UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Programa de Apoyo al Egreso de Profesionales en Actividad

TÍTULO: Buenas prácticas en la Automatización de pagos electrónicos mediante la API de Red Link

AUTOR: Leonardo José Rey

DIRECTOR ACADÉMICO: Ariel Pasini

DIRECTOR PROFESIONAL: Javier Petruccelli

CARRERA: Licenciatura en Sistemas

Resumen

La Tesorería General de La Nación (TGN) administra, a través del sistema e-SIDIF, las órdenes de pago de diferentes agencias gubernamentales, entre las que se encuentran los pagos correspondientes a aportes, contribuciones y retenciones a la AFIP que se pagan por VEP. Este proceso consumía muchos recursos y dependía de la intervención humana para ser realizado. A través de nuevas tecnologías de despliegue y la interacción mediate API con Red Link se pudo automatizar este proceso de pago.

Palabras Clave

e-SIDIF, AFIP-VEP, microservicios, OKD, API- Red Link

Conclusiones

Este nuevo desarrollo ayuda a automatizar un proceso lento y tedioso que involucraba diferentes acciones del usuario en diferentes momentos del día, además de permitirnos confiar en la tecnología de microservicios para acotar aplicaciones monolíticas y de gran escala como e-SIDIF, rediseñando el circuito de pago. Haciendo uso de buenas prácticas tales como salidas controladas a producción o la integración y entrega continua.

Trabajos Realizados

Se modificó el circuito de los pagos a AFIP, automatizando el mismo a través de la interacción con la API pública de Red Link, Desplegando los módulos que interactúan entre éste y el sistema e-SIDIF utilizando microservicios y comunicación asincrónica a través de cola de mensajes, desacoplando e-SIDIF de la comunicación con la API de pago LINK.

Trabajos Futuros

- *Incorporar un nuevo tipo de pago a través de Transferencias Inmediatas del BNA.*
- *Analizar los pagos, que antes se realizaban a través del circuito de Archivos de Lotes Red CUT usando el SNP, para que ahora se realicen de manera inmediata a través de la interacción del e-SIDIF con la API de BNA.*
- *Investigar si es posible replicar la mecánica de solución para la comunicación con Red Link, reutilizando la computación en la nube y las buenas prácticas aplicadas, como las integraciones y entregas continuas y la salida a producción controlada con features flags.*

Fecha de la presentación: 09/2022

UNIVERSIDAD NACIONAL DE LA PLATA

Facultad de Informática

Buenas prácticas en la automatización de pagos electrónicos mediante la API de Red Link

Alumno: Leonardo José Rey

Director: Ariel Pasini

Agradecimientos

Principalmente a mis padres, que hicieron posible que viniera a estudiar a La Plata y siempre estuvieron para guiarme en este proceso.

A mi director de Tesina, Ariel Pasini, por su dedicación e interés para resolver las inquietudes que surgieron a lo largo del desarrollo.

A mi coordinador en el grupo de trabajo de Pagos, Ezequiel Primón, que me ayudó con sus conocimientos tecnológicos.

Al taller de escritura de Tesina, que me permitió descubrir nuevas y adecuadas forma de redacción.

A Guada y Leti, que me ayudaron con su experiencia en el comienzo de este trabajo.

Al LIFIA, que no sólo me permitió comenzar mi actividad laboral en el mundo informático, sino que también me dio las herramientas para que pudiera terminar la carrera.

A la Facultad de Informática por darme la oportunidad de cumplir una asignatura pendiente.

Contenido

Índice de figuras

Índice de tablas

Índice de acrónimos

AFIP: Administración Federal de Ingresos Públicos

AMQP: Advanced Message Queuing Protocol

API: Application Programming Interface

APN: Administración Pública Nacional

BCRA: Banco Central de la República Argentina

BNA: Banco de la Nación Argentina

CDR: Comprobante de Devengado de Retención

CGN: Contaduría General de la Nación

CUT: Cuenta Única del Tesoro

e-SIDIF: Sistema Integrado De Información Financiera Internet

FIFO: First In, First Out

JSON: JavaScript Object Notation

LAF: Ley de Administración Financiera

MAP: Medidas de Afectación Patrimonial

OKD: Optimized Kubernetes Distribution

ONP: Oficina Nacional de Presupuesto

OP: Orden de Pago

PE: Pago Electrónico

Red CUT: Medio de pago emitido a través de la Cuenta Única del Tesoro

SAF: Servicio de Administración Federal

SLU: Sidif Local Unificado

SNP: Sistema Nacional de Pagos

SOAP: Simple Object Access Protocol

SNP: Sector Público Nacional

TGN: Tesorería General de la Nación

Capítulo 1 - Introducción

Objetivo

El objetivo de esta Tesina es automatizar el pago electrónico entre la Tesorería General de la Nación [1] y la Administración Federal de Ingresos Públicos [2] mediante el uso servicios prestados por la interfaz de programación de aplicaciones [3] de Red Link.

Motivación

La TGN administra las órdenes de pago de diferentes agencias gubernamentales, entre las que se encuentra los pagos correspondientes a aportes, contribuciones y retenciones a la AFIP.

