|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | *Proyecto:* VUCE  Diseño de la Arquitectura del Sistema | | | | Logo de everis NTT Data | |
| Resumen | | | | | | | |
| [Introducir resumen] | | | | | | | |
| Registro de modificaciones | | | | | | | |
| **Versión** | **Descripción [o descripción de cambios]** | | **Autor** | **Fecha creación** | **Aprobado por** | | **Fecha aprobación** |
| 1.0 | Versión Inicial | | Jorge Nicolau | 13/04/2018 |  | |  |

Contenido

**Tipo de Información:** confidencial

**Compañía:** everis

**Propietario de la Información:** Secretaria Comercio Exterior

1. Descripción de la arquitectura software 2

2. Descripción de la infraestructura tecnológica 7

3. Descripción de los entornos tecnológicos 7

4. Condicionantes de la arquitectura elegida 7

Índice de figuras

[Diagrama 1 - Esquema General de Arquitectura 2](#_Toc511404713)

[Diagrama 2 - Detalle Arquitectura de Presentación 3](#_Toc511404714)

[Diagrama 3 - Detalle Clúster Público de Presentación 4](#_Toc511404715)

[Diagrama 4 - Detalle Clúster Privado de Presentación 5](#_Toc511404716)

[Diagrama 5 - Detalle de Capa de Servicios API REST VUCE 6](#_Toc511404717)

[Diagrama 6 - Detalle Parcial de Servicios e-goveris 6](#_Toc511404718)

Índice de tablas

# Descripción de la arquitectura software

Descripción de la arquitectura software del sistema presentado desde un punto de vista técnico.

- Tecnologías a utilizar para implementar las funcionalidades requeridas

- Estructura del sistema de información identificando cada uno de los subsistemas, tanto internos como externos al contexto del proyecto. Utilizar diagramas para su representación. A modo de ejemplo se pueden utilizar diagramas UML (Package Diagram/Component Diagram.)

- Descripción de las interacciones entre los diferentes subsistemas.

- Referencias a estándares y modelos de arquitectura software.

Para la implementación de la solución de VUCE, se utilizará una arquitectura de software modular que permita el escalamiento horizontal de los diferentes componentes de modo de poder absorber concurrencia de usuarios sin afectar la performance.

El diseño modular orientado a crecimiento horizontal, permite también la alta disponibilidad de la solución redundando los componentes afectado; esto es procurando evitar los puntos únicos de fallo (SPOF).

Con estas premisas generales, se orienta el diseño de la arquitectura a una aplicación multicapa, con separación funcional de servicios de presentación, de negocio y de integración. De ahí el primer diagrama general de arquitectura:

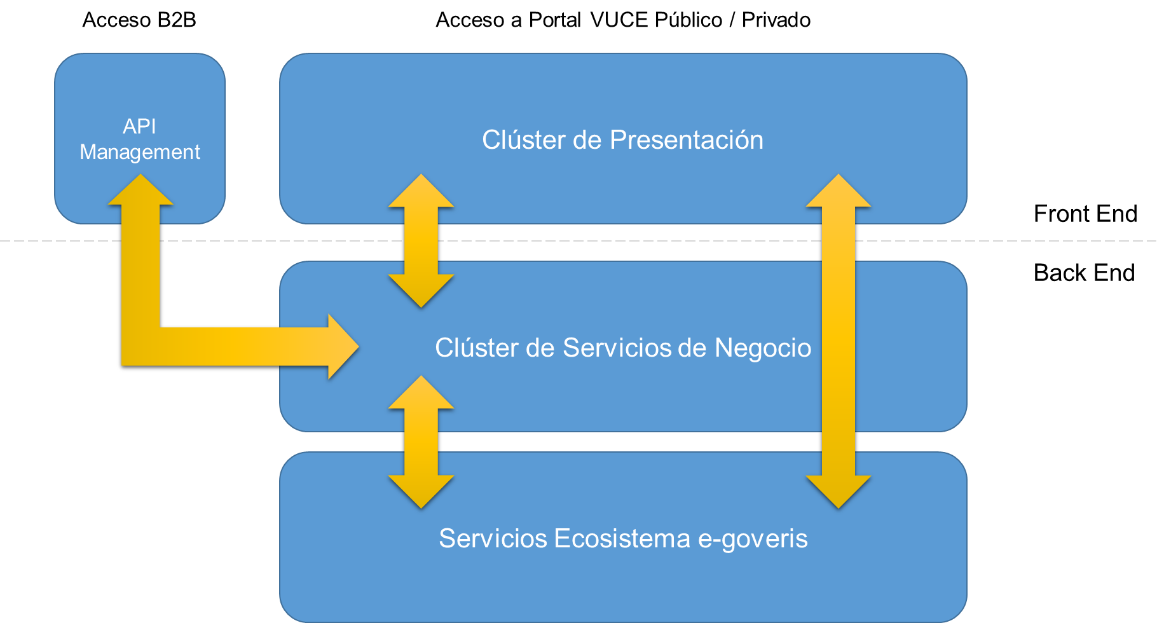


Diagrama 1 - Esquema General de Arquitectura

Se prevé dos tipos de acceso:

