la Web. |
| Delegate | Patrón Proxy que concentrara el acceso a los servicios de negocio desde la capa de presentación. |
| VO | Objetos patrón Value Object, encargados de encapsular información que viajara desde la presentación y servicios REST hacia la lógica de negocio (EJB). |
| POJOS | Objetos Entitys que mapearan las tablas de la base de datos hacia la lógica de Negocio. |

Tabla , Descripción de Elementos de Vista Lógica Detallada.

# Vista de Despliegue

La siguiente vista describe los nodos involucrados en la solución, considerando un ambiente productivo de la plataforma VUCE

## Diagrama



Ilustración : Diagrama de Vista Despliegue

| **Nombre** | **Descripción** |
| --- | --- |
| Servidor de Aplicaciones | Contenedor WebLogic 10.3.3 de aplicaciones JEE. En particular corresponde a un cluster de servidores WebLogic independiente del cluster transaccional. |
| Bus de Servicio | Contenedor para la gestión los servicios de la plataforma VUCE, como de otros sistemas de los miembros de la red.. |
| Terracota | Grilla de datos en cluster de configuración datos de rápido acceso. |
| Base de Datos | Servidor(es) de base de datos transaccional. |

Tabla , Descripción de los Nodos de Despliegue

# Consideraciones Generales

## Herramientas

Para la construcción del proyecto VUCE como plataforma de integración se propone utilizar herramientas de clase mundial en la línea J2EE. En la línea de ReaHat

| **Herramienta** | **Descripción** |
| --- | --- |
| JBOSS ESB | Esta herramienta es la base para la definición y ejecución de los procesos de negocios u orquestación de servicios de negocio. |
| JBOSS Application Server | Herramienta destinada a contener las aplicaciones o soluciones desarrolladas. |
| Terracota | Infraestructura de caché de datos distribuida en una grilla de servidores. |
| Postgres SQL | En esta herramienta se implementarán tablas relacionales de Aplicaciones de la Plataforma VUCE. |

Tabla , Herramientas a utilizar en el Sistema

## Desarrollo

Para el desarrollo se definen tres capas fundamentales como acceso a datos, Lógica empresarial, Presentación. En este apartado se define que componentes y herramientas del servidor se utilizaran para la definición de cada una de estas capas.

### Acceso a datos

Para el desarrollo de la capa de persistencia se define la utilización de componentes y patrones que definen el acceso a datos. Los datos a consumir estarán asociados a un motor de base de datos o a servicios expuestos por cada integrante de la red.

| **Categoría** |  | **Elemento** |
| --- | --- | --- |
| Componentes Lógicos de Acceso a Datos | JDBC | Clases DAO |
| Eclipse Link 1.1.3 (JPA) |
| Patrones | Data Access Object. |
| Agentes de Servicios | Servicios Web WSDL 1.0 |
| FTP |

Tabla , Herramientas/Componentes para el Acceso a Datos

### Lógica Empresarial

Para el desarrollo de la lógica empresarial o capa de negocio se define la utilización de entidades, componentes, flujos de trabajo e interfaces de servicios para dar sustentabilidad y consistencia de las reglas del negocio establecidas en el proceso de levantamiento y análisis de requerimientos.

| **Categoría** | **Elemento** |
| --- | --- |
| Entidades Empresariales | POJOs |
| Componentes Empresariales | Procedimientos de Almacenados, para la conectividad con lógica de sistemas actuales. |
| Sessions EJB Local |
| Flujos de Trabajo Empresarial | BPMN |
| BPEL |
| Interfaces de Servicios | Web Servicie |
| Sessions EJB Remotos |
| Componentes RMI |
| REST 1.3 |

Tabla , Herramientas/Componentes para la Lógica Empresarial

### Presentación

Para el desarrollo de la capa de presentación se definen componentes para el desarrollo web, móvil y de reportes, los cuales se describen en la siguiente tabla.

| **Categoría** | **Elemento** |
| --- | --- |
| Web | HTML5 |
| XHTML |
| CSS3 |
| Spring MVC |
| JQuery, JavaScripts |
| Móvil | HTML5 |
| Reportes | Jasper Reports 5.0 |
| Reportes Personalizados |

Tabla , Herramientas/Componentes para la Capa de Presentación

## Pruebas

Las pruebas serán apoyadas por herramientas, con el fin de optimizar y mejorar el resultado de los desarrollos. Para mejorar se verificara el estilo de codificación, pruebas unitarias, validación con pruebas automatizadas y pruebas de rendimiento.

