TRABAJO FINAL DE MÁSTER
EN INVESTIGACIÓN EN INGENIERÍA DE SOFTWARE Y SISTEMAS

TRABAJO FIN DE MÁSTER
CURSO 2020-2021

AUTOR
JOATHAM PÉREZ EXPÓSITO

DIRECTORA

DRA. ELENA RUIZ LARROCHA

TRABAJO FINAL DE MÁSTER
EN INVESTIGACIÓN EN INGENIERÍA DE SOFTWARE Y SISTEMAS

UNED | UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

TÍTULO

TRABAJO FINAL DE MÁSTER
EN INVESTIGACIÓN EN INGENIERÍA DE SOFTWARE Y SISTEMAS

INGENIERÍA DE SOFTWARE Y SISTEMAS INFORMÁTICOS

AUTOR
JOATHAM PÉREZ EXPÓSITO

DIRECTORA
DRA. ELENA RUIZ LARROCHA

CONVOCATORIA: SEPTIEMBRE 2021
CALIFICACIÓN:

TRABAJO FINAL DE MÁSTER
EN INVESTIGACIÓN EN INGENIERÍA DE SOFTWARE Y SISTEMAS
UNED | UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

DEDICATORIA

A todos aquellos que les he robado tiempo, y he descuidado durante la realización del máster, en especial a ti Dani, costilla que forma parte de mi.

AGRADECIMIENTOS

Quiero agradecer a todos los profesores que he tenido, al igual que los alumnos de los que me he nutrido en este año de COVID, que de alguna forma han aportado en mi formación. También quiero agradecer a aquellos que me han ayudado a crecer como profesional, en proyectos que han supuesto un reto, profesional y humano, como (Platido), Tarife y Carlos seréis referentes, en el aspecto profesional y personal. También a mi familia, a los que están presentes y a aquellos que desde algún lugar me pueden estar viendo y que han dedicado todo su tiempo, su esfuerzo y sus recursos para educarme y formarme lo mejor posible para afrontar la vida. En especial a mi padre, que donde quiera que esté, se sienta orgulloso, y que ante un trabajo nos repetía: "Esto lo hacemos jugando, como el que no quiere la cosa", así estuviera lloviendo o tronando.

Por último a ti abuelo, que fuiste quien primero nos inculcó la cultura del trabajo y el esfuerzo como medio para avanzar: "Una persona no se queda trabada en una hierba mala".

RESUMEN

PLATÓN

Plataforma Digital de Servicios de la Administración Local Interoperable con la Administración General.

El presente proyecto final pretende mostrar una solución de interoperabilidad básica para las administraciones locales, de distinto tamaño, o comunidades autónomas, basada en documentos y firma electrónica bajo el Esquema Nacional de Interoperabilidad, en adelante ENI.

Tiene como fin, minimizar los tiempos de desarrollo, siendo un punto de inicio para el desarrollo de cualquier solución basada en servicios y que beneficie al ciudadano de a pie. Para ello se hará uso de un conjunto de servicios e interfaces proporcionadas por el Portal de Administración Electrónica[1], en adelante PAe.

Para la integración de los distintos servicios propuestos se hará uso de herramientas actuales que facilitan la integración, y comunicación, además de tecnologías que facilitan la interoperabilidad entre sistemas. Todo ello se describe de forma detallada en puntos posteriores.

Palabras clave

ESB, Interoperabilidad, Camel, Integración, Local, Estatal.

[1] https://www.boe.es/eli/es/l/2015/10/01/40/con

ABSTRACT

PLATÓN

Digital Platform of Services of the Local Administration Interoperable with the General Administration.

The aim of this final thesis is to propose a basic solution of interoperability for local administrations of different sizes or autonomous communities, based on documents and the use of the electronic signature under the National Interoperability Scheme, hereinafter ENI.

Its purpose is to minimize development times, being a starting point for the development of any service-based solution that benefits the ordinary citizen. To do this, a set of services and interfaces provided by the Electronic Administration Portal, hereinafter PAe, will be used.

For the integration of the different services proposed, the tools that currently facilitate integration and communication, as well as technologies that facilitate interoperability between systems will also be employed . All of this will be described in detail in later points.

Keywords

ESB, Interoperability, Camel, Integration, Local, State.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

Capítulo 1 - Introducción

1.1 Motivación

Siempre se ha considerado a la administración pública lenta y poco fluida. El ciudadano es el encargado de aportar toda la documentación en cada uno de los trámites solicitados, obrando ésta ya en la propia administración en numerosos casos, proveniente de trámites anteriores u otros organismos. La inclusión de los sistemas de información en los organismos públicos y el cruce de datos, supuso un avance respecto al pasado, solventando algunos de los problemas que se le presentaban entre los diferentes organismos.

La gestión de la información que proviene del ciudadano no resulta una tarea simple. Hemos de tener en cuenta que la documentación genera documentos, y estos a su vez componen expedientes. Es por ello, que la gestión de los documentos, cobre día a día, una mayor importancia, y su correcta gestión suponga un desafío. Todo esto, supone solventar problemas como la gestión documental o la firma electrónica. Estos problemas pueden provocar una resistencia al cambio de los organismos y los responsables de estos organismos ante el desconocimiento.

La primera acción que debe resolver un organismo que desea realizar la integración, y promover su transformación a tramitación electrónica, es conocer aquellos servicios que resultan imprescindibles para esta. Además se deberá realizar un análisis de las características y necesidades específicas de este y comenzar todos los trámites burocráticos para realizar el desarrollo e implementación de las soluciones necesarias.

El artículo 3 "Principios generales" de la Ley 40/2015, de 1 de octubre, de Régimen Jurídico del Sector Público[2], incluye la interoperabilidad entre los principios de actuación de las App, en adelante AAPP, de forma que estas deben relacionarse

[2] https://www.boe.es/buscar/act.php?id=BOE-A-2007-12352

entre sí a través de medios electrónicos que faciliten la fluidez y coordinación de la información, esta mejora la cooperación en el desarrollo y prestación de servicios públicos; así como una mayor eficacia y eficiencia en el despliegue y en la prestación de los servicios.

El artículo 156 de la Ley 40/2015, recoge el ENI. Este esquema comprende el conjunto de criterios y recomendaciones en materia de seguridad, conservación y normalización de la información, de los formatos y de las aplicaciones que deberán ser tenidos en cuenta por las App para la toma de decisiones tecnológicas que garanticen la interoperabilidad.

El ENI fue establecido anteriormente en el artículo 42 de la Ley 11/2007[3] y se materializa en el Real Decreto 4/2010, de 8 de enero[4], por el que se regula el ENI en el ámbito de la Administración Electrónica.

Las normas técnicas de interoperabilidad, en adelante NTI, concretan detalles para facilitar los aspectos más prácticos y operativos de la interoperabilidad entre las App y el ciudadano. Se acompañan de guías de aplicación y de otros documentos complementarios. En la elaboración del ENI y de sus normas técnicas se tuvieron presentes las recomendaciones provenientes de la Unión Europea, particularmente del Marco Europeo de Interoperabilidad.

Su objetivo es:

- Comprender los criterios y recomendaciones.
- Realizar puestas en común para tomar las decisiones correctas en decisiones tecnológicas que garanticen la interoperabilidad.
- Proporcionar los elementos comunes que han de guiar la actuación de las App en materia de interoperabilidad.
- Facilitar la inclusión de las políticas de seguridad comunes, al contribuir a un escenario de mayor racionalidad técnica y de economías de escala.

[3] https://www.boe.es/buscar/act.php?id=BOE-A-2010-1331
[4] https://administracionelectronica.gob.es/pae_Home/.

- Todo este conjunto de normas y reglas, promueven una línea de desarrollo en las administraciones, con el único fin de garantizar los derechos de los ciudadanos y agilizar sus trámites dentro de la administración.

Fuente: PAe_Esquema_Nacional_de_Interoperabilidad.html

1.2 Objetivos

El objetivo general del Trabajo Fin de Máster (TFM) es la realización de una solución básica, que permita implantar el ENI en la administración que lo requiera, sirviendo como punto de partida para el desarrollo.

Se describirán ciertos aspectos del ENI, relacionados con la recepción de documentación electrónica, haciendo uso de microservicios y además se propondrán herramientas y arquitecturas para el desarrollo de los distintos servicios que conformen la solución final, siguiendo una arquitectura de microservicios y que permita exponer estos de distintas formas.

Se irá detallando el proceso de desarrollo de una solución interoperable, teniendo como punto de partida el subconjunto de los servicios expuestos en el PAe.

Dicho objetivo general se puede desglosar en varios subobjetivos:

- Estudio de la Ley 11/2007, de acceso electrónico de los ciudadanos a los servicios públicos.
- Análisis del ENI así como sus Normas Técnicas asociadas.
- Desarrollo de una solución que englobe dos de los servicios más importantes como son el servicio de Firma y Repositorio Documental, siguiendo el ENI.
- Realización de propuestas de implantación y mejoras.

Resulta notable el concepto "interoperabilidad" como "l*a capacidad de los sistemas de información y de los procedimientos a los que éstos dan soporte, de*

compartir datos y posibilitar el intercambio de información y conocimiento entre ellos"[5].

Fuente: PAe_Interoperabilidad_Inicio.html

1.3 Plan de trabajo

Tras realizar un estudio de las soluciones que presenta el PAe como punto de partida en el desarrollo de servicios interoperables entre las App, pretendemos realizar la implementación del ENI, bajo una arquitectura robusta basada en microservicios versionables.

Para abordar cada uno de los aspectos procederemos a emplear la siguiente metodología:

- Estudio bibliográfico, investigación en el PAe sobre casos de éxito, así como diversos fondos bibliográficos.
- Análisis de la Ley 11/2007, de Acceso electrónico de los ciudadanos a los servicios públicos así como estudio del ENI y su Normas Técnicas asociadas.
- Realización de una solución acotada como base para cualquier administración, basada en software libre.
- Redacción de propuestas de mejora, atendiendo a los resultados obtenidos.
- Redacción de conclusiones y estudios futuros.

De esta forma podemos decir que este Trabajo Fin de Máster tiene los componentes esenciales. Por un lado, el componente teórico donde se realizará el estudio del ENI y Firma electrónica, y posteriormente realizar una solución práctica, implantando la solución en un contenedor de aplicaciones de forma ágil, dando una visión real y útil al proyecto.

5 https://administracionelectronica.gob.es/pae_Home/pae_Estrategias/pae_Interoperabilidad_Inicio.html

1.4 Contexto

Para la realización de este Trabajo Fin de Máster en un contexto real, útil y práctico es recomendable llevarlo a cabo en un entorno profesional, utilizando herramientas y frameworks actuales, así como metodologías ágiles. Este TFM tiene como objetivo final ceder el análisis y la solución creada al PAe, con el fin que sea un posible punto de inicio en un posible desarrollo de una administración local o cualquier organismo público en la implantación de una solución basada en el ENI.

1.4.1 Problema a resolver

Como se ha comentado anteriormente, desde siempre la Administración Pública se ha considerado tediosa y poco fluida durante la realización de trámites. El ciudadano debe aportar documentación existente, procedente de trámites anteriores, lo que conlleva duplicar la información. La llegada de las tecnologías de la información y la comunicación a la Administración Pública han supuesto una revolución desde el momento en que se han puesto al servicio de los sistemas de información. Es importante indicar que la Administración Pública se caracteriza por gestionar información y es lógico que dicha información se gestione correctamente, no siendo generalmente así.

La utilización de sistemas de información supone acabar con la burocracia existente. Esto sucede por diferentes causas:

- La automatización de los procesos.
- Utilizar estándares como el ENI, supone la posibilidad de compartir información entre las administraciones y evitar la solicitud de documentación innecesaria al ciudadano.

Numerosas administraciones carecen de los sistemas de información necesarios, o de un ejemplo práctico para el comienzo en el desarrollo de sus soluciones propias.

La propuesta de solución a la problemática detectada se analizará en el siguiente punto. Además como se ha indicado anteriormente se realizará y se desplegará la

solución, para finalmente proporcionar al PAe el análisis y código desarrollado, dotando al trabajo con una cierta importancia y utilidad para las administraciones.

1.4.2 Propuesta de solución

Como propuesta para solventar estas dificultades se realizará un análisis de las distintas soluciones que se encuentran disponibles en la web del PAe. Además se realizará un análisis del ENI, centrándose en los metadatos básicos para realizar una tramitación electrónica y una interoperabilidad de datos.

Por último, y como demostración, se realizará una propuesta de arquitectura de microservicios, su desarrollo y despliegue a través de una arquitectura ágil.

Capítulo 2 - Estado del Arte

Existe un gran desconocimiento por parte de las AA.PP. del conjunto de servicios que provee el PAe para la tramitación electrónica en la actualidad.

Además de esto, existe otro gran problema y es por donde comenzar. Es por esto, que los responsables de las AA.PP, en un primer instante deben de realizarse las siguientes preguntas:

- ¿Qué es lo que necesita mi AA.PP?

  Ante esta pregunta, por lo general los responsables de las AA.PP. responderán tramitación electrónica en el mejor de los casos, o simplemente conceptos teóricos que indica una ley o norma.

- ¿Qué servicios suministra el PAe para resolver ese problema ?

  Existe un gran desconocimiento, además que muchos no se encuentran en los canales de los que dispone la administración para mantenerse actualizados.

  Esto supone, en muchos casos, que las AA.PP. locales no sean capaces de adelantarse a los problemas de la sociedad de la información, o a los plazos que determina la administración general.

- ¿Qué arquitecturas similares existen, y cuales se me suministran como punto de partida?

  Se desconoce, ya que las páginas web de la administración general son complejas, y esta información no llega a los responsables locales. Además algunas de ellas no se encuentran mantenidas.

- ¿Cómo integrar los distintos servicios que necesita mi organización?

  Se desconoce, dado que el responsable tiene el conocimiento teórico pero no técnico de la solución.

Este trabajo trata de ayudar a responder cada una de estas preguntas, identificando el problema común de todas las administraciones (administración electrónica), su marco teórico/técnico y la implementación de una solución, basada en sus elementos más importantes: documentos, expedientes y firma electrónica.

Todo esto se realizará a través de la definición de los distintos conceptos importantes, el uso de frameworks actuales, y por último un ejemplo funcional de la solución.

Capítulo 3 - Ley 11/2007, 22/06, Acceso electrónico de los ciudadanos a los Servicios Públicos.

En este capítulo se describen, a grandes rasgos, los aspectos más importantes de la Ley de 11/2007, de acceso electrónico de los ciudadanos a los Servicios Públicos.

3.1 Introducción

El 23 de Junio del a□o 2007 se publica□ en el Boletín Oficial del Estado y un día después entra□ en vigor la Ley 11/2007, de Acceso Electrónico de los Ciudadanos a los Servicios públicos, cuya finalidad es promover el uso de las tecnologías de la Información y las comunicaciones en las relaciones entre la Administración Pública y los ciudadanos y entre las diferentes App. Los ciudadanos podrían realizar todas sus gestiones administrativas por medios electrónicos. De esta forma, las AAPP quedan obligadas a ofrecer sus servicios por Internet, dispositivos móviles o cualquier medio electrónico futuro.

Anteriormente a esta ley, la Ley 30/1992, de 26 de noviembre tuvo como objetivo abrir la posibilidad de mantener relaciones telemáticas con la Administración, pero la demanda era otra, que supuso el análisis, estudio y la implantación del Esquema Nacional de Interoperabilidad en todas las AA.PP Las nuevas realidades, exigencias y experiencias que se han ido poniendo de manifiesto, el propio desarrollo de la sociedad de la información y el cambio de circunstancias tecnológicas y sociales exigen actualizar el contenido, muy diferente al de 1992. Esta regulación exige reconocer el derecho de los ciudadanos, y no sólo la posibilidad, de acceder mediante comunicaciones electrónicas a la administración.

3.2 Ámbito de aplicación y los principios generales

3.2.1 Objetivo de la Ley

- La presente Ley reconoce el derecho de los ciudadanos a relacionarse con las AAPP por medios electrónicos y regula los aspectos básicos de la utilización de las tecnologías de la información en la actividad administrativa, en las relaciones entre las AAPP, así como en las relaciones de los ciudadanos con las mismas con la finalidad de garantizar sus derechos, un tratamiento común ante ellas y la validez y eficacia de la actividad administrativa en condiciones de seguridad jurídica.
- Las AAPP utilizarán las tecnologías de la información de acuerdo con lo dispuesto en la presente Ley, asegurando la disponibilidad, el acceso, la integridad, la autenticidad, la confidencialidad y la conservación de los datos, informaciones y servicios que gestionen en el ejercicio de sus competencias.

