**ARQUITECTURA DE SISTEMAS DE INFORMACION**

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

**EVOL**

para AFP Integra y Prima AFP



Índice

[1 Propósito 4](#_Toc521685488)

[2 Glosario de Términos 5](#_Toc521685489)

[3 Stakeholders 6](#_Toc521685490)

[4 Objetivos y Restricciones 7](#_Toc521685491)

[4.1 Objetivos de Negocio 7](#_Toc521685492)

[4.2 Objetivos de TI 7](#_Toc521685493)

[4.3 Capacidades 8](#_Toc521685494)

[5 Restricciones 10](#_Toc521685495)

[6 Cumplimiento 11](#_Toc521685496)

[6.1 Principios Estratégicos 11](#_Toc521685497)

[6.2 Principios de Datos 11](#_Toc521685498)

[6.3 Principios de Aplicaciones 12](#_Toc521685499)

[6.4 Políticas y Estándares 13](#_Toc521685500)

[7 Arquitectura de Datos To-Be 14](#_Toc521685501)

[7.1 Modelo de Datos Conceptual 14](#_Toc521685502)

[7.2 Matriz Entidad/Aplicación 14](#_Toc521685503)

[8 Arquitectura de Aplicaciones To-Be 15](#_Toc521685504)

[8.1 Arquitectura de Aplicación 15](#_Toc521685505)

[8.1.1 Front-End 15](#_Toc521685506)

[8.1.2 Back-End 20](#_Toc521685507)

[8.1.3 Integración 23](#_Toc521685508)

[8.1.4 Mapa de Interacción 26](#_Toc521685509)

Información de Documento

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre de Proyecto:** | Proyecto Kadabra | | |
| **Preparado Por:** | Nestor Cayllahua, Juan Pablo Gonzales | **Número Versión:** | 1.0 |
| **Título:** | Arquitectura de Sistemas de Información | **Fecha:** | 07/08/2018 |
| **Revisado Por:** | José Luis Rentería | **Fecha de Revisión:** | 07/08/2018 |

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 |

# 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 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 |

# 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 todo tipo de procesos durante el día. |
| 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 componentes o funcionalidades. |
| 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. |
| 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. **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 Datos

|  |  |
| --- | --- |
| **Nombre** | **Los datos son un activo** |
| **Código** | PAD01 |
| **Declaración** | Los datos son tratados como un activo que tiene valor para la empresa y se gestionan con coherencia en KADABRA. |
| **Justificación** | Los datos son un recurso de la organización para gestionar sus servicios y cuya explotación permite el análisis de la información para la toma de decisiones. |
| **Implicancias** | Asignar un administrador para la gestión de los datos.  Asegurar la calidad de los datos.  Establecer políticas y procedimientos para la gestión de los datos. |

|  |  |
| --- | --- |
| **Nombre** | **Los datos son accesibles** |
| **Código** | PAD02 |
| **Declaración** | Los datos son accesibles para cada una de las áreas del negocio permitiendo a los usuarios ejercer sus funciones. |
| **Justificación** | Brindar información oportuna, confiable y centralizada facilita la toma de decisiones. |
| **Implicancias** | Identificar la necesidad de información.  Brindar acceso a fuentes de información confiables y centralizadas mediante los mecanismos definidos por KADABRA. |

|  |  |
| --- | --- |
| **Nombre** | **Confiabilidad de los datos** |
| **Código** | PAD03 |
| **Declaración** | Se debe proteger la confiabilidad de la información almacenada. |
| **Justificación** | La confiabilidad de los datos almacenados garantiza que éstos no han sufrido manipulaciones no autorizadas. |
| **Implicancias** | Implementar medidas de control de calidad para garantizar la confiabilidad de los datos. |

|  |  |
| --- | --- |
| **Nombre** | **Definiciones de vocabulario y datos comunes** |
| **Código** | PAD04 |
| **Declaración** | Los datos son definidos de manera uniforme en KADABRA. Las definiciones son comprensibles y disponibles a todos los usuarios. |
| **Justificación** | Los datos que se utilizarán en el desarrollo de las aplicaciones deben tener una definición común.  Un vocabulario común facilitará las comunicaciones y ayuda a la integración de los sistemas e intercambio de datos. |
| **Implicancias** | Establecer un vocabulario común de datos para KADABRA.  Implementar los mecanismos para mantener actualizado el glosario de datos de KADABRA. |

|  |  |
| --- | --- |
| **Nombre** | **Seguridad de datos** |
| **Código** | PAD05 |
| **Declaración** | Se garantiza que los datos no podrán ser revisados, copiados o modificados sin las autorizaciones correspondientes. |
| **Justificación** | Se debe impedir el uso inapropiado de los datos y la información, ya sea por personas internas o externas de las organizaciones asociadas. |
| **Implicancias** | Crear políticas de acceso a la información que obliguen a realizar revisiones periódicas de las mismas.  Implementar políticas de seguridad para restringir el acceso de datos sensibles. |

