PLIEGO DE PRESCRIPCIONES TÉCNICAS

ADQUISICION DE LA ACTUALIZACIÓN NECESARIA DEL SOFTWARE DE TRADUCCIÓN DE TEXTOS CLÍNICOS DE EUSKERA A CASTELLANO ITZULBIDE

NÚMERO DE EXPEDIENTE 2022/02606

INDICE

1. OBJETO DEL CONTRATO 4
1.1. Justificación de necesidad e idoneidad del contrato 4
1.2. Justificación de insuficiencia de medios en contratos de servicios 5
1.3. Procedimiento de adjudicación 5
2. ÁMBITO Y ALCANCE DEL CONTRATO 5
2.1. Organizaciones implicadas o lugar de prestación y/o ejecución 5
3. LOTES 6
3.1. Lotes: Objeto / Descripción / Códigos. 6
3.1.1. Objeto 6
3.1.2. Descripción 6
4. REQUISITOS NORMATIVOS Y ETIQUETAS O CERTIFICACIONES GENERALES 7
5. REQUISITOS NORMATIVOS Y ETIQUETAS O CERTIFICACIONES POR LOTES 7
5.1. Requisitos normativos por lote 7
LOTE 1: 7
5.1.1. Lote 1: Requisitos normativos y etiquetas 7
6. REQUISITOS TÉCNICOS GENERALES 8
6.1 Requisitos técnicos generales aplicables a todos los lotes: 8
Debido a las características funcionales y requisitos expuestos, la solución requerirá de: 8
7. REQUISITOS TÉCNICOS POR LOTES 9
7.1. Requisitos técnicos excluyentes por lote 9
7.2. Requisitos técnicos no excluyentes por lote 9
7.3. Número máximo de lotes a los que licitar 9
7.4. Número máximo de adjudicatarios por lotes 9
8. VARIANTES TÉCNICAS 9
9. MUESTRAS / PRESENTACIÓN DE PRODUCTO / VISITA DE OBRA 9

9.1. Muestras 9
9.1.1. Forma de presentación 9
9.1.2. Normas para la identificación de las muestras 9
9.1.3. Presentación de las muestras (unidades y destinos) 9
9.2. Presentación de producto: 9
9.3. Visita de obra 10
10. LUGAR Y PLAZOS DE ENTREGA / PLAN DE PUESTA DE MARCHA 10
11. MODELO DE GESTIÓN 10
11.1. Equipo organizativo 10
11.2. Medios personales adscritos a la ejecución del contrato 10
11.3. Medios materiales adscritos a la ejecución del contrato 10
12. EJECUCIÓN / FACULTAD DE INSPECCIÓN / PLAN DE TRABAJO / PROGRAMA DE TRABAJO 11
12.1. Condiciones de ejecución 11
12.2. Modelo de control y seguimiento 11
12.3. Programa de trabajo 11

1. OBJETO DEL CONTRATO

El presente pliego de condiciones técnicas tiene por objeto definir el alcance de la adquisición de la actualización necesaria del software de traducción de textos clínicos de euskera a castellano Itzulbide.

1.1. Justificación de necesidad e idoneidad del contrato

Siendo Osakidetza un ente público de una comunidad autónoma bilingüe, el usuario tiene derecho a que se garantice el uso de cualquiera de las lenguas oficiales de la Comunidad Autónoma de Euskadi (CAE) en su relaciones con las instituciones públicas y en concreto en su relación con el sistema sanitario, en los términos previstos en la Ley 6/2003, de 22 de diciembre, de Estatuto de personas consumidoras y usuarias y en el Decreto 67/2003, de 18 de marzo de normalización del uso del euskera en Osakidetza-Servicio vasco de salud. Más recientemente, en el Decreto 147/2015, de 21 de julio, por el que se aprueba la Declaración sobre Derechos y Deberes de las personas en el sistema sanitario de Euskadi, se recoge de nuevo el derecho a utilizar cualquiera de las lenguas oficiales en la CAE.

El Marco Estratégico 2021-2024 del Departamento de Salud, establece el objetivo de incrementar el uso oral y escrito del euskera y que la relación del sistema de salud con el usuario debe desarrollarse en el idioma de preferencia del paciente. Por otro lado, el III Plan de Normalización del Uso del Euskera en Osakidetza (2022-2028), recoge entre sus objetivos que los profesionales que lo deseen trabajen en euskera en las aplicaciones informáticas corporativas, en la historia clínica, etc.

Cada vez hay una mayor demanda en relación al uso del euskera en la historia clínica (HC) tanto a nivel social como por parte de los profesionales. Más de la mitad de los empleados activos en plazas estructurales es bilingüe, al igual que las últimas promociones de grados sanitarios de la Universidad del País Vasco-Euskal Herriko Unibertsitatea (UPV/EHU). Todo esto, conlleva un aumento natural en el uso del euskera tanto en la comunicación oral entre profesionales como entre profesional y paciente, siendo aplicable en el ámbito escrito y lo previsible es que esta progresión se acelere. Actualmente gran parte de la información estructurada de la HC así como los documentos para entregar al paciente son bilingües pero Osakidetza no puede dar respuesta a uno de los pilares de la HC, el texto libre.

En el momento actual no existe en el mercado ningún traductor automático de textos clínicos de euskera a castellano o equivalente. Los traductores genéricos no cubren las necesidades específicas del ámbito de la salud con suficiente fiabilidad y calidad ya que el lenguaje/texto libre utilizado en la HC no responde a un lenguaje estándar, dado que se da un gran uso de siglas, acrónimos (muchos de ellos ni siquiera consensuados), frases inacabadas o sin verbo. No disponiendo de medios materiales para satisfacer esta necesidad, se considera esencial la contratación externa.

En enero de 2019, se publicó mediante licitación un expediente de contratación para el desarrollo de un motor de traducción de textos clínicos que fue adjudicado a la UPV/EHU. El grupo de investigación Ixa de la UPV/EHU, creó la primera versión de un motor de traducción neuronal de textos clínicos de euskera a castellano que se llamó “Itzulbide”.

1.2. Justificación de insuficiencia de medios en contratos de servicios

Debido a que la inversión incluida en este expediente no ha sido declarada de carácter estratégico y de adquisición centralizada en el Acuerdo de Consejo de Administración de Osakidetza y se estima conveniente considerar la avocación como opción más adecuada para su contratación, se propone avocar la presente contratación.”

Por todo lo expuesto, ruego el seguimiento de las actuaciones para la ejecución del expediente administrativo mencionado.

1.3. Procedimiento de adjudicación

Tal y como se justifica en el informe de exclusividad y en virtud del artículo 168 apartado c punto 2 de la Ley de Contratos del Sector Público se solicita la tramitación de este expediente por el procedimiento de **NEGOCIADO EN EXCLUSIVIDAD**:

Universidad del País Vasco-Euskal Herriko Unibertsitatea certifica mediante escrito que acompaña el presente pliego que, por motivos técnicos y comerciales, es la única entidad capaz suministrar la actualización del sistema en los términos que ha sido proyectado.

2. ÁMBITO Y ALCANCE DEL CONTRATO

El presente contrato se adjudicará para cubrir la necesidad de traducción del texto libre de euskera a castellano en la historia clínica a nivel corporativo. Además de la actualización del software de traducción tanto a nivel funcional como de calidad del mismo, se creará un banco de terminología sanitaria propia de Osakidetza que se integrará en el traductor y se diseñará el mantenimiento futuro del traductor.

2.1. Organizaciones implicadas o lugar de prestación y/o ejecución

La/s organizaciones destinatarias son todas las organizaciones de servicios y centros sanitarios de Osakidetza.

3. LOTES

3.1. Lotes: Objeto / Descripción / Códigos.

El presente contrato no se divide en lotes, se considera que el objeto es una única unidad funcional, no susceptible de ser separada en lotes y no admitiéndose variantes .

Se trata de la adquisición de una actualización cuyas partes son indivisibles por lo que está justificada su licitación en un único lote

3.1.1. Objeto

El servicio a contratar tiene por finalidad la compra de la actualización del software de traducción de textos clínicos de euskera a castellano Itzulbide. No procede dividir por razones de mayor eficiencia y coordinación en la ejecución de la prestación.

3.1.2. Descripción

En la actualidad, la interfaz está instalada junto con el modelo neuronal Itzulbide en los recursos de Eusko Jaurlaritzaren Informatika Elkartea – Sociedad Informática del Gobierno Vasco, S.A. (EJIE). Se ha realizado una prueba piloto con un reducido número de profesionales para verificar su adecuado funcionamiento y adaptación a las necesidades de la práctica asistencial. Para poder implantarlo con total seguridad en la HC es necesario actualizar el software.

La actualización además de una mejor calidad del traductor al reentrenarlo con nuevo corpus lingüístico bilingüe, conllevará la incorporación de nuevas funcionalidades como la traducción en función del sexo del paciente y la creación de un banco de terminología propia de Osakidetza para facilitar la escritura y la estandarización del euskera en el dominio clínico.

La calidad de los traductores automáticos actuales, depende directamente de la cantidad y la calidad de las traducciones utilizadas en el entrenamiento. Tal y como se ha descrito con anterioridad, los profesionales realizarán las traducciones de textos clínicos de euskera a castellano en la interfaz de Itzulbide alojado en los servidores de EJIE.

Estos textos, al ser fragmentos de evolutivos o informes de alta, son datos anonimizados, no contienen datos personales de pacientes dado que estos se encuentran directamente en la HC. Estas traducciones o nuevo corpus lingüístico, queda almacenado en el servidor de EJIE en el que se encuentra instalada la interfaz web.

El reentrenamiento de un traductor neuronal requiere de muchos datos y software muy específico que se actualiza con frecuencia, además de requerimientos específicos de

hardware como unidad de procesamiento gráfico (del inglés *graphics processing unit*, GPU). El reentrenamiento con nuevos datos puede ejecutarse utilizando distintas técnicas. Se deberán abordar las más prometedoras técnicas utilizadas en la literatura académica y así planificar la mejor estrategia de reentrenamiento para las actualizaciones futuras.

Debido al dinamismo en la investigación sobre los traductores neuronales, y las características mencionadas previamente, los reentrenamientos del traductor automático se realizarán en los servidores del grupo Ixa.

Para ello, de forma periódica, se extraerán únicamente las correcciones realizadas por los profesionales en los informes médicos anonimizados (a determinar la frecuencia en función del uso del traductor, es decir, de la cantidad de nuevo corpus creado) para entrenar y mejorar Itzulbide. Se deberá definir el procedimiento adecuado de extracción, revisión y actualización de textos que deberá garantizar, entre otros aspectos, la anonimización y trazabilidad de la información. El procedimiento a implantar deberá ser aprobado por Osakidetza.

El resultado final consistirá en disponer de un motor de traducción de textos clínicos de euskera a castellano que podrá ser integrado en la HC con garantía.

4. REQUISITOS NORMATIVOS Y ETIQUETAS O CERTIFICACIONES GENERALES

El marco regulatorio de los trabajos objetos de esta contratación está conformado, en particular, por las siguientes normativas:

- Normativa en materia de Salud.
- Normativa de Protección de Datos.
- Normativa de Seguridad de la Información.
- Normativa de Administración Digital.

5. REQUISITOS NORMATIVOS Y ETIQUETAS O CERTIFICACIONES POR LOTES

5.1. Requisitos normativos por lote

LOTE 1:

5.1.1. Lote 1: Requisitos normativos y etiquetas

No procede

6. REQUISITOS TÉCNICOS GENERALES

6.1 Requisitos técnicos generales aplicables a todos los lotes:

Debido a las características funcionales y requisitos expuestos, la solución requerirá de:

- La solución propuesta (aplicación web), debe **integrarse con el Directorio Activo de Osakidetza**.
- La solución propuesta, deberá **integrarse con los sistemas de información de Osakidetza**, en base a las tecnologías indicadas en el ANEXO I: Directrices y Especificaciones Técnicas (DET)
- El nuevo sistema se consolidará como **una solución centralizada y ubicada en el Centro de Proceso de Datos**, en adelante CPD, de la Dirección General, ubicado en Vitoria-Gasteiz.
- El sistema deberá mantener y facilitar **información de auditoría y trazabilidad** de:
    1. **Traducciones consultadas** a través de la API de integración desde la estación clínica. Al menos deberá disponer de la siguiente información:
        - DNI del profesional
        - Fecha y Hora de acceso
        - Texto seleccionado para traducir
        - Traducción de salida
    2. **Propuestas de Traducciones realizadas por el profesional** a través de la API de integración desde la estación clínica. Al menos deberá disponer de la siguiente información:
        - DNI del profesional
        - Fecha y Hora de acceso
        - Texto seleccionado para traducir
        - Traducción de salida
        - Texto PROPUESTO

La herramienta debe cumplir con la regulación europea de protección de datos, conocida como Reglamento General de Protección de Datos (**RGPD**), en cuanto a la metodología para incluir la privacidad desde el diseño. Se incluirá la realización del **Análisis de Riesgos para Derechos y Libertades** de las personas y si procede, la **Evaluación de Impacto sobre la Protección de Datos**.

7. REQUISITOS TÉCNICOS POR LOTES

7.1. Requisitos técnicos excluyentes por lote

No procede

7.2. Requisitos técnicos no excluyentes por lote

No procede.

7.3. Número máximo de lotes a los que licitar

No procede

7.4. Número máximo de adjudicatarios por lotes

No procede

8. VARIANTES TÉCNICAS

No procede

9. MUESTRAS / PRESENTACIÓN DE PRODUCTO / VISITA DE OBRA

9.1. Muestras

9.1.1. Forma de presentación

No procede

9.1.2. Normas para la identificación de las muestras

No procede

9.1.3. Presentación de las muestras (unidades y destinos)

No procede

9.2. Presentación de producto:

No procede

9.3. Visita de obra

No procede.

10. LUGAR Y PLAZOS DE ENTREGA / PLAN DE PUESTA DE MARCHA

El plazo de ejecución total del contrato será de 24 meses detallado en el plan de trabajo.

11. MODELO DE GESTIÓN

11.1. Equipo organizativo

El trabajo se realizará en coordinación con la Subdirección de Calidad, el Servicio Corporativo de Euskera y la Subdirección de Informática de la Dirección General.

Para la realización del trabajo descrito en este pliego el adjudicatario dispondrá de un equipo de trabajo compuesto por el siguiente personal:

Responsables del proyecto

Subdirección de Informática

Servicio Corporativo de Euskera

El equipo de trabajo estará respaldado por toda la organización en el caso de que fuera requerido.

11.2. Medios personales adscritos a la ejecución del contrato

El adjudicatario deberá disponer de suficientes recursos materiales, equipos, medios y personal con la cualificación necesaria para ejecutar las tareas objeto del presente pliego con efectividad.

11.3. Medios materiales adscritos a la ejecución del contrato

El adjudicatario deberá disponer de suficientes recursos materiales, equipos y medios para ejecutar las tareas objeto del presente pliego.

12. EJECUCIÓN / FACULTAD DE INSPECCIÓN / PLAN DE TRABAJO / PROGRAMA DE TRABAJO

12.1. Condiciones de ejecución

Todos los trabajos se planificarán con los diferentes departamentos implicados, coordinando las prioridades

12.2. Modelo de control y seguimiento

La Dirección General podrá establecer las medidas de control que estime necesarias para asegurar el correcto cumplimiento de los términos del contrato, así como comprobar el nivel técnico de los trabajos efectuados.