3.2.2 Ámbito de aplicación

- La presente Ley, en los términos expresados en su disposición final primera, será de aplicación:
    - A las AAPP, entendiendo por tales la Administración General del Estado, las Administraciones de las Comunidades Autónomas y las Entidades que integran la Administración Local, así como las entidades de derecho público vinculadas o dependientes de las mismas.
    - A los ciudadanos en sus relaciones con las AA.PP
    - A las relaciones entre las distintas AA.PP
- La presente Ley no será de aplicación a las AAPP en las actividades que desarrollen en régimen de derecho privado.

3.2.3 Finalidades de la Ley

Son fines de la presente Ley:

- Facilitar el ejercicio de derechos y el cumplimiento de deberes por medios electrónicos.
- Facilitar el acceso por medios electrónicos de los ciudadanos a la información y al procedimiento administrativo, con especial atención a la eliminación de las barreras que limiten dicho acceso.
- Crear las condiciones de confianza en el uso de los medios electrónicos, estableciendo las medidas necesarias para la preservación de la integridad de los derechos fundamentales, y en especial los relacionados con la intimidad y la protección de datos de carácter personal, por medio de la garantía de la seguridad de los sistemas, los datos, las comunicaciones, y los servicios electrónicos.
- Promover la proximidad con el ciudadano y la transparencia administrativa, así como la mejora continuada en la consecución del interés general.
- Contribuir a la mejora del funcionamiento interno de las AAPP, incrementando la eficacia y la eficiencia de las mismas mediante el uso de las tecnologías de la información, con las debidas garantías legales en la realización de sus funciones.
- Simplificar los procedimientos administrativos y proporcionar oportunidades de participación y mayor transparencia, con las debidas garantías legales.
- Contribuir al desarrollo de la sociedad de la información en el ámbito de las AAPP y en la sociedad en general.

3.2.4 Conclusión de la Ley

La comunicación entre los ciudadanos y las administraciones, y la agilidad de ésta, es un factor sumamente importante desde hace décadas. Como podemos

observar, desde los organismos gubernamentales se lleva trabajando desde antaño, ya que mejora notablemente la tramitación y los tiempos entre administraciones, o entre ciudadanos y administración.

Fuente : https://www.boe.es/buscar/act.php?id=BOE-A-2007-12352

Capítulo 4 - Real Decreto 4/2010, Regulación del Esquema Nacional de Interoperabilidad en el ámbito de la Administración Electrónica.

En este capítulo se describen, a grandes rasgos, los aspectos más importantes del Real Decreto 4/2010, Regulación del ENI, en el ámbito de la Administración Electrónica.

4.1 Introducción

La interoperabilidad, es la capacidad de los sistemas de información y de los procedimientos a los que pertenecen, de compartir datos, y posibilitar el intercambio de información entre ellos. Resulta necesaria para la cooperación, el desarrollo, y la prestación de servicios conjuntos; para la transferencia de tecnología y la reutilización de aplicaciones en beneficio de una mejor eficiencia; para la cooperación entre diferentes AAPP, todo ello facilitando el desarrollo de la administración electrónica y de la sociedad de la información.

El intercambio de datos e información, por parte de las AAPP, es una tarea constante y diaria, en el ámbito de la Administración electrónica.

4.2 Ámbito de aplicación

El ámbito de aplicación del presente real decreto será el establecido en el artículo 2 de la Ley 11/2007, de 22 de junio.

El ENI y sus normas de desarrollo, prevalecerán sobre cualquier otro criterio en materia de política de interoperabilidad en la utilización de medios electrónicos para el acceso de los ciudadanos a los servicios públicos.

4.3 Principios básicos del Esquema Nacional de Interoperabilidad.

La aplicación del ENI se desarrollará de acuerdo con los principios generales establecidos en el artículo 4 de la Ley 11/2007, de 22 de junio, y con los siguientes principios específicos de la interoperabilidad:

- La interoperabilidad como cualidad integral.
- Carácter multidimensional de la interoperabilidad.
- Enfoque de soluciones multilaterales.

4.4 La interoperabilidad como cualidad integral.

La interoperabilidad se tendrá presente de forma integral desde la concepción de los servicios y sistemas y a lo largo de su ciclo de vida: planificación, diseño, adquisición, construcción, despliegue, explotación, publicación, conservación y acceso o interconexión con los mismos.

4.5 Carácter multidimensional de la interoperabilidad.

La interoperabilidad se entenderá contemplando sus dimensiones organizativa, semántica y técnica. La cadena de interoperabilidad se manifiesta en la práctica en los acuerdos interadministrativos, en el despliegue de los sistemas y servicios, en la determinación y uso de estándares, en las infraestructuras y servicios básicos de las App y en la publicación y reutilización de las aplicaciones de las App, de la documentación asociada y de otros objetos de información. Todo ello sin olvidar la dimensión temporal que ha de garantizar el acceso a la información a lo largo del tiempo.

4.6 Enfoque de soluciones multilaterales.

Se favorecerá la aproximación multilateral a la interoperabilidad de forma que se puedan obtener las ventajas derivadas del escalado, de la aplicación de las arquitecturas modulares y multiplataforma, de compartir, de reutilizar y de colaborar.

4.6.1 Conclusión

El Real Decreto tiene como objeto establecer las líneas a seguir tanto en interoperabilidad semántica y cómo técnica, definiendo los modelos de datos en el intercambio de información, así como los estándares, siempre abiertos, para su implantación.

Fuente: https://boe.es/buscar/doc.php?id=BOE-A-2010-1331

Capítulo 5 - Esquema Nacional de Interoperabilidad

5.1 Introducción

El artículo 156 de la Ley 40/2015 recoge el ENI que "comprende el conjunto de criterios y recomendaciones en materia de seguridad, conservación y normalización de la información, de los formatos y de las aplicaciones que deberán ser tenidos en cuenta por las App para la toma de decisiones tecnológicas que garanticen la interoperabilidad".

El ENI fue establecido anteriormente en el artículo 42 de la Ley 11/2007 y se materializa en el Real Decreto 4/2010, de 8 de enero, por el que se regula el ENI en el ámbito de la Administración Electrónica.

Las NTI concretan detalles para facilitar los aspectos más prácticos y operativos de la interoperabilidad entre las App y el ciudadano. Se acompañan de guías de aplicación y de otros documentos complementarios.

El ENI es el resultado de un trabajo coordinado por el Ministerio de Asuntos Económicos y Transformación Digital. Todas las App participaron en la elaboración del ENI a través de los órganos colegiados con competencia en materia de administración digital; y se contó también con la opinión de las asociaciones de la industria del sector de tecnologías de la información y las comunicaciones.

En la elaboración del ENI y de sus normas técnicas se tuvieron presentes las recomendaciones provenientes de la Unión Europea, particularmente del el Marco Europeo de Interoperabilidad disponible a la fecha; por tanto, el ENI se ubica en el contexto europeo de políticas, actos, documentos y servicios relativos a la interoperabilidad; además, contempla la noción del enlace con los diversos instrumentos equivalentes del ámbito de la Unión Europea, sean redes de comunicaciones, componentes elementales (building blocks), o servicios para la cooperación en interoperabilidad o la reutilización.

5.2 Objetivos

El ENI persigue los objetivos siguientes:

- Comprender los criterios y recomendaciones que deberán ser tenidos en cuenta por las App para la toma de decisiones tecnológicas que garanticen la interoperabilidad y que eviten la discriminación a los ciudadanos por razón de su elección tecnológica. Y que por tanto contribuyen a crear las condiciones necesarias para la interoperabilidad en el uso de los medios electrónicos que permitan a los ciudadanos y a las administraciones el ejercicio de derechos y el cumplimiento de deberes a través de estos medios.
- Proporcionar los elementos comunes que han de guiar la actuación de las App en materia de interoperabilidad, para facilitar la interacción de las App, así como la comunicación de los requisitos de interoperabilidad a la industria.
- Facilitar la implantación de las políticas de seguridad, al contribuir a un escenario de mayor racionalidad técnica y de economías de escala.
- La interoperabilidad se concibe, al igual que la seguridad, desde una perspectiva integral, de manera que no caben actuaciones puntuales o tratamientos coyunturales, debido a que la debilidad de un sistema la determina su punto más frágil y, a menudo, este punto es la coordinación entre medidas individualmente adecuadas pero deficientemente ensambladas.

5.3 Elementos del Esquema Nacional de Interoperabilidad.

Los elementos principales del ENI son:

- Los principios básicos de la interoperabilidad: la interoperabilidad como cualidad integral presente desde la concepción de los servicios y sistemas y a lo largo de su ciclo de vida; el carácter multidimensional de la interoperabilidad; y el enfoque de soluciones multilaterales.

- La interoperabilidad organizativa: incluye los aspectos relativos a la publicación de servicios a través de la Red de comunicaciones de las AA.PP. (Red SARA), con las condiciones asociadas; la utilización de nodos de interoperabilidad; y el mantenimiento de inventarios de información administrativa.
- La interoperabilidad semántica: a través de la publicación y aplicación de los modelos de datos de intercambio, horizontales y sectoriales, así como los relativos a infraestructuras, servicios y herramientas comunes.
- La interoperabilidad técnica: a través del uso de estándares en las condiciones previstas en la normativa para garantizar la independencia en la elección, la adaptabilidad al progreso y la no discriminación de los ciudadanos por razón de su elección tecnológica.
- Las infraestructuras y los servicios comunes, elementos de dinamización, simplificación y propagación de la interoperabilidad, a la vez que facilitadores de la relación multilateral.
- La utilización, preferentemente, de la Red de comunicaciones de las App españolas para comunicarse entre sí y a la que conectarán sus redes y nodos de interoperabilidad, aplicando el Plan de Direccionamiento de la Administración . La Red SARA presta la citada Red de comunicaciones.
- La reutilización: incluye condiciones de licenciamiento de las aplicaciones, de la documentación asociada y de otros objetos de información que las AA.PP. pongan a disposición de otras administraciones y de los ciudadanos; enlace entre los directorios de aplicaciones reutilizables y consulta por parte de las AA.PP. de las soluciones disponibles para libre reutilización; así como publicación del código de las aplicaciones.
- La interoperabilidad de la firma electrónica y de los certificados: la política de firma electrónica y de certificados de la Administración General del Estado como herramienta que podrá ser utilizada como referencia por otras AA.PP.; aspectos relativos a la validación de

certificados y firmas electrónicas, las listas de confianza, las aplicaciones usuarias, los prestadores de servicios de certificación y las plataformas de validación de certificados y firma electrónica.

- La recuperación y conservación del documento electrónico, como manifestación de la interoperabilidad a lo largo del tiempo, y que afecta de forma singular al documento electrónico.
- Se crean las NTI y los instrumentos para la interoperabilidad.

5.4 Ámbito de aplicación.

El ámbito de aplicación del Esquema Nacional de Seguridad es el establecido en el artículo 2 de las leyes 39/2015[6] y 40/2015[7] sobre el ámbito subjetivo y lo indicado sobre el sector público institucional.

5.5 Adecuación al Esquema Nacional de Interoperabilidad

Para la adecuación al ENI es de interés la guía de auditoría de cumplimiento del ENI que tiene por objetivo facilitar que se pueda realizar una valoración del cumplimiento de las medidas de interoperabilidad, señalando una lista de controles sobre el ENI. Dichos controles se estructuran en tres categorías siguiendo el modelo del ENS:

- Marco organizativo, referido a aquellos controles cuyo cumplimiento exige medidas horizontales, como los aspectos jurídicos, de políticas de actuación o determinadas decisiones, frecuentemente referidas a la gobernanza de la interoperabilidad.
- Marco operacional, referido a aquellos controles cuyo cumplimiento requiere la adopción de prácticas, procedimientos y medidas alineadas con la administración de la interoperabilidad como un conjunto,

[6] https://www.w3.org/XML/
[7] https://www.json.org/json-en.html

incluyendo el diseño, la implementación, la configuración y explotación de sistemas interoperables.

- Medidas técnicas, que suponen requisitos concretos que permiten garantizar la interoperabilidad, incluyendo formatos, vocabularios o protocolos.

Fuente: https://www.boe.es/eli/es/l/2015/10/01/40/con

Capítulo 6 - Normas Técnicas de Interoperabilidad

El ENI establece la serie de NTI que son de obligado cumplimiento por las AA.PP. y que desarrollan aspectos concretos de la interoperabilidad entre las AA.PP. y con los ciudadanos. Se trata de las normas que siguen junto con sus guías de aplicación y otros documentos de apoyo:

- Catálogo de estándares.
- Documento electrónico .
- Digitalización de documentos.
- Expediente electrónico .
- Política de firma electrónica y de certificados de la Administración .
- Protocolos de intermediación de datos.
- Relación de modelos de datos.
- Política de gestión de documentos electrónicos.
- Requisitos de conexión a la Red de comunicaciones de las AA.PP. españolas.
- Procedimientos de copiado auténtico y conversión entre documentos electrónicos, así como desde papel u otros medios físicos a formatos electrónicos.
- Modelo de Datos para el intercambio de asientos entre las Entidades Registrales.
- Reutilización de recursos de información.
- Reutilización y transferencia de tecnología.
- Declaración de conformidad con el ENI.
- URL´s de esquemas XML.

Toda la información detallada de cada uno de estos puntos se encuentra en https://administracionelectronica.gob.es/pae_Home/pae_Estrategias/pae_Interoperabilidad_Inicio/pae_Normas_tecnicas_de_interoperabilidad.html, y dado que excesivo

solo se hará incapie en los aspectos más destacados sobre los que recae todo el peso de este trabajo final de master. Esos son los siguientes:

6.1 Documento electrónico

En la Resolución de 19 de julio de 201, de la Secretaría de Estado para la Función Pública, es donde se aprueba la Norma Técnica de Interoperabilidad de Documento Electrónico.

Las diferencias a la NTI, se han desarrollado con el objetivo de cubrir las necesidades derivadas de la normativa aplicable en un planteamiento de partida basado en mínimos, de forma que se garantice la interoperabilidad entre las distintas administraciones favoreciendo su implantación y aplicación en un corto plazo con un impacto mínimo, pero sin perder una orientación de desarrollo y perfeccionamiento a lo largo del tiempo, en paralelo al progreso de los servicios de Administración Electrónica, de las infraestructuras que los apoyan y de la evolución tecnológica.

6.1.1 Definición

Un documento administrativo electrónico es, el objeto digital administrativo que contiene la información objeto (datos y firma) y los datos asociados a ésta (metadatos).

Este reside, desde el mismo instante de su captura, en el sistema de gestión de documentos de una determinada organización, donde permanecen inalteradas las características de autenticidad, fiabilidad e integridad, que confieren al documento valor probatorio, en el marco de la política de gestión de documentos electrónicos correspondiente.

Aunque pueden definirse diferentes dimensiones y componentes digitales, para la gestión, tratamiento y conservación de un documento electrónico, éste se considera de forma funcional o conceptual como una unidad.

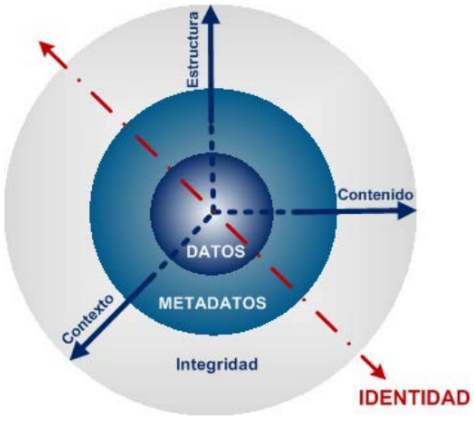


Figura 1: Dimensiones de un documento digital

Fuente: Administración Electrónica PAe

Entre estos componentes y dimensiones de alto nivel, destacaría los siguientes:

- El contenido, entendido como el conjunto de datos en que se sustenta la información de un documento electrónico.
- La firma electrónica, que:
  - Permite detectar cualquier cambio de los datos firmados.
  - Está vinculada al firmante de manera única y a los datos a los que se refiere.
  - Ha sido creada por medios que el firmante puede mantener bajo su exclusivo control.
- Los metadatos, elemento que proporciona contexto al contenido, estructura y firma de un documento, contribuyendo al valor probatorio y fiabilidad de éste a lo largo del tiempo como evidencia electrónica de las actividades y procedimientos.

