**ARQUITECTURA DE APLICACIONES Y DATOS**

**Modelo para el Desarrollo y Gobierno de Nuevo Core de Pensiones**

**EVOL**

para AFP Integra y Prima AFP



Índice

[1 Propósito 4](#_Toc522546777)

[2 Glosario de Términos 5](#_Toc522546778)

[3 Stakeholders 6](#_Toc522546779)

[4 Objetivos y Restricciones 7](#_Toc522546780)

[4.1 Objetivos de Negocio 7](#_Toc522546781)

[4.2 Objetivos de TI 7](#_Toc522546782)

[4.3 Capacidades 8](#_Toc522546783)

[5 Restricciones 10](#_Toc522546784)

[6 Cumplimiento 11](#_Toc522546785)

[6.1 Principios Estratégicos 11](#_Toc522546786)

[6.2 Principios de Negocio 11](#_Toc522546787)

[6.3 Principios de Aplicaciones 13](#_Toc522546788)

[6.4 Estándares 14](#_Toc522546789)

[7 Arquitectura de Aplicaciones TO-BE 15](#_Toc522546790)

[7.1 Arquitectura de Aplicación 15](#_Toc522546791)

[7.1.1 Front-End 15](#_Toc522546792)

[7.1.2 Experiencia del Usuario 21](#_Toc522546793)

[7.1.3 Back-End 22](#_Toc522546794)

[7.1.4 Integración 25](#_Toc522546795)

[7.1.5 Mapa de Interacción 29](#_Toc522546796)

[7.2 Arquitectura de Integración 30](#_Toc522546797)

Información de Documento

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre de Proyecto:** | Proyecto Kadabra | | |
| **Preparado Por:** | Nestor Cayllahua, Juan Pablo Gonzales | **Número Versión:** | 1.3 |
| **Título:** | Arquitectura de Sistemas de Información | **Fecha:** | 07/08/2018 |
| **Revisado Por:** | José Luis Rentería | **Fecha de Revisión:** | 20/08/2018 |

Lista de Distribución

| **Desde** | **Fecha** | **Teléfono/Correo Electrónico** |
| --- | --- | --- |
|  |  |  |
|  |  |  |

| **Para** | **Acción\*** | **Fecha Fin** | **Teléfono/Correo Electrónico** |
| --- | --- | --- | --- |
|  |  |  |  |

\* Tipos de Acción: Aprobar, Revisar, Informar, Archivo, Otros (por favor, especificar)

Historial de Versiones

| **Número Versión** | **Fecha** | **Revisado Por** | **Descripción** | **Nombre de Archivo** |
| --- | --- | --- | --- | --- |
| 1.0 | 07/08/2018 | José Luis Rentería | Creación | KADABRA - C\_Arquitectura\_Sistema\_Informacion\_V1\_0.docx |
| 1.1 | 14/08/2018 | José Luis Rentería | Observaciones en principios, aplicaciones e integraciones | KADABRA - C\_Arquitectura\_Sistema\_Informacion\_V1\_1.docx |
| 1.2 | 15/08/2018 | José Luis Rentería | Justificación para componentes de capa de Experiencia de Usuario. | KADABRA - C\_Arquitectura\_Sistema\_Informacion\_V1\_2.docx |
| 1.3 | 16/08/2018 | José Luis Rentería | Modificaciones en arquitectura de aplicaciones. | KADABRA - C\_Arquitectura\_Sistema\_Informacion\_V1\_3.docx |
| 1.4 | 16/08/2018 | José Luis Rentería | Modificaciones según observaciones Integra | KADABRA - C\_Arquitectura\_Sistema\_Informacion\_V1\_4.docx |
| 1.5 | 20/08/2018 | José Luis Rentería | Modificaciones según observaciones Integra y Prima | KADABRA – C\_Arquitectura\_Sistema\_informacion\_v1\_5.docx |

# Propósito

Este documento muestra la arquitectura de sistemas de información para el Proyecto Kadabra. Contiene los artefactos de arquitectura creados durante el proyecto. Provee una vista cualitativa de la solución y apuntan a comunicar la intención de los arquitectos.

# Glosario de Términos

|  |  |
| --- | --- |
| **Nombre** | **Significado** |
| API | Application Program Interface. Conjunto de objetos encapsulados que sirven de interfaz con otro componente o programa. |
| REST | REpresentational State Transfer. Protocolo estándar utilizado en servicios web. |
| SOAP | Simple Object Access Protocol. Protocolo estándar utilizado en servicios web. |
| JMS | Java Message Service. Solución para manejo de cola de mensajes. |
| JPA | Java Persistence API. API para interactuar con la base de datos por medio de objetos Java. |
| JDBC | Java Database Connectivity. |
| MVC | Model-View-Controller. Patrón de arquitectura donde se separa los datos, la lógica de negocio y la presentación en tres capas. |
| JSF | Java Server Faces. Framework para simplificar el desarrollo de aplicaciones web hechas en Java. |
| JS | JavaScript |
| HSM | Hardware Security Module |

# Stakeholders

Esta sección muestra las responsabilidades de los stakeholders para la arquitectura de datos y aplicaciones. Los stakeholders son personas que participan activamente en el proyecto, o cuyos intereses pueden verse afectados positiva o negativamente por la ejecución o la finalización del proyecto.

|  |  |
| --- | --- |
| Rol/Grupo | Nombres |
| Arquitecto Empresarial/EVOL | José Luis Rentería |
| Arquitecto de Aplicaciones/EVOL | Juan Pablo Gonzales |
| Arquitecto de Datos/EVOL | Nestor Cayllahua |
| Arquitecto de Datos/EVOL | Jorge Guerreros |
| Analista Funcional/EVOL | Maura Cieza |
| Arquitecto TI/Integra | Pedro Tapia |
| Arquitecto TI/Prima | Rayner Huamantumba |
| Líder Técnico/Integra | Marcelo Meza |
| Líder Técnico/Prima | Nancy Bohorquez |
| Analista Senior Desarrollo/Integra | William Ccucho |
| Analista Senior Desarrollo/Integra | Luis Polo |
| Arquitecto TI/Prima | Marco Begazo |
| Arquitecto TI/Prima | Mauricio Soto |
| Arquitecto TI/Prima | Oscar Macedo |

# Objetivos y Restricciones

En esta sección describe los objetivos, tanto de negocio como de TI, las restricciones y las capacidades que la arquitectura To-Be debe cumplir.