## 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. |

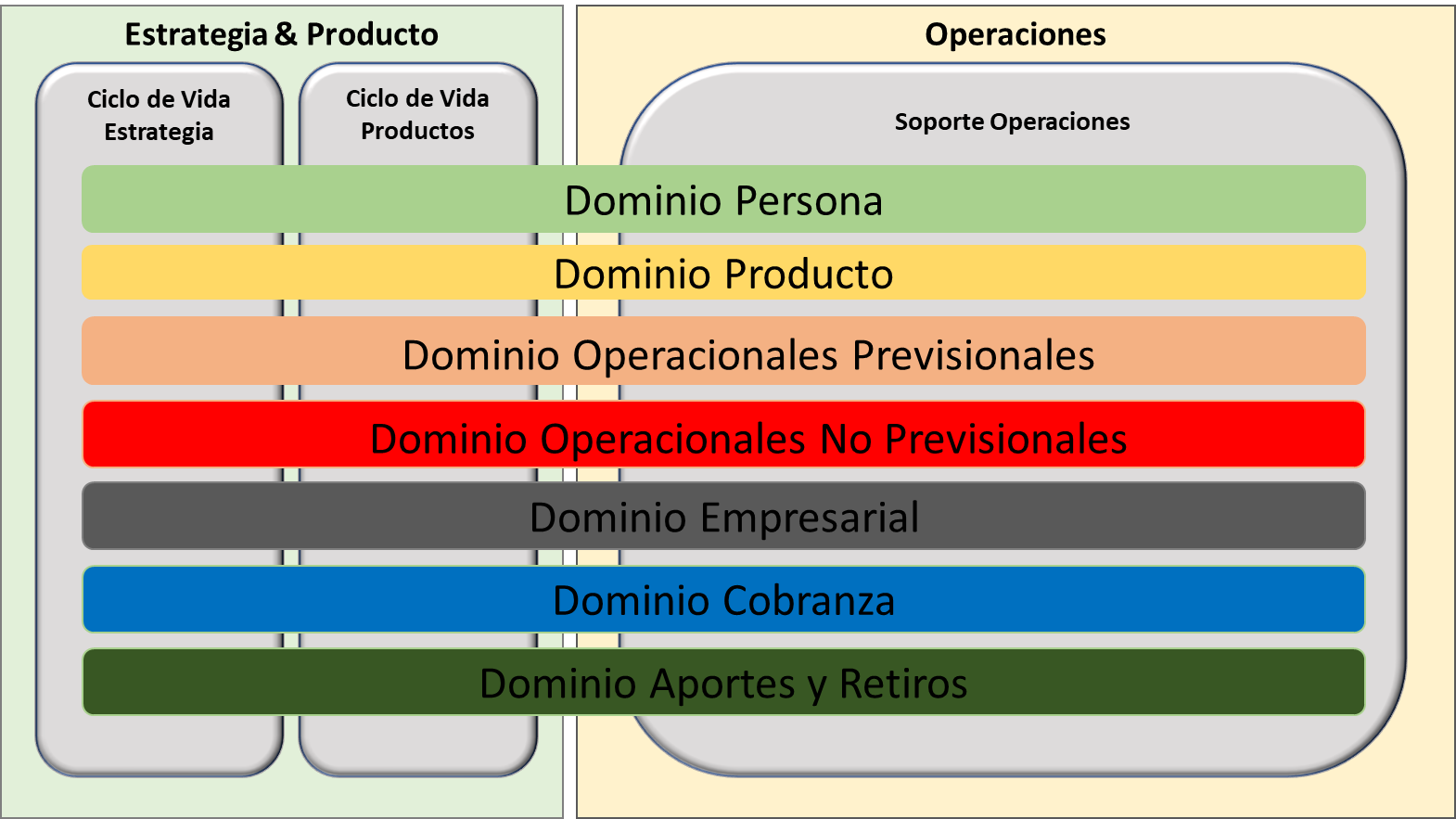
## Políticas y Estándares

* 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.
* El modelado de una arquitectura de aplicaciones forma parte del concepto de Metodología de Diseño, Desarrollo y Evaluación de Software soportado por la norma ISO 12207 que a su vez son los estándares y políticas regidas por la norma 1074 IEEE, la cual define al ciclo de vida del software como “una aproximación lógica a la adquisición, el suministro, el desarrollo, la explotación y el mantenimiento del software.