6.1.2 Estructura lógica

En base a la definición formal del documento electrónico expresada en el punto anterior, el documento electrónico constituye una unidad documental estructurada

con datos y metadatos que contribuyen a cada una de las dimensiones del documento electrónico.

Para facilitar la interpretación y correcta aplicación de las condiciones establecidas para los documentos electrónicos en la NTI de Documento Electrónico y resto de NTIs de contenido relacionado, a nivel lógico, el documento electrónico se representa, en cuanto a su gestión, como una unidad lógica de tratamiento o contenedor con la siguiente estructura lógica:

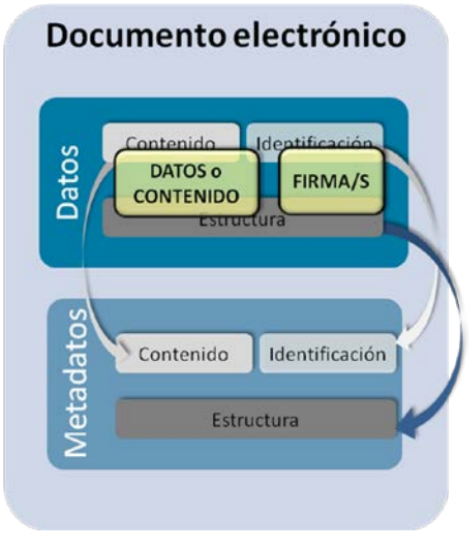


Figura 2: Estructura de un documento electrónico

Fuente: Administración Electrónica PAe

Veamos cada uno de los elementos.

6.1.2.1 Datos

Formado por los datos de contenido e identificación del documento.

- Datos o contenido informativo: fichero que soporta el contenido.
- Datos de identificación para la autenticación y validación del documento: firma electrónica o firmas asociadas al fichero de contenido anterior.

6.1.2.2 Metadatos

Bloque de metadatos asociados al contenido, que definen su estructura e identificación este de forma única.

6.2 Expediente electrónico

6.2.1 Definición

El expediente administrativo es el conjunto ordenado de documentos y actuaciones que sirven de antecedente y fundamento a la resolución administrativa, así como las diligencias encaminadas a ejecutarla, estableciendo además que *los expedientes tendrán formato electrónico*.

6.2.2 Estructura

Para garantizar la vinculación de los documentos electrónicos que conforman un expediente, la Ley define el *índice electrónico como un índice numerado y autentificado de todos los documentos que contenga cuando se remita*, estableciendo además que *la autenticación del citado índice garantizará la integridad e inmutabilidad* del expediente electrónico generado desde el momento de su firma y permitirá su recuperación siempre que sea precisa.

Además del citado índice electrónico el expediente estará formado por:

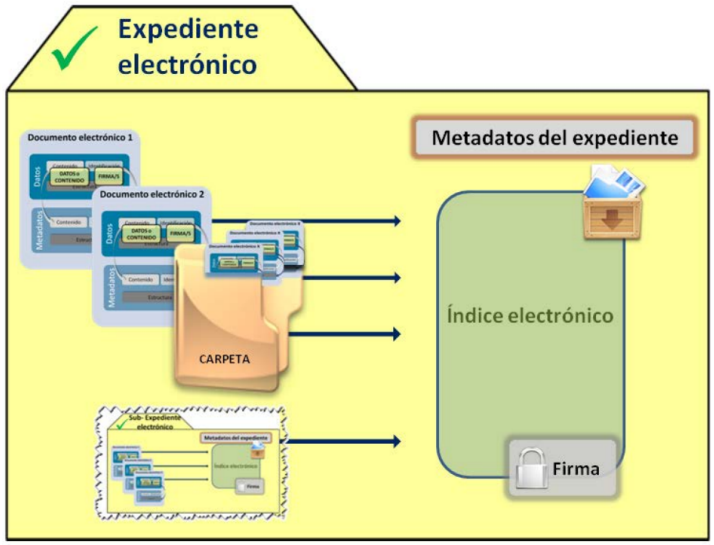


Figura 3: Estructura de un expediente electrónico

Fuente: Administración Electrónica PAe

- Documentos electrónicos: objetos digitales administrativos de cada una de las actuaciones administrativas que integran el expediente, y que contienen la información (contenido y firma) y los datos asociados a ésta (metadatos) que, como tales, han de cumplir lo establecido en la NTI de Documento Electrónico.
- Índice electrónico: que constituye un objeto digital que contiene la identificación sustancial de los documentos electrónicos que componen el expediente debidamente ordenada para reflejar la disposición de los documentos, así como otros datos con el fin de preservar la integridad y permitir la recuperación del mismo
- Firma del índice electrónico: firma electrónica que garantiza la autenticidad e integridad del contenido del índice, y por extensión, de los documentos que conforman el expediente electrónico así como de su estructura.
- Metadatos del expediente: constituyen un conjunto de datos que proporciona contexto al contenido, estructura y firma del expediente,

contribuyendo al valor probatorio de éste a lo largo del tiempo. Además, los metadatos del expediente podrán incluir particularidades procedimentales de cara a facilitar su gestión.

6.2.3 *Ciclo de Vida*

Las fases del ciclo de vida del expediente electrónico se desarrollan como un ciclo continuo, por lo que los procesos de gestión de documentos y expedientes podrán renovarse en función de las necesidades que se produzcan a lo largo del ciclo de vida.

Figura 4: Ciclo de vida de un expediente electrónico

Fuente: Administración Electrónica PAe

6.2.3.1 Apertura

La fase de apertura o creación del expediente electrónico conlleva, al menos, la realización de las siguientes acciones:

- Creación del objeto administrativo digital expediente.
- Creación del índice electrónico.
- Inclusión en el índice de las referencias de cada uno de los documentos.
- Asignación de metadatos mínimos obligatorios del expediente electrónico. Estos metadatos mínimos se describirán más adelante.

6.2.3.2 Tramitación

Durante la fase de tramitación tendrán lugar actuaciones administrativas relacionadas con el expediente. Estas actuaciones podrían tener como consecuencia:

- La inclusión de nuevos documentos en el expediente.
- Cambios de estado del expediente. Esto se refleja a través de los cambios que se producen en los metadatos del expediente.
- Creación de sub-expedientes, dentro del expediente principal, o relación de este con otro expediente con el que guarde relación.

6.2.3.3 Cierre

El cierre del expediente no debe ser entendido en el sentido estricto asociado al cierre del procedimiento administrativo correspondiente, sino como proceso que permite obtener y conservar las características, contenido y estado de un expediente en un momento del tiempo.

En el proceso de cierre de un expediente cabe contemplar, al menos, los siguientes aspectos:

- Compleción del índice del expediente. Asegurar que se incluyen todas las referencias a los documentos electrónicos del expediente, así como de los metadatos mínimos en el momento del cierre.
- El foliado o indizado del expediente. Cuando en virtud de una norma sea preciso remitir el expediente electrónico, se hará de acuerdo con lo previsto en el ENI y en las correspondientes NTIs, y se enviará completo, foliado, autentificado ya acompañado de un índice, asimismo autentificado, de los documentos que contenga. La autenticación del citado índice garantizará la integridad e inmutabilidad del expediente electrónico generado desde el momento de su firma y permitirá su recuperación siempre que sea preciso, siendo admisible que un mismo documento forme parte de distintos expedientes electrónicos. El proceso contempla:

- Identificadores de los documentos electrónicos.
- Huellas digitales de dichos documentos electrónicos.
- La fecha de incorporación del documento electrónico al expediente.
- El orden del documento dentro del expediente.
- Otro elemento que indique el patrón correspondiente, en su caso, para el proceso de indexado o foliado.
- Datos generales sobre el expediente, como puede ser la fecha de apertura o de cierre del expediente.
- Una vez que el índice electrónico ha sido conformado por los elementos anteriormente definidos, se procede a firmar el índice electrónico al objeto de que quede debidamente garantizada la autenticidad e integridad de su contenido, y por extensión del expediente.
- La siguiente figura ilustra el procedimiento de indizado descrito.

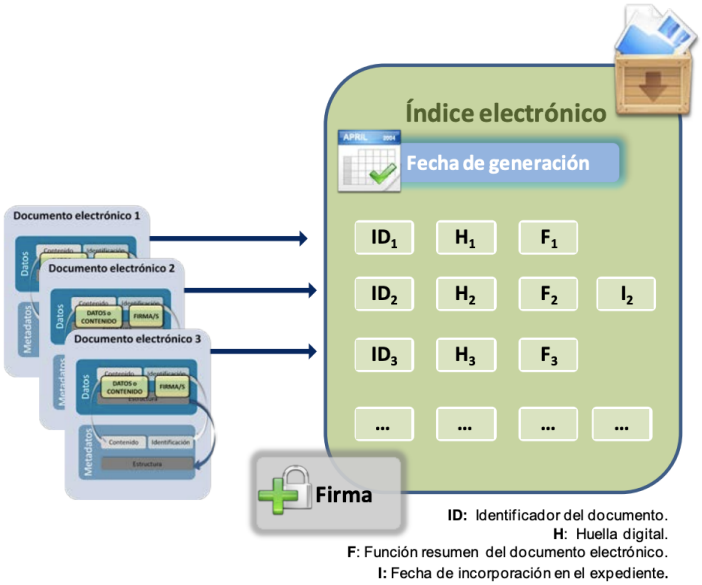


Figura 5: Estructura de un índice electrónico

Fuente: Administración Electrónica PAe

6.2.3.4 Conservación y selección

En la fase de conservación y selección, una vez transcurrido el periodo de validez administrativa de un expediente, que puede resultar de un período sin actividad sobre los documentos, aquellos expedientes o documentos integrantes de los mismos con valor efímero se eliminan reglamentariamente, en tanto que los que tienen valor a largo plazo en atención a su utilidad administrativa, jurídica, archivística, histórica o de investigación y social, y según establezcan las autoridades competentes, se conservan permanentemente, en cualquier caso atendiendo a lo establecido en la NTI.

6.3 Firma Electrónica

Los documentos administrativos electrónicos, y aquellos susceptibles de formar parte de un expediente, tendrán siempre asociada al menos una firma electrónica de acuerdo con la normativa aplicable. Como resulta evidente estos documentos serán los propios del expediente, así como el índice electrónico.

De forma general, para garantizar la autenticidad e integridad de cualquier documento electrónico cabe contemplar la asociación de, al menos, una firma electrónica. *Nunca un documento puede existir dentro del sistema sin poseer al menos una firma electrónica asociada.*

Respecto a la firma electrónica asociada a cualquier tipo de documento, hemos de distinguir dos procedencias:

- Firma electrónica por parte del ciudadano, cuya procedencia debe estar certificado electrónico emitido por una entidad válida.
- Firma electrónica por parte de las organizaciones, la cual puede tener dos procedencias. Certificado electrónico al igual que los ciudadanos o Sistemas de Código Seguro de Verificación, en adelante **CSV**. En caso de existir, el valor del CSV y la referencia a la Orden o Resolución que regula su generación se debe de incorporar al documento electrónico correspondiente como metadato mínimo obligatorio.

En ambos casos es un conjunto de datos electrónicos que acompañan o que están asociados a un documento electrónico y cuyas funciones básicas son:

- Identificar al firmante de manera inequívoca.
- Asegurar la integridad del documento firmado. Asegura que el documento firmado es exactamente el mismo que el original y que no ha sufrido alteración o manipulación.
- Asegurar el no repudio del documento firmado. Los datos que utiliza el firmante para realizar la firma son únicos y exclusivos y, por tanto, posteriormente, no puede decir que no ha firmado el documento.

La firma electrónica basada en certificado, es la más utilizada, tanto por el ciudadano, al ser la única disponible, como por parte de las organizaciones, al disponer de distintas entidades emisoras válidas.

6.3.1 Firma Electrónica basada en certificado

El proceso básico de firma basado en certificado es el siguiente:

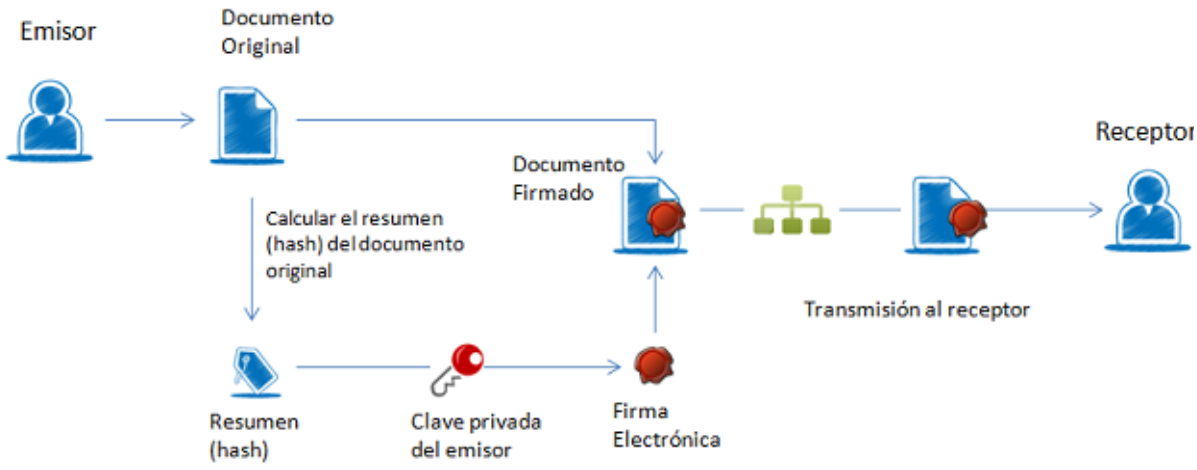


Figura 6: Proceso de firma basado en certificado electrónico

Fuente: Administración Electrónica PAe

- El usuario dispone de un documento electrónico (una hoja de cálculo, un pdf, una imagen, incluso un formulario en una página web) y de un certificado que le pertenece y le identifica.
- La aplicación o dispositivo digital utilizados para la firma realiza un resumen del documento. El resumen de un documento de gran tamaño puede llegar a ser tan solo de unas líneas. Este resumen es único y cualquier modificación del documento implica también una modificación del resumen.
- La aplicación utiliza la clave privada para codificar el resumen.
- La aplicación crea otro documento electrónico que contiene ese resumen codificado. Este nuevo documento es la firma electrónica.

El resultado de todo este proceso es un documento electrónico obtenido a partir del documento original y de las claves del firmante. La firma electrónica, por tanto, es el mismo documento electrónico resultante.

Fuente: Ley 40/2015, de 1 de octubre, de Régimen Jurídico del Sector Público

Capítulo 7 - Integración de Servicios

7.1 Introducción

La arquitectura orientada a los servicios, en adelante SOA, es un tipo de diseño que tiene como eje principal reutilizar sus elementos. Eso se logra gracias a la implementación de interfaces de servicios que se comunican a través de una red con un lenguaje común.

Un servicio es una unidad autónoma de una o más funciones diseñada para realizar una tarea específica. Incorpora la interrelación de código y datos que se necesita para llevar a cabo una función.

Expresado de otra forma, la SOA integra los elementos del software que se implementan y se mantienen por separado, y permite que se comuniquen entre sí con el fin de construir elementos más complejos.

Fuente: RedHat

En el libro de Thomas Erl "Service-Oriented Arquitecture: Concepts, Technology and Design" se hace una revisión de los principales conceptos de la SOA.

La orientación a servicios son paradigmas de implementación agnóstica que pueden ser llevados a cabo con cualquier plataforma tecnológica.

Fuente:Service-Oriented Arquitecture: Concepts, Technology and Design

7.2 Ventajas frente al enfoque monolítico

Existen distintas ventajas de las implementaciones SOA frente a las arquitecturas monolíticas del pasado. Algunas son:

- Flexibilidad. La posibilidad de reutilizar los servicios que ofrece la SOA agiliza y simplifica el proceso.
- Menor coste, debido a la flexibilidad y agilidad en el desarrollo.

- Escalabilidad sencilla y eficiente.
- Modularidad supone mejorar el mantenimiento de la plataforma.
- La disponibilidad se produce gracias a la independencia de sus módulos.
- Fácil monitorización al tratarse de elementos independientes.
- Testeo funcional sencillo.