## Objetivos de Negocio

|  |  |
| --- | --- |
|  | Objetivo de Negocio |
| Desafíos del SPP | Mejorar la imagen del Sistema Privado de Pensiones |
| Dar mejor soporte a la centralización de procesos del SPP |
| Reducir costos de administración de fondos de pensiones |
| Necesidades del Negocio de AFP’s | Fidelización y diferenciación |
| Diversificar oferta de productos |
| Simplificar procesos |
| Mejorar servicios |
| Reducir el riesgo operativo automatizando procesos manuales |

## Objetivos de TI

|  |
| --- |
| Objetivo de TI |
| Reducir obsolescencia tecnológica. |
| Disminuir los costos de desarrollo, mantenimiento y operación del Core de pensiones. |
| Mejorar Time-To-Market para implementación de cambios. |
| Aprovechar sinergias tecnológicas entre AFP’s. |
| Incrementar la calidad y servicio al cliente interno y externo. |
| Contar con un sistema seguro y confiable. |
| Definir políticas y procedimientos de TI. |
| Facilitar la implementación futura de un modelo BPO basado en el nuevo Core de pensiones |

## Capacidades

|  | Capacidad | Descripción |
| --- | --- | --- |
| Aplicaciones | Soporte de Gestión Documental | Posibilidad de integración con herramienta de Gestión Documental. |
| Workflow, Control de datos y Alertas | Posibilidad de crear y monitorear flujos de trabajo, así como alertas y notificaciones de estos. |
| Modular | Soporta componentes de aplicación independientes y desacoplados. |
| Parametrizable | Alto nivel de parametrización de los módulos del sistema.   * Cuentas/Clientes con atributos flexibles. Posibilidad de agregar información adicional a clientes o cuentas sin realizar modificaciones al sistema. * Multicuenta: Manejo de diferentes cuentas por afiliado y asignación de múltiples afiliados por cuenta. * Multiproducto/Multiservicio: Tener la capacidad de definir y soportar distintos productos o servicios. * Multimoneda: Manejo de múltiples monedas. * Multifondo: Posibilidad de distribuir o asignar cuentas a distintos tipos de fondo. |
| Multicompañía | Manejo de varias empresas, permitiendo el ingreso de nuevas. |
| Operaciones a demanda | Soporte de procesos en línea o en batch a demanda. |
| Cero papeles | Prescindir de la impresión de documentos. Deseable contar con soporte de firmas digitales. |
| Usabilidad | Aplicación fácil de usar. |
| Auditable | Contar con control de procesos que deje rastros de auditoría. |
| Flexible | Facilidad de agregar o quitar componentes o funcionalidades. |
| Seguridad | Implementar políticas y procedimientos de seguridad en todas las capas del sistema. |
| Escalabilidad | Facilidad de aumentar la cantidad de usuarios o transacciones que se procesan sin afectar la performance del sistema |
| Orientado a servicios | Sistema preparado para exponer y consumir servicios. |
| Datos | Alta calidad de datos | Asegurar confiabilidad de datos. |
| Información histórica | Mantener disponible en todo momento la información histórica. |
| Seguridad | Asegurar la confidencialidad, integridad, disponibilidad y privacidad de los datos |
| Homologación de datos | Diccionario de datos único. |
| Gobierno de datos | Políticas de gestión de datos. Catálogo y modelo de datos actualizado. |
| Trazabilidad | Seguimiento de información manejado de forma interna o externa usando una herramienta. |
| Explotación de información de negocio | Proveer servicios y brindar mecanismos de extracción de información para explotación de datos. |

# Restricciones

Esta sección describe las restricciones que se deben tomar en cuenta en cualquier trabajo de arquitectura.

* 1. **Escalas**

|  |  |  |
| --- | --- | --- |
| Tipo | Valor | Descripción |
| Severidad | Bajo | Impacto Menor. Se pueden tomar acciones rápidas para resolverlas. |
| Moderado | Impacto Medio. Si no se realiza acción podría causar traer consecuencias a corto plazo. |
| Grave | Impacto ­alto. Se requiere acción urgente. |
| Crítico | Impacto muy alto. Se requiere acción urgente. |
| Probabilidad | Raro | Podría ocurrir sólo en circunstancias excepcionales. |
| Improbable | Probabilidad baja de ocurrencia. |
| Posible | Probabilidad media de ocurrencia. |
| Probable | Probabilidad alta de ocurrencia. |
| Casi seguro | Probabilidad muy alta de ocurrencia. |

* 1. **Restricciones de Financieras y Presupuesto**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Restricción | Severidad | Probabilidad | Actividades de Mitigación |
| C1 | Presupuesto del Proyecto | Grave | Probable | Revisar políticas y procedimientos de compra del proyecto o de ambas AFP’s para que el presupuesto pueda ser utilizado con control. |

* 1. **Restricciones Externas y Negocio**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Restricción | Severidad | Probabilidad | Actividades de Mitigación |
| C2 | Cambios deben ser realizados teniendo en cuenta la normativa y regulación de la SBS. | Crítico | Probable | Revisar normativa y regulación para asegurar su cumplimiento. |

* 1. **Otras Restricciones**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Restricción | Severidad | Probabilidad | Actividades de Mitigación |
| C3 | Estándares de seguridad y protección de datos | Crítico | Probable | Tomar en cuenta Ley de Protección de Datos Personales, Circular G-140-2009, Circular G-193-2017 |

# Cumplimiento

Esta sección muestra los principios a los que se adhiere la Arquitectura Empresarial, así como las políticas y otros estándares que ésta debe seguir.

## Principios Estratégicos

|  |  |
| --- | --- |
| **Nombre** | **La arquitectura KADABRA debe soportar un esquema de multiempresa.** |
| **Código** | PAS01 |
| **Declaración** | La arquitectura KADABRA debe soportar un esquema de asociación entre varias AFP’s. |
| **Justificación** | Este principio está asociado a la estrategia de negocio definida por KADABRA que deja abierta la posibilidad de incluir a otras AFP’s. |
| **Implicancias** | Implementar una arquitectura que soporte Multiempresa.  Asegurar una arquitectura que permita el desarrollo, mantenimiento y operación común del Core de Pensiones de KADABRA  Alinear las definiciones propias de cada empresa con las definidas para KADABRA. |