Para ello podrá realizar reuniones periódicas con el adjudicatario a fin de comprobar el desarrollo de los trabajos y el cumplimiento de lo ofertado.

Además, se prevén reuniones y conversaciones periódicas con algunos miembros del personal sanitario usuario del traductor cada 5-6 meses para realizar un seguimiento de la evolución de las actualizaciones y mejoras del mismo.

12.3. Programa de trabajo

Se identifican las siguientes tareas en el proyecto:

1. Línea de investigación para que el traductor mediante el dato “sexo del paciente” de la ficha personal del paciente pueda realizar la traducción correcta en función del mismo.

En euskera los pronombres son neutros, por lo tanto, el motor de traducción desconoce el sexo del paciente sobre el que se está escribiendo el texto, cosa que no ocurre en castellano. Esta nueva funcionalidad pretende mejorar la calidad de las traducciones forzando al traductor a generar la traducción en función del sexo del paciente.

2. Desarrollo de API (“Application Programming Interface” *o* interfaz de programación de aplicaciones). A día de hoy, el traductor neuronal está en funcionamiento en una interfaz web, fuera de la HC. Para una integración completa en la misma, es imprescindible el desarrollo de una API. Está API tendrá que cumplir con los requerimientos tecnológicos de Osakidetza detallados en apartado 6 “REQUISITOS TÉCNICOS GENERALES” del documento.

3. El motor de traducción ofrecerá la posibilidad de corregir la traducción propuesta de forma automática por Itzulbide y guardarla.

Como ya se ha mencionado, el impacto de los datos en la calidad de un traductor neuronal es clave en la calidad de las traducciones del mismo. De esta forma, se irá aumentando el

corpus de textos clínicos bilingües (corpus paralelo) y permitirá reentrenamientos del traductor para aumentar su calidad.

4. El traductor tendrá la posibilidad de añadir un check/señal si la traducción ha sido verificada/validada por algún usuario. De esta manera, aumenta la calidad y fiabilidad de las traducciones al estar validadas de forma manual por los profesionales sanitarios.

5. Itzulbide identificará el idioma de origen del texto a traducir o de partes del mismo. Esto evitará errores en el hipotético caso de que el profesional se equivocara al seleccionar el texto a traducir o la traducción de partes que ya son bilingües (por ejemplo, análisis clínicos).

6. Servicio de asesoramiento e implementación de la terminología en el entorno clínico y en euskera y línea de investigación para su integración en el traductor Itzulbide. Por un lado, se trabajará en la creación de un banco terminológico propio de Osakidetza con el apoyo del servicio corporativo de euskera, respondiendo a las necesidades de los profesionales clínicos. Por otro lado, se investigará en la integración de terminología específica y controlada en el traductor automático, y su posible personalización por servicios y/o profesionales.

ANEXO I. DIRECTRICES Y ESPECIFICACIONES TECNICAS

ÍNDICE

1 OBJETO 15
2 ESPECIFICACIONES PARA DESARROLLO 12
2.1 Arquitectura de Referencia 17
2.1.1 Vista lógica 18
2.2 Directrices de Desarrollo 20
2.2.1 Construcción de Aplicaciones por Capas 20
2.2.2 Capa de Presentación 21
2.2.3 Capa de Negocio 22
2.2.4 Capa de Persistencia 25
3 MODELO DE APLICACIONES MODERNAS 28
3.1 Especificaciones de desarrollo 38
3.2 Entorno tecnológico 40
3.2.1 Contenedores 40
3.2.2 Orquestacion de contenedores 41
3.2.3 Plataforma operativa y seguridad 42
3.2.4 Plataformas horizontales 43
4 DIRECTRICES TECNOLÓGICAS DE IOC 45
4.1 Entorno tecnológico 45
4.2 ESPECIFICACIONES DEL CLIENTE (MAQUETA) 49
4.3 Dispositivos de entrada/salida 51
4.4 Seguridad, acceso, autenticación y autorización 52
4.5 Monitorización 55
4.6 Backup/Restore 56
5 INTEROPERABILIDAD 57

5.1 Servicios Web 58
5.2 Gestor de Eventos 59
5.2.1 Estándares de comunicación 59
5.2.2 Mensajería. Definición de Evento 61
5.3 Arquitectura API REST 63
6 EXPLOTACIÓN INFORMACIÓN- BUSINESS INTELLIGENCE 65
7 SOLUCIONES MÓVILES 66
7.1 Tecnologías Cliente 67
7.1.1 Angular 67
7.1.2 Ionic Framework 5 o superior 67
7.2 Tecnologías Servidor 68
7.2.1 Nginx 68
7.2.2 Node.js 68
7.2.3 MongoDB 68
8 CALIDAD DE USO 69

1 OBJETO

Este documento, DET (**Directrices y Especificaciones Técnicas),** contiene las Directrices y Especificaciones Técnicas de los Sistemas de Información de Osakidetza en el ámbito TIC.

El objeto es incluirlo en los Pliegos de Bases Técnicas (PBTs) de los expedientes de contratación para la adquisición, mejora, evolución o adaptación de las soluciones (producto/aplicación/desarrollo) que conforman los Sistemas de Información de Osakidetza y/o cualquier otra contratación que requiera recursos TIC que sean competencia de la Subdirección de Informática de la Dirección General de Osakidetza.

2 ESPECIFICACIONES PARA DESARROLLO

Este apartado refleja los principales aspectos del diseño técnico de la arquitectura de referencia para aplicaciones web, definida por Osakidetza.

Una arquitectura de referencia es un conjunto de patrones diseñados y probados para ser usados en un contexto específico, en nuestro caso, en los sistemas de misión crítica de Osakidetza. Es una guía de ayuda para ser usada por los arquitectos y desarrolladores, que permite conceptualizar nuevas aplicaciones rápidamente reutilizando un diseño suficientemente probado.

La arquitectura de referencia refleja las decisiones y aspectos técnicos de diseño basados en la experiencia de Osakidetza en el desarrollo, mantenimiento y evolución de aplicaciones web. El principal propósito de la arquitectura de referencia para aplicaciones web es por lo tanto, homogeneizar el desarrollo, despliegue y explotación en producción de las aplicaciones.

En este sentido, la implementación de la arquitectura de referencia aportará los siguientes beneficios perseguidos por Osakidetza:

- Mejorar el rendimiento de las aplicaciones.
- Garantizar la escalabilidad y la tolerancia a fallos.
- Facilitar las integraciones entre sistemas.
- Aprovechar las tecnologías estándares y capacidades proporcionadas por los recursos hardware y software existentes en la organización.
- Reducir el coste del mantenimiento (evolutivo y correctivo) mediante la simplificación del diseño de la solución y la implementación del mismo.

2.1 Arquitectura de Referencia

Una de las características generales de los proyectos con éxito es que existe un diseño previo bien fundamentado que define la arquitectura general del sistema, las soluciones de infraestructura y plataforma (por ejemplo alta disponibilidad, recuperación frente a desastres, copia de seguridad y restauración, monitorización), y realiza un mapeo de requerimientos a elementos de la arquitectura.

Desde el punto de vista de sistemas de información, este requerimiento estratégico debe tener una base de arquitectura y diseño previo que permita cumplirlo durante la evolución del ciclo de vida.

Los **principios de la arquitectura de referencia**, describen los principios fundamentales que se han considerado para el diseño de dicha arquitectura. Los principios representan los preceptos o reglas básicas que guían el proceso de diseño.

La arquitectura de referencia se describe usando tres vistas diferentes:

- **Vista lógica** o conceptual, en la que se presenta la arquitectura mediante una visión de alto nivel que permite entender los diferentes componentes que intervienen y las interacciones existentes entre ellos.
- **Vista de implementación**, en la que se proporcionan detalles adicionales sobre la tecnología y productos para la construcción de los diferentes componentes.
- **Vista de despliegue**, en la que se describe de forma pormenorizada la información del sistema relativa a la instalación y explotación que se realiza por entornos.

2.1.1 Vista lógica

El propósito de la vista lógica es permitir un entendimiento de alto nivel de la arquitectura de referencia y de las diferentes capas y componentes en las que está estructurada. También se define la integración con otros sistemas.

Sobre esta vista lógica se deberían plasmar todos los elementos que deben ser considerados cuando se diseña una nueva aplicación.

Las aplicaciones deben de tomar este diseño como referencia e indicar:

- Las capas de la arquitectura de referencia que se implementarán
- Servicios externos con los que interactúa la aplicación
- Clientes que usarán la aplicación
- Servicios de persistencia de datos

En la siguiente figura se muestra la **vista lógica** de la arquitectura de referencia para aplicaciones web.

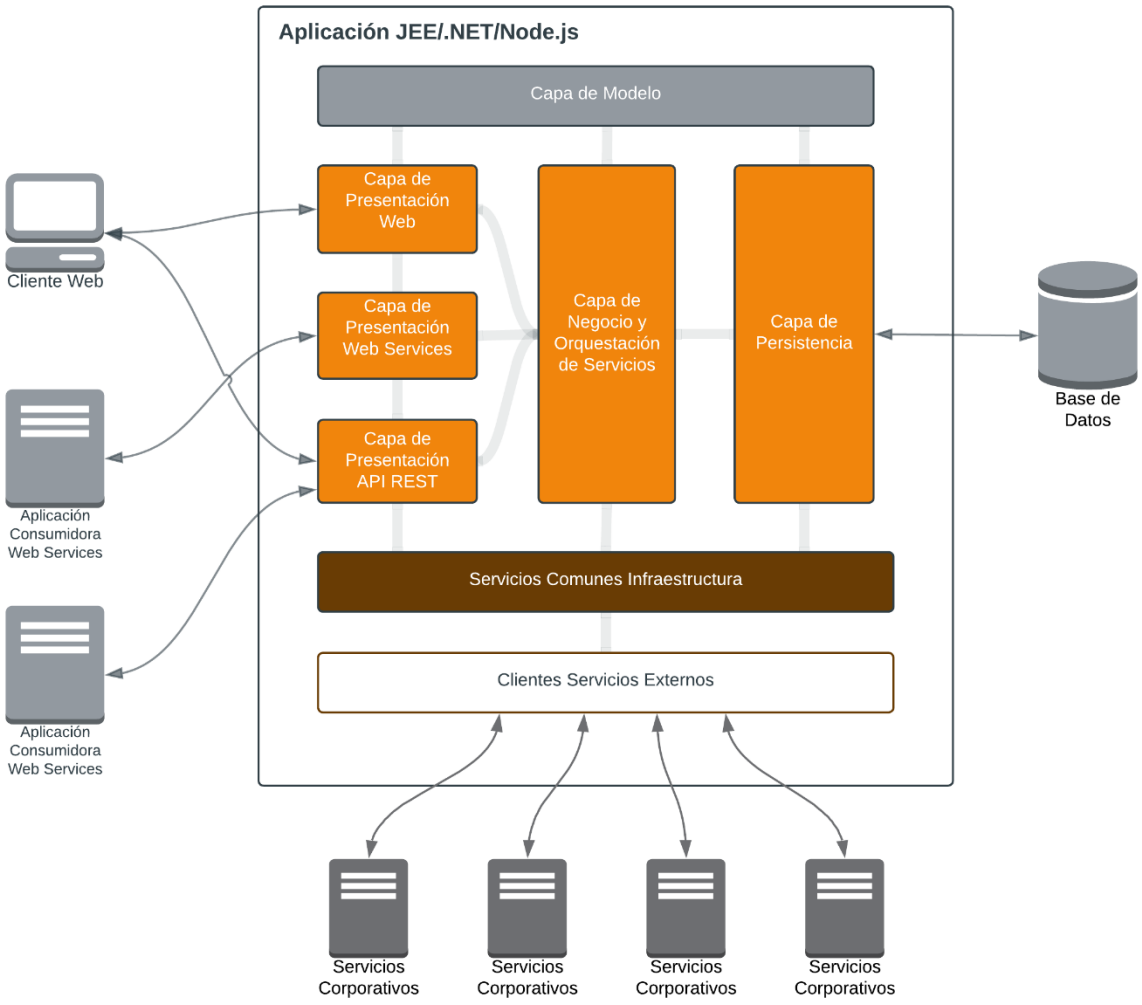


Figura 1. Vista lógica de la arquitectura de referencia

2.2 Directrices de Desarrollo

Se contemplan como directrices de desarrollo, las **normativas**, **estándares** y **recomendaciones** para la elaboración de un **código fuente homogéneo**, **estandarizado** y de **calidad**, con el objeto de **minimizar las tareas de mantenimiento** y la **deuda tecnológica**.

También se incorporan las especificaciones para la obtención de **sistemas de información seguros**, con un **rendimiento óptimo** y adaptado a las necesidades de las tecnologías definidas por las Arquitecturas de Software que se tomarán como referencia en los desarrollos.

A continuación se enumeran algunas de las **directrices de desarrollo** que deberán cumplir las soluciones desarrolladas para Osakidetza:

2.2.1 Construcción de Aplicaciones por Capas

El principal objetivo de la construcción de aplicaciones por capas, es la **separación de la lógica de negocio, de la lógica de diseño**. Su principal ventaja, es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sea necesario llevar a cabo algún cambio, sólo se necesita intervenir al nivel requerido.

El ejemplo clásico de este tipo de práctica y el que se contempla a nivel de directriz, es el consistente en separar las aplicaciones en tres capas: **Presentación**, **Negocio** y **Persistencia**.

Por lo tanto**, las aplicaciones serán desarrolladas en capas**; es decir, se diseñará las aplicaciones **separando entre los modelos de datos, la lógica de aplicación, y la interfaz de usuario**. Si optamos por esta separación de responsabilidades, podremos compartir totalmente el modelo de datos y sólo tendremos que adaptar la interfaz de usuario.

Por cada capa se consideran una serie de **directrices de desarrollo basadas en el modelo de tres capas propuesto por la Arquitectura de Referencia** de Osakidetza.

2.2.2 Capa de Presentación

Mediante la capa de presentación se separa la interacción del usuario respecto a la lógica de negocio.

El uso de la arquitectura en tres capas en el desarrollo de aplicaciones, ha favorecido la aparición de tecnologías que facilitan la implantación de esta capa, además de un conjunto de buenas prácticas que mejoran el proceso de su implementación.

Directrices de Desarrollo para la Capa de Presentación:

Finalidad: *Presentar la información al usuario*.

La capa de presentación, también llamada "capa de usuario", deberá presentar el sistema al usuario, comunicar la información necesaria y capturar la misma en un proceso que realiza un filtrado para validar los datos respecto al formato. **No deberá procesar datos ni tomar decisiones**.

Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (comprensible y fácil de usar) para el usuario.

Modularidad: *Controlar la relación entre las vistas.*

En una vista sólo debe hacerse referencia a otras vistas relacionadas con su funcionalidad. En ejecución, se recomienda integrar la vista con un **sistema de plantillas**, para poder aumentar de esta manera la **reusabilidad** y el **encapsulamiento de funcionalidades**.