Las diferencias entre una arquitectura y otra la podemos ver a través de la siguiente imagen:

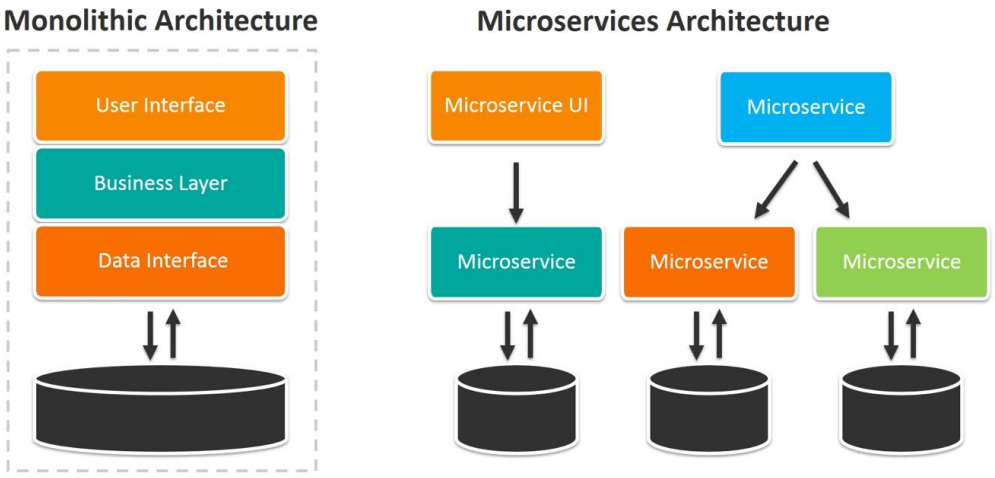


Figura 7: Arquitectura Monolítica y Arquitectura basada en Microservicios

Fuente: Aplicación monolítica o distribuida

Podemos observar lo sencillo que será que se cumplan cada una de las ventajas de SOA frente a las arquitecturas monolíticas.

7.3 Diseño de Servicios

De la misma forma que la orientación a objetos, la orientación a servicios incluye principios comúnmente aceptados. Entre otros, se hace hincapié en el diseño de los componentes y la colocación de estos en la arquitectura.

Algunos de los aspectos claves de los principios de la orientación a servicios son descritos a continuación:

- Relación débil: Los servicios mantienen un acoplo mínimo de dependencias. Se debe de tender a una relación entre ellos a través de una API de elementos.
- Contrató: Los servicios mantienen una relación de comunicación a través de un contrato, definido previamente.
- Autonomía: Cada servicio mantiene su encapsulación de forma independiente al resto de servicios con los que se relacionan.
- Abstracción: La capacidad de abstracción de cada uno de los servicios es de tal magnitud, que se esconde del resto de servicios o de lo que se describe en sus contratos.
- Reutilización: Se promueve la reutilización de funciones con el fin de evitar la duplicidad de funciones o código.
- Compatibilidad: Un conjunto de servicios o funciones puede ser agrupado para formar nuevos servicios más complejos.
- Carencia de estado: Se minimiza la información de estado relacionada con una actividad realizada en una función específica.
- Detectabilidad: Los servicios están implementados mediante tecnologías que permiten su descubrimiento y evaluación.

Con el conocimiento de los elementos que forman la arquitectura y un conjunto de principios de diseño que pueden ser usados para dar forma y estandarizar estos componentes. Con todo esto, falta una plataforma de implementación que permita unificar todas estas piezas para construir soluciones orientadas a servicio.

7.4 Microservicios

Los microservicios son tanto un estilo de arquitectura como un modo de programar. Con los microservicios, las aplicaciones se dividen en elementos más pequeños e independientes entre sí. A diferencia del enfoque tradicional y monolítico

de las aplicaciones, en el que todo se compila en una sola pieza, los microservicios son elementos independientes que funcionan en conjunto para llevar a cabo diferentes tareas. Cada uno de esos elementos o procesos comentados anteriormente son un microservicio.

Fuente: Redhat

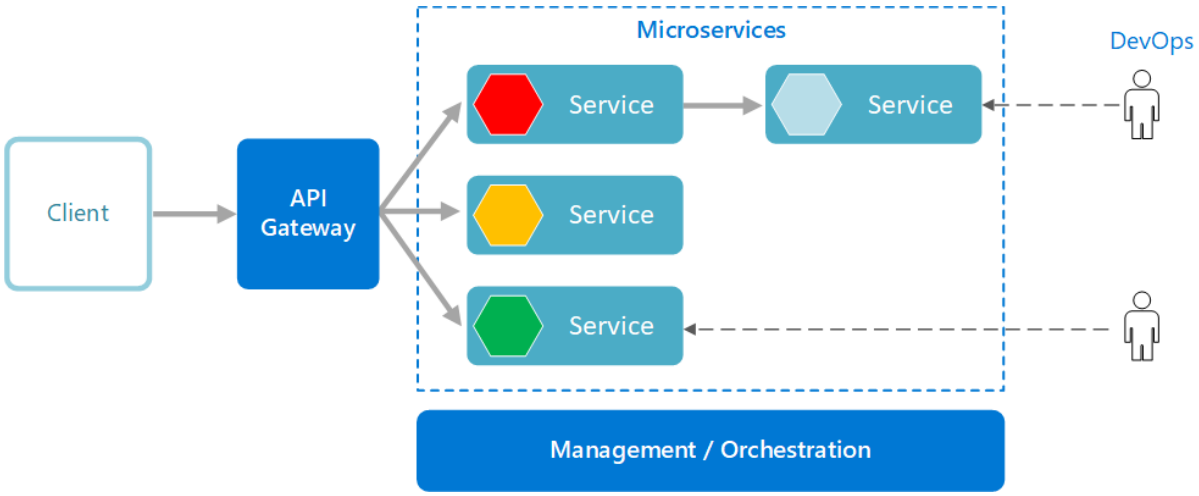


Figura 8: Arquitectura de Microservicios

Fuente:Microsoft

7.4.1 Características

Alguna de las características más importantes son:

- Los microservicios son pequeños e independientes, de modo que el desarrollo lo puede realizar un equipo pequeño.
- Los detalles de la implementación interna de cada servicio se ocultan frente a otros servicios.
- No es necesario que los servicios compartan la misma tecnología, o bibliotecas.
- La comunicación se realiza a través de una API bien definida. Este será uno de los ejes principales en el desarrollo de servicios.

Fuente:Joatham Pérez Expósito

7.4.2 Ventajas

Las principales ventajas de su uso son:

- Desarrollar micro servicios, repercute en poder aislar los errores, cuando se producen y facilitar su resolución.
- Se produce una mezcla de tecnologías, lo cual, enriquece el conjunto.
- Se facilita la escalabilidad cuando las necesidades lo requieren.
- Cada micro servicio puede aislar sus datos, si la lógica de negocio así lo indicará.

Fuente:Joatham Pérez Expósito

7.4.3 ¿Cómo lograrlo?

Para lograrlo, en primer lugar hemos de hacer uso de frameworks que nos permiten la construcción de pequeños módulos, que se puedan comunicar, y que sean independientes los unos de los otros. Existen diferentes para lograrlo, y en este post se describirán las características básicas de dos de estos.

Fuente:Joatham Pérez Expósito

7.4.4 OSGI (Open Services Gateway Initiative)

Es un framework que nace en marzo de 1999, orientado a Servicios e implementado en Java, que define una forma de crear módulos y la forma en que estos interactúan entre sí en tiempo de ejecución. Este es apoyado por la OSGi Alliance, es un consorcio de empresas tecnológicas a nivel mundial que trata de asegurar la interoperabilidad de las aplicaciones y servicios. Algunos ejemplos de miembros: Motorola, Nokia, Mitsubishi Electric Corporation, Vodafone Group Services, LinkedIn, LG Electronics, etc. Como en otros proyectos, un grupo de empresas apoyan un proyecto para solventar sus necesidades.

Este framework provee al desarrollador de un entorno Java gestionado y seguro que permite el despliegue de aplicaciones denominadas bundles. El framework de OSGI permite la descarga, instalación y borrado de bundles en tiempo de ejecución.

OSGi intenta solventar los problemas del tradicional "classloader" de la máquina virtual y de los servidores de aplicaciones Java. Para ello, en OSGI cada bundle tiene su propio classpath separado del resto de classpath de los demás módulos. Destacaría esta como la principal característica del desarrollo a través de bundles, siendo totalmente opuesto a los .jar característicos de Java.

La arquitectura OSGI se divide en capas, tal y como se muestra en la siguiente figura, las cuales se detallan a continuación:

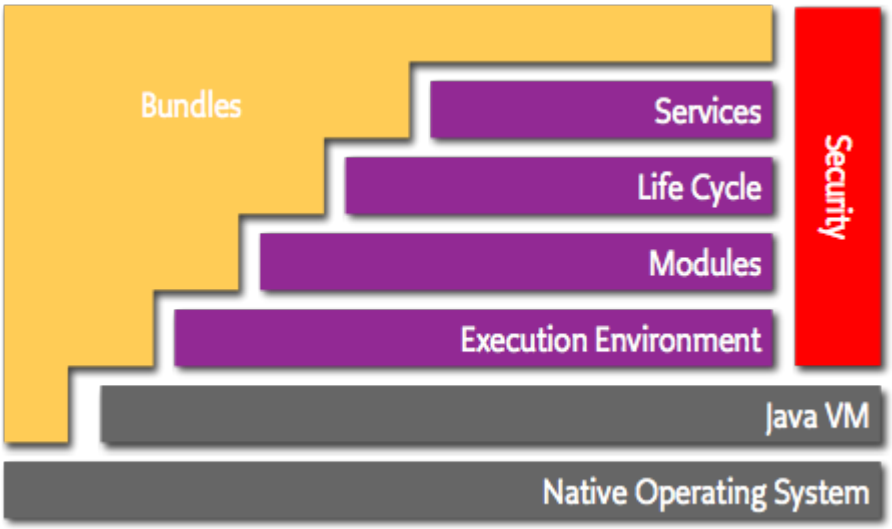


Figura 9: Arquitectura OSGI

Fuente: Osgi-architecture

- Bundles: Componentes OSGI creados por los desarrolladores.
- Servicios: Capa encargada de conectar distintos bundles de manera dinámica.
- Ciclo de vida: API que permite instalar, iniciar, parar, actualizar y desinstalar bundles sin necesidad de reiniciar el framework.

- Módulos: Capa que define cómo importar/exportar código fuente de un bundle.
- Seguridad: Capa que administra los aspectos de seguridad del framework.
- Entorno de ejecución: Especifica qué métodos están disponibles en la plataforma.

Fuente:Joatham Pérez Expósito

7.4.4.1 Resolución de dependencias

OSGI utiliza un versionado semántico para la resolución de dependencias. Con él, OSGI siempre encuentra la mejor opción en tiempo de ejecución.

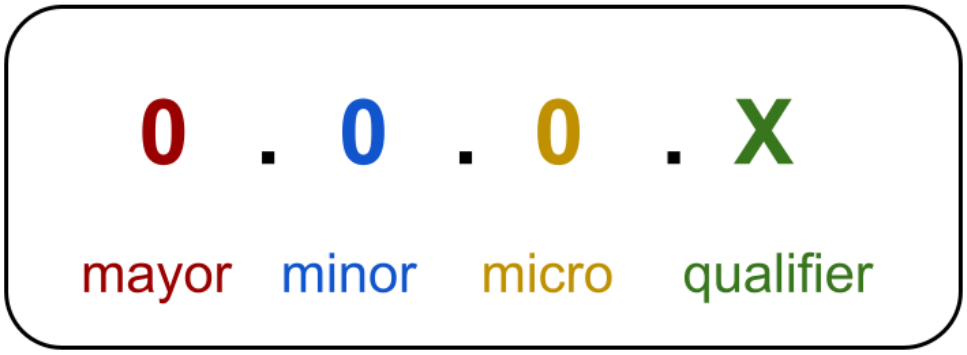


Figura 10: Versionado de librerías

Fuente: Propia

- Major: Se rompe la retrocompatibilidad de la API. Supone una actualización tanto para consumidor como el publicador de la API.
- Minor: Se mantiene la compatibilidad para el consumidor pero no para el proveedor de la API.
- Micro: Pequeña actualización en la API, que puede suponer por ejemplo la corrección de un bug.

- Qualifier: Un identificador único, como puede ser la fecha de creación (timestamp).

El consumidor de una API consumida a través de esta especificación de dependencias, debe importar un rango de versiones. Por ejemplo: [4.2,5).

7.4.4.2 Manifiest

La gestión de las dependencias entre bundles (módulos), así como toda la información e información de una librería se almacena dentro del fichero MANIFIEST que a su vez se almacena dentro de la librería.

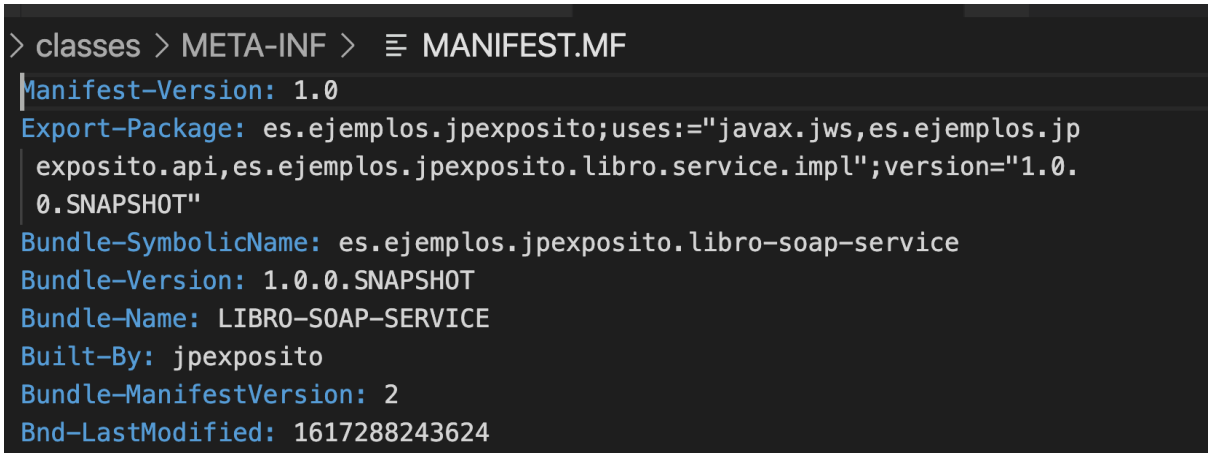


Figura 11: Manifiest.mf de una librería osgi

Fuente: Propia

Todo lo que se ha expuesto hasta el momento podemos simplificarlo a través de la siguiente imagen:

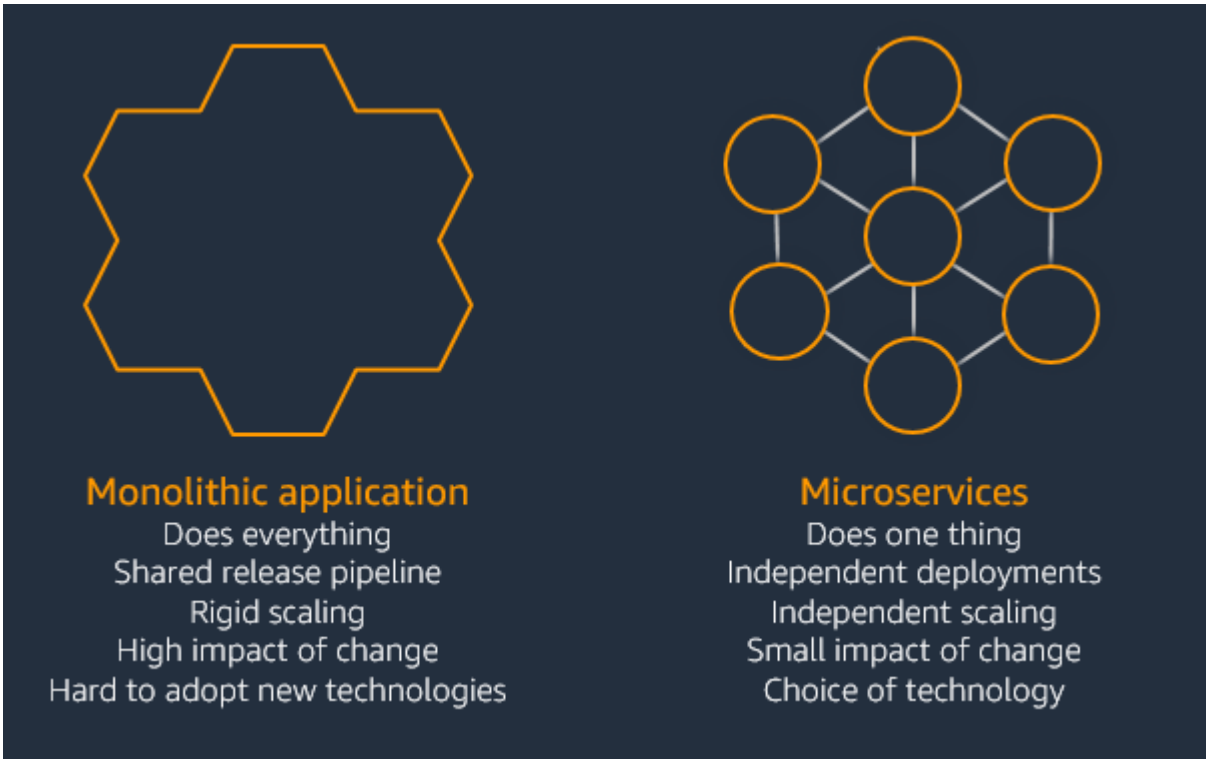


Figura 12: Aplicación monolítica - microservicios

Fuente: Amazon - Design Your Workload Service Architecture

7.5 APIS REMOTAS

Las API remotas están diseñadas para interactuar en una red de comunicaciones. Generalmente las API remotas son web, ya que hacen uso del protocolo HTTP. Los mensajes de comunicación a través de estas API tienen estructura XML o JSON, debido a que presentan los datos en una manera fácil de manejar para otras aplicaciones.