# Arquitectura de Datos To-Be

Esta sección contiene los esquemas y gráficos que representan la arquitectura de información To-Be.

## Modelo de Datos Conceptual

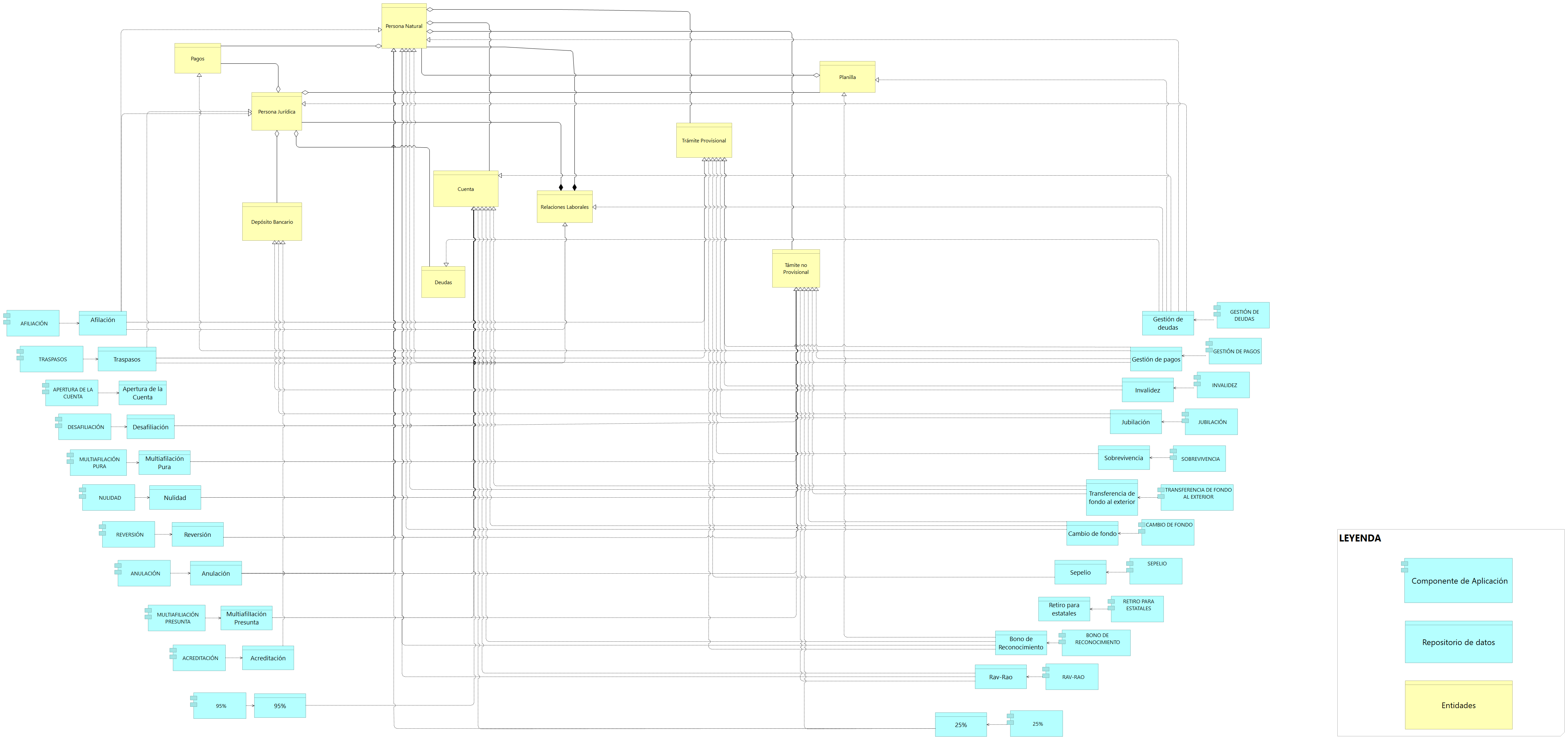


* **AGRUPACIÓN VERTICAL:** Involucra todas las operaciones tanto en línea como en batch que serán soportadas por KADABRA.
  + **Estrategia y Producto**: cubre la planificación y la gestión del ciclo de vida de la estrategia y el producto, además de dar soporte en línea a los clientes o afiliados.
  + **Operaciones**: cubre el núcleo de la gestión operacional, además de dar soporte en batch a los usuarios internos de KADABRA.
  + **Gestión Empresarial**: cubre la gestión de soporte corporativo o empresarial.
* **AGRUPACIÓN HORIZONTAL:** Esta agrupada por los dominios de KADABRA que involucran los procesos, datos y las aplicaciones.