Internacionalización y localización: *Preparar las aplicaciones para diferentes idiomas y convenciones.*

Las aplicaciones estarán preparadas para que puedan adaptarse a diferentes idiomas y convenciones (formatos de fecha, moneda, etc.), sin necesidad de realizar cambios de relevancia en el código.

Validaciones: Realizar validaciones sobre los datos introducidos por los usuarios.

La vista será la responsable de la validación inicial de los datos introducidos por el usuario. La validación debe ser obligatoria sólo para el caso de campos y formato. **Nunca deberá realizarse una validación de negocio desde esta capa**.

Catálogo de controles: *Elaborar un catálogo con la lista de controles.*

Se especificará, si es posible, la función del control, los posibles validadores/conversores que se le puedan asociar, los eventos que pueda lanzar y los atributos que puedan cambiarse para manipular su aspecto externo.

Los controles deberían permitir el tratamiento de componentes y eventos que posibiliten extraer de la vista toda la lógica de interfaz. La consecuencia principal de este enfoque, es que **desde la vista nunca se manipularán los controles**. El objetivo es que **la construcción de la vista sea una tarea totalmente autónoma del resto de componentes de la función de negocio**.

Uso de plantillas: Facilitar el mantenimiento haciendo uso de plantillas.

Se debe utilizar un **framework de plantillas (templates) en la capa de presentación**, ya que de esta manera se facilita el mantenimiento del diseño de los sistemas de información. Existen frameworks que posibilitan aislar el diseño de la aplicación en un único fichero de plantilla (o más, si se necesitan). **Esto facilita la tarea de hacer modificaciones en el diseño**.

Controles de interfaz: Utilizar un identificador único para los controles de interfaz.

Todos los controles de interfaz, deben mantener **un identificador único** para facilitar su **compresión**, **reutilización** y **batería de pruebas** del mismo.

Independencia entre capas: Evitar el acoplamiento entre distintas capas.

Cada capa debe tomar la responsabilidad que le corresponde. En este caso, debe evitarse que la capa de presentación realice atribuciones que le correspondan a otras capas.

2.2.3 Capa de Negocio

La capa de negocio expone la lógica necesaria a la capa de presentación para que el usuario, a través de la interfaz, interactúe con las funcionalidades de la aplicación.

Se utilizarán **componentes de negocio** para abstraer la lógica de negocio de otros problemas generales de las aplicaciones, como la concurrencia, transacciones, persistencia, seguridad, etc…

Directrices de Desarrollo para la Capa de Negocio:

Reglas de negocio: *Implementar las reglas de negocio en esta capa.*

Los módulos que requieran una funcionalidad deberán encontrarla en un solo lugar y única versión. No debe haber réplica de funcionalidades en ninguna de las otras capas (Persistencia y/o Presentación)

Validaciones de datos: Garantizar el valor correcto de los datos.

Esta capa debe garantizar que los datos requeridos para procesarla hayan sido debidamente validados, y sólo se puede iniciar el flujo del proceso de negocio si la validación resultó correcta.

Comunicación entre capas: Definir la estrategia de comunicación entre capas.

La comunicación entre capas se debe establecer mediante objetos del modelo. Se crearán entidades sin métodos, sólo tendrán propiedades, campos y atributos, que nos servirán para almacenar información, como puede ser la asociación de las propiedades a las columnas de la base de datos.

Transacciones: Tratar convenientemente las transacciones de los datos.

La capa de modelo de datos proporciona los datos al sistema, pero **las transacciones** que involucren a dichos datos, **se deben manejar desde la capa de negocio.**

Excepciones: Encapsular en la capa de negocio las excepciones y trasladarlas de forma correcta a la capa de presentación.

El objetivo es **simplificar el manejo de excepciones** de la aplicación. Se recomienda tener bien definida una **política para el tratamiento de excepciones y disponer de una jerarquía propia de excepciones**. Como norma general, **las aplicaciones no deberán nunca extender las excepciones del sistema**, sino las de su propia jerarquía de clases.

Servicios encapsulados: Encapsular la capa de negocio en servicios.

Se debe encapsular la capa de negocio en servicios, para de esta forma **aislar la base de datos respecto de una interacción directa con el usuario**. Estos servicios de negocio conforman un "puente" entre el usuario y los datos. Responden a peticiones de usuarios u otros servicios de negocio aplicando procedimientos formales y reglas de negocio a datos relevantes.

Acceso a servicios: *Controlar el acceso a los servicios en la capa de negocio.*

El control de acceso al servicio de negocio debe hacerse en la capa de negocio, puesto que podemos tener distintas capas de presentación, como por ejemplo, una capa de presentación web y una capa de presentación Web Services.

El control de acceso puede realizarse tanto a nivel de método, dentro de cada servicio, como a nivel de servicio vertical.

Inyección de Dependencias: Uso del patrón de diseño 'Inyección de Dependencias' (DI), como forma de 'Inversión de control' (IoC)

La **Inversión de Control (Inversion of Control en inglés, IoC)** es un **método de programación** en el que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales. En la Inversión de Control se especifican respuestas deseadas a sucesos o solicitudes de datos concretas, dejando que algún tipo de entidad o arquitectura externa lleve a cabo las acciones de control que se requieran en el orden necesario y para el conjunto de sucesos que tengan que ocurrir. Es el principio subyacente a la técnica de Inyección de Dependencias, siendo términos frecuentemente confundidos.

La **Inyección de Dependencias (Dependency Injection en inglés, DI)** es **un patrón de diseño orientado a objetos**, en el que se suministran objetos a una clase en lugar de ser la propia clase quien cree el objeto. La forma habitual de implementar este patrón es mediante un "Contenedor DI" y objetos planos o simples, por ejemplo los llamados POJO. El contenedor inyecta a cada objeto, los objetos necesarios según las relaciones plasmadas en un fichero de configuración. Típicamente este contenedor es implementado por un framework externo a la aplicación.

La implementación de un patrón de diseño como la 'Inyección de Dependencias' permite gestionar de una manera eficiente las dependencias entre los diferentes componentes de una aplicación, sin tener que recurrir a un acoplamiento no deseable entre estos.

2.2.4 Capa de Persistencia

La necesidad de vincular los datos guardados en una base de datos relacional, con los objetos de una aplicación orientada a objetos, determina la aparición del concepto de **persistencia de objetos**. Siguiendo el paradigma de desarrollo en tres capas, **la persistencia queda recogida en su propia capa, separada de la lógica de negocio y de la interfaz de usuario**.

La persistencia de la información es probablemente la parte más crítica en una aplicación de software y permite ejecutar operaciones en el almacenamiento persistente (base de datos). El modelo de objetos de un sistema, difiere en muchos aspectos del modelo relacional. La interfaz que une esos dos modelos se llama asociación objeto-relacional (ORM en inglés). La capa de persistencia encapsula el comportamiento necesario para mantener los objetos.

Directrices de Desarrollo para la Capa de Persistencia:

Asociación Objeto-Relacional: Usar un mapeador Objeto Relacional para implementar la capa de persistencia.

Se debe utilizar un mapeador Objeto-Relacional para implementar la capa de persistencia, ya que es una técnica que permite convertir los tipos de datos utilizados en un lenguaje

de programación orientado a objetos y los utilizados en una base de datos relacional, lo que posibilita el uso de las características propias de la orientación a objetos.

Uso del patrón DAO: Crear una clase DAO por cada objeto de negocio del sistema

El **patrón CRUD**, reconocido como el patrón más importante del acceso a datos, indica que **cada objeto debe ser creado en base de datos para que sea persistente**. Para esto es necesario asegurar que existen operaciones que permiten a la capa inferior (de acceso a datos) leerlo, actualizarlo o borrarlo.

Mediante la implementación de DAO's pueden proporcionarse las operaciones CRUD necesarias para cada aplicación. No es obligatorio que cada DAO implemente todas las operaciones CRUD (puede no ser necesario en la lógica funcional del sistema). Por lo tanto, **se recomienda crear un DAO distinto por cada objeto de negocio en el sistema**.

Manejo de la caché: Usar la caché en las aplicaciones para reducir tiempos de lectura.

Se recomienda **utilizar un sistema de caché** en las aplicaciones, para reducir tiempos de lectura, ya que la mayoría de accesos de lectura acceden a una pequeña parte de los datos de la aplicación. Normalmente, hay un conjunto de datos que son relevantes a todos los usuarios y que, por lo tanto, son accedidos con más frecuencia.

Concurrencia de usuarios: Permitir la concurrencia de usuarios.

Se debe permitir que varios usuarios trabajen en la misma base de datos, **protegiendo los datos de ser escritos erróneamente**. En la mayoría de los casos, se utilizará el bloqueo optimista, soportado por la mayoría de los motores de persistencia como la opción por defecto.

Referencias circulares entre objetos: Evitar las referencias circulares entre objetos de la capa de persistencia.

Se deben evitar las referencias circulares, facilitando la localización de los objetos. De esta manera, se devuelve el objeto solicitado sin necesidad de realizar un recorrido para localizarlo, lo que supone un acceso más efectivo.

Actualización en cascada: Utilizar la actualización en cascada siempre que sea posible.

Utilizar la posibilidad que ofrecen algunos frameworks para la actualización en cascada de objetos. Una actualización en cascada **permite que las modificaciones hechas a un objeto se repliquen en los objetos relacionados**. De esta manera se mejora el mantenimiento de los objetos, se asegura que los cambios introducidos se replican de manera eficiente y se mantiene la **integridad de los datos**.

3 MODELO DE APLICACIONES MODERNAS

Osakidetza se encuentra en un proceso de renovación de la arquitectura tecnológica, con el objeto de adaptarla a las prácticas más actualizadas.

La arquitectura de aplicación tradicional de tres niveles (que comprende un navegador, una aplicación monolítica implementada en un servidor de aplicaciones y una base de datos) es inadecuada para respaldar nuevas demandas comerciales digitales. Las aplicaciones monolíticas inflexibles, la complejidad arquitectónica y la deuda técnica sobrecargan a los equipos de desarrollo e impiden la agilidad llegando a frustrar a los usuarios. La modernización de la cartera de aplicaciones facilitará la transformación del negocio digital, este esfuerzo de modernización comienza con una arquitectura de aplicación moderna.

Las demandas de "negocios digitales" conducen hacia una rápida transformación de aplicaciones, sistemas y procesos, así como la necesidad de aplicaciones innovadoras de múltiples experiencias y obligan a brindar soluciones flexibles que puedan adaptarse rápidamente a los cambios en los requisitos y los avances tecnológicos.
Una arquitectura ágil basada en tecnologías modernas de interfaz de usuario (UI), API y servicios poco acoplados brindan la agilidad y flexibilidad para satisfacer estos requisitos. Este enfoque de arquitectura es lo que el mercado llama MASA (Mesh App and Service Arquitecture), arquitectura de aplicaciones y servicios en malla.
MASA (Mesh App and Service Arquitecture) proporciona tres capacidades arquitectónicas para potenciar el negocio digital:
multiexperiencia (múltiples experiencias de usuario), este enfoque brinda la oportunidad de crear aplicaciones específicas más pequeñas que son más fáciles de diseñar, desarrollar e implementar. También permite que la aplicación se concentre en cualquier canal que prefiera el usuario al realizar tareas, ya sea una aplicación basada en la web que se ejecuta en una computadora portátil, una aplicación móvil en un teléfono o una interfaz conversacional.

- mediación de API (manejo de las API programáticas), un principio crítico en MASA es garantizar que las partes individuales de la aplicación puedan cambiar según sea necesario sin afectar otras partes de la aplicación. Debido a que las API pueden causar dependencias estrechas entre los componentes de la aplicación, todas las interacciones de la API deben abstraerse para garantizar un acoplamiento flexible.

  Monitorización de interacciones: la mediación de API puede monitorear interacciones desde perspectivas técnicas y comerciales.
  Protección de las interacciones: la mediación de la API puede hacer cumplir las políticas de autenticación y autorización y evitar las intrusiones de malware.
  Gestión del flujo de tráfico: la mediación de API puede admitir el

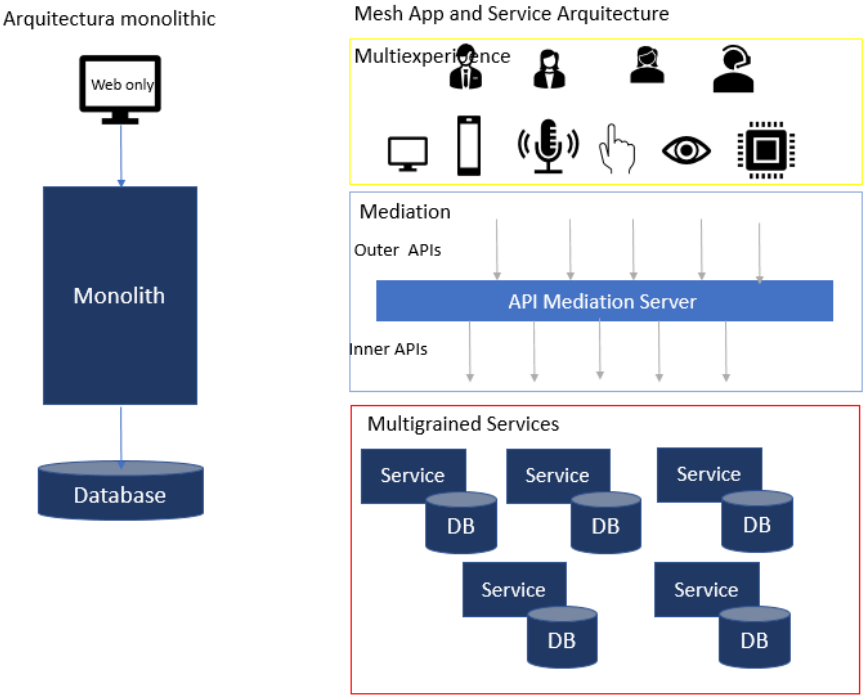
enrutamiento de solicitudes, el equilibrio de carga, la limitación, la redirección y la recuperación automática de interacciones fallidas.
Gestión de transformaciones: la mediación de API puede traducir entre diferentes protocolos: Puede transformar mensajes, por lo que admite formatos personalizados para diferentes consumidores y asignaciones entre diferentes versiones de API.

- servicios multigrano (microservicios, mini servicios o web services), cada servicio encapsula una capacidad comercial y expone esa capacidad a los consumidores a través de una API. El tipo de API más común utilizado en MASA es una API REST, pero MASA también admite otros tipos de API de solicitud/respuesta (como SOAP y gRPC), así como API basadas en mensajes y eventos.

  Macroservicios granulares que encapsulan y exponen algunas o todas las capacidades de una aplicación monolítica sin refactorización.

  Miniservicios de implementación independiente y granulares que implementan la funcionalidad a nivel de dominio.

  Microservicios detallados y de implementación independiente que implementan características individuales.