A medida que se han difundido las API, se desarrolló una especificación de protocolo para permitir la estandarización del intercambio de información; se llama Protocolo de Acceso a Objetos Simples, más conocido como SOAP. Las API diseñadas con SOAP usan XML para el formato de sus mensajes y reciben solicitudes a través de HTTP o SMTP. Con SOAP, es más fácil que las aplicaciones que funcionan en entornos distintos o están escritas en diferentes lenguajes compartan información.

Otra especificación es la Transferencia de Estado Representacional (REST). Las API web que funcionan con las limitaciones de arquitectura REST se llaman API de RESTful. La diferencia entre REST y SOAP es básica: SOAP es un protocolo, mientras que REST es un estilo de arquitectura. Esto significa que no hay ningún estándar oficial para las API web de RESTful.

Fuente:Redhat

Existe aún una tendencia de los servicios SOAP frente a REST en lo que se refiere a plataformas de integración de servicios dentro de las soluciones propuestas por el PAe. Esta tendencia y los motivos se describirán más adelante mostrando casos de éxito en este tipo de plataforma y mostrando información relevante de alguna de ellas.

En los siguientes puntos se describe el concepto SOAP y REST, así como de distintos conceptos que nos ayudará a comprender las necesidades que tendrá la solución a construir.

7.5.1 SOAP

SOA simplifica un modelo arquitectónico, donde se realiza una descomposición de los elementos en varias unidades lógicas pequeñas. Consiste en evitar fuertes conexiones entre los elementos que componen la solución, así como su descomposición en elementos más simples, eliminando entre ellas dependencias muy restrictivas.

SOA propone la existencia de las unidades lógicas autónomas pero no aisladas del resto. Las unidades lógicas deben evolucionar a través de estándares reconocidos, que les permita evolucionar de forma independiente. Cada unidad se conocerá como servicio o función de este.

Indicar, por último que SOAP presenta como modelo de conexión entre elementos el estándar XML[8], tanto en clientes que consumen la plataforma o la comunicación que se realiza entre elementos dentro del servicio.

Algunas de las características de SOAP son las siguientes han sido descritas en puntos anteriores:

- Encapsulación de la lógica en servicios. Descrita en la autonomía en la fase de diseño de los servicios.
- Relación entre servicios. Definida en el contrato durante la fase de diseño de los servicios. Los servicios intercambian información para interactuar y realizar una acción. Es por esto, que se necesitan Frameworks que fomenten el acopló débil entre unidades .
- Comunicación entre servicios. La comunicación entre los elementos que forman parte del servicio se realizará a través de protocolos establecidos, con el fin de simplificar la transferencia de información entre estos, favoreciendo siempre el acoplo débil entre los elementos.

7.5.2 REST

REST representa una evolución respecto a SOAP, siendo cualquier interfaz de comunicación entre sistemas que se comuniquen a través de HTTP, ampliando los formatos posibles, e incluyendo JSON[9]. Es una alternativa que se ha consolidado frente a SOAP, que dispone de una gran capacidad pero también de mucha complejidad.

REST presenta una serie de características que lo diferencias frente a SOAP, como son:

- Carencia de estado en el protocolo cliente/servidor sin estado: cada petición contiene toda la información HTTP necesaria para su ejecución. Esto evita almacenar información previa para la ejecución correcta.

[8] https://www.boe.es/eli/es/l/2015/10/01/40/con
[9] https://www.boe.es/buscar/act.php?id=BOE-A-2015-10565

- Las operaciones básicas sobre los datos: POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar).
- Las peticiones se modifican a partir de un identificador único de cada recurso en el sistema (URI).
- Interfaz uniforme para la transferencia de información, a través de operaciones (POST, GET, PUT y DELETE) y sobre los recursos, identificados con una URI.
- Sistema basado en capas. Existe una arquitectura jerárquica entre los componentes. Cada capa lleva a cabo una funcionalidad dentro del sistema.

7.6 Camel

7.6.1 El problema

Uno de los principales problemas de la integración de servicios es la gestión de las dependencias entre todos aquellos que forman parte de la arquitectura. Un ejemplo lo representa la siguiente imagen:

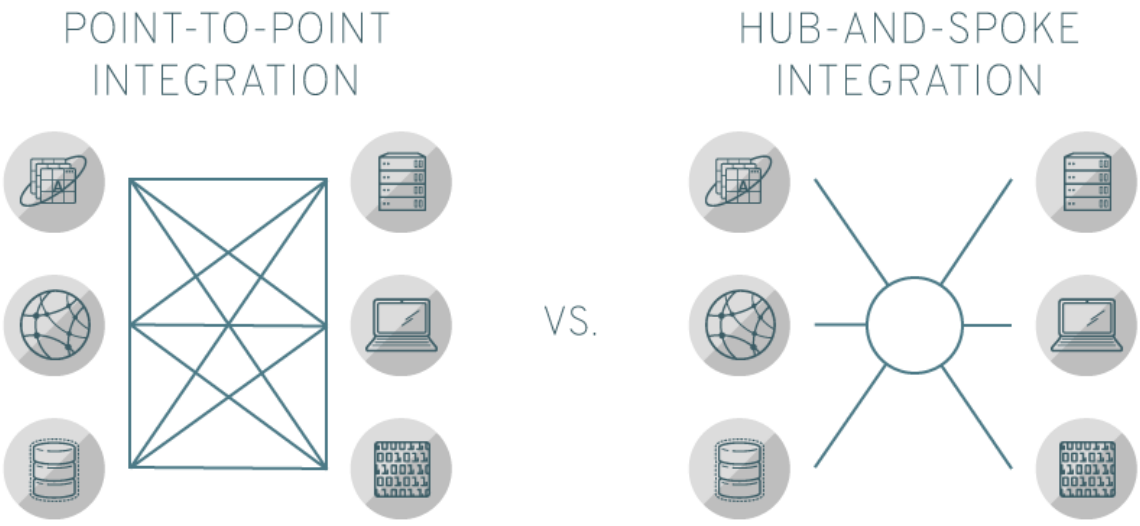


Figura 13: Aplicación punto a punto - bus de integración

Fuente:RedHat

Hemos de utilizar un framework que permita limitar las dependencias entre servicios. Este debe permitir, entre otras cosas: una sencilla comunicación, desarrollos paralelos independientes, y una API conocida y robusta.

7.6.2 Una buena solución

Figura 14: Orquestación Camel

Fuente:camel-apache

Apache Camel es un "enrutador", orientado a mensajes. Gracias al uso de estos patrones, admite todos los protocolos de transporte comunes y tiene un amplio conjunto de adaptadores útiles incluidos. Además permite realizar el desarrollo de otros adaptadores, si no existieran o no se adaptarán a las necesidades.

Con esta simple definición podemos hacer uso de camel como elemento que integra distintos micro servicios, a través de un API bien definida, como se ha comentado anteriormente, y haciendo uso de rutas. Estas rutas se pueden escribir en Scala DSL (XML) o Java.

El concepto básico, pero al igual el más importante son las rutas. Una ruta en camel ó "Camel Route" es el elemento específico del framework, que permite mover un dato desde A hasta B.

Figura 15: Route Camel

Fuente:Redhat

Podemos verlo más gráficamente en la siguiente imagen, donde identificamos cada uno de los elementos que forman parte de una ruta camel.

- Endpoint: Elemento de inicio/final desde donde viene el mensaje y hacia donde va dirigido.
- Route: Ruta específica que realiza una determinada acción.
- Contexto: Contexto sobre el que se ejecuta la ruta. Puede ser más de una, pero lo recomendable es que sea única.

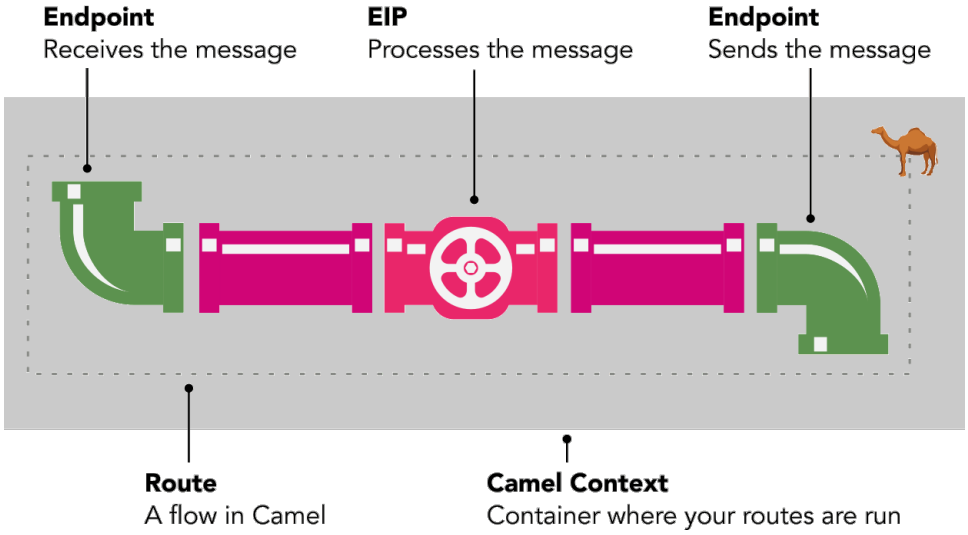


Figura 16: Elementos route camel

Fuente:Tutorial-Camel

La interacción es sincrónica entre componentes y se gestiona a través de la memoria de la aplicación. Existen más operaciones que se permiten hacer cuando realizamos trabajamos con Camel:

- Intercambio de archivos. Una aplicación produce datos compartidos para que otra aplicación los consuma.
- Base de datos común. Hacer que las aplicaciones almacenen los datos que desean compartir en un esquema común, teniendo lugar en una o varias bases de datos. El diseño debe manejar el acceso concurrente, siendo uno de los aspectos más importantes en cualquier implementación de servicios. En este tfm haremos uso de una Api común, que se describe en más adelante, que hace uso de una base de datos única al ser un solución simplificada que forma parte de una solución.
- Llamada de API remota. Proporciona un interfaz para permitir que una aplicación interactúe con otra aplicación en ejecución, como una llamada de método típica.
- Mensajería. Se proponen un funcionamiento a través de mensajes, que fomente que tanto el generador del mensaje, como quien lo consume no deba estar permanentemente conectado, permitiendo destinar recursos a otras acciones.

Esta solo una de las muchas acciones que podemos realizar con Camel, ya que como se ha descrito posee un amplio número de conectores, y en caso de la no existencia, una api para la construcción de aquella que necesitemos. Otra característica importante es que permite la integración con Kubernetes, Kafka, AWS EC2, etc. Esto es debido a la continua actualización del framework.

Capítulo 8 - Entorno de Desarrollo

El entorno de desarrollo girará en torno a herramientas de software libre y que favorezcan la integración y el despliegue continuo[10].

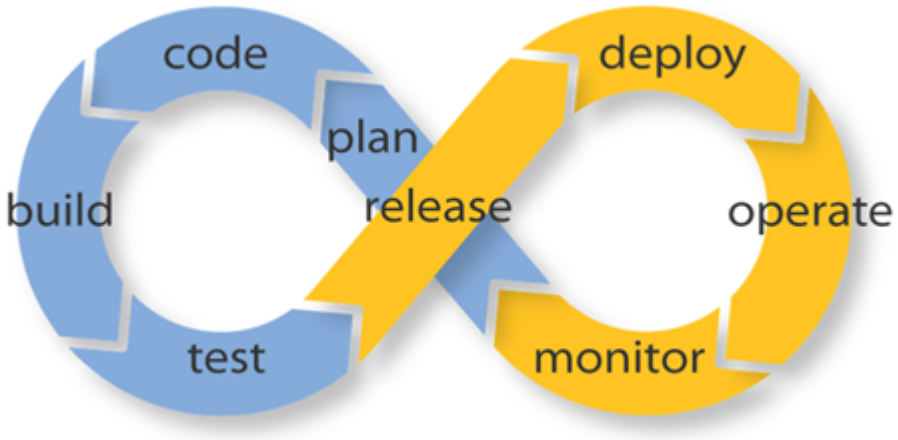


Figura 17: Entorno de desarrollo

Fuente:azurecomcdn

Por limitaciones, no se hará uso de todas las herramientas, pero sí de las más importantes. Además se hará inclusión de un estilo de desarrollo que favorece la calidad del código y el mantenimiento de este.

8.1 Apache Maven

Apache Maven[11] o comúnmente conocido como "Maven" es una herramienta esencial para los programadores en Java. Ésta nos ayudará a construir nuestro proyecto, y nos dará diferentes recursos para fabricar código de calidad.

Presenta un antes y un después en el desarrollo java, ya que ayuda a solventar numerosos problemas relacionados con la gestión de dependencias de nuestros

[10] https://www.eclipse.org/jetty/
[11] Libro: Maven, The Definitive Guide

proyectos, la generación automática de documentación, despliegue en contenedores ligeros, como Jetti[12], el empaquetado automatizado de librerías, etc.

Durante el desarrollo, cada programador poseerá un repositorio[13] maven local (dentro de su equipo) con las dependencias que requiere sus proyectos y aquellas que construya.

Maven utiliza una convención de nombres para el desarrollo de software descrita en este documento. Todas estas cualidades, han hecho que Java adopte gustosamente a Maven como herramienta que ayuda en su desarrollo.

¨La mayor parte de los proyectos open-source, son proyectos maven, lo que promueve que se comparta código entre los desarrolladores.¨

Fuente:Joatham Pérez Expósito

8.1.1 Evolución

Maven nace en 2002 como necesidad para el desarrollo de software, descrito en la instrucción anteriormente. Con el paso de los años, y entre otros motivos, su carácter open-source, ha tenido un crecimiento exponencial. Como ejemplo, veamos el incremento de artefactos (librerías) en su repositorio principal.