## Principios de Negocio

|  |  |
| --- | --- |
| **Nombre** | **Continuidad del negocio** |
| **Código** | PAN01 |
| **Declaración** | Las operaciones de KADABRA se mantienen a pesar de las interrupciones del sistema. |
| **Justificación** | KADABRA debe contar con la capacidad de continuar la operativa, independientemente de los eventos externos (desde caídas de sistema hasta todo tipo de desastres). |
| **Implicancias** | Evaluar la criticidad e impacto de las aplicaciones a fin de determinar qué nivel de continuidad se requiere y qué plan de recuperación es necesario.  Implementar revisiones periódicas, pruebas de vulnerabilidad y exposición o, el diseño de servicios de misión crítica para garantizar la continuidad del funcionamiento del negocio, a través de capacidades redundantes o alternativas.  Implementar procesos de continuidad de negocio.  Asegurar la ejecución de los procesos, incluyendo la gestión de riesgos en situaciones en donde se vea afectada la operación de KADABRA. |

|  |  |
| --- | --- |
| **Nombre** | **Orientación a servicios** |
| **Código** | PAN02 |
| **Declaración** | La arquitectura KADABRA se basa en un diseño orientado a servicios que refleje las actividades de los procesos de negocio. |
| **Justificación** | La orientación de servicio ofrece agilidad y asegura el flujo de información entre procesos. |
| **Implicancias** | Implementar un gobierno que permita catalogar los servicios y medir impacto de los cambios a realizar en los mismos. |

|  |  |
| --- | --- |
| **Nombre** | **Responsabilidad de TI** |
| **Código** | PAN03 |
| **Declaración** | El área de TI es responsable de implementar procesos e infraestructura que permitan que las soluciones construidas cumplan con los requerimientos definidos por el usuario en términos de funcionalidad, niveles de servicio, costo y tiempo de entrega. |
| **Justificación** | La definición de estos procesos permitirá cumplir con las expectativas del negocio y con las capacidades que se implementarán para KADABRA. |
| **Implicancias** | Crear procedimientos que gestionen la demanda de KADABRA, cumpliendo los estándares y mejores prácticas de gobierno, desarrollando KPI’s que permitan medir el cumplimiento de este principio. |

|  |  |
| --- | --- |
| **Nombre** | **Implementación de un lenguaje común** |
| **Código** | PAN04 |
| **Declaración** | Se debe promover la elaboración de documentación y modelamiento de los procesos de KADABRA mediante un lenguaje común que facilite su entendimiento y comunicación. |
| **Justificación** | Una mejor comunicación entre las distintas áreas (negocio y TI) permite agilizar la interacción entre las mismas. |
| **Implicancias** | Implementar el uso de plantillas y modelamiento de procesos con notación BPMN. |

|  |  |
| --- | --- |
| **Nombre** | **Gobernabilidad de procesos** |
| **Código** | PAN05 |
| **Declaración** | KADABRA adopta el concepto de gobernabilidad de procesos de punta a punta (E2E), incluyendo la información que fluye a través de estos. |
| **Justificación** | La gobernabilidad de procesos permite la generación de valor asociada a la identificación de responsabilidades, gestión y control integral de los procesos. |
| **Implicancias** | Definir los responsables de los procesos (Process Owner).  Gestionar los procesos a través de indicadores y programas de mejoramiento continuo.  Identificar y definir controles de mitigación de riesgos de proceso. |

|  |  |
| --- | --- |
| **Nombre** | **La gestión de requerimientos de negocio es un asunto de todos** |
| **Código** | PAN06 |
| **Declaración** | La alta dirección, representantes de arquitectura y Stakeholders participan en las decisiones de gestión de requerimientos de negocio para lograr los objetivos de KADABRA. |
| **Justificación** | Este principio garantiza que la gestión de requerimientos de negocio esté alineada con la visión de KADABRA. |
| **Implicancias** | Implementar roles y responsabilidades para abordar los requerimientos.  Formar grupos de trabajo entre expertos de negocio y personal TI para definir las metas y objetivos de KADABRA |

|  |  |
| --- | --- |
| **Nombre** | **Aplicaciones de uso común** |
| **Código** | PAN07 |
| **Declaración** | Se desarrollan aplicaciones de uso común para todo el entorno de KADABRA y no aplicaciones aisladas a la funcionalidad, capacidad y expectativa de ésta. |
| **Justificación** | El permitir y mantener capacidades o funciones duplicadas es costosa, y alienta la aparición de datos contradictorios e inconsistentes. |
| **Implicancias** | Crear una política de gobierno que norme el uso de aplicaciones comunes que asegure su alineamiento a las expectativas funcionales de KADABRA. |

|  |  |
| --- | --- |
| **Nombre** | **Cambios basados en requerimientos de negocio** |
| **Código** | PAN08 |
| **Declaración** | Se realizan cambios en las aplicaciones y datos en respuesta a las necesidades de negocio. |
| **Justificación** | La base de cualquier cambio propuesto es principalmente dar soporte a los objetivos estratégicos y operacionales del negocio de KADABRA. |
| **Implicancias** | Alinear los procesos de gestión de cambio a este principio.  Implementar mejoras o cambios en los sistemas de información basados en una necesidad de negocio documentada. |

## Principios de Aplicaciones

|  |  |
| --- | --- |
| **Nombre** | **Independencia Tecnológica** |
| **Código** | PAA01 |
| **Declaración** | Las aplicaciones deben ser independientes de la plataforma tecnológica. |
| **Justificación** | La independencia tecnológica de las aplicaciones permite que se desarrollen, actualicen y operen de la manera más rentable y oportuna, además de garantizar que no dependa de un hardware, componente o software específico. |
| **Implicancias** | Definir o proponer estándares que soporten la portabilidad entre tecnologías.  Promover el desarrollo y mantenimiento de las aplicaciones independiente de la plataforma en la cual se utilice. |

|  |  |
| --- | --- |
| **Nombre** | **Facilidad de uso** |
| **Código** | PAA02 |
| **Declaración** | Las aplicaciones deben ser fáciles de usar. |
| **Justificación** | Una aplicación, con una interfaz intuitiva y fácil de usar, facilita la transición y adaptabilidad del usuario, ayudando a que él mismo realice sus tareas con un menor esfuerzo, reduciendo la curva de aprendizaje del usuario. |
| **Implicancias** | Definir un "look-and-feel" común y considerar requisitos ergonómicos.  Desarrollar criterios de prueba de usabilidad. |

|  |  |
| --- | --- |
| **Nombre** | **Flexibilidad de aplicaciones** |
| **Código** | PAA03 |
| **Declaración** | La arquitectura de aplicaciones de KADABRA debe ser modular, escalable y de fácil desacoplamiento. |
| **Justificación** | Permite optimizar la agilidad y minimizar la complejidad de integración, simplificar la implementación y el mantenimiento. Así mismo, gestionar los cambios en las soluciones de negocio con un impacto bajo en los procesos. |
| **Implicancias** | Establecer un método de integración común.  Implementar arquitectura basada en servicios.  Establecer estrategias de integración de aplicaciones. |

|  |  |
| --- | --- |
| **Nombre** | **Duplicidad de funcionalidades** |
| **Código** | PAA04 |
| **Declaración** | La arquitectura de aplicaciones debe promover la racionalización en el portafolio de soluciones de negocio, maximizando su aprovechamiento y evitando la implementación de funciones ya existentes. |
| **Justificación** | La correcta identificación y clasificación funcional en el portafolio de aplicaciones de KADABRA evitará que se propongan e implementen soluciones que cubran funcionalidades ya existentes en las aplicaciones actuales. |
| **Implicancias** | Gestionar portafolio de aplicaciones y funcionalidades para KADABRA.  Gestionar los requerimientos comparando las funcionalidades solicitadas con las existentes en las aplicaciones actuales mediante un portafolio de aplicaciones y servicios.  Establecer trazabilidad en la identificación de necesidades de negocio.  Reusar componentes de otras aplicaciones. |

|  |  |
| --- | --- |
| **Nombre** | **Sistema escalable** |
| **Código** | PAA05 |
| **Declaración** | El sistema KADABRA debe soportar la escalabilidad a nivel de capacidades para atender los cambios en la demanda. |
| **Justificación** | Permite que el sistema pueda operar de manera óptima según cambios en la carga operativa y demanda del negocio. |
| **Implicancias** | Monitorear el comportamiento de las aplicaciones o servicios para asegurar la atención óptima de la operación diaria. |