Aplicaciones multiexperiencia

La estrategia de aplicaciones front-end en MASA aplica un enfoque modular y granular para crear aplicaciones de clientes especializadas. Este es un cambio que se aleja del enfoque tradicional donde la intención es poner tanta funcionalidad como sea posible en las aplicaciones de clientes individuales. En lugar de una sola aplicación de cliente compleja, la experiencia del usuario de una aplicación de malla se compone de aplicaciones enfocadas estrechamente alineadas con flujos de trabajo específicos requeridos por diferentes personas que trabajan a través de varias interfaces .
Esto brinda la oportunidad de crear aplicaciones más pequeñas y con un enfoque limitado que pueden usar múltiples servicios desde el back-end. Esto también le permite enfocar las aplicaciones en los canales preferidos del usuario para realizar sus tareas, ya sea una aplicación web en una computadora portátil, una aplicación móvil en un teléfono o una aplicación en una interfaz conversacional.
En una aplicación grande y monolítica, los usuarios deben navegar para encontrar la parte de la funcionalidad que necesitan para realizar las tareas específicas para las que están usando la aplicación. Sin embargo, en una aplicación multiexperiencia adecuada, la aplicación debe ser intuitiva, hacer que las tareas del usuario sean lo más eficientes posible y admitir el canal en el que el usuario desea utilizar la aplicación.

El diseño de la aplicación utilizando personas y escenarios implica un proceso de cuatro pasos:

1. Crear personajes de usuario para establecer una comprensión precisa de sus usuarios. Use las personas para identificar cosas como preferencias de dispositivos, nivel de experiencia, comportamientos y limitaciones de tiempo.
2. Identifique escenarios para comprender mejor las tareas que requieren los usuarios. Los escenarios deben revelar el orden de las tareas requeridas, los pasos para realizar las tareas y las expectativas específicas que los usuarios pueden tener al realizar las tareas en una aplicación.
3. Diseñe flujos de trabajo basados en las personas y los escenarios. Estos flujos de trabajo deben identificar el flujo natural de trabajo, así como las posibles desviaciones que pueden experimentar los usuarios. Tenga en cuenta que la aplicación puede admitir varias personas y flujos de trabajo.
4. Utilice los flujos de trabajo para impulsar el diseño de la aplicación. Implementación de diseño de páginas web/móviles u otras interfaces, así como la navegación entre ellas, para soportar de manera óptima los flujos de trabajo definidos.

Además de diseñar las aplicaciones para que proporcionen una apariencia coherente en varias aplicaciones, la apariencia debe ser coherente en varios canales. Se utilizarán marcos, plataformas y herramientas comunes que garanticen la coherencia entre las aplicaciones y simplificar su implementación:

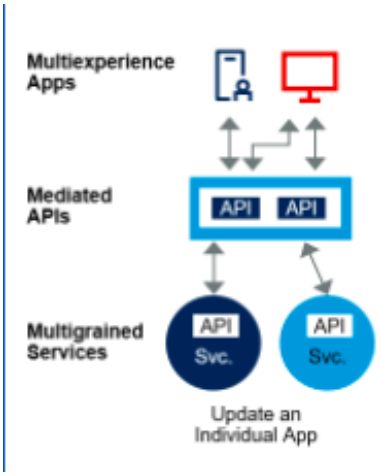
Frameworks de JavaScript como Angular, React, Vue.js y Polymer para aplicaciones web personalizadas
Frameworks móviles como Xamarin, React Native e Ionic para aplicaciones móviles personalizadas
Plataformas de experiencia digital (DXP)
Plataformas de aplicaciones de código bajo (LCAP) y plataformas de desarrollo multiexperiencia (MXDP)

Mediación de servicios, API y eventos

Las API mediadas permiten ofrecer API optimizadas para satisfacer las necesidades de consumidores específicos. El enfoque de API mediadopermitirá abstraer las complejidades en la implementación del servicio, así como para implementar un control de acceso, seguridad y administración consistentes.

Un enfoque de API mediada implica la implementación de tres capas distintas:

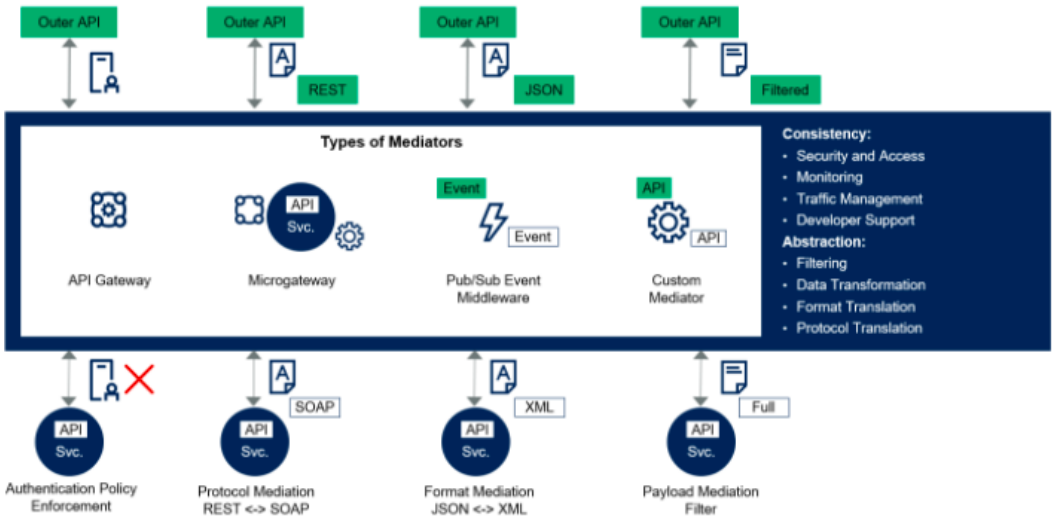
- API internas habilitan el dominio nativo del servicio y los modelos de interacción.
- API externas proporcionan una capa uniforme para que las aplicaciones accedan a la funcionalidad de back-end. Las aplicaciones pueden compartir una API externa o tener una API única. Las aplicaciones usan API externas como parte de su plataforma y, por lo tanto, las tratan como productos de consumo individuales.
- La mediación de API asigna las API externas a las API internas, monitorea y registra las interacciones y aplica políticas de tiempo de ejecución para la seguridad y la gestión del tráfico. Esto incluye actuar como intermediario para traducir protocolos, formatos, estructuras de datos y otras transformaciones a medida que las solicitudes pasan entre las API internas y las API externas.



Implementar una capa de mediación para administrar las API

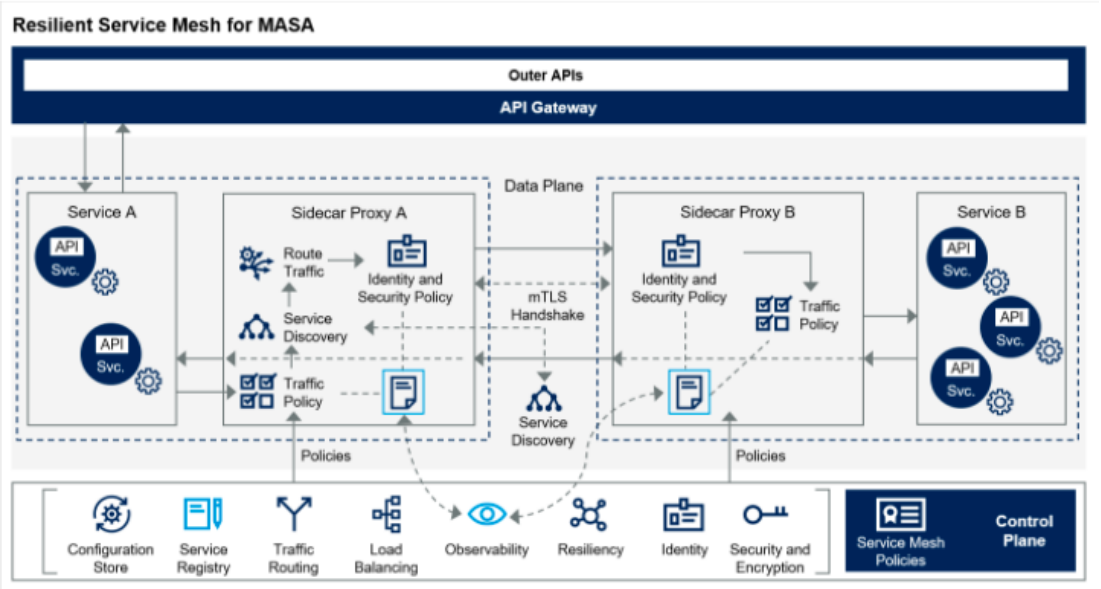
La capa de mediación permite abstraer las complejidades de los servicios de las preocupaciones del cliente de la API externa. La capa de mediación puede usar uno o más mediadores, tecnologías que se ubican entre las API externas que usan las aplicaciones cliente y los servicios. Los contratos técnicos definido en la API externa y el diseño de la API interna ayudan al diseño y la implementación de la capa de mediación. Mapee los requisitos técnicos de la API externa con las capacidades actualmente expuestas por el servicio. Por ejemplo, la API externa puede requerir una interfaz REST mientras que el servicio expone una interfaz SOAP.

Una puerta de enlace API(API gateway) es el lugar óptimo para comenzar con la mediación porque proporciona de manera efectiva una capa consistente para aplicar políticas de administración de seguridad, identidad, monitoreo y tráfico. Además, algunas puertas de enlace API pueden usar políticas para filtrar o transformar datos.



Implementar capacidades de Service Mesh para admitir servicios multigrano

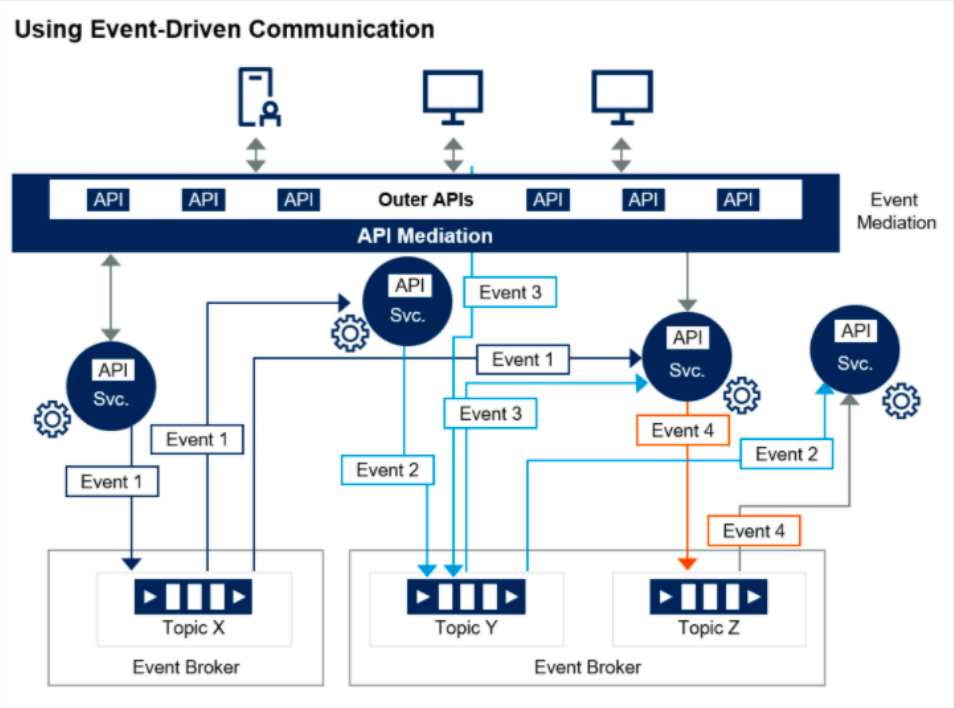
Construir un MASA con servicios multigrano requiere capacidades para habilitar y gobernar las interacciones entre servicios. Una red de servicios (service mesh) es una combinación de componentes de infraestructura, arquitectura y configuración, como puertas de enlace internas o proxies y mediación ligera que proporciona un modelo de comunicación coherente entre servicios débilmente acoplados. Una red de servicios (service mesh) es parte de la capa de mediación de API y es un componente importante de MASA que permite un modelo consistente para definir, implementar y administrar las complejidades de la comunicación entre múltiples servicios desacoplados.



Modelo basado en eventos para la comunicación desacoplada

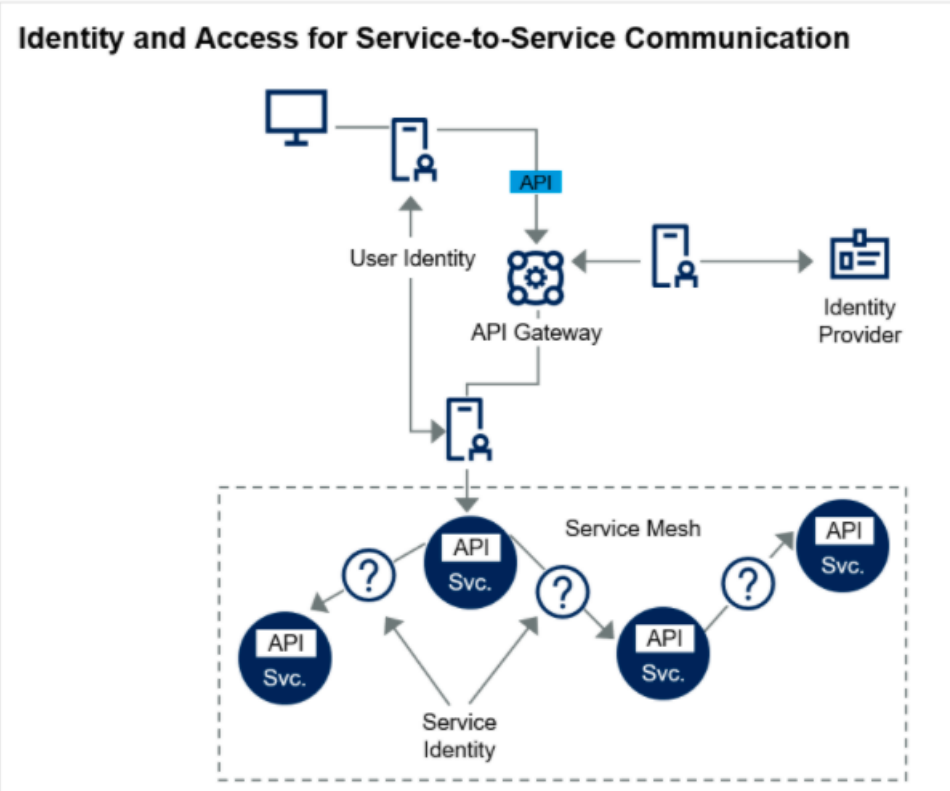
Un modelo de comunicación basado en eventos proporciona un método altamente desacoplado y eficiente para comunicar cambios de estado y otros eventos entre servicios. En un modelo de comunicación basado en eventos, los editores de eventos envían eventos a un tema y los consumidores de eventos se suscriben a esos temas y ejecutan la funcionalidad cuando reciben un evento.

Un modelo de comunicación basado en eventos proporciona un mayor grado de desacoplamiento que un modelo de comunicación tradicional de solicitud/respuesta. Los productores de eventos no tienen conocimiento directo de los consumidores intermedios y no dependen del procesamiento de eventos: simplemente notifican al corredor que se ha producido un evento. Los consumidores de eventos tampoco tienen dependencias directas de los productores de eventos: simplemente ejecutan la funcionalidad cuando reciben un evento.



Aplicar una estrategia de identidad y acceso para API y servicios multigrano

Aprovechar la mediación para externalizar la autenticación y la autorización de los componentes de la aplicación. Una aplicación de malla que utiliza la comunicación de servicio a servicio agrega una dimensión adicional a la identidad y el acceso. La protección solo en la puerta de enlace es un antipatrón. Los servicios necesitan protegerse a sí mismos; no asuma que una red es segura.