* el acceso a la Web presentada en el portal VUCE. Este sitio tiene además una parte pública (navegación anónima) y una parte privada (es decir, requiere autenticación del usuario). El desarrollo de estos accesos a VUCE son los previstos en las etapas iniciales del proyecto, implementado la parte pública con CMS y la parte privada con una aplicación Web.
* el acceso B2B expuesta con una interfaz de API Management. Este acceso se diseña tanto para integración con sistemas externos como para reutilizar la lógica de negocio de VUCE desde nuevas aplicaciones (futura aplicación móvil, por ejemplo), desde trabajos entre sistemas orquestados en entornos de integración (un *workflow* entre reparticiones, por ejemplo) o simplemente consulta externa del estado de trámites (como incentivo a terceros desarrolladores a usar VUCE). Este acceso no está previsto en las etapas actuales del desarrollo, pero puesto que el desarrollo de servicios debe darse con este enfoque, es importante definirlo en esta etapa.

En cuanto al “Clúster de Presentación” estará compuesto de dos plataformas: el CMS Drupal para la parte pública y la Webapp VUCE para la parte privada.

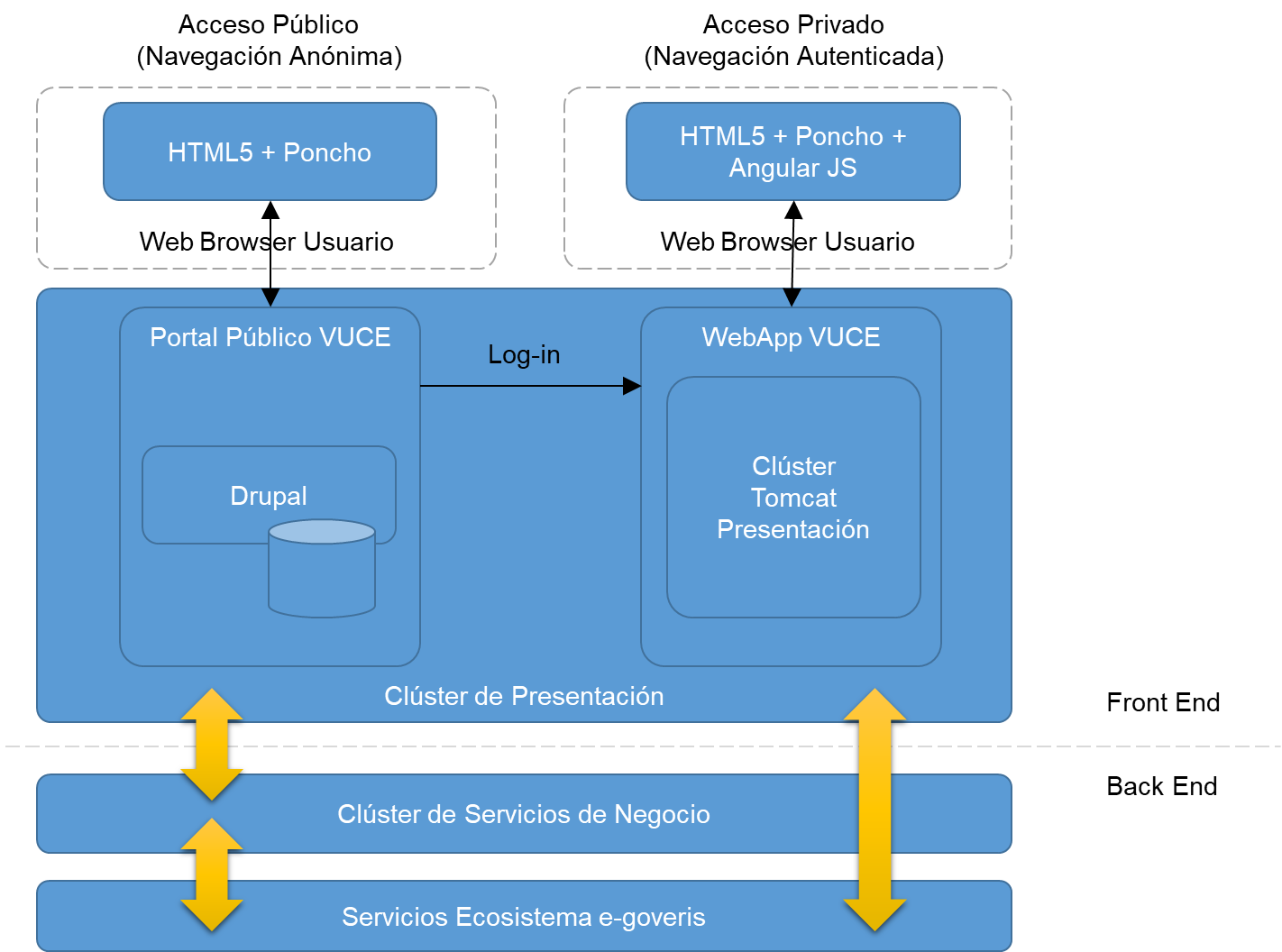


Diagrama 2 - Detalle Arquitectura de Presentación

Para permitir presentar las noticias de forma dinámica y que estas puedan administrarse sin necesidad de modificar el código de las páginas del sitio público, se utilizara Drupal como *Content Management System* (CMS). Además se tratar de una versión de Drupal que se configurará con plantillas preparadas para HTML5 que implementen Poncho (<http://argob.github.io/poncho/>) de modo de seguir los lineamientos visuales de sitios pertenecientes a la Administración Pública Nacional de la República Argentina. El uso de Poncho además permite generar contenido con diseño responsivo (adaptado a dispositivos móviles y a equipos de portátiles y de escritorio) por estar basado en Bootstrap (<http://getbootstrap.com/>)

En cuanto a la aplicación Web de VUCE, su capa de presentación se implementará usando también HTML5 y Poncho, pero para la interacción con los controladores de componentes de UI se utilizará AngularJS (<https://angular.io/>). Los controladores de presentación estarán implementados sobre el framework Spring (<https://spring.io/>) y utiliza todos los servicios adicionales del Ecosistema e-goveris (Security, Sesión, etc).