|  |  |
| --- | --- |
| DOMINIO | DESCRIPCIÓN |
| PERSONA | Relacionado con la gestión de Personas Naturales y Jurídicas. |
| PRODUCTO | Relacionado a la gestión de cuentas de afiliados y sus movimientos. |
| OPERACIONES PREVISIONALES | Gestión de trámites con Fin Previsional. Por ejemplo, Trámites de beneficios, jubilación, invalidez y sobrevivencia. |
| OPERACIONES NO PREVISIONALES | Gestión de trámites sin fin previsional. Por ejemplo, Solicitud de afiliación, de traspaso, cambios de fondo, retiro de 25%. |
| EMPRESARIAL | Abarca las capacidades de: contabilidad, gestión documental, gestión legal y gestión del riesgo |
| COBRANZA | Gestión de la deuda tanto administrativa como judicial. |
| APORTES Y RETIROS | Relacionado a los aportes y retiros que afectan las cuentas de los afiliados. |

## 

## Matriz Entidad/Aplicación



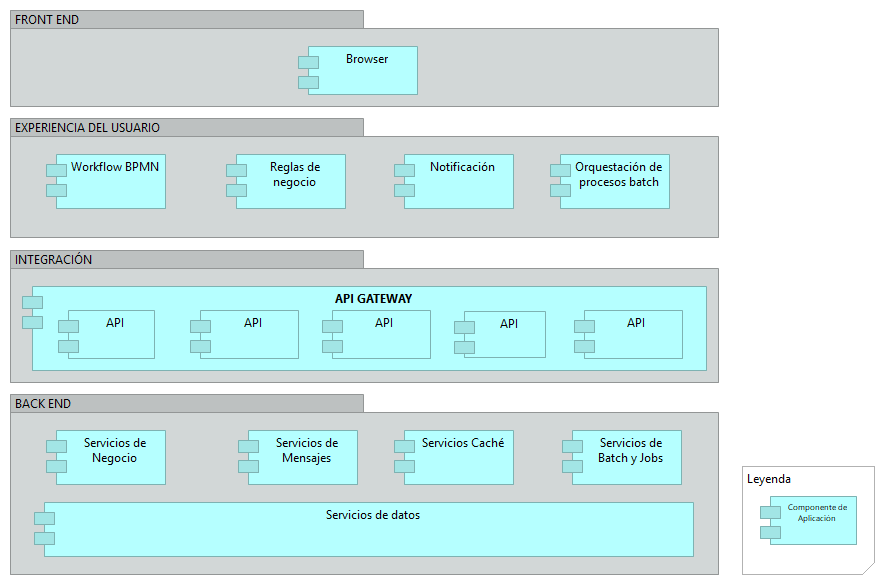
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **GRUPO DE FUNCIONALIDADES      ENTIDADES** | **Afiliación** | **Traspasos** | **Apertura de cuenta** | **Desafiliación** | **Multiafiliación Pura** | **Nulidad** | **Reversión** | **Anulación** | **Multiafiliación Presunta** | **Acreditación** | **95%** | **25%** | **RAV - RAO** | **Bono de Reconocimiento** | **Retiro para estatales** | **Sepelio** | **Cambio de Fondo** | **Transferencia de fondo al exterior** | **Sobrevivencia** | **Jubilación** | **Invalidez** | **Gestión de Pagos** | **Gestión de Deuda** |
|
|
|
| Persona Natural | X | X |  | X | X | X | X | X |  |  | X | X | X | X |  |  | X | X |  |  |  | X | X |
| Persona Jurídica | X | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X |
| Cuenta | X | X |  | X | X | X | X | X | X |  | X | X | X | X |  |  | X | X |  |  |  |  | X |
| Planilla |  |  |  |  |  |  |  |  |  |  |  |  |  | X |  |  |  |  |  |  |  |  | X |
| Pagos |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X |  |
| Tramites Previsionales | X | X |  |  |  |  |  |  |  |  |  |  |  | X |  | X |  |  | X | X | X | X |  |
| Tramite No Previsional |  |  |  | X | X | X | X | X | X |  | X | X | X |  |  |  | X | X |  |  |  | X |  |
| Deuda |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X |
| Relaciones Laborales | X | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X |
| Depósito bancario |  |  |  |  |  |  |  |  |  | X |  |  |  | X |  |  |  |  |  | X | X |  |  |

# 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:



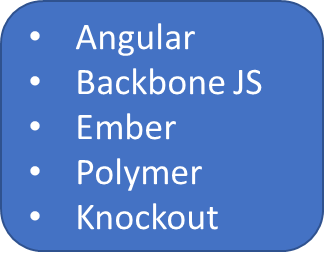
### Front-End

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.