La estrategia de identidad y acceso admite la autenticación para las solicitudes de servicio a servicio, además de las solicitudes de las aplicaciones. Usar un servicio de puerta de enlace para convertir los tokens de autenticación de los usuarios (como contraseñas) en tokens de seguridad y páselos entre los servicios. Utilizar la mediación en la red de servicios para administrar la identidad y el acceso de la solicitud en la comunicación de servicio a servicio.

Implementar servicios multigrano

Los servicios son un componente clave para implementar una aplicación utilizando MASA. Los servicios proporcionan la lógica comercial, el acceso a datos y recursos, y otras funciones de back-end requeridas por las aplicaciones. Los servicios exponen esta funcionalidad a través de las API internas que reflejan su modelo de dominio interno y su modelo de interacción nativo, es decir, sus API internas.
El propósito de las API internas es exponer la funcionalidad de los servicios para que la use la capa de mediación. Las API internas desacoplan y abstraen la funcionalidad del servicio para proporcionar flexibilidad, agilidad y escalabilidad en la implementación y despliegue de los servicios. Las API internas deben centrarse en el dominio del servicio para el consumo general de otros servicios y mediadores. Esto es diferente de las API externas, que se enfocan en las necesidades particulares de las aplicaciones de los clientes.
La funcionalidad de aplicaciones empresariales, las aplicaciones heredadas, SaaS y los servicios de integración habilitados a través de API son ejemplos de implementaciones de servicios potencialmente disponibles para las aplicaciones MASA.
Los patrones comunes de integración de servicios personalizados incluyen:

- Orquestación y agregación: las aplicaciones desacopladas en MASA a menudo requieren interacciones con múltiples sistemas de back-end. Los servicios de orquestación personalizados coordinan las solicitudes a múltiples sistemas de back-end para proporcionar un acceso unificado a los datos y el procesamiento de esas aplicaciones. Esto reduce la cantidad de solicitudes que las aplicaciones deben enviar al back-end. Un uso común de la orquestación son los servicios de agregación, que combinan datos de varios sistemas back-end y pueden aplicar proyecciones, filtros y otras operaciones para optimizar los datos.

- Coreografía: el patrón de coreografía y orquestación le permite realizar una única solicitud a un servicio personalizado que inicia el procesamiento en varios servicios de back-end en un orden específico o a un conjunto específico de sistemas de back-end.

- Backend para frontend: las aplicaciones multiexperiencia a veces requieren una API de back-end especializada que se aplique específicamente a sus necesidades. En lugar de diseñar e implementar un back-end único y completo, un servicio personalizado que abstrae las necesidades específicas de la aplicación de la implementación del back-end le permitirá utilizar las capacidades del back-end existentes.

- Procesamiento por lotes: los servicios de integración personalizados proporcionan un método adaptable para implementar el procesamiento por lotes de datos de una variedad de fuentes de datos en un sistema de back-end.

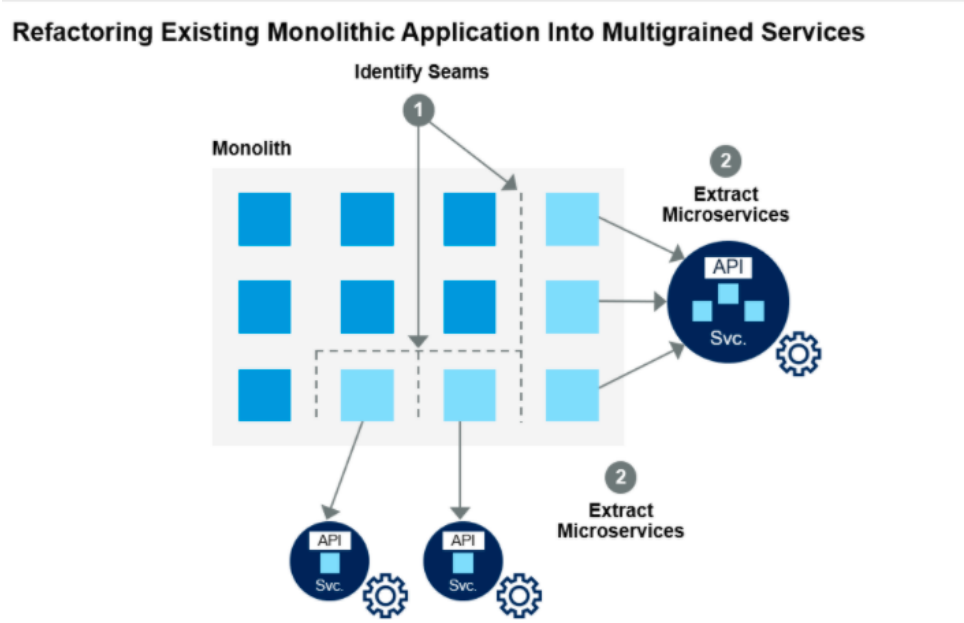
- Integración de servicio a servicio: los servicios de integración personalizados le permiten conectar servicios de back-end de varias maneras, como sincronizar cambios de estado entre dos sistemas de back-end diferentes o ampliar el procesamiento comercial al iniciar procesos comerciales adicionales en otros sistemas.

- Integración basada en eventos: los eventos proporcionan un mecanismo efectivo para integrar los cambios de estado que ocurren en un servicio para procesarlos en un servicio separado con un acoplamiento mínimo.

Refactorizar los sistemas existentes:

Refactorice los sistemas existentes que limitan el rendimiento, la escalabilidad o la cadencia de entrega requerida por la aplicación MASA. Esto evita la fragilidad posterior, las limitaciones de flexibilidad y los problemas de escalabilidad. No necesita refactorizar toda

la aplicación, solo las partes que necesita para la aplicación de malla. La refactorización debe ser un proceso iterativo en el que refactorice el monolito extrayendo piezas.



Implemente nuevos servicios con la granularidad adecuada:

Implemente nuevos servicios para agregar funcionalidades que no existen y que no se pueden refactorizar. Los nuevos servicios también incluyen funciones que pueden orquestar o agregar funciones de otros servicios existentes para admitir nuevas funciones requeridas por la aplicación.
Considere la granularidad de la funcionalidad del servicio MASA como un espectro con macroservicios en un extremo, microservicios en el otro y miniservicios en el medio:

- Macroservicios: servicios detallados que habilitan algunas o todas las funciones de una aplicación monolítica; los cambios requieren reconstruir y volver a implementar todo el monolito.
- Miniservicios: servicios de granularidad gruesa que brindan funcionalidad a nivel de dominio, lo que le permite realizar e implementar cambios en un solo servicio de dominio en la aplicación de malla sin afectar necesariamente a los otros servicios de dominio.
- Microservicios: servicios detallados con capacidades específicas que se implementan de forma independiente y brindan la mayor cantidad de flexibilidad.

Con esta renovación se persiguen los siguientes objetivos:

- Contar con una base tecnológica plenamente actualizada
- Profundizar en la modularización del sistema, donde cada módulo mantenga una total independencia, tanto en su versionado, como en cuanto al equipo responsable

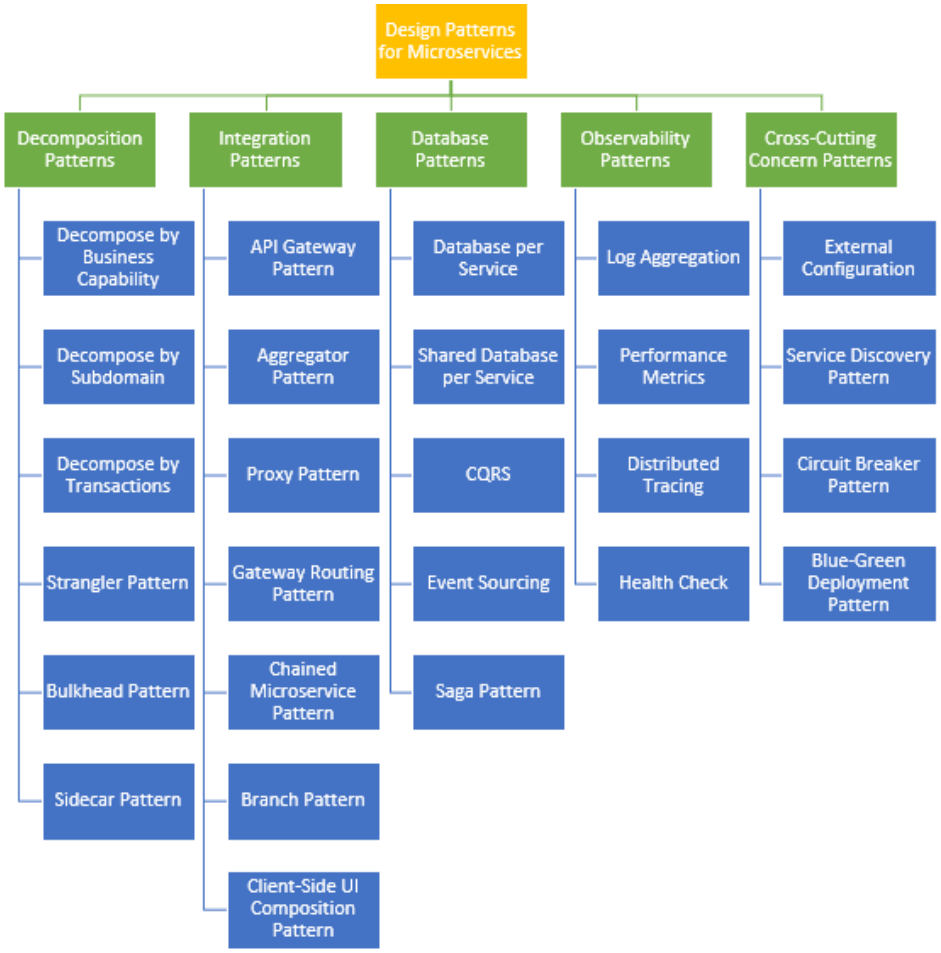
de desarrollo y mantenimiento

- Mejorar el Time to Market de los nuevos desarrollos
- Posibilitar una correcta gobernanza del sistema en Producción, optimizando la relación infraestructuras / rendimiento
- Maximizar la reutilización de código
- Automatizar al máximo las tareas de integración, pruebas, despliegue y control de calidad.

3.1 Especificaciones de desarrollo

Microservicios se refiere a servicios independientes e implementables adjuntos a un dominio particular. Los microservicios se comunican a través de redes. La arquitectura de microservicio consta de múltiples microservicios que colaboran y es una forma de arquitectura orientada a servicios (SOA).

https://learncsdesign.medium.com/microservices-design-patterns-91fe56a33a47



Patrones de descomposición:

- Descomponer por capacidad empresarial

Los microservicios tienen que ver con hacer que los servicios estén poco acoplados, aplicando el principio de responsabilidad única. Se descompone por capacidad empresarial. Definir los servicios correspondientes a las capacidades del negocio. Una capacidad empresarial es un concepto del modelado de arquitectura empresarial, es algo que hace una empresa para generar valor.

- Descomponer por subdominio

Descomponer una aplicación usando capacidades comerciales puede ser un buen comienzo, pero hay algunas clases que no serán fáciles de descomponer. Estas clases serán comunes entre múltiples servicios. Defina los servicios correspondientes a los subdominios de diseño controlado por dominio (DDD). DDD dominio consta de varios subdominios. Cada subdominio corresponde a una parte diferente del negocio.

- Patrón de descomposición por transacciones

Puede descomponer los servicios sobre las transacciones. Entonces habrá múltiples transacciones en el sistema.

- Patrón de estrangulador

Los patrones de diseño por los que pasa descomponen aplicaciones para Greenfield, pero el 80 % del trabajo que realiza es con aplicaciones brownfield, que son aplicaciones grandes y monolíticas (base de código heredada). El patrón Strangler viene al rescate o solución. Esto crea dos aplicaciones separadas que viven una al lado de la otra en el mismo espacio URI. Con el tiempo, la aplicación recién refactorizada "estrangula" o reemplaza la aplicación original hasta que finalmente puede cerrar la aplicación monolítica.

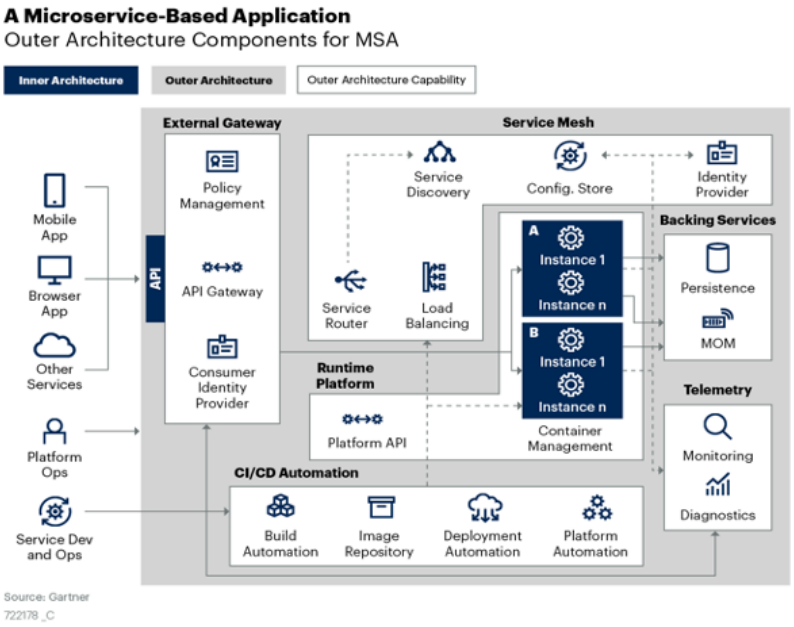
- La malla de servicio (Service Mesh)

patrón informático distribuido, implementado como middleware, que optimiza las comunicaciones entre servicios de aplicaciones.

Proporciona mediación de proxy y/o liviana para comunicaciones de servicio a servicio y admite funciones como autenticación, autorización, encriptación, descubrimiento de servicios, enrutamiento de solicitudes, equilibrio de carga e instrumentación de servicios.

- Sidecar proxies

- API gateways
- OAuth 2.0
- Microservices orchestration



3.2 Entorno tecnológico

Existen 2 estrategias a nivel de utilización de imágenes base:

3.2.1 Contenedores

3.2.1.1 Modo no gestionado

En este modo todo el peso del contenedor recae en el desarrollo. Se proveen flujos de CI/CD para el despliegue de la imagen, pero ésta no es gestionada desde Osakidetza.

A pesar de eso puede ser exigible el cumplimiento de ciertos requisitos de seguridad durante el despliegue.

3.2.1.2 Modo gestionado

En este modo desde Osakidetza se proveen imágenes base de plataformas, asegurando el ciclo de vida de las mismas, y los parcheos, tanto evolutivos, como de seguridad.

Desde la aplicación a su vez existen 2 opciones para adaptarse al ciclo de vida de estas imágenes base.

- Utilizar la version “latest” de la misma, de forma que en cada despliegue siempre se utilice la ultima versión publicada de la imagen base
- Utilizar una versión concreta de la misma, de forma que el control de la versión a utilizar en cada despliegue, dependa de desarrollo.