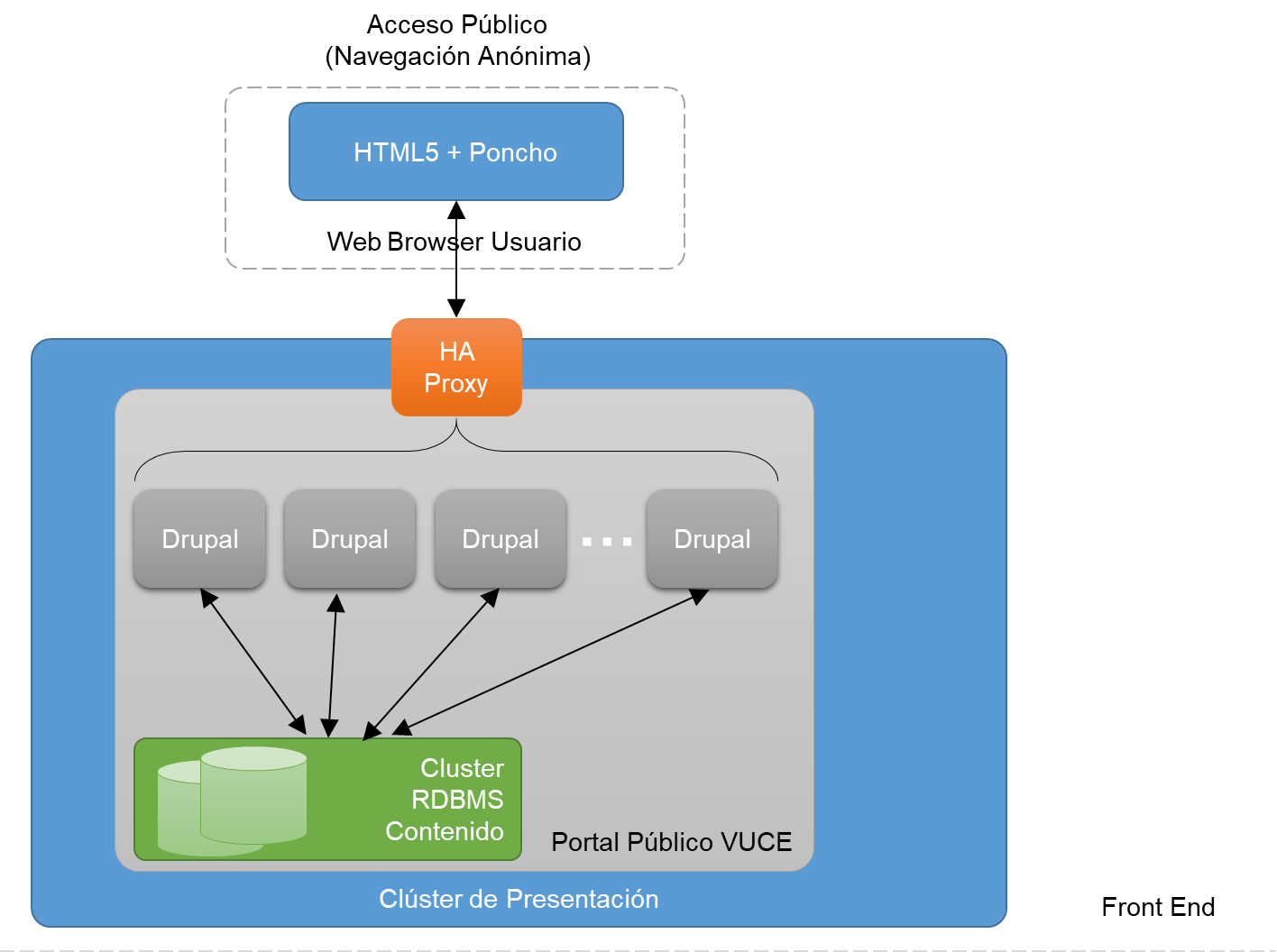


Diagrama 3 - Detalle Clúster Público de Presentación

Para la implementación en alta disponibilidad del CMS Drupal, se propone un conjunto de nodos Drupal (cada nodo Drupal es un Web Server, por ejemplo Apache, con PHP-FastCGI y la aplicación Drupal desplegada en el mismo) en balanceo Round-Robin detrás de HAProxy (también puede reemplazarse con una solución de Hardware, por ejemplo F5, u otra de software como NGINX). Todos los nodos tiene acceso al RDBMS que persiste los contenidos a presentar, también configurado en un Clúster; el servidor de base de datos puede ser algo pequeño como MySQL (con capacidad de clúster) o una infraestructura compartida de Oracle RDBMS (en configuración Rack).

Este despliegue se prevé para absorber mucho tráfico concurrente y puede ser escalado desde la configuración mínima un único nodo Drupal con una única instancia de base de datos de Contenidos.