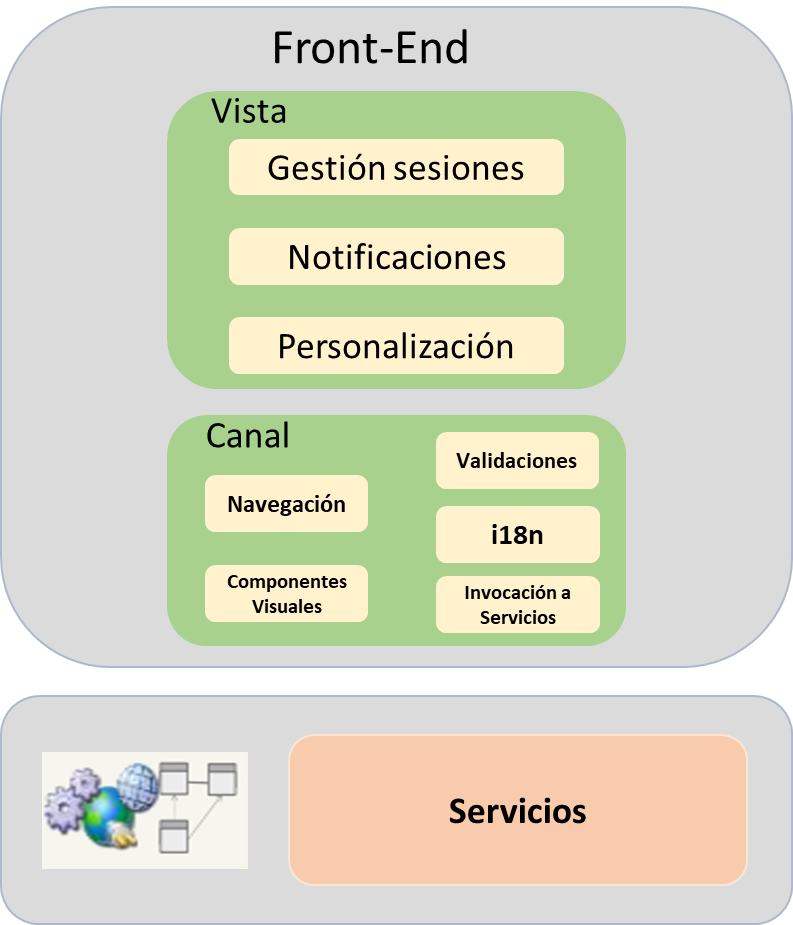
#### Estándares

* + - View
* Alto costo de personalización
* Necesidad de un servidor de aplicaciones
* Integración sencilla con framework MVC
* Madurez
  + - Server MVC
* Alta flexibilidad de integración con diversos framework (Spring, JQuery y JSF)
* Necesidad de un servidor de aplicaciones
* Alto nivel de madurez y soporte
  + - Cliente MVC
* Alto desacoplamiento entre los datos y la presentación
* Reducción del consumo de CDP
* Servidores web (thin server)
* Facilita el despliegue en Cloud (Recursos estáticos)
* Alto nivel de madurez
* La mayoría de framework se apoyan en Angular y JQuery