[12] https://netbeans.apache.org/
[13] https://www.eclipse.org/downloads/packages/release/kepler/

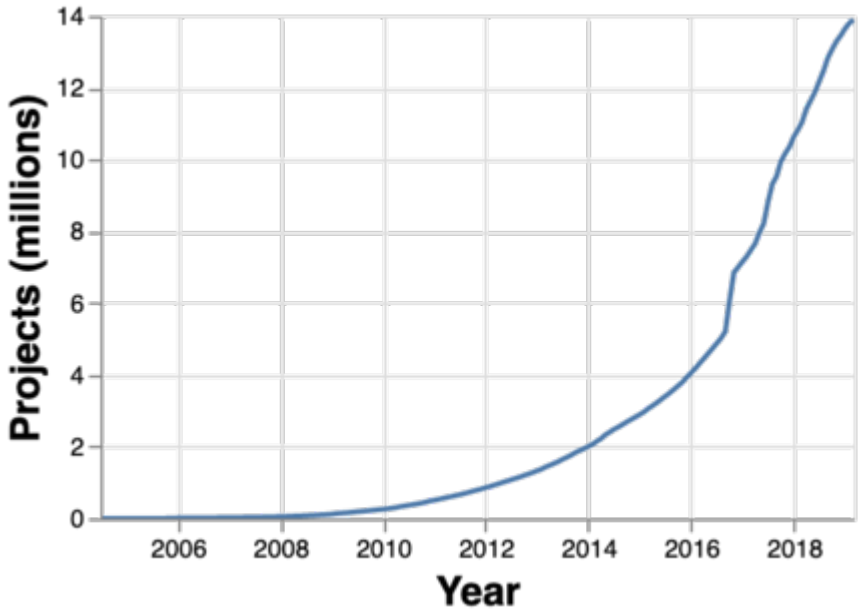


Figura 18: Evolución maven

Fuente: Wikipedia (The number of artifacts on Maven's central repository has grown rapidly)

Indicar por último que maven se convirtió en sustituto de Ant[14], como herramienta para la construcción de artefactos (librerías).

8.1.2 Ciclo de vida

El ciclo de vida de maven se puede resumir a través de la siguiente imagen:

[14] https://www.jetbrains.com/

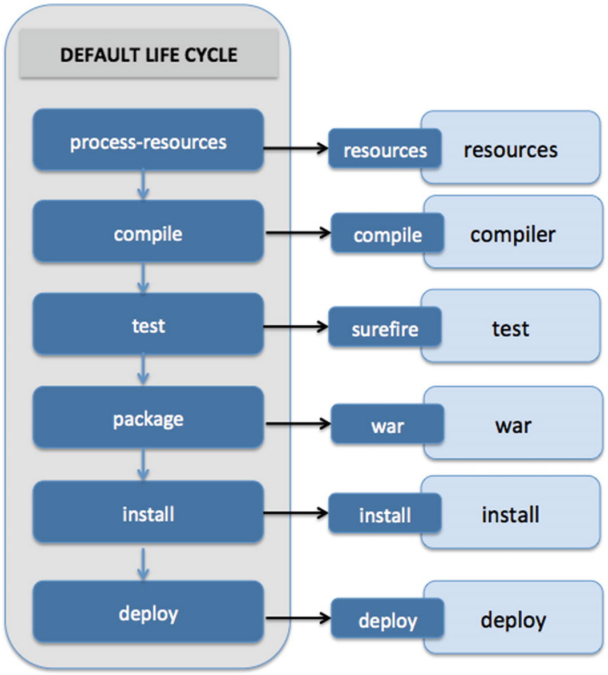


Figura 19: Ciclo de vida maven

Fuente : Maven: The Definitive Guide[15]

Para la ejecución de los distintos comandos que supondrán la ejecución hasta la fase que deseemos realizar, hemos de situarnos en el directorio donde se encuentre el fichero pom.xml.

Comandos más destacados:

- mvn clean.- Realiza una limpieza de las clases generadas hasta el momento.
- mvn compile.- Compila nuestro proyecto.
- mvn package.- Genera el empaquetado del proyecto, generalmente un jar si no se especifica otro tipo.
- mvn install.- LLeva el empaquetado (jar) a nuestro repositorio local. Queda "visible" para otros proyectos maven en nuestro ordenador.
- mvn deploy Lleva el empaquetado (jar) a nuestro servidor de librerías. Queda "visible" para otros proyectos maven en otros ordenadores. Este comando necesita que a maven se le haya indicado dónde está dicho servidor.
- mvn javadoc:javadoc Genera la documentación javadoc de nuestro proyecto.
- mvn site Genera documentación html del proyecto. Por supuesto, necesitamos haber escrito en determinados ficheros el contenido de esa documentación.

[15] https://insights.stackoverflow.com/survey/2019

Fuente:Joatham Pérez Expósito

8.1.3 Plugin

Los plugins en maven, los podríamos definir como herramientas que nos permiten realizar distintas funcionalidades, es decir, nos permiten: compilar, empaquetar, copiar librerías, desplegar en servidores embebidos, análisis de código, etc.

Es por ello que los plugins esconden uno de los grandes poderes en maven. A continuación se enumeran algunos de los más interesantes y de los que se hacen uso en el proyecto:

- maven-compiler-plugin.- Es el plugin utilizado para compilar el código fuente del proyecto. En él, entre otras cosas podemos configurar la versión de JDK con la que compilamos el proyecto.
- maven-jar-plugin.- Es el plugin que provee la capacidad de construir Jars. Entre sus características a destacar, podemos configurar el fichero Manifiest que genera el proyecto.
- maven-dependency-plugin.- Es el plugin encargado de realizar la manipulación en las jar generados.
- jacoco-maven-plugin.- Plugin que permite realizar el análisis de código desarrollado en base a los test que se hayan realizado de forma paralela al código.
- maven-jetty-plugin.- El plugin de jetty es muy utilizado para un rápido desarrollo y pruebas durante el desarrollo de aplicaciones web que requieren un contenedor web. Es posible iniciar el contenedor de jetty utilizando un goal de maven mvn jetty:run, o recargar la aplicación sin necesidad de reiniciar el contenedor.

Fuente:Joatham Pérez Expósito

8.1.4 Identificación de un proyecto Maven

En el archivo pom.xml se declaran, entre otras cosas, un identificador único de nuestro proyecto/artefacto, que resulta de la unión de tres identificadores:

- groupId: representa la organización autora/dueña del artefacto.
- artifactId: este campo define el nombre por el que se conoce al proyecto en sí mismo.
- versión: identificador X.X.X.

Este tipo de construcción hace que se unifique con la convención de nombres descrita anteriormente.

Fuente:Joatham Pérez Expósito

8.1.5 Independencia del Ide

Maven permite trabajar de forma independiente al IDE[16], con su propio modelo de proyecto, que se guarda en el archivo pom.xml (de Project Object Model)[17].

Entre las diferentes alternativas que existen debemos de escoger uno que reúna todo lo que necesita un programador en su día a día. Entre esas características estarían:

- Editor de código ligero.
- Debugger.
- Compilador.
- Completado de código.
- Extensiones.
- ....

Es por ello, que ante esta tesitura y tras haber probado otros ides como:

[16] https://visualstudiomagazine.com/articles/2019/11/21/fb-vs-code.aspx#:~:text=Facebook%20announced%20it's%20adopting%20Visual,tool%20in%20major%20development%20surveys.

[17] https://marketplace.visualstudio.com/items?itemName=vscjava.vscode-java-pack

- Netbeans[18]. Nada recomendado, y obsoleto en muchos aspectos.
- Eclipse[19]. Consumo elevado de recursos.
- Intelli[20]j. Hasta no hace mucho mi elección.

Me he decantado por VS-Code[21], siendo un Ide actual y con una interfaz agradable.

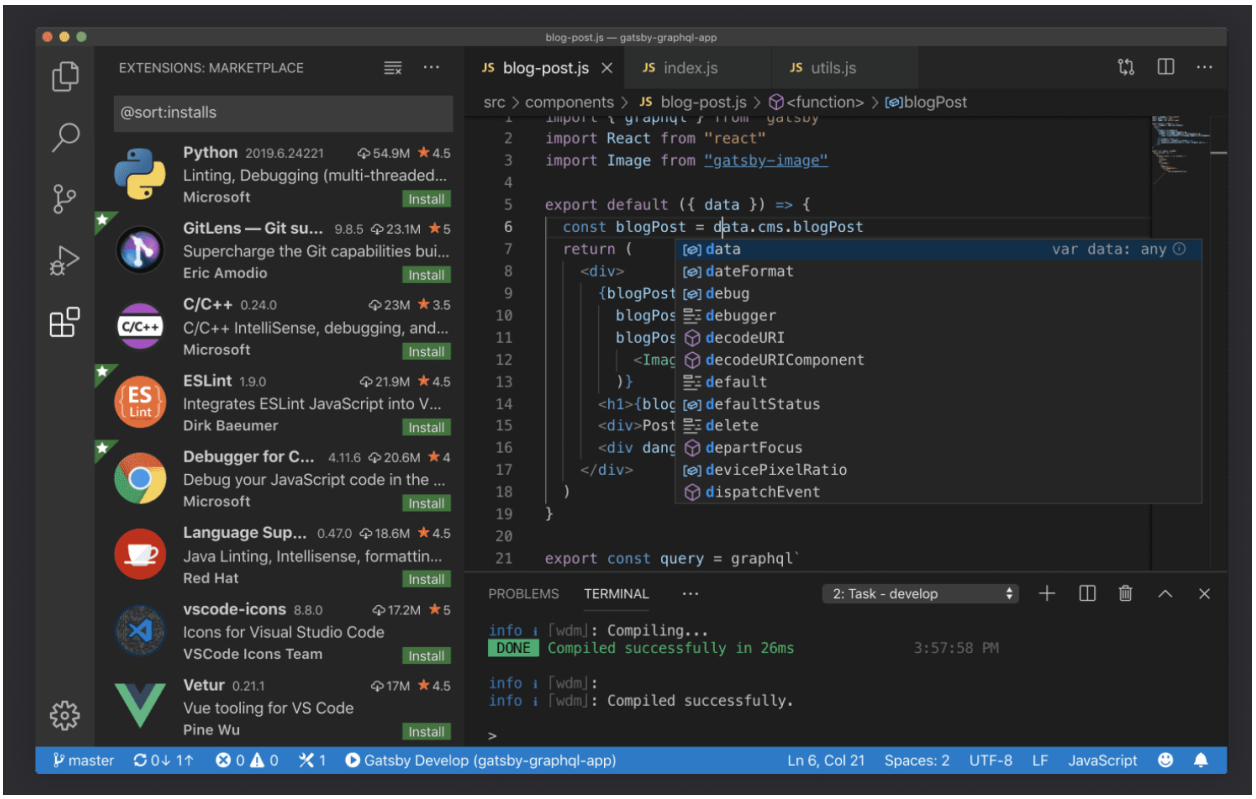


Figura 20: Visual Studio Code

Fuente: Visualstudio

Son varios los motivos de esta elección, pero uno de los principales es el escaso consumo de recursos de los que hace uso, lo cual me recuerda a Intellij Idea. Además de este motivo, se suman otros que me gustaría destacar:

[18] https://karaf.apache.org/

[19] https://docs.microsoft.com/es-es/sharepoint/dev/spfx/toolchain/implement-ci-cd-with-azure-devops

[20] https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html

[21] http://www.juntadeandalucia.es/servicios/madeja/contenido/libro-pautas/146

- Es un editor de código fuente sencillo, potente y multiplataforma.
- Permite el desarrollo en cualquier lenguaje de programación gracias a un amplio abanico de extensiones.
- En 2019 fue el IDE más popular según stackoverflow[22].
- Facebook lo adopta como IDE de desarrollo gracias a sus características[23].
- …

Extensiones

Una vez instalado Visual Studio Code todavía hemos de realizar una serie de pasos antes de comenzar a trabajar.

VS Code es un editor muy versátil, gracias a su diseño modular, podemos añadir soporte para JAVA mediante extensiones. Para facilitar más las cosas, disponemos de un "Java Extension Pack[24]", que contiene las extensiones más populares usadas por los desarrolladores JAVA:

- Language Support for Java(TM) de Red Hat
- Debugger for Java
- Java Test Runner
- Maven for Java
- Java Dependency Viewer

Fuente:Joatham Pérez Expósito

[22] https://ant.apache.org/
[23] https://maven.apache.org/guides/introduction/introduction-to-the-pom.html #What_is_a_POM
[24] https://www.redhat.com/es/topics/middleware/what-is-ide

Capítulo 9 - Entorno de despliegue y ejecución

Existen diferentes entornos de despliegue y ejecución para este tipo de arquitecturas, pero haré uso Apache Karaf[25].

Apache Karaf es un entorno de ejecución que destaca por su sencillez de uso y la flexibilidad que nos ofrece, además de permitir el uso de Felix o Eclipse Equinox como contenedores OSGi[26].

Figura 21: Apache Karaf

Fuente:karaf-apache

Karaf facilitará el despliegue de micro servicios a través de las distintas herramientas que se han ido describiendo. Además nos ayudará para la integración de soluciones de terceros para facilitar la escalabilidad, y robustez de la solución conjunta.

Los sistemas alternativos pueden ser, los siguientes:

- Spring Integration, es un framework ligero que permite construir sistemas desacoplados, pero que resulta un tanto tedioso por los múltiples ficheros de configuración que se deben de escribir.
- Mule ESB es un bus de servicio empresarial, muy robusto y funcional.
- RedHat Fuse, un producto similar basado en Apache Karaf con un amplio conjunto de características.

[25] https://code.visualstudio.com/
[26] https://www.osgi.org/

- Servicemix, el cual fue base de Redhat Fuse y que en la actualidad se encuentra descontinuado.

La elección de Karaf es el resultado del estudio de las anteriores ya que:

- Permite la integración de Spring.
- Solo se instalarán los sistemas necesarios a través de Features. No será necesario la instalación de un bus de mensajería asíncrono como Mule ESB, Redhat Fuse o Servicemix.
- Permite la construcción de sistemas robustos basados en cluster.
- Existe una comunidad amplia de ejemplos para distintos tipos de soluciones.

Además de todo lo anterior, Karaf posee características destacables para el desarrollador, que hacen de ella una herramienta sumamente interesante:

- Implementación en caliente basada en carpetas
- Shell integrada en la herramienta web. (org.Apache.karaf.Shell)
- Acceso remoto SSH a esa consola. (org.Apache.karaf.Shell.ssh)
- Sistema de registro centralizado. (org.Apache.karaf.log)
- Tiene su propia forma de aprovisionar paquetes y niveles de inicio. (org.Apache.karaf.features)
- Repositorio de maven propio.

Como se ha mencionado, el aprovisionamiento de paquetes o módulos se realizará a través de *features*. En resumen, un feature será un fichero XML, que contiene todas las referencias de las librerías que se van a desplegar dentro del contenedor. El contenido de feature.xml es el siguiente:

```xml
<?xml version="1.0" encoding="UTF-8"?>

<features name="karaf-osgi-${project.version}" xmlns="http://karaf.apache.org/xmlns/features/v1.4.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://karaf.apache.org/xmlns/features/v1.4.0">



</features>

```

Figura 22: Feature en Karaf

Fuente: Propia

Para realizar dicho aprovisionamiento, van a interactuar los siguientes elementos:

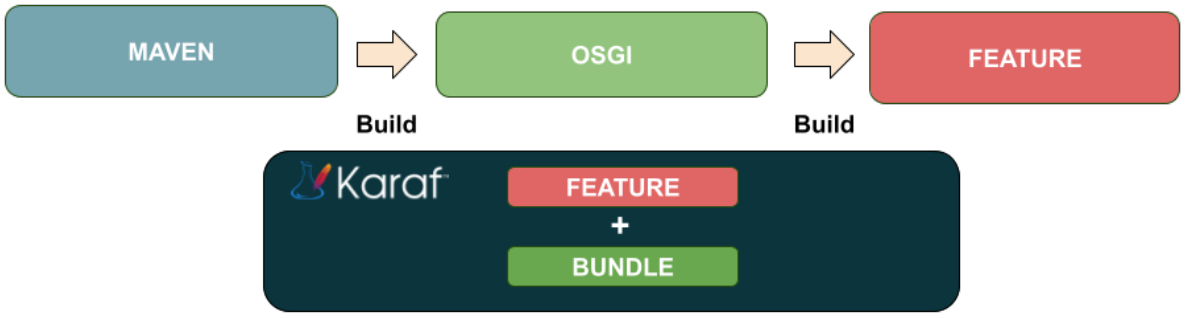


Figura 23: Ciclo desarrollo en Karaf

Fuente: Propia

- Maven: Como herramienta para la construcción de artefactos.
- OSGI: Arquitectura para la construcción de artefactos. El elemento más simple es el bundle, y sobre este tema se habló en la primera entrada dedicada a microservicios.
- FEATURE: Conjunto, o herramienta, por definirlo de alguna manera, que permitirá el aprovisionamiento en los contenedores OSGI, como KARAF. Su estructura viene definida por un xml.
- KARAF: Contenedor OSGI de bundle y features de nuestras aplicaciones/servicios así como de las librerías necesarias para el funcionamiento de estas.