## Estándares

Se utilizarán los siguientes estándares en el presente entregable:

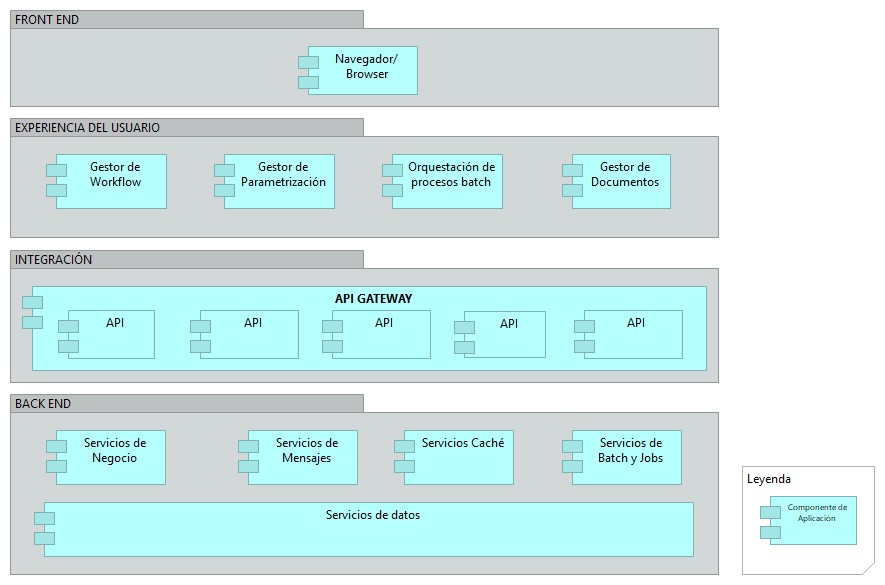
* ARCHIMATE un es lenguaje de notación unificado utilizado para modelamiento de arquitectura empresarial de alto nivel, el cual permite visualizar e identificar las aplicaciones negocio, sub-aplicaciones, componentes, bases de datos, servicios, etc. y sus respectivas interacciones.

# Arquitectura de Aplicaciones TO-BE

Esta sección contiene los esquemas y gráficos que representan la arquitectura de aplicaciones TO-BE.

## Arquitectura de Aplicación

A continuación, se muestra la arquitectura de aplicación y el detalle de cada uno de sus componentes:



### Front-End

Es la capa que se encarga de recibir las peticiones por parte del usuario, invocar la lógica de negocio y devolver los resultados al usuario mediante la interfaz.

#### Patrones Tendencias

* + - Desacoplamiento entre la capa de presentación y servicios.
    - Diseño de arquitecturas Cloud.
    - Omnicanalidad.
    - Multicanalidad y multidispositivo.
    - Cache y persistencia de datos en el cliente.

#### Estándares

A continuación, se presenta un cuadro comparativo entre las distintas tecnologías usadas en la actualidad:

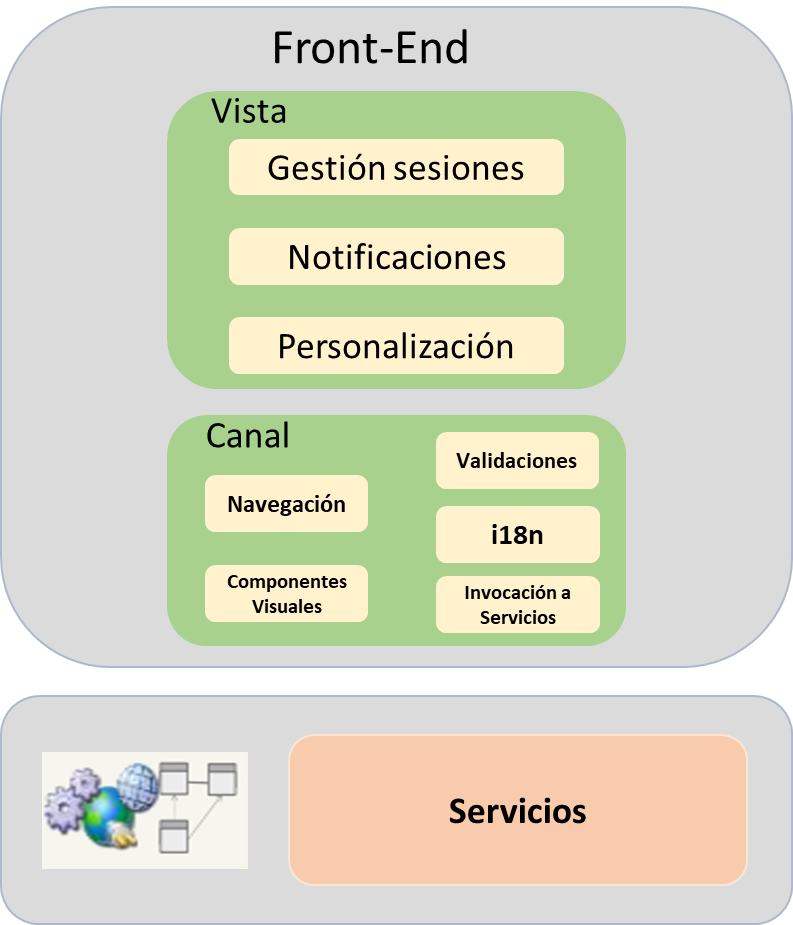
|  |  |  |
| --- | --- | --- |
| **View** | **Server MVC** | **Cliente MVC** |
| Alto costo de personalización.  Necesidad de un servidor de aplicaciones.  Integración sencilla con framework MVC.  Madurez. | Alta flexibilidad de integración con diversos frameworks (Spring, JRuery y JSF).  Necesidad de un servidor de aplicaciones.  Alto nivel de madurez y soporte. | **Alto desacoplamiento entre los datos y la presentación.**  **Reducción del consumo de CDP.**  **No necesita de servidores de aplicaciones.**  Alto nivel de madurez.  La mayoría de framework se apoyan en Angular y JQuery. |

* + - Se recomienda el uso de Cliente MVC por las siguientes razones:
* Alto desacoplamiento entre los datos y la presentación. Esto permite:
  + Realizar cambios en servicios de negocio sin afectar la operación en la capa de presentación y viceversa.
  + Eliminar la necesidad de mantener una conexión constante al origen de datos, consumiendo API’s para interactuar con los Servicios de Negocio solo cuando la aplicación lo necesite.
  + Optimización de tiempos de desarrollo paralelizando tareas asignadas al equipo responsable de la capa de aplicación y capa de negocio.
* No se necesita servidor de aplicaciones (Weblogic, WAS, etc.). Solo se necesitan servidores delgados (thin server) para contenido estático (HTML, JavaScript y CSS).
  + Posibilidad de llevar este servidor a la nube.