A nivel tecnológico existen imágenes base de las siguientes plataformas:

- Tomcat
- Nginx (basada en la imagen oficial)
- NodeJS (basada en la imagen oficial)
- Spring boot

- MongoDB
- Redis (en construcción)

3.2.2 Orquestacion de contenedores

La aplicación funcionará sobre una plataforma para aplicaciones modernas (*cloud-native application platform*) basada en contenedores orquestados por Kubernetes.

Esta plataforma se apoya en el portfolio de soluciones de VMware Tanzu y aporta:

- El entorno de ejecución nativo e integrado en el cloud privado sobre el que está desplegada el resto de la infraestructura de Osakidetza.
- Capacidades de gestión, operación y obervabilidad para los contenedores.
- Prestaciones de seguridad de la información, pudiéndose establecer además políticas transversales con las partes basadas en máquinas virtuales
- Gestión del ciclo de vida del software
- Soluciones de continuidad como backup y recuperación y posibilidad de despliegue en CPDs de respaldo.
- Capacidad para que la aplicación pueda funcionar en diferentes proveedores de cloud pública del mercado sin precisar de adaptaciones e hibridada también con el entorno on-premise estando federados entre sí.

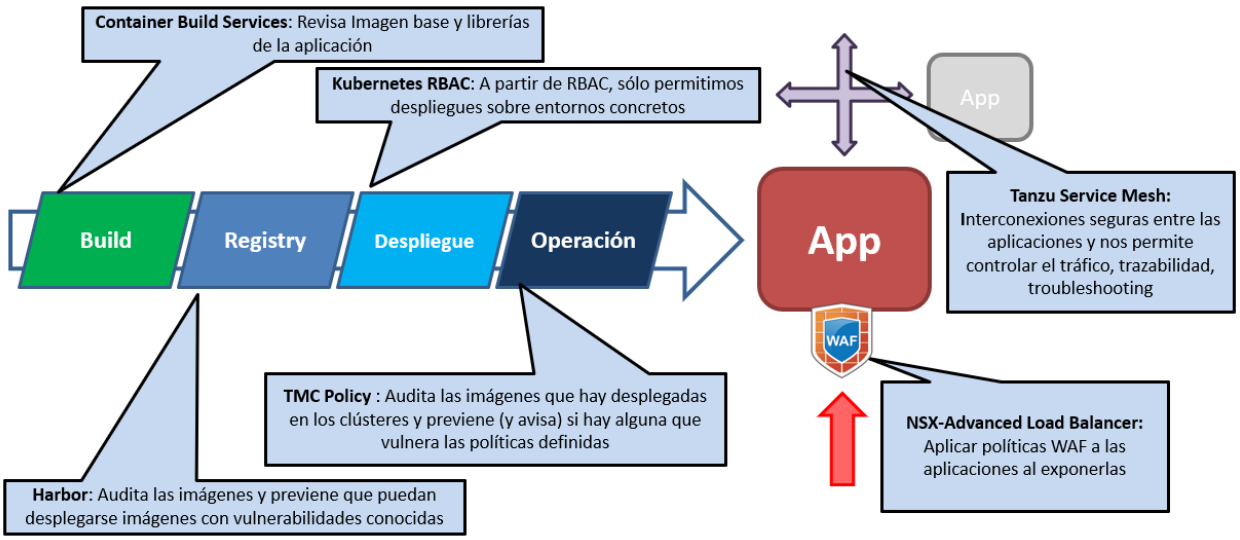
En cada una de esas ubicaciones se podrá disponer a su vez de entornos de no-producción y producción según se precise por criterios de calidad del software, testing, conveniencia o agilidad.

La plataforma en general estará operada a distintos niveles según responsabilidades por el equipo que da servicio a Osakidetza de Plataforma (PlatOps), Infraestructura (CloudOps) y Seguridad (SecOps) siendo el equipo de desarrollo de la aplicación un usuario consumidor de la misma. El equipo de PlatOps será su principal punto de interacción primándose la automatización y las capacidades de autoservicio.

3.2.3 Plataforma operativa y seguridad

Apoyándose en componentes de la plataforma descrita en el apartado anterior, el alineamiento con buenas prácticas de seguridad de la información en todas las etapas del ciclo del software se articula en cada etapa de un pipeline CI/CD concebido específicamente para aplicaciones modernas.

Este pipeline que tiene el acento en la parte “CD” del desplegue continuo y sirve como base para formar una cadena de suministro segura extremo a extremo, con difererentes momentos de actuación (build vs runtime).



Los controles básicos de segmentación, autenticación y autorización de microservicios se basarán en un service mesh mesh en la mayoría de los casos, para que su aplicación sea realista y abordable. Esta service mesh se debe conciliar y/o con la capa de mediación de API en los casos que esta última sea de aplicación

Aunque se prioricen los controles que estén incorporados en la plataforma de orquestación, se podrán usar herramientas externas por excepción cuando la singularidad, criticidad o particularidad del escenario lo requiera.

El desarrollador de la aplicación debe seguir las políticas, reglas y estandarizaciones establecidas meticulosamente y ser partícipe dentro de su diligencia debida y ámbito de responsabilidad del alinamiento con las buenas prácticas.

3.2.4 Plataformas horizontales

Plataformas distribuidas de transmisión de datos, como Kafka, almacenamiento de objetos, etc.

En múltiples ocasiones existen escenarios mixtos en los que la aplicación se evoluciona a un modelo sobre contenedores, pero el dato sigue persistiendo en BBDD’s “tradicionales”.

En estos casos las versiones estándarizadas son:

- Oracle Database Single Instance 19c
- Oracle RAC Database 19c
- MySQL

4 DIRECTRICES TECNOLÓGICAS DE IOC

Osakidetza dispone de múltiples aplicaciones desarrolladas en diversas tecnologías (principalmente Java y .Net), y albergadas en diferentes plataformas (principalmente Apache HTTP Server, Microsoft IIS y VMware Horizon como soluciones de presentación, Oracle WebLogic, Apache Tomcat y Microsoft IIS como servidores de aplicación y Oracle RDBMS, Microsoft SQL Server, MongoDB como servidores de BBDD).

El modelo de referencia de las aplicaciones es una solución Web (sin plug-in's), con **arquitectura** de n-capas preferentemente **3 capas**: presentación, negocio y persistencia, compatible con tecnología de virtualización y almacenamiento SAN y NAS.

Dicha arquitectura es **escalable horizontalmente** en cuanto a las capas de negocio y persistencia (reparte la carga equitativamente entre los diversos servidores que ejecutan la lógica de negocio y estos a su vez, mediante un pool de conexiones acceden a la capa de persistencia), y es **tolerante a fallos** garantizando la continuidad de negocio.

Los servidores que atienden a las capas de negocio y persistencia están ubicados en los dos CPDs de la Dirección General de Osakidetza siendo el reparto de los mismos proporcional en un CPD u otro; la capa de presentación está distribuida en los puestos de trabajo fijos, portátiles, móviles y tabletas en los diversos centros de Osakidetza (Hospitales, Centros de salud, etc..).

De modo que la solución (producto/aplicación/desarrollo) a suministrar deberá ser compatible con el modelo de arquitectura indicado en el párrafo anterior y con la relación de productos y tecnologías que se indican a continuación (las que apliquen).

4.1 Entorno tecnológico

En conjunto, el entorno tecnológico de las soluciones actualmente en producción es:

- Plataformas de desarrollo:
  - .NET sobre Windows 2016*
  - Java sobre Red Hat Enterprise Linux 8
  - Aplicaciones móviles:
    - Node.js, Express, AngularJS, MongoDB sobre Red Hat Enterprise Linux 8
    - Ionic, Cordova sobre Android, iOS

(*)En la actualidad 2019 en construcción

- Plataforma de virtualización:
  - VMWare vSphere 7.0 U2

- Almacenamiento:
  - SAN: EMC VMax
  - NAS: EMC Isilon

- Sistemas Operativos:
  - Servidores:
  - Windows 2016 (en la actualidad en construcción 2019)
  - Red Hat Enterprise Linux 8
  - Puesto de trabajo fijo: Windows 10, Enterprise x64
  - Puesto de trabajo móvil: Android, iOS

- Servidores Web:
  - Apache Webserver 2.4.X (en la actualidad 2.4.6)
  - MS IIS 10 (S.O. Windows Server 2016)
  - SAP Web Dispatcher 7.73 (actual 7.20 obsoleta)
  - Nginx 1.18.0

- Servidores de Aplicación:
  - Apache Tomcat 9.0.32 + JDK 8
  - Oracle Weblogic Server 12c + JDK 8
  - MS IIS 10 + .NET Framework 4.6.2
  - Application Server de SAP Netweaver 7.5
  - Node.js 12.17.0
  -
- Sistemas de Gestión de Bases de Datos (*)
  - Oracle RAC 19c (para réplicas se usa Active Data Guard)
  - Oracle 19c Single instance
  - Microsoft SQL Server 2016 (En la actualidad 2019 en construcción)
  - MongoDB 4.4.1

(*) Continuidad de negocio y Alta disponibilidad: Oracle RAC, SQL Server Always ON solo para aplicaciones críticas

Gestión de contenidos:

- MS SharePoint Online (O365)
- Documentum 20.4

- Infraestructura SOA:
    - Oracle Service Bus 12c (12.2.1.4) + JDK 8.0

- Plataforma de firma (PKI):
    - Izenpe

- Autenticación:
    - Directorio Activo de MS (AD)

- Modo de ejecución/escritorio:
    - local: puesto fijo o móvil
    - Cliente de Horizon 5.3

- Dispositivos Inalámbricos:
    - Seguridad:
        - Utilización del estándar IEEE 802.11 b/g/n
        - Capacidad de asociarse a un SSID de la red WIFI correspondiente
        - Soporte SSID oculto
        - Soporte WPA2 Enterprise
        - Cifrado de datos AES
        - Autenticación EAP basada en certificado de máquina
        - Soportar certificados:
            - Los certificados emitidos son de 1024 bits
            - El certificado de la CA raíz es de 4096
            - Soporten certificados con hash SHA-2
            - El tipo de certificado: x.509
            - Protocolo: RSA
    - Señal:
        - Ganancia de la antena: Mínimo 3 dBi
        - Sensibilidad <= -70 dBm a 11Mbps

Consideraciones en relación a los de productos y tecnologías indicados anteriormente:

- En cuanto al navegador, se debe **evitar** el uso de Applets (Java), componentes ActiveX (.NET), Flash u otras tecnologías que impliquen la **descarga y ejecución de software embebido en el navegador**.
- Al respecto de Office 2010, **no está permitido** el uso, dependencia o interrelación con Access.
- Al respecto de los dispositivos inalámbricos se han especificado los **requisitos mínimos** que deben cumplir.

4.2 ESPECIFICACIONES DEL CLIENTE (MAQUETA)

Actualmente, la versión de maqueta que se instala en los equipos de **Osakidetza** es la *2.1,* aunque convive con versiones anteriores instaladas.

- Hardware (mínimo existente):
    - PC arquitectura Intel o equivalente
    - Procesador con un mínimo de Tecnología de 45 nm, Caché L2 de 3 Mb. y FSB de 1.066 MHz.
    - Memoria RAM 2 Gb (4 GB en W10)
    - Disco duro 250 Gb (500 Gb en W10)
    - Monitor 19 pulgadas con resolución 1440x900 mayoritaria o 1280x1024. (21-22" 1920x1080 en W10)

- Restricciones de escritura en disco C: para usuarios standard, excepto en su perfil de usuario (C:|users\)
    - En cuanto al navegador, se debe evitar el uso de Applets (Java), componentes ActiveX (.NET), Flash u otras tecnologías que impliquen la descarga y ejecución de software embebido en el navegador.
    - Al respecto de Office 2016, no está permitido el uso, dependencia o interrelación con Access.
    -
    - Mismas restricciones (navegador y Office) que W7.

Sistema Operativo que instala esta imagen		
X	Actual - Windows 10, Enterprise x64 1709 Actualizando - Windows 10, Professional x64 1909	
Idiomas Disponibles:	Castellano (es-ES); Euskera (eu-ES)	
Hotfixes:		
Descripción		Clasificación
Compatible desde 1703 hasta el update y actualizaciones de seguridad de W10 en vigor en todo momento		Updates Críticas y de seguridad

Parámetros de Configuración de Software				
Fabricante	Producto	Versión	Idioma	Plataforma
Microsoft	Silverlight	5.5.1	Castellano	x64
Microsoft	Visual C++ 2008	9.0.30729.4148	Castellano	X64,x86
Microsoft	Visual C++ 2010	10.0.30319	Castellano	x64;x86
Microsoft	Visual C++ 2012	11.0.60610	Castellano	x64;x86
Microsoft	Visual C++ 2013	12.0.30501	Castellano	x64;x86
Microsoft	Visual C++ 2017	14.11.25325	Castellano	x64;x86
Oracle	Java	8.1510.12	Castellano	X64;x86
Adobe	Acrobat Reader	15.020.20039	Castellano	x64
Adobe	Shockware	12.2.4.194	Castellano	x64
7zip Org	7zip	16.04	Castellano	x64
FNMT	DNIe	14.0	Castellano	x64
FNMT	Certificados Raiz			
FNMT	TC-FNMT	6.1.3	Castellano	x64
Xuxen	Corrector Xuxen	5.2	Euskera	X86
LibreOffice	LibreOffice	5.4.5.1	Multiligual	X64
Firefox	Firefox	58.0.2	Castellano	X64
Microsoft	RSTA Cliente	1709	Castellano	X64
Microsoft	Cliente One Drive	17.3.6816.0313	Castellano	x64
IZENPE	Middleware_Izenpe	4.0.2.436	Castellano	X64
IZENPE	Idazki-desktop-win	3.3.1	Castellano	X64
Vmware	Horizon Client	4.7	Multilingual	x64
Parámetros de Configuración de Software Personalización de Osakidetza				
Oracle	Cliente Oracle	12.1.0.2	Castellano	x64
VMWare	Cliente Horizon	5.0.0		

Microsoft	Office Standard	2016	Castellano	x86
Microsoft	Office ProPlus	2016	Castellano	x86
Microsoft	Access Runtime (Con versión Standard 2016)	2016	Castellano	X86
Business Objects	Crystal Reports	10.5.0.0.sp16	Ingles	X86
Business Objects	Crystal Reports runtime engine	13.0.22	Ingles	X86
NDDigital	nddDigital DCS Server (Pago por impresion)	2.0.0.6	Ingles	X64
NDDigital	nddDigital nddPrint Agent	5.20.5	Ingles	X64
BitDefender	Bitdefender Endpoint Security Tools	6.6.11	Castellano	X86
Microsoft (integrado con el SO)	NET Framework	3.5 (incluye .NET 2.0 y 3.0)		X64
Microsoft (integrado con el SO)	Servicios avanzados de .NET Framework	4.7(ASP.NET 4.7)		X64
Unify	Team Developer 5.2 Deployment	5.2.0000		x86

4.3 Dispositivos de entrada/salida

La solución deberá ser capaz de interactuar con diversos dispositivos de entrada y salida de diversos fabricantes; siendo dichos dispositivos: pantallas táctiles, impresoras térmicas, monitores, por tanto el interface con los mismos será en base a estándares del mercado.