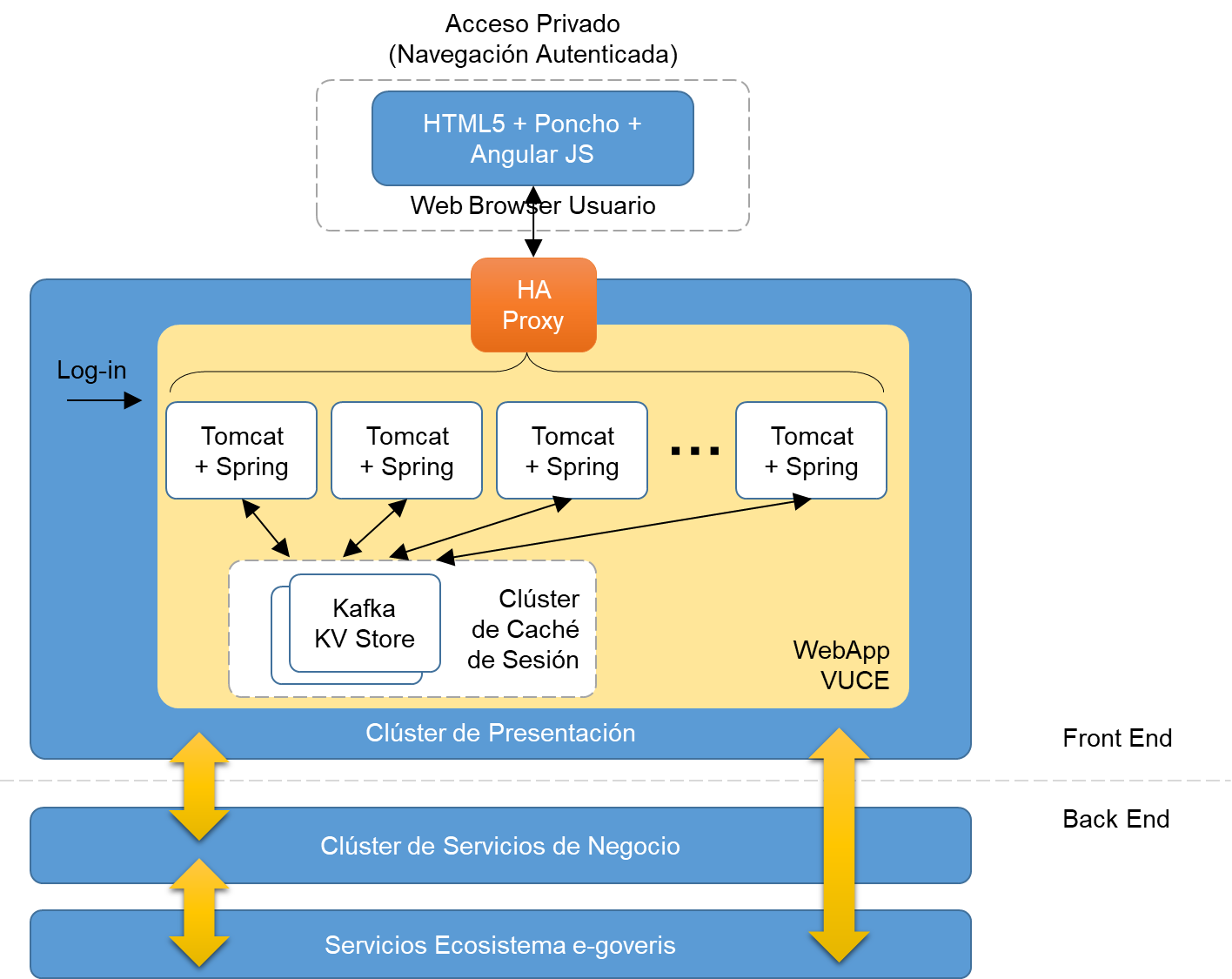


Diagrama 4 - Detalle Clúster Privado de Presentación

Para la implementación en alta disponibilidad de la Capa de Presentación de la Webapp VUCE, se propone la utilización de nodos Tomcat que desplieguan el front end de la aplicación hecha con Spring y son balanceados en Round-Robin por un HAProxy. Para que cualquier nodo puede atender la sesión de un usuario dado en cualquier momento, la información de sesión se almacena fuera de la memoria del nodo Tomcat en un clúster Kafka con el adaptador Key Value distribuído en las diferentes réplicas del servicio (que es coordinado desde un server ZooKeeper)

Además este clúster de presentación utiliza los servicios del Ecosistema e-goveris para, entre otras cosas, autenticar los usuarios (utilizando el componente Security, que implementa la autenticación y autorización vía PAEC); asimismo utiliza los servicios de negocio en la capa implementada en el Back End mediante protocolo REST. Estos servicios, a su vez, utilizan el acceso a la persistencia de datos y a la integración entre sistema también provistos por componentes de e-goveris.

Este despliegue puede escalarse desde la configuración mínima de una instancia Tomcat, pero ya preparada para almacenar externamente su sesión en una instancia Kafka.