#### Requisitos No Funcionales

#### 

#### Modelo Lógico



* + - Escritorio/Vista:
* **Gestión de sesiones**: Módulo encargado de manejar la persistencia de la sesión en el cliente (navegador web), utilizando el patrón de persistencia de HTML5. Se utilizará tokens para autenticarse con API’s.

Para almacenar tokens se utilizaría dos tipos de almacenamiento:

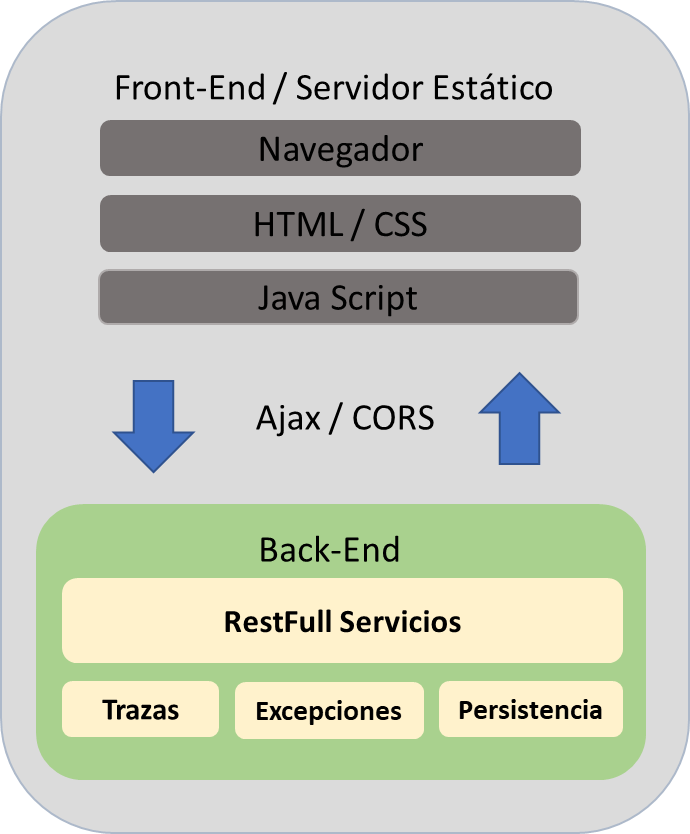
|  |  |
| --- | --- |
| **SQLite** | **Local Storage** |
| Almacenamiento completo en memoria.  Disponible solo en Chrome y navegadores basados en Webkit (Safari, Opera, etc.). | Almacenamiento compartido entre memoria y disco.  Disponible en la mayoría de los navegadores. |

Se recomienda utilizar Local Storage por la compatibilidad con cualquier navegador.

Se recomienda utilizar PC’s con las siguientes características:

* + - RAM: 1GB
    - Procesador: Core i3
    - Espacio en Disco: 50 GB
* **Personalización**: Permitir la configuración y manejo de estilos y logotipos en la aplicación a nivel de la empresa, perfil o usuario.
* **Notificaciones**: Módulo encargado del manejo de mensajes, errores y notificaciones que serán enviados desde la capa de integración.
  + - Canal
      * **Navegación**: mecanismo que permite al usuario acceso a diversos contenidos y funcionalidad.
      * **Validaciones:** capacidad de comprobar la corrección del formato y el contenido de los datos introducidos por el usuario.
      * **Componentes visuales:** creación de componentes de presentación simples o complejos que ofrecen funcionalidad facilitando al usuario el manejo de la aplicación y acelerando el desarrollo.
      * **I18n:** módulo encargado de permitir la traducción de mensajes dependiendo del lenguaje requerido por el usuario.
      * **Invocación de servicios:** módulo encargado de invocar a los servicios necesarios para recuperar la información requerida.

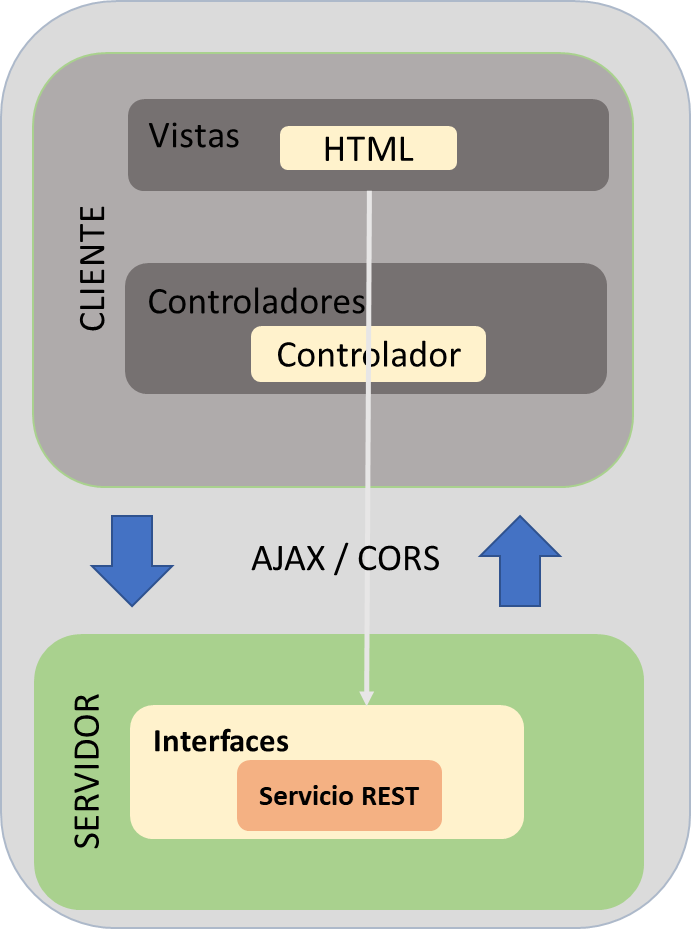
#### Modelo Tecnológico



* **Navegador**: En él encontramos los componentes que forman el patrón Cliente MVC. En el que podemos diferenciar:
  + **Vista**: Clásico modelo web que utiliza los recursos estáticos HTML5 y CSS3.
  + **Model-Controller**: Formado por framework JavaScript a los que se ha trasladado el controlador que se ubicaba en el servidor de aplicaciones. Se comunica con los servicios de negocio a través de API’s disponibles en la capa de integración.
* **Servidor liviano:** Servidor encargado de gestionar el contenido estático como HTML, JavaScript y hojas de estilo (CSS). Puede ser fácilmente migrado a la nube.