#### Requisitos No Funcionales

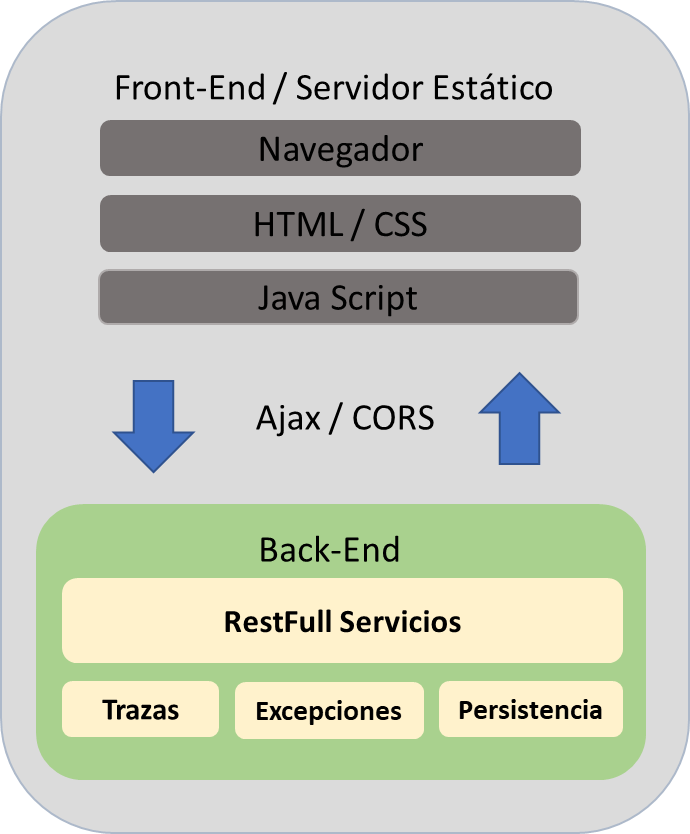
#### 

#### Modelo Lógico



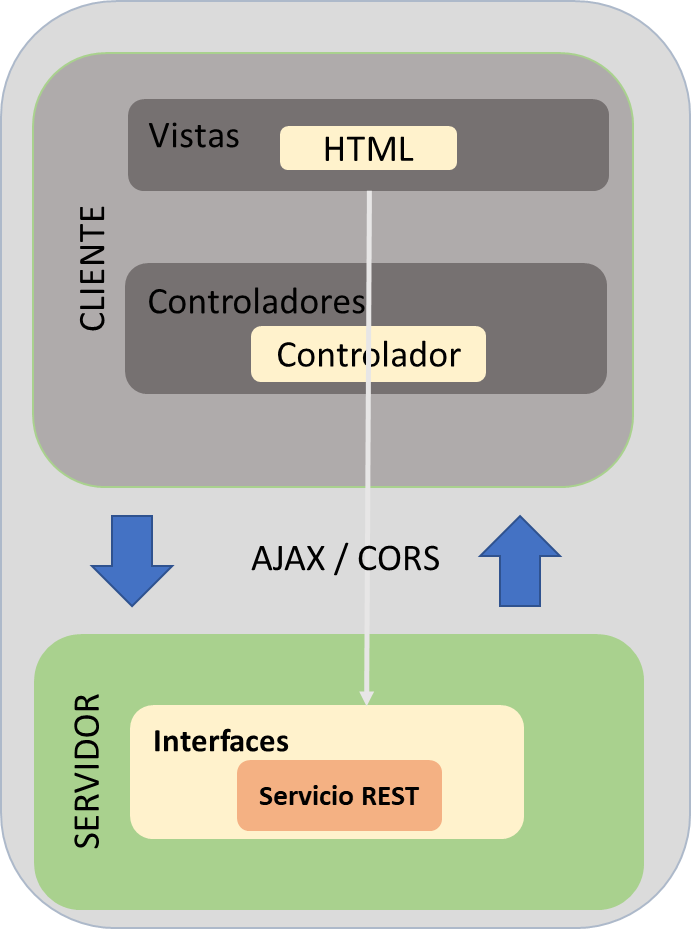
* + - Escritorio/Vista:
* **Gestión de sesiones**: módulo encargado de las operaciones de apertura, cierre y gestión de la información compartida para las distintas sesiones.
* **Personalización**: mecanismos asociados para permitir la configuración de la aplicación dependiendo de perfiles o a nivel de usuario.
* **Notificaciones**: módulo encargado del envío y recepción de mensajes.
  + - 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 el encontramos les 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.
* **Servidor ligero:** Corre en modo local tanto en PC’s como dispositivos y permite la ejecución de un alto throughput de operaciones I/O (entrada salida) locales y remotas.
* **Modelo de despliegue:** Debido a que los recursos utilizados en el navegador son estáticos, se facilita la adopción del despliegue en cloud al no contener datos propietarios.

#### 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
  + Autorización
  + 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.

### Back-End

Es la capa donde se va a alojar la lógica de negocio, es decir, es el encargado de que todo funcione según el giro de negocio de la organización.



#### Estándares y Patrones

##### Servicios de Negocio

* + Será manejado por Microservicios.
    - Mayor modularidad y descentralización de las aplicaciones.
    - Entregas rápidas de funcionalidades.
    - Soporta transacciones en línea y exposición de API’s para los canales digitales.
    - Mayor aprovechamiento de herramientas tecnológicas de vanguardia (DevOps).
    - 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
  + Por un lado, para la comunicación entre microservicios de manera síncrona será utilizando el patrón request/response, para consultas puntuales donde no se dé una cadena larga y compleja de llamadas entre estos. Para ello se usará el patrón Gateway Aggregation. Por otro lado, para la comunicación asíncrona se utilizará el patrón Publish/Subscribe para el manejo de eventos mediante el patrón Event Bus.
  + Se utilizará el patrón de Service Discovery para obtener en tiempo de ejecución la configuración de los microservicios.
  + Para el manejo de fallos y rollbacks se utilizarán las mejores practicas para la resillency:
    - Implementar el patrón de reintentos para manejo de estados.
    - Implementar el patron Circuit Breaker para aliviar una possible denegación de servicios en algun microservicio.
    - Incluir timeouts en cada llamada de microservicios.
    - Utilizar el patrón de compensación con logica para ejecutarla en el momento de un fallo.