Fuente:Joatham Pérez Expósito

9.1 Arquitectura

Los servicios desarrollados tendrán una arquitectura homogénea, de modo que sea sencillo identificar cada uno de sus módulos. Así mismo, todo el desarrollo estará basado las siguientes características:

- API COMÚN. Una API es un conjunto de definiciones y protocolos que se utilizan para desarrollar e integrar el software de las aplicaciones. *API significa interfaz de programación de aplicaciones*. Las API permiten que sus productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados. Las API le otorgan flexibilidad; simplifican el diseño, la administración y el uso de las aplicaciones, y proporcionan oportunidades de innovación, lo cual es ideal al momento de diseñar herramientas y productos nuevos (o de gestionar los actuales). A veces, las API se consideran como contratos, con documentación que representa un acuerdo entre las partes: si una de las partes envía una solicitud remota con cierta estructura en particular, esa misma estructura determinará cómo responderá el software de la otra parte.

Fuente:Redhat

- INTERFACE DE SERVICIO. Todo servicio que forme parte de la solución propuesta, dispondrá de una INTERFACE, que podrá ser extendida por otros servicios. Hemos de tener claro que un interfaz es *una funcionalidad abierta a la reutilización y extensibilidad*.
- IMPLEMENTACIÓN DE SERVICIO. Cada una de las INTERFACES creadas deberán ser implementadas para su posterior exposición a través de microservicios. Con esto, se realizará una reutilización y extensibilidad del código con el fin de minimizar el duplicado de código desarrollado.
- ENDPOINT SOAP. Se desarrollará un módulo que exponga el servicio bajo la API SOAP, de modo que la comunicación se realice a través de XML.

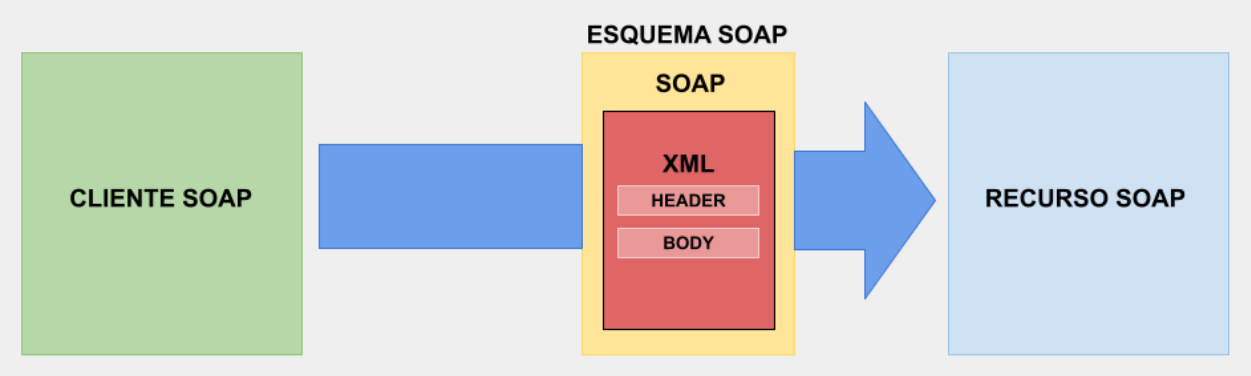


Figura 24: Esquema Soap

Fuente: Propia

- ENDPOINT REST. Se desarrollará un módulo que exponga el servicio bajo la API REST, de modo que la comunicación se realice a través de XML. *Cabe destacar que exponer la API a través de JSON sería muy sencillo*.

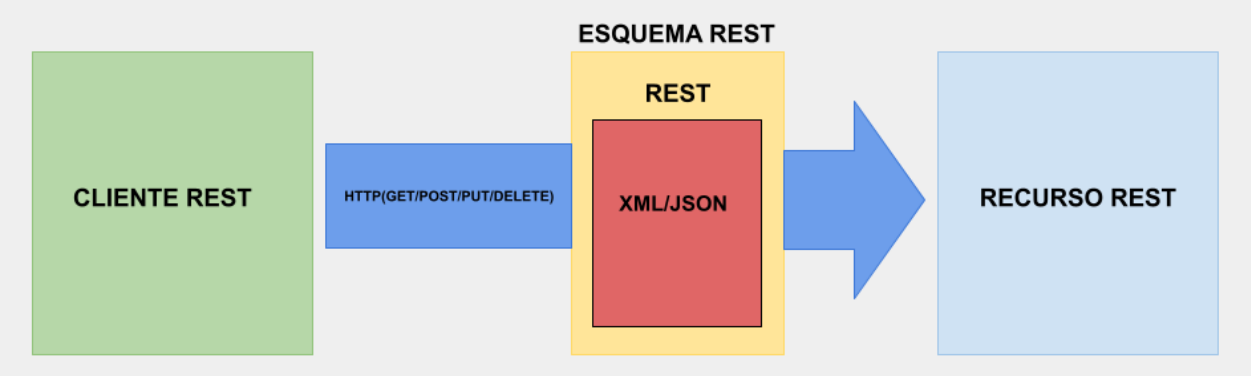


Figura 25: Esquema Rest

Fuente: Propia

- COMUNICACIÓN DE MÓDULOS. La comunicación entre módulos se realizará a través de Camel.

9.2 Ejemplo de construcción de un servicio

Supongamos que vamos a crear un servicio llamado ENIService. Nuestro servicio se compondrá de los siguientes elementos:

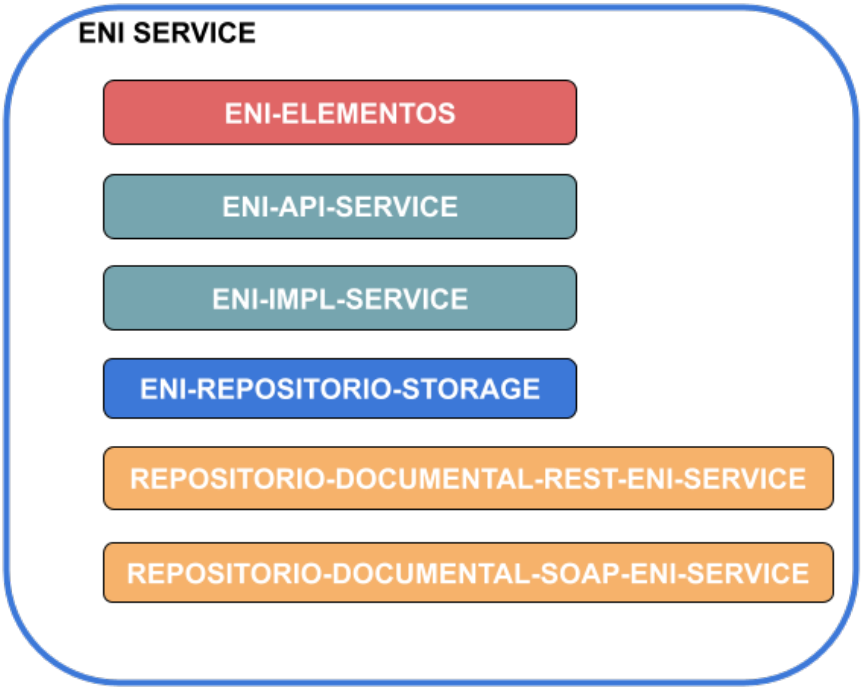


Figura 26: Módulos servicio Eni

Fuente: Propia

El servicio que desarrollaremos esta compuesto por:

- Eni-elementos: Módulo que contiene la definición de los elementos para la comunicación de los distintos elementos de los distintos servicios (documentos/expediente/firma).
- Eni-api-service: Módulo que contiene la definición de la api de los servicios documento y expediente basado en el ENI.
- Eni-impl-service: Módulo que implementa la api Eni-api-service.
- Repositorio-documental-rest-eni-service: Módulo que permite mostrar el módulo eni-impl-service a través de la api rest.
- Repositorio-documental-soap-eni-service: Módulo que permite mostrar el módulo eni-impl-service a través de la api soap.
- Eni-repositorio-storage: Módulo que permite realizar la persistencia de los documentos y expedientes a través de BBDD (H2 embebed).

Como resulta evidente, la implementación del servicio se realiza una única vez. Haciendo uso del concepto de módulos (cajas), evitamos el duplicado de código, y la

reutilización del mismo de forma correcta. La comunicación de cada uno de estos módulos se realizará a través de rutas Camel.

Existe un módulo final, denominado eni-features-service que contiene la invocación a los módulos, así como a las dependencias transitivas de estos, y que garantiza el despliegue correcto.

Esta estructura transformada en un contexto xml será como sigue:

9.2.1 Dependencias comunes de todos los servicios.

```
<feature name="eni-dependency-system" version="${project.version}">
      <feature>http</feature>
      <feature>http-whiteboard</feature>
      <feature dependency="true">web-console</feature>
      <feature>cxf-http-netty-client</feature>
      <feature>cxf-http-netty-server</feature>
      <feature>camel-cxf</feature>
   </feature>
```

Figura 27: Dependencias comunes servicio soap - rest

Fuente: Propia

El feature eni-dependency-system contiene todas las dependencias de los servicios eni (documento o expediente) necesarios para su despliegue de forma correcta, así como para la comunicación a través del Camel.

9.2.2 Dependencias comunes de todos los servicios.

```
<feature name="eni-common-dependency" version="${project.version}">
      <feature version="${project.version}">eni-dependency-system</feature>
      <bundle>mvn:org.osgi/org.osgi.core/6.0.0</bundle>
      <bundle>mvn:es.uned.tfm/eni-api-elementos/${project.version}</bundle>
      <bundle>mvn:es.uned.tfm/eni-api-service/${project.version}</bundle>
      <bundle>mvn:es.uned.tfm/eni-impl-service/${project.version}</bundle>
   </feature>
```

Figura 28: Dependencias comunes servicio Eni

Fuente: Propia

Se puede observar la existencia de una variable significativa denominada ${proyect.version} que indica la versión del proyecto, y que permitirá ir realizando las distintas versiones de nuestra solución. Además y como resulta evidente contiene los elementos comunes a todos los servicios que se desplegarán.

9.2.3 Dependencias y módulo de persistencia en BBDD.

```
<feature name="eni-storage-common" version="${project.version}">
    <feature>jdbc</feature>
    <feature>transaction</feature>
    <feature>jndi</feature>
    <feature>pax-jdbc-config</feature>
    <feature>pax-jdbc-h2</feature>
    <feature>pax-jdbc-pool-dbcp2</feature>
    <feature dependency="true">aries-blueprint</feature>
    <feature>jpa</feature>
    <feature>eclipselink</feature>
    <feature>hibernate</feature>
    <feature>openjpa</feature>
</feature>
<feature name="eni-storage" version="${project.version}">
    <feature version="${project.version}">eni-storage-common</feature>
    <bundle
dependency="true">mvn:org.mapstruct/mapstruct/${org.mapstruct.version}</bundle>
    <bundle>mvn:es.uned.tfm/eni-repositorio-storage/${project.version)</bundle>
</feature>
```

Figura 29: Feature eni-storage-common

Fuente: Propia

Podemos observar como este feature contiene todas las dependencias para el despliegue de la BBDD embebida, así como su conexión a través de jndi.

9.2.4 Servicios Eni en arquitectura SOAP/REST

Figura 30: Feature eni documental soap - rest

Fuente: Propia

9.2.5 Desplegando un Feature

Una vez construido el feature, el siguiente paso como resulta evidente es su despliegue.

En su despliegue haremos uso de karaf, concretamente la versión 4.3.0, como contenedor OSGI.

Prerrequisitos:

- OpenJdk 8.
- Maven 3.

Una vez instalado Karaf tan solo debemos de iniciarlo, para obtener la siguiente imagen:

```
jpexposito@MacBook-Pro-de-Joatham bin % ./karaf start
        __ __                  ____
       / //_/____ _____ ____ _/ __/
      / ,<  / __ `/ ___/ __ `/ /_
     / /| |/ /_/ / /  / /_/ / __/
    /_/ |_|\__,_/_/   \__,_/_/

  Apache Karaf (4.3.0)

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown Karaf.

karaf@root()> list
START LEVEL 100 , List Threshold: 50
ID | State  | Lvl | Version | Name
---|--------|-----|---------|-----------------------------------------------
31 | Active |  80 | 4.3.0   | Apache Karaf :: OSGi Services :: Event
karaf@root()>
```

Figura 31: Arranque plataforma Karaf

Fuente: Propia

Para la instalación de todo servicio tan solo hemos de lanzar el siguiente comando:

feature:repo-add mvn:es.uned.tfm/eni-features-service/LATEST/xml, que incluye el repositorio en karaf y realizar la instalación del feature:

feature:install eni-services.

Cada uno de los features realizados bajo esta arquitectura, los contenedores OSGI, KARAF en este momento, los denomina repositorios, con lo que cada uno de los microservicios que realicemos, estarán disponibles en un repositorio que debemos proveer (feature:repo-add ...).

Obtendremos tras ejecutar el comando list la siguiente salida:

```
karaf@root()> list
START LEVEL 100 , List Threshold: 50
 ID | State  | Lvl | Version          | Name
----|--------|-----|------------------|-----------------------------------------------
 31 | Active |  80 | 4.3.0            | Apache Karaf :: OSGi Services :: Event
 55 | Active |  80 | 1.9.3.1          | Apache ServiceMix :: Bundles :: jasypt
 56 | Active |  80 | 1.4.4            | OPS4J Pax JDBC Generic Driver Extender
 57 | Active |  80 | 1.4.4            | OPS4J Pax JDBC Config
 58 | Active |  80 | 1.4.4            | OPS4J Pax JDBC Pooling Support Base
 59 | Active |  80 | 1.0.0.201505202023 | org.osgi:org.osgi.service.jdbc
 60 | Active |  80 | 1.5.1            | ClassMate
 68 | Active |  50 | 2.9.2            | com.github.ben-manes.caffeine
 69 | Active |  80 | 1.4.199          | H2 Database Engine
 72 | Active |  80 | 1.0.0            | ENI-ELEMENTOS
 73 | Active |  80 | 1.0.0            | ENI-API-SERVICE
 74 | Active |  80 | 1.0.0            | ENI-IMPL-SERVICE
 75 | Active |  80 | 1.0.0            | ENI-REPOSITORIO-STORAGE
 76 | Active |  80 | 1.0.0            | REPOSITORIO-DOCUMENTAL-REST-ENI-SERVICE
 77 | Active |  80 | 1.0.0            | REPOSITORIO-DOCUMENTAL-SOAP-ENI-SERVICE
```

Figura 32: Despliegue de los servicios eni

Fuente: Propia

Como vemos los distintos bundle implicados se encuentran activos. Esto nos indica que el despliegue se ha desarrollado de forma correcta. Vamos a consultar, visualmente que todo el proceso ha funcionado de forma correcta a través de la url http://localhost:8181/cxf, por defecto, la url de KARAF donde se expone los servicios SOAP/REST.

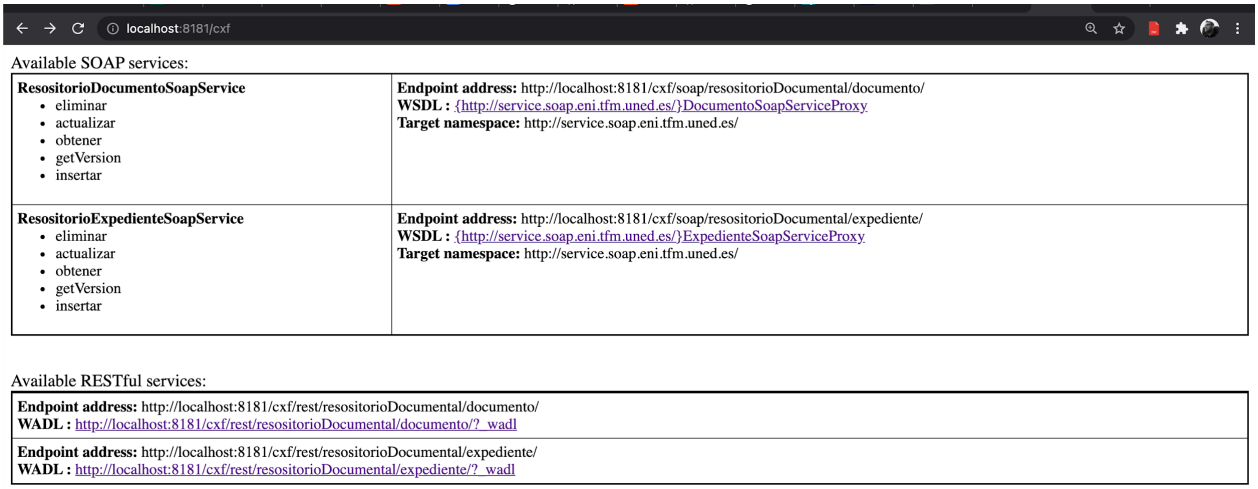


Figura 33: Visualización de los servicios eni soap - rest

Fuente: Propia

La construcción y el despliegue de aplicaciones/servicios en OSGI, resulta óptima, gracias al versionado semántico, y la gestión de dependencias en la arquitectura.