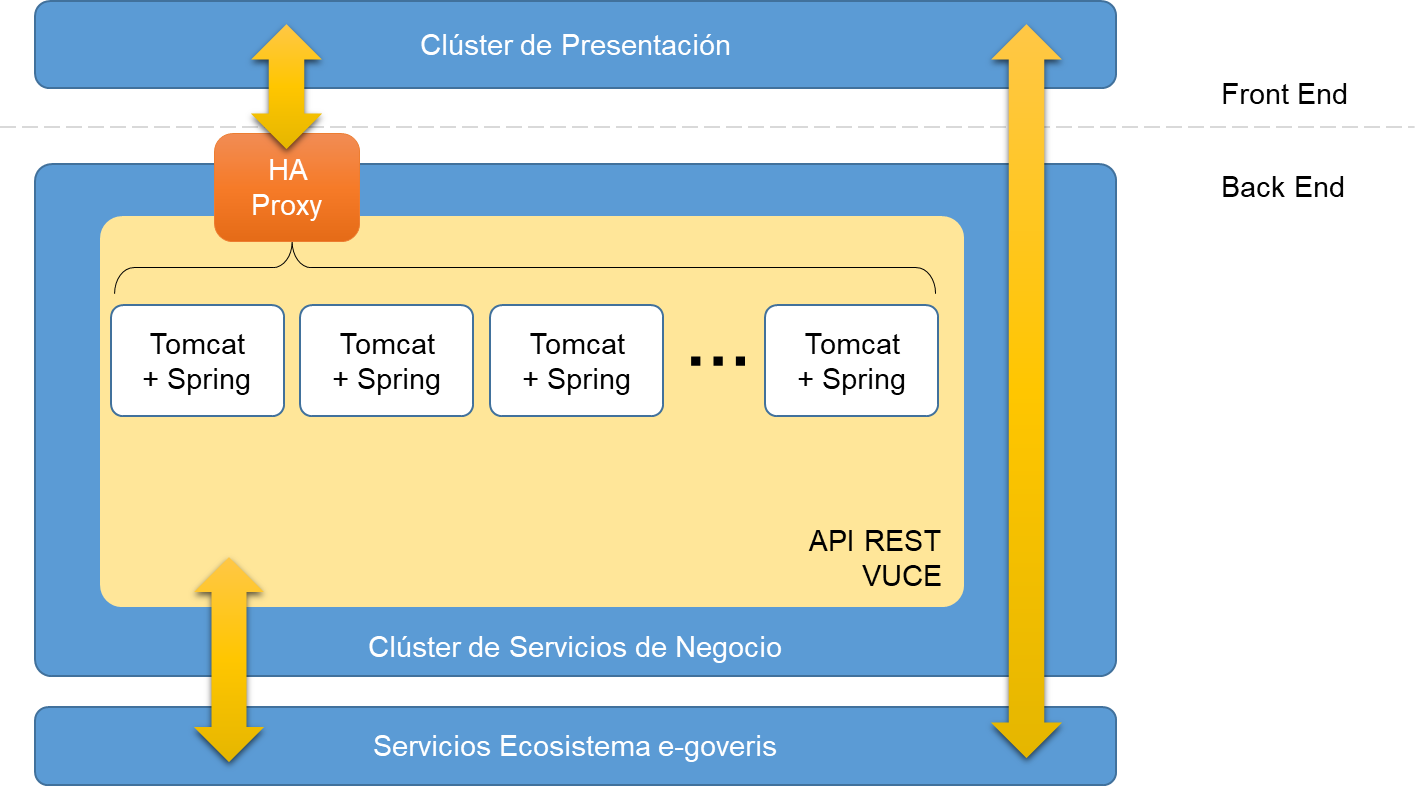


Diagrama 5 - Detalle de Capa de Servicios API REST VUCE

Para implementar los servicios de negocio de VUCE, se implementarán sobre un conjunto de nodos Tomcat balanceados detrás de un HAProxy para soportar mayor concurrencia de llamadas. Los servicios implementados con el auxilio del framework Spring serán accedidos exclusivamente vía REST. Estos servicios REST serán desarrollados de modo que sean completamente *stateless*, por lo que no se almacenará sesión ni en los nodos Tomcat ni en otro almacenamiento externo.

Para acceder a la persistencia y a mecanismos de integración con otros sistemas, estos servicios a su vez utilizarán los servicios provistos por el Ecosistema e-goveris.