#### Modelo de Seguridad



* **Cliente**: Al ser recursos estáticos que se ejecutan en el navegador deben contener un conjunto de normas y utilidades que permitan evitar vulnerabilidades de seguridad.
* **Servidor**: Debe asegurar la fachada de servicios publicados para su consumo. Los mecanismos son los siguientes:
  + Autenticación. Los atributos de autenticación son enviados desde el Front-End hacia la capa de integración y seguridad.
  + Autorización. Los atributos de autorización son enviados desde el Front-End para acceder a los recursos del Back-End a través de la capa de integración. Estos atributos también serán usados para lograr una granularidad de autorización en la aplicación a nivel de página, opciones de menú, regiones y botones; esto debe tomarse en cuenta en la construcción de la aplicación.
  + Tratamiento de datos.
* **Comunicación**: Comunicación con el servidor HTTPS.
* **Ataques XSS**: Se deben utilizar mecanismos que eviten los ataques XSS, en este caso, dado que se va a gestionar con token deben utilizarse mecanismos para ataques CSRF.
  + Herramientas propias de los framework: *angular-sanitize*.
  + Métodos POST/PUT.
  + Gestión de cabeceras.
    - XFrameOptions: deny para protegerse contraataques de clickjacking.

### Experiencia del Usuario

Es la capa que permite la interacción directa con el usuario y KADABRA, sin intervención de personal de TI, mejorando y automatizando sus actividades.

#### Gestor de Workflow

* + - Componente que permite organizar y controlar tareas, recursos y reglas para soportar un proceso de negocio.
    - Los procesos identificados necesitan la ejecución y orquestación de tareas, así como el manejo de aprobaciones, escalamientos y notificaciones internas.
    - Es necesaria la implementación o desarrollo de este componente, debido a que, en la mayoría de los procesos de negocio TO-BE, se necesitan validaciones y aprobaciones para continuar con el flujo establecido.



#### Gestor de Parametrización

* + - Componente que permite configurar reglas de negocio que pueden ser utilizadas por procesos o tareas, sin necesidad de realizar cambios en la aplicación.
    - Permite al usuario crear o cambiar las reglas de negocio directamente, sin involucrar a personal de TI.

#### Orquestación de Procesos Batch

* + - Componente que permite controlar la ejecución de procesos en lote. Posee las siguientes características:
      * **Priorización**. Otorga una prioridad a cada proceso. Al programar varios procesos, se ejecuta el más importante primero.
      * **Programación**. Se puede programar la ejecución una sola vez o de manera repetitiva.
      * **Dependencia**. La ejecución de un proceso puede ser programada luego del término de otro. Creación de mallas de procesos.
      * **Visualización.** Interfaz gráfica que permita el monitoreo de procesos. Revisión de estado

#### Gestor de Documentos

* + - Componente que permite rastrear, gestionar y almacenar documentos.
    - Mantiene un registro de las diversas versiones creadas y modificadas.
    - Será utilizado para gestionar los documentos de sustento que sean requeridos en los flujos de procesos creados en el componente Gestor de Workflow.
    - Se encontró, en la mayoría de los procesos de negocio TO-BE, el uso de un gestor de documentos.



### Back-End

Es la capa donde se va a alojar la lógica de negocio, es decir, es el encargado de que todo funcione según lo requerido por kadabra.



#### Estándares y Patrones

##### Servicios de Negocio

* + Será soportado por una arquitectura híbrida (monolítico y microservicios) dependiendo de los siguientes criterios:

|  |  |  |
| --- | --- | --- |
| **Monolítico** |  | **Microservicios** |
| Corresponde en el caso que se requiera una integración con aplicaciones legados, satélites y externos.  Corresponde en el escenario de procesos en lotes y de alta duración.  Corresponde cuando se espera un bajo nivel de transaccionalidad o tráfico entre capas de la arquitectura. | Se utiliza para una mayor modularidad y descentralización de las aplicaciones.  Se utiliza para entregas rápidas de funcionalidades.  Se utiliza para soportar las transacciones en línea y exposición de API’s para los canales digitales.  Mayor aprovechamiento de herramientas tecnológicas de vanguardia (DevOps).  Se utiliza para un mejor enfoque de arquitectura basada en dominios. |

* + La seguridad será delegada a la capa superior o API Layer que será manejada por el patrón API Gateway.
  + El API Gateway implementará los siguientes patrones:
    - Gateway Routing
    - Gateway Aggregation
    - Gateway Offloading
  + Se utilizará el patrón de Service Discovery para obtener en tiempo de ejecución la configuración de los servicios.

##### Servicios de Datos

* + Los servicios de datos utilizarán JDBC y JPA para mapear las entidades de negocio, mediante el patrón DAO (Data Access Object).
  + Contiene el modelo canónico de datos.
  + Contiene el modelo de datos representado en un modelo de objetos.
  + Los servicios de datos de tipo consulta, podrían utilizar como origen de datos la instancia de réplica para no impactar en el rendimiento de la instancia transaccional.

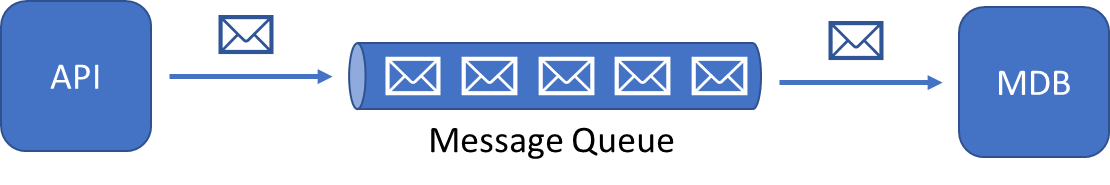
##### Servicios de Batch con Jobs

* Son de naturaleza de procesamiento Batch y son los que se ejecutan a una determinada hora de ejecución configurada mediante un scheduler y cron.
* No necesitan una condición previa para que se ejecuten.
* Serán construidos con una arquitectura monolítica utilizando el manejo de colas asíncronas para su ejecución, debido a que necesitan varias instrucciones juntas para su ejecución.
  + Utilizado en escenarios de procesos en batch masivos, lotes y de alta duración.
  + Utilizado para aplicaciones con bajo nivel de transaccionalidad o tráfico desde la capa de Front-End hacia la capa de Back-End.
  + Existe alta transaccionalidad en el interior de los procesos ejecutados en el Back-End.