##### Servicios de Datos

* + Los servicios de datos los microservicios implementados utilizarán JDBC y JPA para mapear la entidad de negocio.
  + Los microservicios utilizarán el patrón de acceso a datos Shared-Data es decir según la ubicación del dominio donde se encuentren accederán a entidades compartidas, dejando la posibilidad según requerimiento de negocio pueda utilizar el patrón de virtualización de datos para un microservicio específico.

##### 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 más de 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

##### 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 en los microservicios

##### 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 Síncronas y Asíncronas | Permite la ejecución de transacciones con procesamiento en segundo plano o en línea. |
| 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. |
| Orquestación de servicios | Permite manejar el estado de varias actividades dentro de un servicio de negocio. |
| 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. |

### 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.



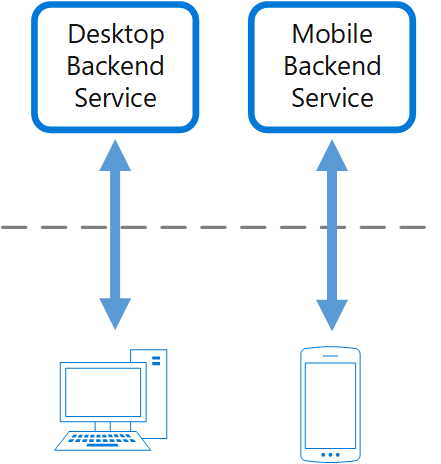
#### Patrones y Estándares

* + API Gateway

Esta tecnología implementa los siguientes patrones:

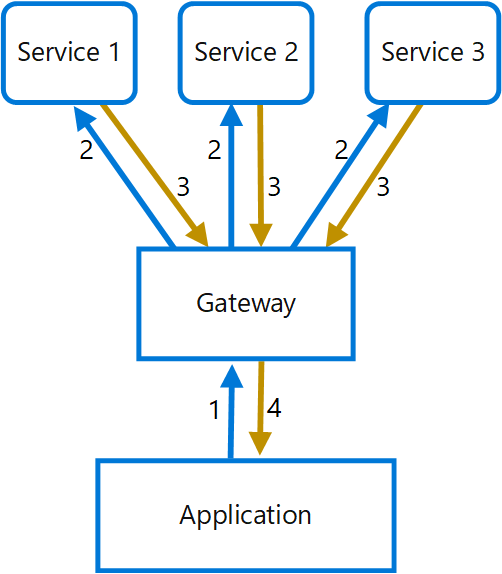
* + - Backends for Frontends

Crear una interfaz Backend por usuario (B2C2B2E), con la finalidad de aumentar la experiencia de usuario y el rendimiento en cada servicio del back-end con la finalidad que cumpla de la mejor manera las necesidades de integración, sin afectar otros canales y experiencias en la capa del Front-End.



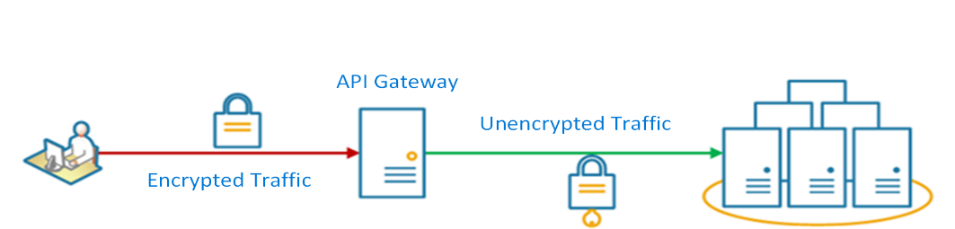
* + - Gateway Aggregation

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.