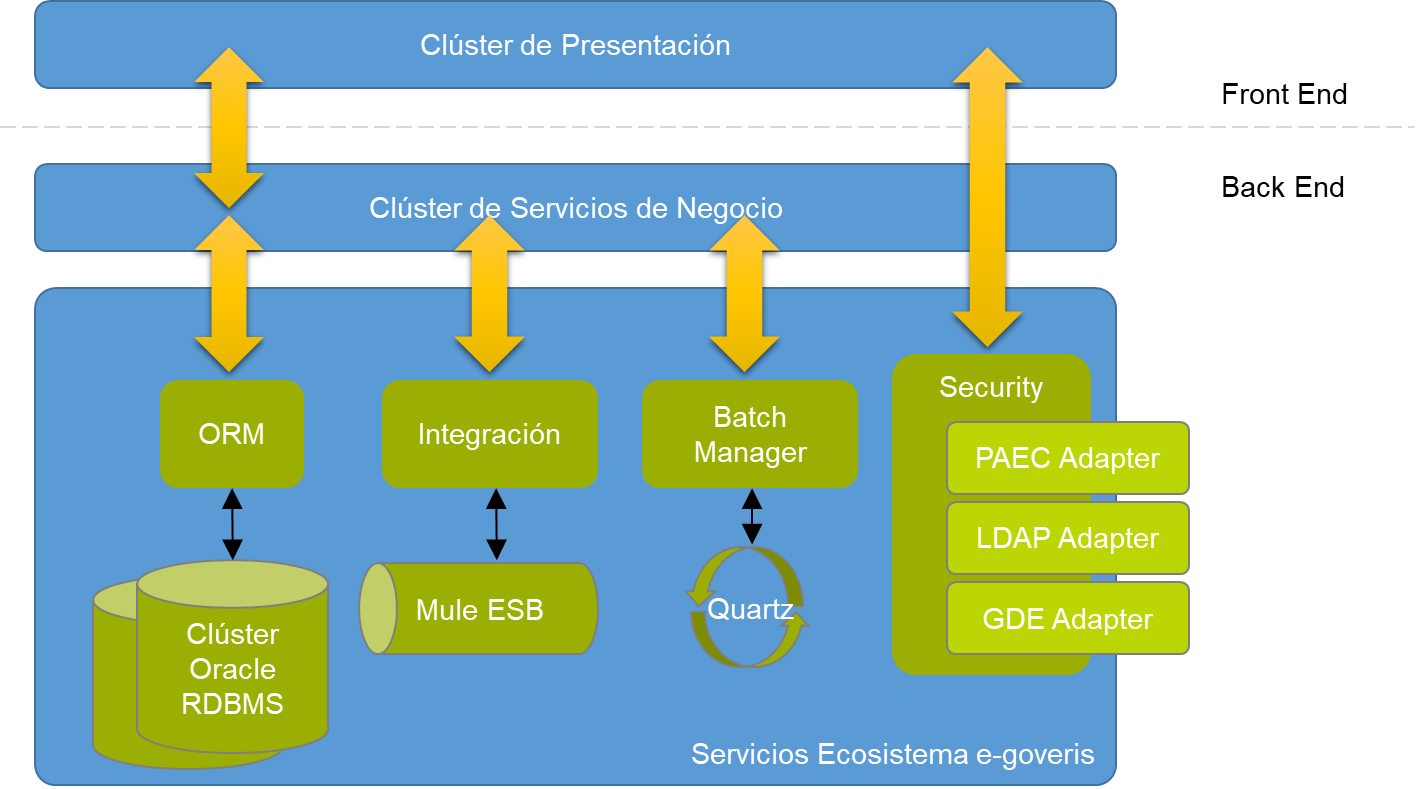


Diagrama 6 - Detalle Parcial de Servicios e-goveris

El Ecosistema e-goveris provee una variedad de servicios; entre los servicios que utilizará la Webapp VUCE están:

* Acceso al modelo de datos mediante los servicios de ORM (Hibernate)
* Acceso a terceros sistemas mediante los servicios de integración (Mule ESB)
* Programación de Tareas utilizando el Batch Manager (implementado con Quartz)
* Autenticación y Autorización de usuario mediante Security, que implementa adaptadores para LDAP, GDE o PAEC. En el caso de VUCE, Security implementará el acceso a PAEC (ver diseño técnico detallado de Login con PAEC).

Algunos de estos servicios se implementan como un servidor separado (como Mule ESB) o como una librería que utilizarán los servicios API REST internamente (como Hibernate).

# Descripción de la infraestructura tecnológica

Descripción de la infraestructura tecnológica del sistema.

Suponiendo que la capa de presentación es la primera de ellas, se realizara con Angular 5 integrando con ng2-charts (https://valor-software.com/ng2-charts/), consumiendo servicios de las capas inferiores.

Para la capa de negocio se estará utilizando Spring-Boot (framework suit, <https://projects.spring.io/spring-boot/>), aquí se expondrán los servicios como, security, ORM, iOc, entre otros, la comunicación entre los mismos será via REST (aunque puede existir algún SOAP pero mediante demanda)

En la última capa encontraremos las bases de datos, donde encontramos un Oracle 12g, la misma será accedida por la capa de negocio.

En cuanto a la web estatica se realizara con PHP utilizando Drupal 7 (<https://www.drupal.org/drupal-7.0>) dicho framework es un CMS (Content Management System), montado sobre un Apache (<https://httpd.apache.org/>)

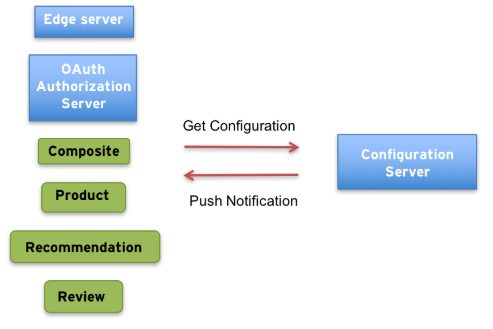
Sobre aspectos visuales se utilizara Poncho (<http://argob.github.io/poncho/>) en *complience* con los estilos del Republica Argentina