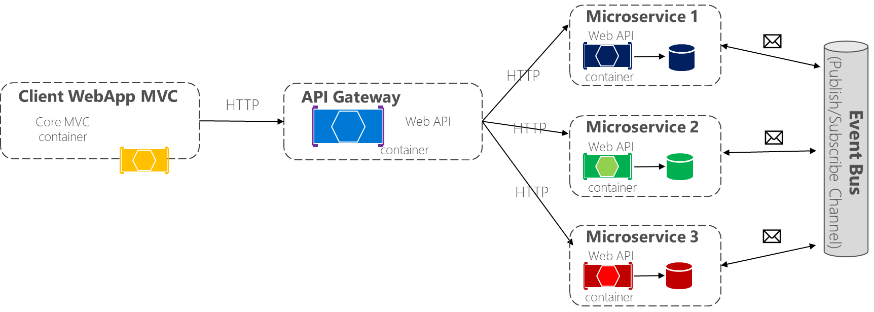
##### Servicios de Mensajería

Serán de dos tipos:

* Basados en el patrón Queue (Punto a Punto con Distribute Queue) para el manejo de mensajería asíncrona para las aplicaciones monolíticas que se utilizarán para la ejecución de Jobs Batcheros.



* Basados en el patrón Topic (Publish/Subscribe) para el manejo de eventos asíncronos en los microservicios.



Será soportado por un componente tecnológico libre con opción a soporte. Por ejemplo: RabbitMQ o Kafka, que utilizan el protocolo AMQP.

##### Servicios de Caché

* + Estos servicios serán utilizados con la finalidad de cargar en memoria los siguientes tipos de información:
* Tablas Paramétricas.
* Información de catálogos.
* Tablas maestras con poco movimiento.

#### Capacidades

| Capacidad | Descripción |
| --- | --- |
| Soporte microservicios | Estilo de programación orientada a soportar capacidades muy granulares |
| Soporte API RESTFul y SOAP | Soporte a servicios web usando protocolos REST y SOAP. |
| Transacciones Asíncronas | Permite la ejecución de transacciones con procesamiento en segundo plano. |
| Mensajería JMS | Java Message Server, protocolo JEE para el manejo de mensajería asíncrona. |
| Gestión de caché | Permite mantener en memoria persistencia de datos. |
| Gestión de errores | Maneja un catálogo de errores para respuesta a los servicios invocados. |
| Gestión de Excepciones | Manejar bifurcaciones para continuar con la aplicación o enviar a una porción de código para el tratamiento de un error. |
| Service Discovery | Permite descubrir en tiempo ejecución la ubicación de un servicio. |
| Eventos (Publish/Suscribe) | Manejador de eventos de tipo TOPIC para la ejecución de acciones asíncronas y/o reactivas. |
| Auditoría | Permite seguir el rastro de una actividad u operación de negocio. |
| Gestión de trazabilidad | Maneja la correlación de eventos para visualizar el end-to-end de una transacción de negocio. |
| Soporte ejecución de job’s | Procesos de alta duración serán soportados por job’s. Podrán ser ejecutados a demanda o programados. |
| Monitoreo Centralizado | Capacidad de recolectar todos los registros de eventos de las API’s y/o servicios para su posterior explotación y visualización. |

### Integración

Es la capa que se encarga de exponer los servicios al Front-End. Se encarga de crear servicios compuestos, lo cual es resultado de integrar llamadas a varios servicios de negocio. Asimismo, contiene “mediaciones”, que transforman datos y también enrutan las peticiones al Back-End correspondiente.



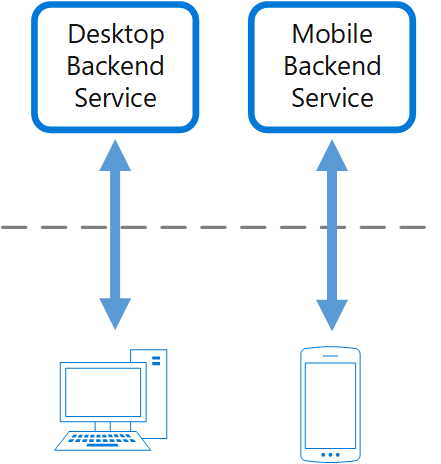
#### Patrones y Estándares

* + **API Gateway**

Esta tecnología implementa los siguientes patrones:

* + - **Backends for Frontends**

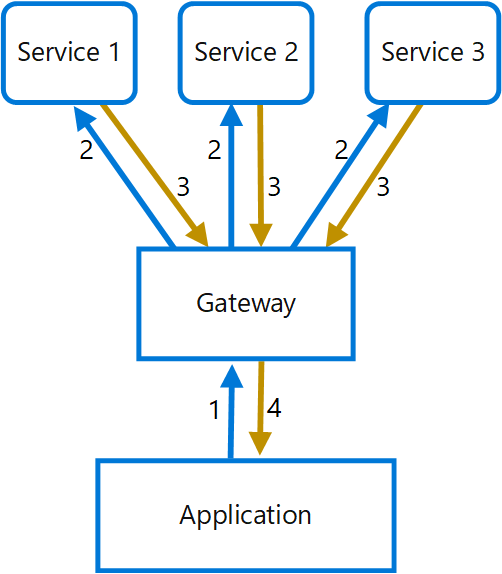
Crear API’s con la finalidad de virtualizar un servicio de Back-End y gestionar el tráfico (throttling) de los diferentes canales que puedan utilizarlo.





* + - **Gateway Aggregation**

Para agregar múltiples solicitudes individuales en una única solicitud. Este patrón es útil cuando un cliente debe realizar múltiples llamadas a diferentes sistemas backend para realizar una operación.