9.2.6 Comunicación de módulos.

Como se ha indicado la comunicación se realizará a través de Camel. Para realizarlo en primer lugar se debe de crear el conjunto de las rutas que contendrá cada módulo, para posteriormente realizar su declaración y por último la comunicación entre cada uno de ellos.

9.2.6.1 Definición de las rutas Camel.

Los procesos y los componentes se conectan en el módulo de integración y permite realizar el ruteo. En estas conexiones se puede filtrar mensajes basándonos en criterios definidos por el usuario. En mi caso me basaré en la operación que deseo realizar, aunque puede ser cualquier otra. Existen varias opciones para escribir estas reglas. Se puede usar Java, Scala, Groovy o incluso XML. Un ejemplo de la definición a través de Java será el siguiente:

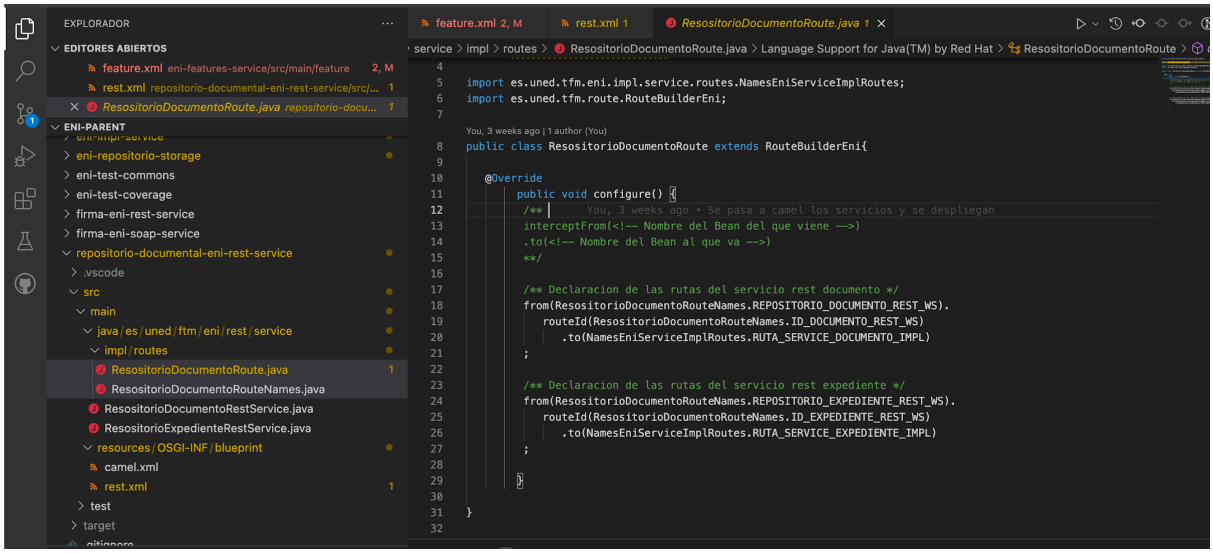


Figura 34: Definición de route camel

Fuente: Propia

9.2.6.2 Exposición de las rutas Camel.

La exposición de cada una de las rutas creadas se realizará a través de ficheros de configuración blueprint. De este modo y como se muestra sólo debemos de indicar el paquete donde se encuentran las creadas.

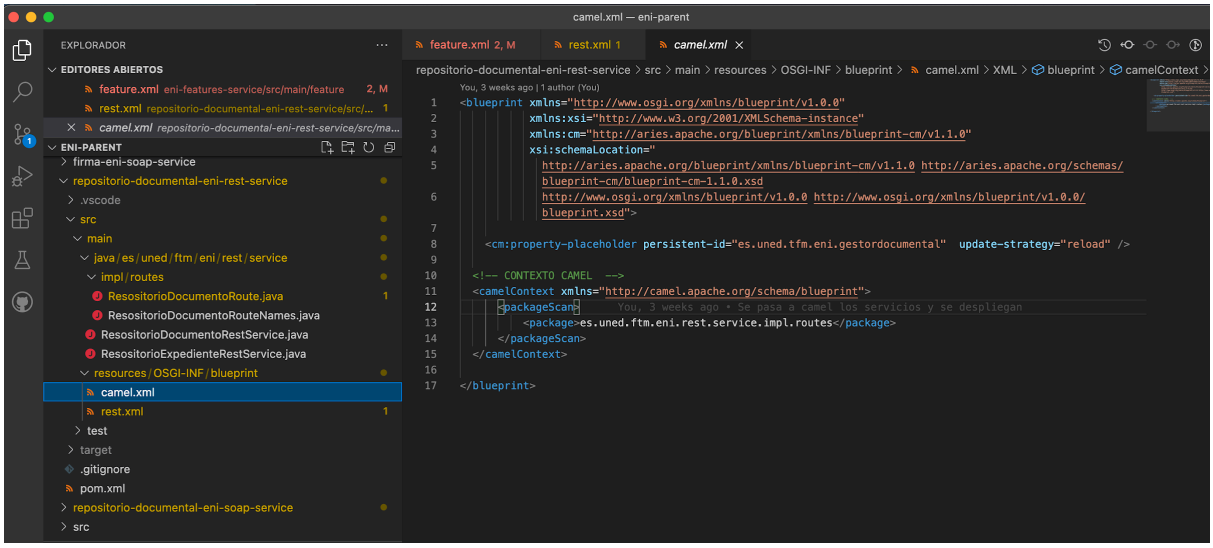


Figura 35: Exposición paquege route camel

Fuente: Propia

9.2.6.3 Despliegue de las rutas Camel en el contenedor Karaf.

El despliegue se realiza durante la instalación del módulo en el contenedor Karaf, gracias a la correcta definición como se ha indicado anteriormente. La demostración del despliegue correcto de las rutas y la disposición de estas podemos verlo en la siguiente imagen:

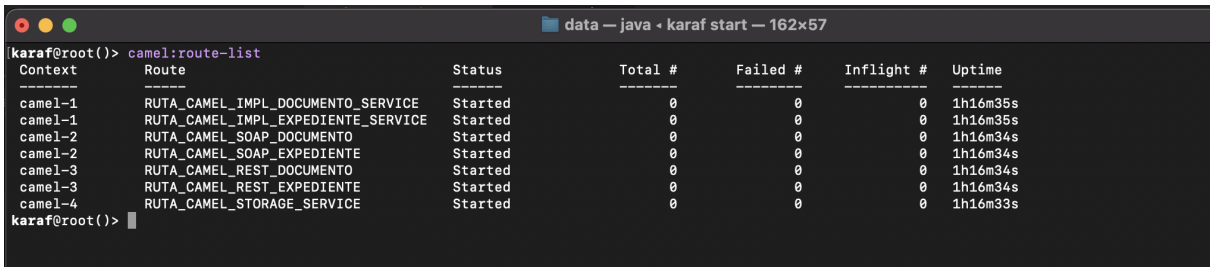


Figura 36: Listado de route camel en karaf

Fuente: Propia

Podemos observar como están arrancadas las distintas rutas que componen la solución de comunicación de la solución.

Visualmente lo explicado supone la comunicación de los distinto módulos que componen los servicios como se muestra:

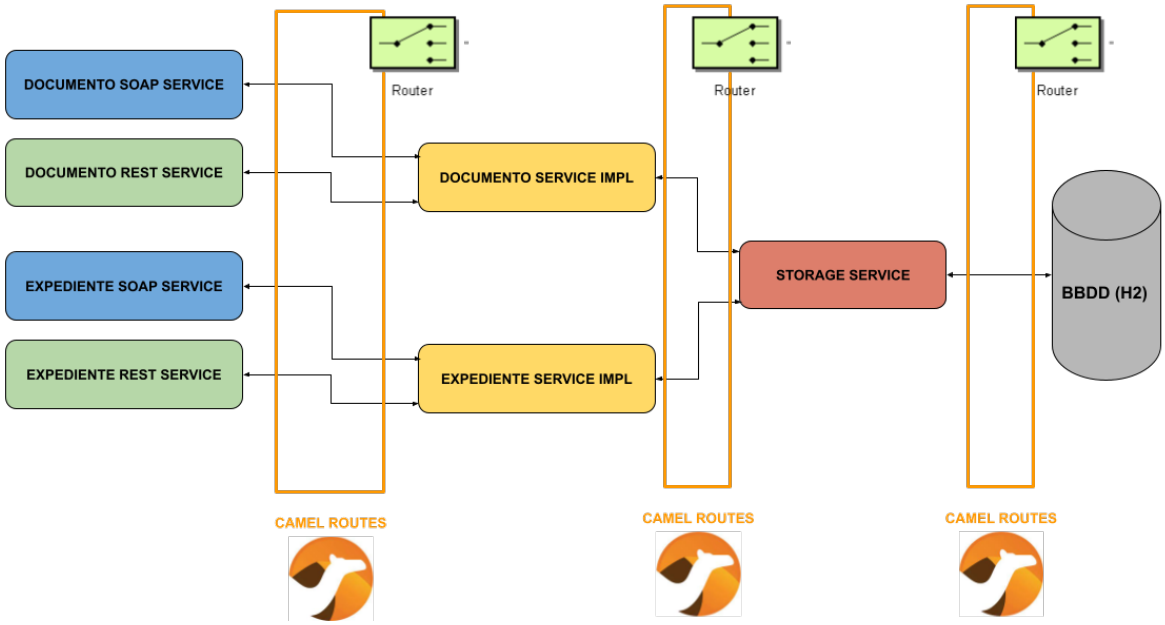
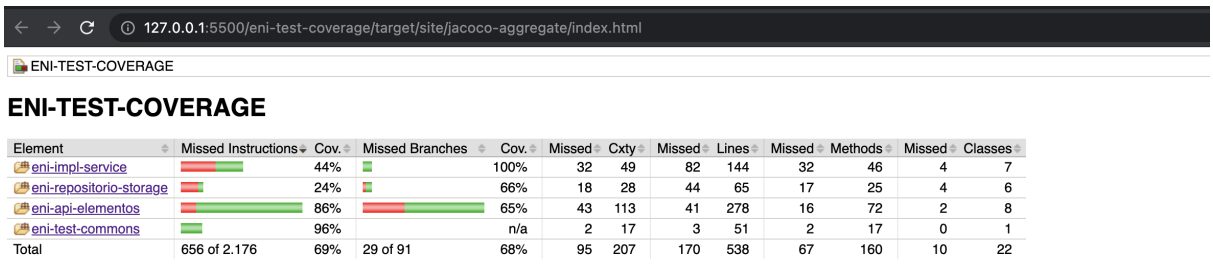


Figura 37: Comunicación de las distintas de route camel de servicio

Fuente: Propia

9.3 Calidad del código

El desarrollo de la solución se ha basado en el desarrollo bajo test unitarios, integrando la herramienta jacoco. El análisis de cobertura de código se realiza de todo el proyecto, con el fin de llegar a una cobertura total del proyecto por encima del 80%.



Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
eni-impl-service		44%		100%	32	49	82	144	32	46	4	7
eni-repositorio-storage		24%		66%	18	28	44	65	17	25	4	6
eni-api-elementos		86%		65%	43	113	41	278	16	72	2	8
eni-test-commons		96%		n/a	2	17	3	51	2	17	0	1
Total	656 of 2.176	69%	29 of 91	68%	95	207	170	538	67	160	10	22

Figura 38: Cobertura de código de los distintos módulos del servicio

Fuente: Propia

Fuente:Joatham Pérez Expósito

Capítulo 10 - Conclusiones y trabajo futuro

En el este capítulo se presentan las principales conclusiones extraídas del análisis y construcción de servicios basado en tecnologías ágiles para el desarrollo y despliegue de servicios que se han utilizado. Además se expondrán los trabajos futuros a realizar.

10.1 Conclusiones del trabajo

Las principales conclusiones relacionadas al trabajo realizado son:

- La administración electrónica estatal, carece de un estándar y frameworks para la construcción de servicios y el desarrollo de soluciones.
- Se debe tender a desarrollo de soluciones ágiles que permita el despliegue rápido de correcciones. Para ello, se debe de hacer uso del desarrollo basado en microservicios.
- Las soluciones disponibles en el Pae carecen de una doble API de integración, que permita una elección u otra en función de sus necesidades.
- En la actualidad la exposición de los servicios de la administración se mantienen bajo servicios SOAP, cuando la industria desde hace ya mucho tiempo tiende a servicios REST.
- Existe una tendencia hacia las soluciones monolíticas frente a los microservicios debido al desconocimiento de la administración y la ventaja de esta arquitectura a medio-largo plazo..
- Se ha propuesto una solución basada en microservicios, basada en estándares y frameworks actuales, a través de una herramienta que permite el despliegue automatizado, como base para la construcción de una solución más extensa.

- Se ha fraccionado la soluciones en distintos módulos que permite que un cambio no afecte a otros, que permite la minimización de los cambios en otros módulos.

  Con estos resultados obtenidos, se pueden dar por cumplidos los objetivos propuestos para este trabajo de investigación.

10.2 Líneas futuras de investigación

Las principales futuras líneas de investigación aplicadas a este trabajo son:

- Análisis e integración de:
  - Nuevos servicios básicos en la administración electrónica, principalmente el servicio de firma. Se plantea el desarrollo e integración en la plataforma propuesta de servicios de Firma Electrónica. Esto está motivado por la estrecha relación que existe entre los documentos/expedientes con la firma electrónica.
  - De la plataforma propuesta y desarrollada en Docker. Se debe de extender el desarrollo para el despliegue en docker y favorecer el crecimiento horizontal de la solución, fomentando el despliegue automático en función del consumo de servicios.
  - De nuevos sistemas de almacenamiento de documentos/expedientes, a través de módulos que permita extender la plataforma. Se crearán nuevos módulos (bundles) que permitan extender o cambiar el sistema de almacenamiento de los documentos/expedientes de la plataforma. Un ejemplo de ello sería la creación de un módulo para el almacenamiento en un repositorio

documental como Alfresco[27]. Este repositorio documental como otros disponen de una API de integración a través de REST.

- En una plataforma de integración continua, dando solución a bugs de forma rápida y eficaz. Para ello se hará uso de software libre como Jenkins[28] o Sonarqube[29]. Ambas herramientas se interrelacionan a través de scripts, jaccoco y test unitarios.

[27] https://www.alfresco.com/es/
[28] https://www.jenkins.io/
[29] https://www.sonarqube.org/

BIBLIOGRAFÍA

[ERL1] http://www.gobiernodecanarias.org/platino/

[ERL2] https://www.boe.es/eli/es/l/2015/10/01/40/con

[ERL3] https://www.boe.es/buscar/act.php?id=BOE-A-2007-12352

[ERL4] https://www.boe.es/buscar/act.php?id=BOE-A-2010-1331

[ERL5] https://www.boe.es/eli/es/l/2015/10/01/40/con

[ERL6] https://administracionelectronica.gob.es/pae_Home/

[ERL7] https://www.w3.org/XML/

[ERL8] https://www.json.org/json-en.html

[ERL9] https://www.osgi.org/

[ERL10] https://camel.apache.org/

[ERL11] https://maven.apache.org/

[ERL12] Maven, The Definitive Guide

[ERL13] https://insights.stackoverflow.com/survey/2019

[ERL14] https://www.redhat.com/es/topics/cloud-native-apps/what-is-service-oriented-architecture