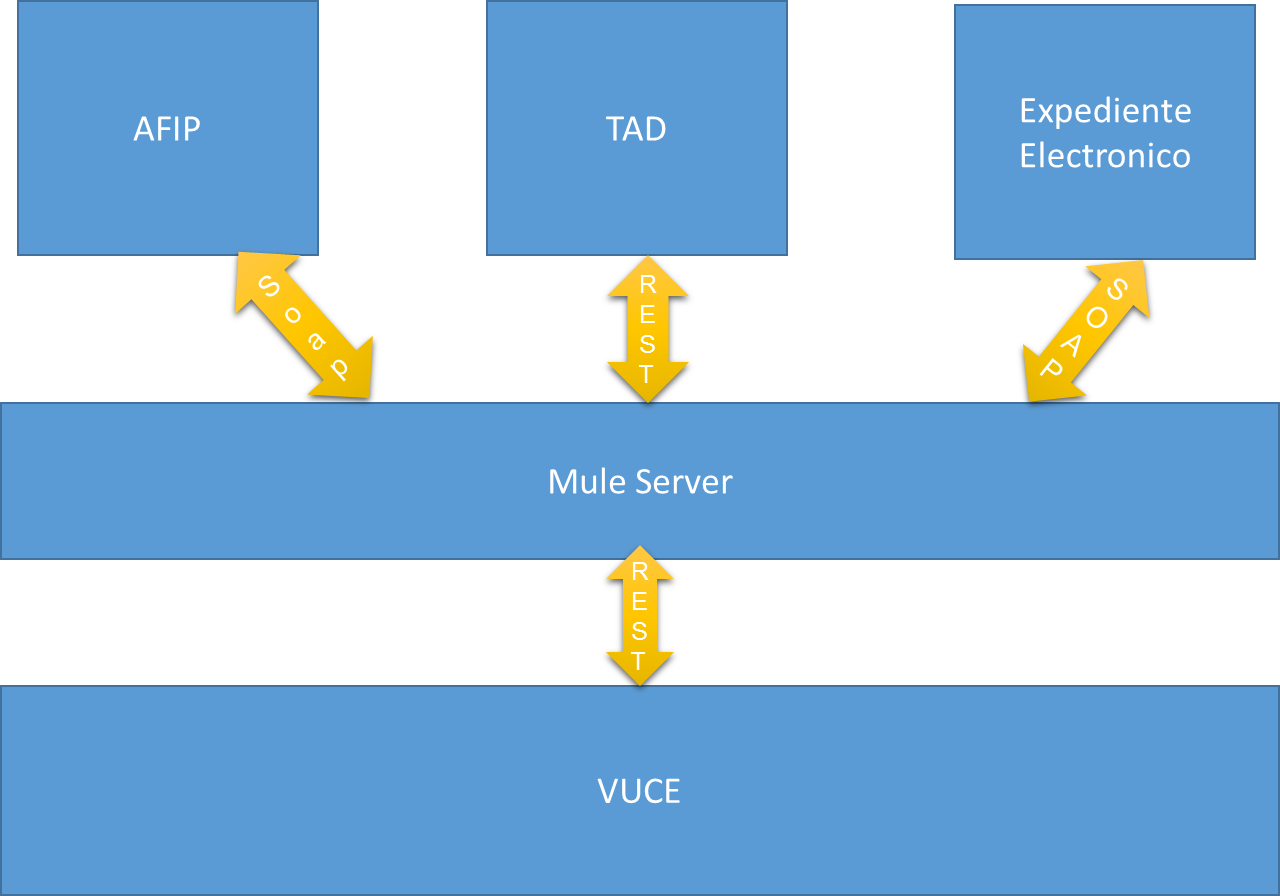
Se estará implementando micro servicios para la capa de negocio la suit de Spring Boot Netflix,

Dentro de la suit de aplicativos se evaluar la positiblidad de utilizar Eureka (Service Discovery Server), Ribbon (Dynamic Routing and Load balancer) *\*\* puede ser reemplazado por el ESB de Emule que gestionaría este modulo \*\** En cuanto a la monitorización se propone utilizar Hystrix (Monitoring Server) para detectar los fallos de los servicios

En cuanto al manejo de reglas se propone utilizar drools framework (<https://www.drools.org/>) debido a que dentro de una pagina del sitio puede contar un usuario con diferentes reglas de negocio.  
  
Para mantener las configuraciones cross a todas las aplicaciones se propone utilizar Spring Cloud Config

Manejando el siguiente concepto





# Descripción de los entornos tecnológicos

Descripción de los entornos tecnológicos que se utilizarán durante el desarrollo del proyecto (desarrollo, pruebas, aceptación, producción…).

Se contara con dos ambientes provistos por Everis y el productivo provisto por DEV (Development) , UAT (User acceptance testing), PROD (Production).

Si bien cada desarrollador estará trabajando en su entorno local, se harán despliegues en un ambiente de desarrollo donde se podrá evaluar la calidad del código, utilizando integración continua y casos de pruebas, asi se podría llevar al ambiente de UAT minimizando los defectos.

Una vez aprobado el UAT se desplegara en PROD.

# Condicionantes de la arquitectura elegida

Enumeración de las ventajas de la arquitectura elegida.

Enumeración de las limitaciones/restricciones de la arquitectura elegida.

Se ha basado la arquitectura sobre el software libre como Java para la capa de negocio, Angular para la presentación, PHP para el sitio estatico publico, para la capa de estilos Poncho.

Se plantea tener capas con separadas donde lo estará orquestando el ESB de mule, en la capa de negocio intervendrá la suit de spring y en la capa de datos Oracle 12g