Así mismo la solución deberá adaptarse a la resolución de las diversas pantallas.

En relación a dispositivos móviles se deben tener en cuenta ciertas particularidades en relación a las características de movilidad, tamaño, comunicación inalámbrica e interacción con las personas:

- Son aparatos pequeños.

- La mayoría de estos aparatos se pueden transportar en el bolsillo del propietario o en un pequeño bolso.
- Tienen capacidad de procesamiento.
- Tienen conexión permanente o intermitente a una red.
- Tienen memoria (RAM, tarjetas MicroSD, flash, etc.).
- Normalmente se asocian al uso individual de una persona, tanto en posesión como en operación, de modo que puede adaptarlos a su gusto.
- Tienen una alta capacidad de interacción mediante la pantalla o el teclado.

En lo que se refiere a dispositivos sobre los que tienen que ejecutar soluciones de movilidad hay dos ámbitos que son las soluciones orientadas a profesionales de Osakidetza y soluciones orientadas al ciudadano.

Las aplicaciones orientadas al ciudadano se ejecutarán en dispositivos de consumo ya sean móviles, tabletas o cualquier otro dispositivo al que vaya destinada la solución. En este caso el desarrollo deberá ejecutarse en dispositivos con los diferentes sistemas operativos que haya en el mercado.

En las aplicaciones orientadas a los profesionales de Osakidetza las aplicaciones deberán de ejecutarse en el dispositivo elegido y homologado para soportar la solución. Habrá que tener en cuenta los siguientes aspectos:

- Ergonomía, tamaño y usabilidad del dispositivo, se refiere a que cumpla las expectativas del usuario en ámbito en el que se va a desarrollar el proyecto.
- Especificaciones de limpieza y resistencia a los golpes.
- Características de comunicaciones como pueden ser conectividad Wifi y 4G.

4.4 Seguridad, acceso, autenticación y autorización

El tratamiento de los datos cumplirá lo establecido por la legislación vigente en materia de **seguridad** y protección de datos.

En el ámbito intranet se permitirá el acceso mediante los **transportes** RMI, JMS, HTTP y HTTPS.

El proveedor de **servicios de certificación** de Osakidetza es Izenpe.

Osakidetza ha dotado a todos sus profesionales de certificados corporativos reconocidos, en soporte tarjeta criptográfica, con capacidad de firma de documentos y mails, cifrado y autenticación en el AD.

Además, para los nuevos SI que necesitan firma electrónica ágil, Osakidetza ha implantado una solución horizontal de firma, basada en certificados alojados en HSMs, llamada "firma centralizada". Son certificados con las mismas características y garantías que los alojados en tarjeta criptográfica. Estos HSM son de última generación y cumplen con las más exigentes normas internacionales (FIPS 140-2 Level 2 and Level 3, Common Criteria at EAL 4+ y RoHS compliant). Permiten la firma a través de CSP propio (applets) y la firma directa basada en una Pasarela de firma propietaria.

Izenpe es también el proveedor de la infraestructura de autenticación, validación y firma basada en certificados electrónicos. Osakidetza dispone de 2 Zain en HA (equipos TrustedX de Safelayer) en propiedad y su backup, en modo servicio, en los Zain de Izenpe. La integración con estos equipos se puede realizar mediante su librería Smartwrapper. Provee diferentes servicios web (validación, evolución de firmas, firma en servidor, etc.) alojados en los OSB de Osakidetza.

Así mismo, Osakidetza dispone de su propia infraestructura de Custodia de documentos firmados, para asegurar la validación de cualquier firma realizada en sus sistemas de información con certificados electrónicos. Está constituida por 2 equipos Siaval en HA, de la empresa SIA, y HSM Ncipher propios.

Para la firma electrónica de documentos in-situ por parte de sus pacientes, Osakidetza dispone de una solución denominada "firma biométrica", que asegura la validez de todas las firmas implicadas. Se apoya en la solución SmartAccess de Telefónica.

Osakidetza está inmerso en la adecuación de toda su infraestructura y sistemas de información al reglamento europeo eIDAS. Tanto en cuestión de certificados como de niveles de aseguramiento (autenticación).

Los sistemas de **autenticación** validos son:

- **Autenticación** basada en **Directorio Activo (Microsoft Active Directory) corporativo,** que permite identificar un empleado de Osakidetza mediante usuario/contraseña o la tarjeta profesional.
- **Token de Kerberos**, que habrá sido emitido por el Directorio Activo corporativo.
- **Autenticación** basada en **infraestructura de certificación digital (PKI)**. Esta autenticación permite la identificación de personas (certificados personales) o de sistemas de información (certificados de servidor)
- **Certificados X.509 de servidor**, que permitirán identificar sistemas (máquinas) y establecer confianza entre ellas.
- **Certificados X.509 de cliente**, que permitirán identificar individuos (personas)
- Para los servicios web, el estándar **WS-Security** permitirá aplicar políticas de autenticación.
- El bus de servicios sólo se utilizará para la autenticación basada en **certificados X.509** de servidor.

El sistema de **autorización** se implementará por medio de la definición de roles de usuario en base a los cuales se gestionarán los permisos y acciones sobre los distintos procesos por los que estará compuesta la solución. De modo que cada usuario en función del rol que tenga asignado podrá acceder a una serie de funcionalidades, así como a los datos relativos a su organización de servicio, centro de trabajo, servicio/área de atención.

Se crearán grupos de autorización del Directorio Activo. Por cada perfil que tenga la aplicación (medico, enfermera, administrativo, administrador de la aplicación,…) se creará un grupo de autorización en el DA. Se mapeará el perfil al grupo.

4.5 Monitorización

Osakidetza dispone de una **plataforma de monitorización** a través de la cual ya tiene establecida una línea de monitorización base de su infraestructura y SI.

En este sentido la solución deberá incluir un apartado en el que se describan los elementos a monitorizar para asegurar un correcto funcionamiento del sistema, servicio o producto suministrado de forma integral dentro de la mencionada plataforma de monitorización.

La monitorización se podrá llevar a cabo a través de alguna combinación de los siguientes productos o métodos que Osakidetza integra dentro de su plataforma de monitorización:

- SCOM mediante Management Pack para sistemas Microsoft.
- Nagios XI para sistemas Linux.
- Enterprise Manager Cloud Control para SGBD Oracle / EM 13 y servidor de aplicaciones Weblogic para rendimiento.
- Instalación de agente Dynatrace (1.219.131) para aplicaciones web/Horizon

Las herramientas de monitorización se integran con Evanios/ITOM (gestión de eventos) para las plataformas Batera.

Con el objetivo de monitorizar la solución que vaya a ser suministrada, deberán detallarse los objetos concretos a monitorizar para garantizar su correcto funcionamiento más allá de los aspectos generales de la infraestructura de sistemas que los alberguen; dicha relación de objetos podrá incluir, aunque no exclusivamente, algunos como:

- Procesos en ejecución
- Puertos de comunicaciones a la escucha
- Eventos concretos en registros del sistema
- Unidades de disco o File Systems

Para cada uno de ellos deberán incluirse los umbrales de consumo que se consideren anormales, especificándose 2 valores a partir de los cuales establecer el correspondiente nivel de alerta preventiva o alerta crítica según proceda.

Adicionalmente podrán suministrarse scripts o procesos para llevar a cabo:

- Pruebas sintéticas de los sistemas cuyo resultado determine el correcto funcionamiento del sistema o el correspondiente nivel de alerta preventiva
- Operativas sobre el sistema que puedan ayudar a determinar o especificar la condiciones concretas de los errores o alertas producidos.
- Operativas sobre el sistema para intentar remediar automáticamente la condición de la alerta o error, como reinicio de procesos o servicios, etc.

Todos estos procesos deberán entregarse con su correspondiente documentación, y su operativa deberá resultar sencilla de modo que pueda ser realizada por un operador.

4.6 Backup/Restore

Osakidetza dispone de una infraestructura para la realización de copias de seguridad y restauración a través de red basada en CommVault.

De modo que la solución deberá especificar los procedimientos necesarios a llevar a cabo para la recuperación del servicio ante la pérdida del mismo, corrupción de sus datos o alguno de sus componentes. En este sentido se plantearán escenarios de recuperación independientes para cada uno de los posibles componentes impactados y el orden de ejecución en caso de que deban ejecutarse secuencialmente para la recuperación de varios componentes o del sistema completo en su conjunto.

A partir de los procedimientos anteriormente descritos y en cada uno de ellos, se inferirá **los elementos concretos** sobre los **que se requiere hacer Backup**, en qué **orden**, con qué **frecuencia** y **política de persistencia** deben llevarse a cabo **para poder satisfacer** los distintos **escenarios de recuperación** del servicio contemplados. Así mismo deberán indicarse las restricciones que pudieran existir para la realización del Backup: posible impacto en el servicio, restricciones horarias, etc.

Los distintos procedimientos de recuperación suministrados con la solución deberán ser validados por Osakidetza para garantizar su compatibilidad con la infraestructura de copia de seguridad y restauración existente.

5 INTEROPERABILIDAD

Los requisitos de integración de diversos sistemas de información de Osakidetza se realizaran de acuerdo a una arquitectura orientada a servicios (SOA) sobre la plataforma Oracle SOA Suite (Oracle BPEL Process Manager, Oracle Service Bus (OSB), Oracle Business Rules, ...).

Dicha arquitectura proporciona una forma bien definida de exposición e invocación de Servicios Web, lo cual facilita la interacción entre diferentes sistemas propios o de terceros.

Sobre la misma arquitectura SOA, Osakidetza implementa una solución de integración orientada a Eventos; es un diseño a medida que gestiona un conjunto de sistemas que publican eventos y un conjunto de aplicaciones que se suscriben a determinados eventos. Dicha solución se denomina Gestor de Eventos-Event Manager y es responsable de recibir los eventos publicados, ejecutar las validaciones adecuadas, enrrutar y almacenar los eventos para su envío a los subscriptores que estén asociados a cada uno de los elementos recibidos.

Relación de estándares de comunicación soportados:

- Protocolos a nivel de mensaje:
  - SOAP 1.1 y SOAP 1.2
  - WSDL 1.1 y WSDL 1.2 Binding
  - SOAP con Attachments
  - JSON – REST (para aplicaciones móviles)
- Protocolos de seguridad a nivel de mensaje:
  - WS-Security 1.0/1.1
  - WS-SecurityPolicy
  - WS-Policy
  - WSPolicyAttachment
  - WS-Security: Username Token Profile 1.0/1.1
  - WS-Security: X.509 Token Profile 1.0/1.1
  - WSSecurity: SAML Token Profile 1.0/1.1
  - WS-Security: KerberosToken Profile 1.1
  - WS-Reliable Messaging 1.0
  - WS-Addressing
  - WS-I Basic Profile 1.1
  - WSSecurity: JWT Token (aplicaciones móviles en internet)
- Protocolos a nivel de transporte:
  - HTTP 1.0, HTTP 1.1
  - TLS, SSL

- Interoperabilidad con registros UDDI v3-compliant
- Sistemas middleware basados en JMS/MQ
- HTTPS

5.1 Servicios Web

Los Servicios Web se implementarán de acuerdo con las especificaciones WSDL v1.1, SOAP v1.1, v1.2, JSON-REST (en el caso de aplicaciones móviles), UDDI v2.XX y XML v1.0, con el objetivo de incorporar las recomendaciones de la WS-I definidas en la especificación Basic Profile v1.0, v2.0 y de esta manera asegurar la interoperabilidad entre los sistemas.

El estándar de codificación que se utiliza en los mensajes XML es UTF-8.

La seguridad aplicada a los servicios web cubrirán los siguientes aspectos:

- Autenticación: Verificar que el cliente (usuario o aplicación) es quien dice ser. La identidad de un usuario se realizará en base a la información presentada por el usuario (usuario/contraseña, certificado, token SAML)
- Autorización: Otorgar acceso a los servicios en base a la identidad del cliente o a los roles asignados.
- Confidencialidad, privacidad: Mantener la información secreta mediante el uso de algoritmos de encriptación estándar de elementos XML.
- Integridad, no repudio: Asegurar que un mensaje permanece inalterado durante la transmisión mediante la firma digital. La firma también valida la identidad del remitente y proporciona una marca de tiempo para garantizar que una transacción no puede ser repudiada más tarde ni por el remitente ni por el destinatario.

Osakidetza usa Oracle Web Service Manager (OWSM) para gestionar y aplicar políticas a los servicios web publicados en la plataforma SOA.

La política estándar que Osakidetza ha definido para los servicios proporciona:

- Autenticación mediante certificado x509
- Protección del mensaje mediante firma (sin encriptado)

Existen dos versiones de la política en OWSM, una para servicios y otra para clientes. Para garantizar la interoperabilidad, cada política tiene su versión compatible con tecnología .NET y Java.

- oracle_wss10_x509_token_with_message_sign_service_policy
- oracle_wss10_x509_token_with_message_sign_service_policy_net
- oracle_wss10_x509_token_with_message_sign_client_policy
- oracle_wss10_x509_token_with_message_sign_client_policy_net
- oracle_http_jwt_token_server_policy

Los clientes y aplicaciones que acceden a través de internet entran a la DMZ a través del firewall de aplicaciones (WAF). El WAF aplica reglas contra ataques y define patrones de seguridad.

- Es responsabilidad de cada aplicación publicada en la DMZ controlar el acceso y autorizar a los usuarios.
- El OSB de internet publica los servicios a los que pueden acceder las aplicaciones de internet.
- El OSB de internet audita todas las llamadas a los Servicios Web mediante una política propietaria de Osakidetza gestionada por OWSM.
- En el OSB de internet se protegen todos los servicios con la política oracle_wss10_x509_token_with_message_sign_service_policy. Esta política autentica a las aplicaciones mediante certificado x509 y firma el mensaje de petición y respuesta.
- El OSB de internet delega la ejecución a servicios publicados en la Intranet.
- En la intranet, se despliegan instancias independientes de servicios web para dar servicio a las peticiones que llegan desde el OSB de Internet.
- Entre el OSB de intranet y el OSB de intranet, se realiza una federación de servicios con las condiciones de seguridad habituales.

Nota: OSB -> Oracle Service Bus 11.1.1.7 -> Paso a OSB 12c

5.2 Gestor de Eventos

A continuaciónse describen los requisitos técnicos que tienen que cumplir las aplicaciones para publicar y/o recibir eventos.

5.2.1 Estándares de comunicación

La mensajería del Servicio Osakidetza se implementará de acuerdo con las especificaciones del estándar HL7 versión 2.XX o con cualquier otro formato propio de Osakidetza; de esta manera se asegura la interoperabilidad entre los sistemas de información.