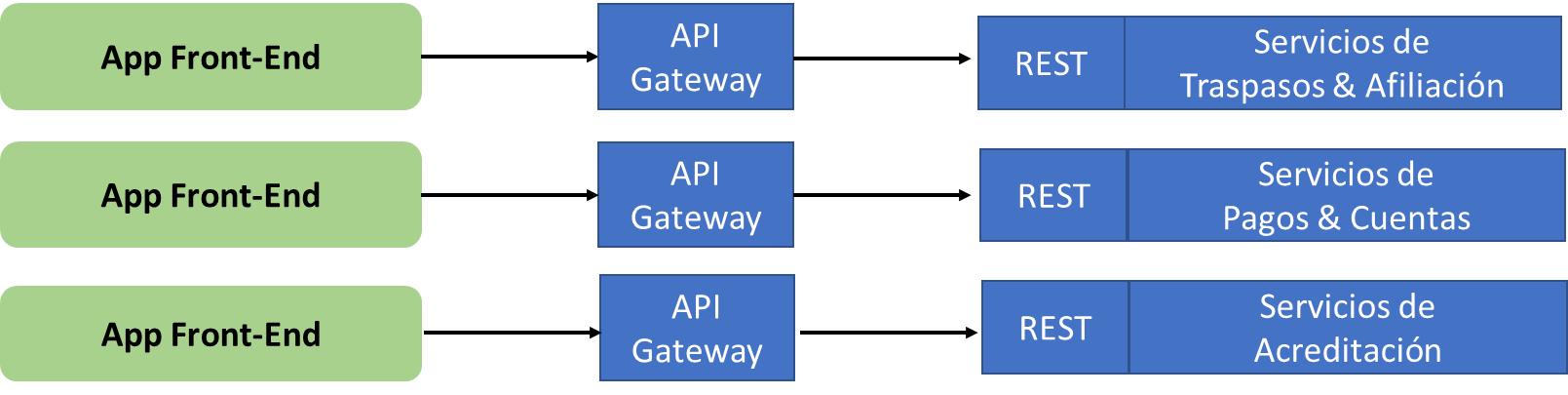
* + - Gateway Offloading

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

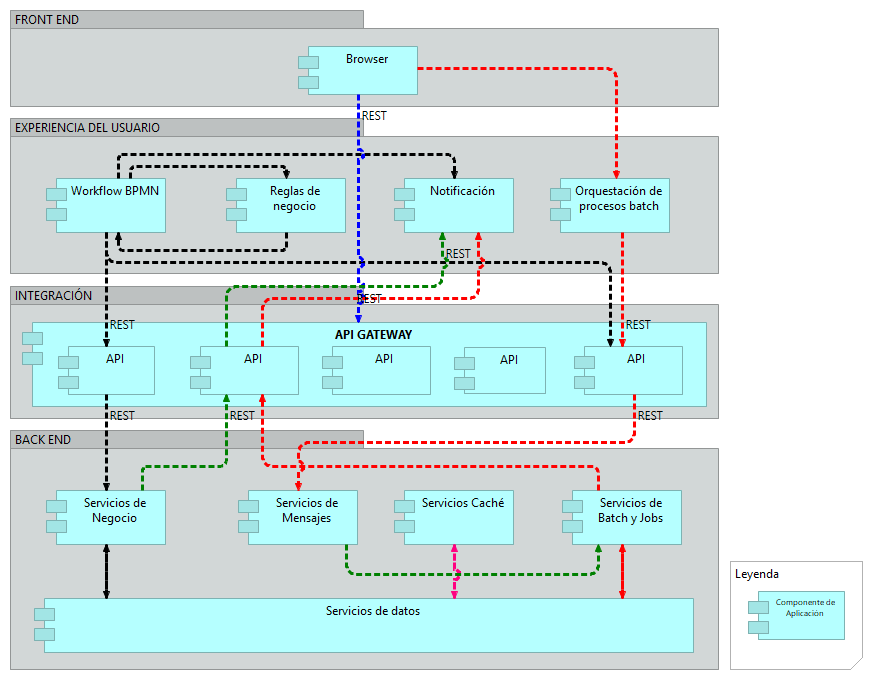
Serie de reglas, especificaciones y herramientas que nos ayudan a documentar nuestras API’s. De esta manera, podemos realizar documentación que sea realmente útil para las personas que la necesitan. Swagger Service nos ayuda a crear documentación que todo el mundo entienda.

#### Capacidades

| Capacidad | Descripción |
| --- | --- |
| Soporte API’s y Microservicios | Estilo de programación orientada a soportar capacidades muy granulares |
| Autenticación y Autorización | Acceso a los recursos de la capa Front-End y Back-End. |
| Transacciones Síncronas y Asíncronas | Permite la ejecución de transacciones con procesamiento en segundo plano o en línea. |
| Seguridad OWASP Top 10 | Los diez riesgos más críticos para Aplicaciones Web y servicios Web |
| Mensajería JMS |  |
| 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 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 datos. |
| 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 enviar notificaciones utilizando el Servicio de Notificaciones o utilizar Reglas de Negocio para validaciones. |
|  | El Servicio de Mensajes se comunica con el Servicio de Batch y Jobs para la ejecución de procesos en segundo plano de larga duración. Puede utilizar el servicio de Notificaciones. |
|  | 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 API’s publicados en la capa de Integración para su funcionamiento. |

## Arquitectura de Integración

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