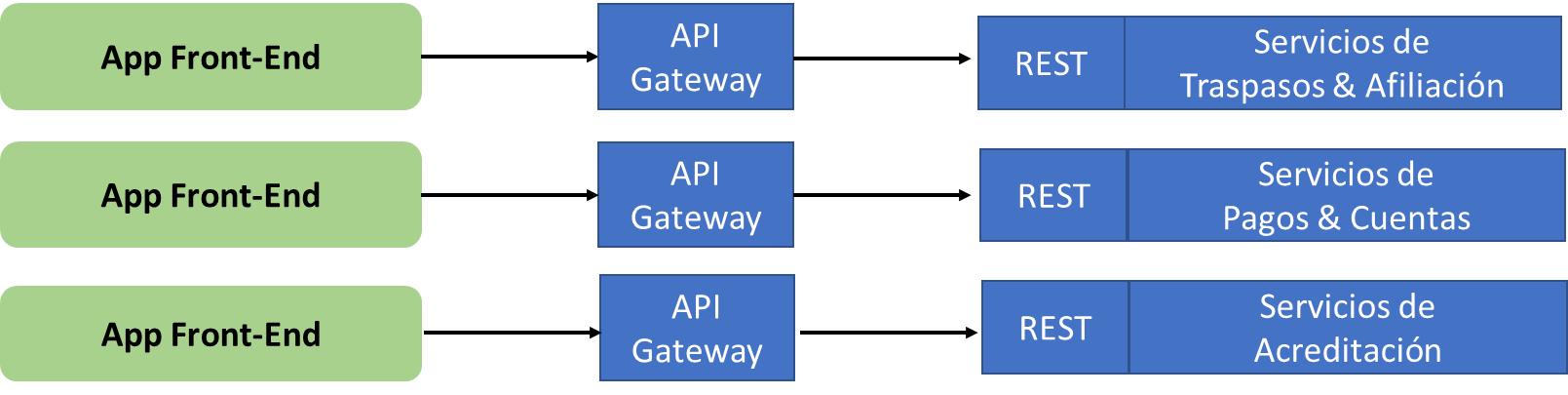
* + - **Gateway Offloading**

Simplificar el desarrollo y configuración de APIs y/o servicios eliminando la dependencia no funcional como la seguridad, registro de eventos, conectividad, transacciones, ya que una configuración o desarrollo más simple da como resultado una administración y escalabilidad más sencillas y simplifica las actualizaciones de las API’s y servicios.



* + - **Gateway Routing**

Con este patrón la aplicación frontend solo necesita conocer y comunicarse con un único punto final, si un servicio se consolida o se descompone, el frontend no necesariamente requiere actualización. Puede continuar realizando solicitudes al API Gateway solo con cambiar el enrutamiento.



* + **Swagger Service Wizard**

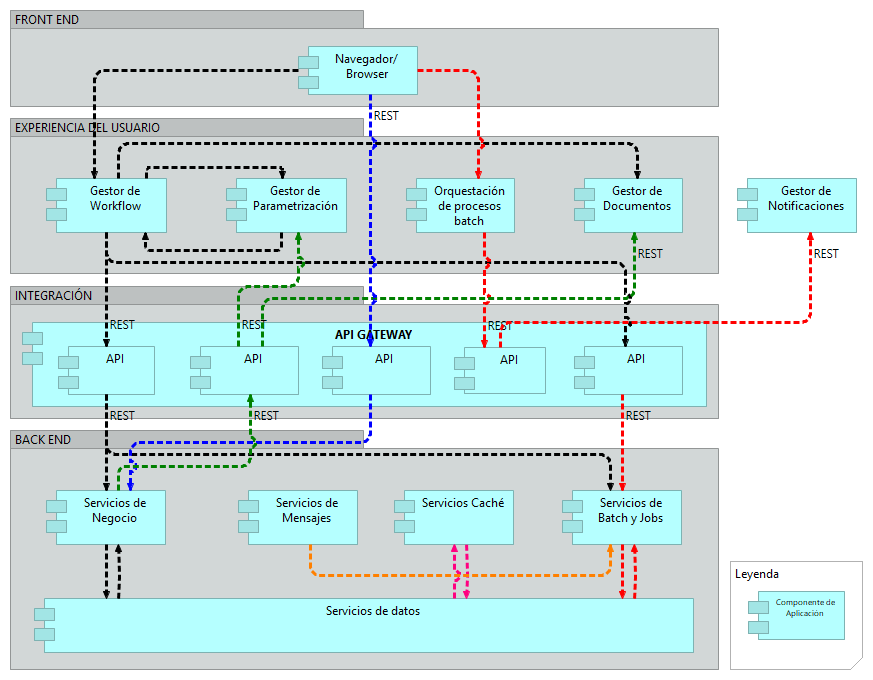
Serie de reglas, especificaciones y herramientas que nos ayudan a documentar nuestras API’s. De esta manera, podemos realizar documentación que sea realmente útil para las personas que la necesitan. Swagger Service nos ayuda a crear documentación que todo el mundo entienda.

#### Capacidades

| Capacidad | Descripción |
| --- | --- |
| Soporte API’s y Microservicios | Estilo de programación orientada a soportar capacidades muy granulares |
| Autenticación y Autorización | Acceso a los recursos desde la capa Front-End y Back-End. |
| Transacciones Síncronas | Permite la ejecución de transacciones síncronas únicamente. |
| Seguridad OWASP Top 10 | Los diez riesgos más críticos para Aplicaciones Web y servicios Web |
| Validación de Esquemas | Acción que valida las estructuras de los mensajes en los servicios web tanto SOAP como REST. |
| Ruteo | Proxy de servicios hacia la capa Back-End |
| Throttling & SLA en Servicios | Capacidad que permite limitar el tráfico de las transacciones a los servicios y API’s. |
| Múltiples protocolos | Permite la ejecución de todos los protocolos estándares de aplicación definidos. |
| Virtualización y Exposición API’s | Permite exponer interfaces fuera de la implementación de los servicios ubicados en el Back-End. |
| Orquestación y Agregación API’s | Permite reducir la cantidad de llamadas a los servicios del Back-End. |
| OAuth 2.0 | Protocolo de seguridad para manejar el accesos y autenticación a los servicios del Back-End. |
| Transformación | Permite la transformación de los esquemas y mensajes de los servicios en cualquier formato de tipo ANSI. |
| Caché | Permite mantener en memoria persistencia de API’s. |
| HSM | Permite almacenar y guardar llaves privadas. |

### Mapa de Interacción

A continuación, se muestra la arquitectura de aplicación con las relaciones entre cada uno de sus componentes:



|  |  |
| --- | --- |
|  | |
|  | El componente Workflow, dependiendo de las actividades configuradas, utiliza algún Servicio de Negocio o Servicio Batch a través de la capa de integración. También puede utilizar el Gestor de Parametrización para validaciones de reglas de negocio. (\*) |
|  | Dependiendo de las funcionalidades implementadas en los Servicios de Negocio, podrían comunicarse con el Gestor de Parametrización y Gestor de Documentos a través de la capa de integración. |
|  | El servicio de Orquestación de procesos Batch utiliza los servicios de Batch y Job’s para ejecutar cada proceso configurado en un orden y priorización establecido. Utiliza la capa de Integración para lanzar la ejecución. Puede utilizar el servicio de Notificaciones. (\*) |
|  | El servicio de Caché se comunica con el Servicio de Datos para refrescar el contenido en memoria. (\*) |
|  | El Front-End utiliza servicios de negocio publicados en la capa de Integración para su funcionamiento. (\*) |
|  | El Servicio de Mensajes se comunica con el Servicio de Batch y Jobs para la ejecución de procesos en segundo plano de larga duración. |
| (\*) Los Servicios de Negocio, Servicio de Caché y Servicios de Batch y Job’s se comunican directamente los Servicios de Datos. | |

## Arquitectura de Integración

Este diagrama muestra el mapa de integraciones y los sistemas que deben sufrir cambios para conectarse a la arquitectura KADABRA.