| **Categoría** | **Elemento** |
| --- | --- |
| Estilo de Codificación | Sonar 3.5 |
| Pruebas Unitarias | JUnit |
| SOAP - UI |
| Validación | Selenium 2.31 |
| Rendimiento | Jmeter 2.9 |

Tabla , Herramientas/Componentes para el Proceso de Pruebas

## Herramienta de Desarrollo

De las herramientas de desarrollo a utilizar se consideran aquellas que ayuden a optimizar el desarrollo, como IDES, Gestión de base de datos, Control de versiones, Automatización y prototipo para tareas de definición funcional.

| **Categoría** | **Elemento** |
| --- | --- |
| IDS’s | NetBeans 7.3, Se propone usar este IDE. |
| JDeveloper 11g |
| Eclipce 4.2 |
| Gestión DB | PgAdmin III |
| Control de Versiones | SVN 1.7.9 |
| Herramientas de Automatización | Jenkins 1.514 |
| ANT 1.8.4 |
| Maven[[1]](#footnote-1) 3.0.3[[2]](#footnote-2) |
| Prototipo | Axure 6.5 |

Tabla , Herramientas de Desarrollo

## Seguridad

Para la seguridad de la aplicación se considera el uso del estándar JAAS que está integrado en el servidor de aplicaciones empresarial, con el fin de aprovechar las ventajas de estas herramientas.

Como propuesta se plantea la utilización de perfilamiento mediante un LDAP.

| **Categoría** | **Elemento** |
| --- | --- |
| Basada en Framework | JAAS |
| Dominio LDAP | Microsoft Active Directory, OPEN LDAP |
| Autenticación | Microsoft Active Directory, OPEN LDAP |
| Autorización | Por Código (Dependencia JAAS) |
| Por Archivos de Configuración (Dependencia JAAS) |
| Seguridad del Canal | VPN |
| SSL (Propuesta) |
| Registro de Auditoría | Log4j – File Adapter |
| Administración de Perfiles | Base de Datos |
| LDAP, (Propuesta) |

Tabla , Herramientas y Componentes de Seguridad

## Administración de funcionamiento

Como requerimientos no funcionales se propone la utilización de dos elementos de seguridad perimetral para el acceso a los servicios. Estos componentes pueden ser de Hardware y Software que se deben utilizaran en la implementación de la solución. Principalmente estos elementos son provistos por los Data Center que guardan la aplicación. Estas se muestran a continuación.

### Hardware

| **Categoría** | **Elemento** |
| --- | --- |
| Servidor Proxy |  |
| Firewall |  |

Tabla , Herramientas/Componentes para el Acceso a Datos

### Software

A continuación se definen el software base que contempla la implementación y puesta en marcha de la solución.

| **Categoría** | **Elemento** |
| --- | --- |
| Sistema Operativo | Linux Centos 6 |
| Servidor de Directorio | Active Directory |
| Servidor de Correo | MS Exchange 2010, Google |
| Plataformas JAVA | Java EE 5.x |
| Java SE 6.x |
| Gestores de Bases de Datos Relacionales | PostgreSQL |
| Navegadores | IE 9.x \* |
| Firefox |
| Chrome |
| Safari |
| Servidor Web | Apache 2.4 |
| Servidor Empresarial | JBOSS Application Server |
| ESB | JBOSS ESB |
| ECM | Alfresco 4.2 |

Tabla , Herramientas/Requerimientos de Software Base

## Comunicación

El sistema interactúa internamente con sus componentes definidas mediante una definición de políticas de sincronía de servicios, formato y protocolos, los que se muestran a continuación.

| **Categoría** | **Elemento** |
| --- | --- |
| Sincronización | JMS 2.0 |
| WS-I 2.0 |
| Formato | XML |
| JSON 3.0 |
| Protocolo | SOAP 1.2 |
| HTTP 1.1 |
| FTP |

Tabla , Componentes para la Comunicación

1. La utilización de Maven es indistinto al IDE de desarrollo que se utilice, por lo que cada equipo (desarrollo, mantenimiento, QA) podrá utilizar la herramienta de desarrollo que más acomode según la tarea a realizar. [↑](#footnote-ref-1)
2. Para la implementación de los entornos de desarrollo, test y producción se recomienda levantar su propio repositorio Maven, para no depender del repositorio del proveedor. [↑](#footnote-ref-2)