Relación de tecnologías soportadas para la publicación y subscripción a eventos.
Tecnologías que soporta el gestor de eventos para la publicación y subscripción a eventos, así como si la modalidad soporta transaccionabilidad y las opciones de seguridad disponibles:

Modalidad	Tecnología	Transaccional	Orden	Seguridad
Publicación	Mensajería JMS	Sí	Sí, si el publicador establece el parámetro UnitOfOrder	Autenticación (user/pass)
Publicación	Servicio web	No	No	Ninguna, Autenticación (user/pass) y WS-Security
Subscripción	Mensajería JMS	Sí	Sí. Event Manager garantiza la entrega en el mismo orden que ha recibido los mensajes	Autenticación (user/pass)
Subscripción	Servicio OSB	Sí	Sí. Event Manager garantiza la entrega en el mismo orden que ha recibido los mensajes	Ninguna
Subscripción	Servicio web	Sí	Sí. Event Manager garantiza la entrega en el mismo orden que ha recibido los mensajes	Ninguna, Autenticación (user/pass) y WS-Security

Por cada modalidad y tecnología se deben cumplir unos requisitos técnicos que serán indicados al adjudicatario para la implementación de esta forma de integración.

5.2.2 Mensajería. Definición de Evento

Un evento es un documento XML definido mediante un XSD, donde:

- ✓ **id**: Es el identificador del tipo de evento. Se genera durante el proceso de alta del evento en el sistema de administración de Event Manager. Durante el procesamiento de un evento se verifica que el id sea válido.
- ✓ **correlation**: Es un campo libre en el que el publicador del evento indica un número correlativo relativo a su sistema.
- ✓ **source**: Es el identificador del publicador. Se genera durante el proceso de alta de un publicador en el sistema de administración de Event Manager. Durante el procesamiento de un evento se verifica que el source sea válido.
- ✓ **timestamp**: Lo establece el publicador del evento en el momento del envío.
- ✓ **metadata**: Puede contener un xml que ayude a describir el contenido del evento. Event Manager puede utilizar esta información para tomar decisiones de enrutado.
- ✓ **payload:** Es el contenido del evento. Puede ser cualquier cadena de texto o XML.

El resultado devuelto cuando se publica un evento en Event Manager es un XML definido por un XSD, donde:

- ✓ **uuid**: Es un identificador único que se asigna a cada evento procesado por Event Manager.
- ✓ **processed**: true o false, si el evento se ha procesado correctamente o con errores.
- ✓ **errorCode**: Si se ha producido un error, aqui se informa el código del error.

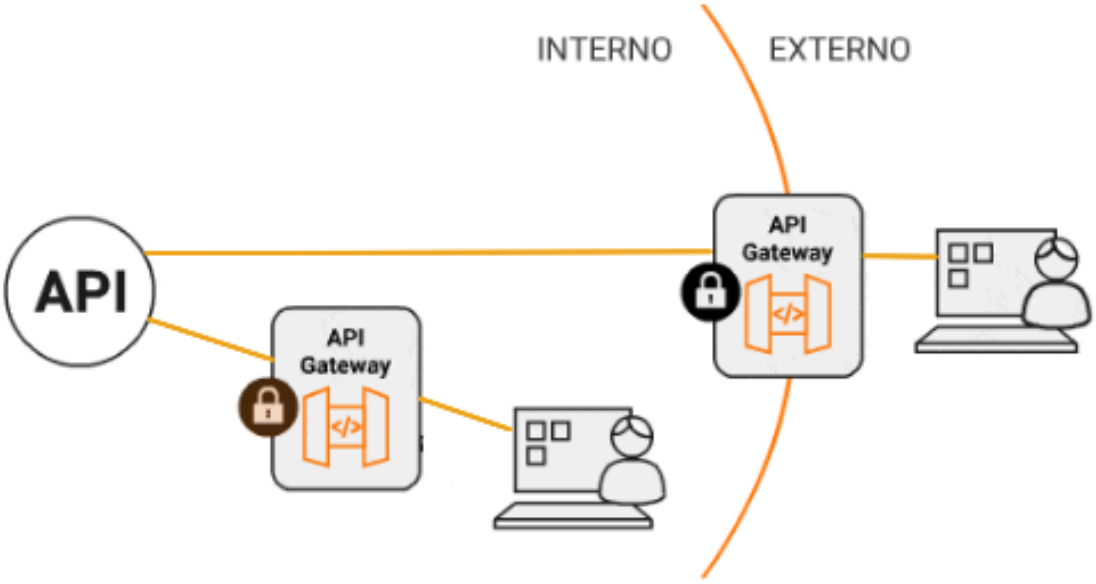
- ✓ **errorDescription**: Si se ha producido un error contiene la descripción de éste.

Se realiza gestión de errores: codificación de errores y descripción de los mismos.

5.3 Arquitectura API REST

De forma adicional a la Arquitectura SOA, Osakidetza permita la interoperabilidad mediante API REST.

El componente centrar de la la arquitectura API REST de Osakidetza el API Manager.



Las funciones del API Manager son:

- Exponer las APIs de manera fácil y segura a consumidores internos y externos.
- Implementar flujos de autorización estándara, como OAuth, OpenID Connect y JWT.
- Gestionar el correcto diseño y el tiempo de ejecución para todo el ciclo de vida de las APIs
- Monitorización y estadísticas de uso.

Osakidetza requiere que las API sigan las buenas de desarrollo tal y como se indica en el documento distribuido a tal fin.

Las API deben tener documentación elaborada mediante Swagger/Open API 3.0.

6 EXPLOTACIÓN INFORMACIÓN- BUSINESS INTELLIGENCE

La herramienta usada para desarrollo de tecnología Business Intelligence es Oracle Business Intelligence Enterprise Edition (OBIEE).

Esta herramienta es una herramienta web, que está soportada por una cada de presentación en Apache, y un servidor Web (Weblogic). En esta capa web de Weblogic se definen los ROLES de usuario (grupos y perfiles de usuario) que luego se usan para la Seguridad de Acceso a Datos y a partes de la aplicación

En este servidor también se alberga el RPD, que es donde se desarrolla toda la metadata y las distintas Áreas accesibles por usuarios.

Todos los datos accesibles desde OBI están almacenados en una única BBDD de Producción.

Para realizar la carga de datos (Incremental), utilizamos la herramienta de extracción (ETL) en ODI.

Para poder realizar las cargas incrementales necesitamos que los orígenes de datos tengan alguna fecha o característica que indique que el dato se ha modificado el día anterior, y así poder hacer cargas de sólo aquello que no existe en destino o se ha modificado. (Un LAST_MODIFIED por ejemplo). Las cargas son normalmente diarias, aunque existen cargas anuales o trimestrales.

<u>Versiones utilizadas:</u>

- OAS (Oracle Analytics Server) 5.5
- OBIEE 12.2.15
- ETL
    - ODI -> Oracle Data Integrator 11g Build ODI_11.1.1.7.0
    - Oracle WarehouseBuilder (DWH) (11.2.0.4)

7 SOLUCIONES MÓVILES

Este apartado refleja las tecnologías para el desarrollo de aplicaciones móviles, definida por Osakidetza.

Comprende dos capas:

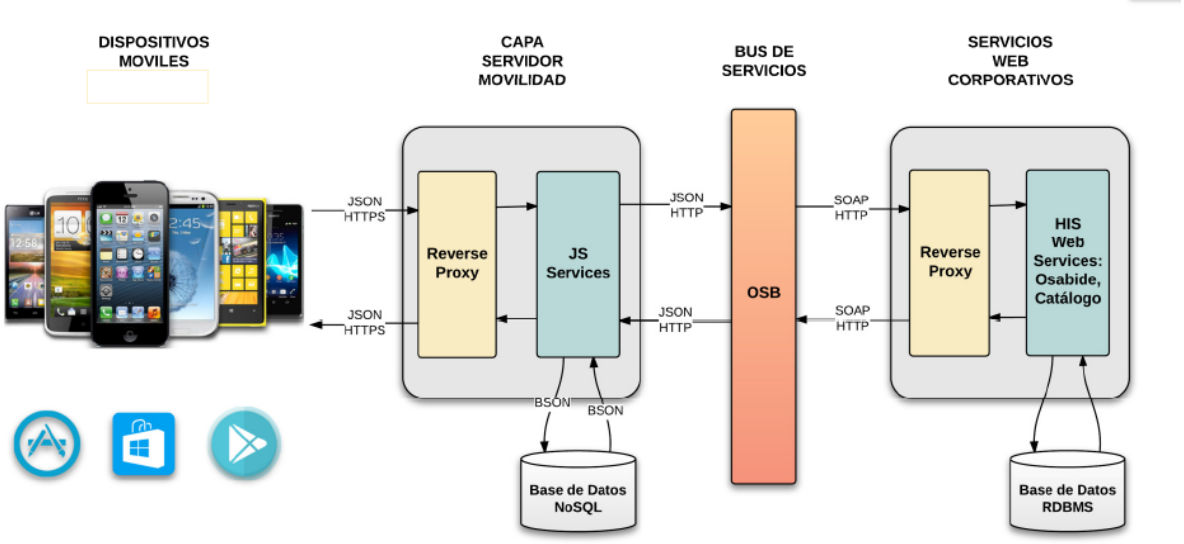
- Tecnologías para el desarrollo en cliente: aplicaciones para dispositivos móviles tales como tablets, smartphones (iOS, Android)

- Tecnologías para el desarrollo en servidor: servicios para la integración con servicios backend, seguridad y lógica de negocio movilidad.

Las características comunes a todas las tecnologías de movilidad son:

- Licencia de código abierto (open source)
- Permiten el desarrollo de aplicaciones multiplataforma
- Lenguajes de programación JavaScript, HTML5, CSS3

En la siguiente figura se muestra la **vista lógica** de la arquitectura de referencia para aplicaciones en movilidad.



7.1 Tecnologías Cliente

7.1.1 Angular

AngularJS es un framework de JavaScript de código abierto, mantenido por Google, que ayuda con la gestión de lo que se conoce como aplicaciones de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

La biblioteca lee el HTML que contiene atributos de las etiquetas personalizadas adicionales, entonces obedece a las directivas de los atributos personalizados, y une las piezas de entrada o salida de la página a un modelo representado por las variables estándar de JavaScript. Los valores de las variables de JavaScript se pueden configurar manualmente, o recuperados de los recursos JSON estáticas o dinámicas.

7.1.2 Ionic Framework 5 o superior

Ionic es un SDK completo de código abierto para el desarrollo de aplicaciones móviles híbridas. Construido sobre AngularJS y Apache Cordova, Ionic ofrece herramientas y servicios para el desarrollo de aplicaciones móviles híbridos utilizando tecnologías web como CSS, HTML5, y Sass.

Las aplicaciones pueden ser construidas con estas tecnologías Web y luego distribuidas a través de las tiendas de aplicaciones nativas para ser instalado en los dispositivos mediante el aprovechamiento de Capacitor.

7.2 Tecnologías Servidor

7.2.1 Nginx

Nginx es un servidor web/proxy inverso ligero de alto rendimiento.

Actualmente Nginx se usa en más de 100 millones de sitios web.

Es un software moderno y su adopción ha sido muy rápida debido a su implementación ligera, que proporciona unos tiempos de respuesta muy bajos con un consumo de recursos hardware mínimo.

7.2.2 Node.js

Node.js es un entorno de programación en la capa del servidor basado en el lenguaje de programación ECMAScript (JavaScript), asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.

Node.js es ligero y eficiente ejecutando aplicaciones que hacen uso intensivo de grandes cantidades de datos en tiempo real y corren en dispositivos distribuidos.

7.2.3 MongoDB

Mongodb es una base de datos No SQL open source. Mongodb está orientada a documentos, en vez de guardar los datos en tablas como se hace en las base de datos relacionales, MongoDB guarda estructuras de datos en documentos tipo JSON con un esquema dinámico.

8 CALIDAD DE USO

Rendimiento

Se exigirá que el rendimiento de la aplicación en uso (producción) cumpla y se ajuste a los requisitos no funcionales y al plan de pruebas diseñado en etapas tempranas. En cualquier caso, el comportamiento de la aplicación deberá cumplir de forma obligatoria los siguientes criterios en la ejecución de las pruebas técnicas.

INDICADOR	DESCRIPCIÓN	DESVIACIÓN
Prueba de escalabilidad	El tiempo de respuesta de la aplicación deberá mantenerse invariable con el incremento de usuarios concurrentes(a menos que exista alguna saturación de recursos del sistema), al menos hasta alcanzar la carga pico estimado.	+/- 10%
Prueba de estrés	Tras someter al sistema al doble de carga nominal, tras la disminución de dicha carga a la carga normal de trabajo (80%), la aplicación deberá garantizar su estabilidad y recuperación sin necesidad de reinicio.	+/- 10%
Prueba de estabilidad	La aplicación deberá garantizar un tiempo de respuesta estable manteniendo una carga nominal normal (80 %) durante al menos 12 h	+/- 10%

Usabilidad

Se exigirá que la aplicación mantenga unos criterios de usabilidad que permitan:

- Facilitar el aprendizaje de la aplicación sin requerir conocimientos técnicos en su uso. El sistema deberá ser fácil de usar, asegurando unos bajos costes de aprendizaje y de asistencia o ayuda al usuario.
- Asegurar la velocidad de uso de la aplicación permitiendo el acceso a la información en el menor número posible de clics de ratón.
- El interfaz de usuario deberá ser amigable de forma que se incremente la satisfacción y la productividad de los usuarios.
- Optimizar los costes de diseño, rediseño y mantenimiento.

La ISO 9241-11:1998 “Guidance on usability”, define la usabilidad como: " *La medida con la que un producto se puede usar por usuarios determinados para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso concreto*"

Osakidetza exigirá el cumplimiento de las directrices de la ISO 9241-151 en cuanto al cumplimiento del grado de usabilidad a exigir en el desarrollo de la aplicación

Accesibilidad

El cumplimiento de la accesibilidad de la aplicación incorporará posibilidades de navegación que garanticen el acceso a la información y a los servicios proporcionados minimizando al máximo las limitaciones y/o restricciones por razón de discapacidad de cualquier carácter o condicionantes técnicos, atendiendo así a la normativa existente.

Se deberán aplicar todas las pautas correspondientes al nivel de accesibilidad AA, llegando a un nivel AAA en las secciones que la configuración de los contenidos y diseño lo permita (Sede electrónica). La accesibilidad se regirá por las pautas WAI indicadas por el Word Wide Web Consortium (W3C).

La accesibilidad no se refiere únicamente a personas discapacitadas. Un sitio web accesible y que se adapte a diferentes tipos de usuario podrá ser visitado por más personas; esta flexibilidad también da beneficios a personas con discapacidades temporales (una convalecencia o enfermedad, problemas de visión), personas de edad avanzada, con una conexión lenta a Internet, que utilizan un dispositivo móvil, etc.

Compatibilidad y portabilidad

Osakidetza exigirá la compatibilidad de la aplicación con los siguientes navegadores web al menos en su última versión estable del fabricante y al menos dos anteriores: Internet Explorer, Firefox, Google Crome, Safari y Mozilla.

La empresa adjudicataria garantizará el funcionamiento de la aplicación con estos navegadores y se responsabilizará de la actualización de la aplicación en caso de necesidad para garantizar la compatibilidad de la misma con estos navegadores durante toda la ejecución del contrato y su periodo de mantenimiento.

El interfaz debe visualizarse independientemente del sistema operativo utilizado por el cliente.

Si procede el diseño deberá ser multi-dispositivo, es decir, accesible y visible desde cualquier dispositivo tipo teléfono móvil, tablet, etc.

Documento firmado por: MARIA ROSA PEREZ ESQUERDO
En la fecha 17.10.2022 11:30:46 CEST