El pago de retenciones como de aportes y contribuciones que tienen como beneficiario a la AFIP se realiza con una transferencia cuyo destino es una cuenta bancaria de la TGN que actúa como una cuenta temporal hasta que se finaliza con el circuito de pago. Cada organismo gubernamental, como parte del circuito, ingresa periódicamente a una página web del Banco de la Nación Argentina para realizar los pasos de carga y firma de los pagos a realizar a través de Volantes Electrónicos de Pago [4]. Finalmente, una vez que el organismo cuenta con la información del pago confirmado, una persona se encarga de mandar un mail al Banco de la Nación Argentina [5], quien culmina la operación debitando de la cuenta de TGN y dando por pagado cada VEP. Gran parte de este circuito requiere la intervención manual de los organismos y del BNA, lo cual hace que en toda la operación participen varios actores e insuman un tiempo considerable y, además, que sea un proceso cuestionado en las auditorías.

Considerando que a lo largo del mes puede haber picos de 500 pagos por día a favor de la AFIP, se presenta necesario generar un procedimiento automatizado, apoyado en buenas prácticas, para facilitar la tarea. Dichas buenas prácticas incluyen las que veníamos aplicando durante todos estos años de aplicación de metodologías ágiles, como las nuevas que surgieron al desplegar aplicaciones basadas en arquitectura de microservicios.

Desarrollo propuesto

- Analizar el circuito de pago de la TGN a la AFIP
- Definir un circuito de pago de la TGN a la AFIP utilizando un nuevo modelado de datos.

[1] Desde ahora en adelante utilizaremos el término TGN para referirnos.
[2] Desde ahora en adelante utilizaremos el término AFIP para referirnos.
[3] Desde ahora en adelante utilizaremos el término API para referirnos.
[4] Desde ahora en adelante utilizaremos el término VEP para referirnos.
[5] Desde ahora en adelante utilizaremos el término BNA para referirnos.

- Relevar y analizar los servicios otorgados por la API Red Link.
- Analizar el servicio de pagos y consultas de VEP mediante la API Red Link
- Desarrollar el módulo de comunicación entre el Sistema Integrado De Información Financiera Internet [6] y el servicio de consulta y pago de VEP de la API de Red Link.
- Relevar y analizar el protocolo de estándar abierto "Advanced Message Queuing Protocol" [7].
- Relevar y analizar la implementación del gestor de colas de mensajes ***RabbitMQ*** (RAbbitMQ, s.f.).
- Implementar en el módulo de interoperabilidad con Red Link el uso de cola de mensajes de **RabbitMQ**.
- Relevar y analizar la arquitectura de microservicios.
- Relevar y analizar la gestión de despliegue de microservicios de **Openshift** (Red Hat, s.f.).
- Aplicar **OpenShift** en la interoperabilidad con la API Red Link
- Relevar y analizar la integración y entrega continua que **OpenShift** nos permite aplicar.
- Relevar y analizar **API REST** (Interface de acceso a servicios)
- Aplicar **API REST**, tanto para la comunicación entre los distintos servicios del módulo de interoperabilidad, como para la comunicación de este último con el sistema e-SIDIF y con Red Link.
- Relevar y analizar el generador de clientes de servicios REST de forma declarativa llamado **OpenFeign** (Spring OpenFeign, s.f.).
- Aplicar **OpenFeign** tanto para "encapsular" el cliente e-SIDIF y el cliente de Red Link y poder invocar sus API.

Resultado esperado

Automatizar el circuito de pagos de la TGN a favor de la AFIP agilizando el proceso, minimizando las demoras por la intervención humana, llegando así a una solución más eficiente.

Contexto.

La presente tesina se presenta en el contexto del Programa de Apoyo al Egreso de Profesionales en Actividad (PAEPA).

Desde el el 20 de julio del año 2005, a través del Laboratorio de Investigación y Formación en Informática Avanzada - LIFIA, ingresé a trabajar en la Dirección General de Sistemas de Administración Financiera - DGSIAF, en la Secretaría de Hacienda con el objetivo de desarrollar y mantener el e-SIDIF. A lo largo de los años fui desempeñando tareas de diseño y desarrollo. Desde el 2020 integro el equipo de interoperabilidad con sistemas externos, como es el de AFIP y actualmente BNA.

[6] Desde ahora en adelante utilizaremos el término e-SIDIF para referirnos.
[7] Desde ahora en adelante utilizaremos el término AMQP para referirnos.

Estructura del documento

En el Capítulo 2 se presentan los conceptos generales relacionados con la evolución del sistema e-SIDIF; se detalla el Sistema de Tesorería, los pagos realizados a través de la Cuenta Única del Tesoro y la utilización de buenas prácticas en el desarrollo y despliegue del e-SIDIF. Luego, en el siguiente capítulo se describirá, de forma minuciosa, los circuitos de órdenes de pago, de las retenciones generadas por el pago de las primeras y los medios de pagos involucrados, específicamente aquellos que utilizan volantes electrónicos de pago destinados a la AFIP, y que se pagaban de forma manual. En el cuarto capítulo se describe un nuevo modelo de transferencia de pagos a la AFIP, detallando su diseño dentro del e-SIDIF para gestionar los pagos realizados a través de la nueva interacción con Red Link. En el penúltimo capítulo se describen todas las tecnologías necesarias para la comunicación e interacción con Red Link y que servirá para desarrollar la solución planteada en el capítulo precedente.

Y por último, en el capítulo 6, se verán las conclusiones y los trabajos futuros.

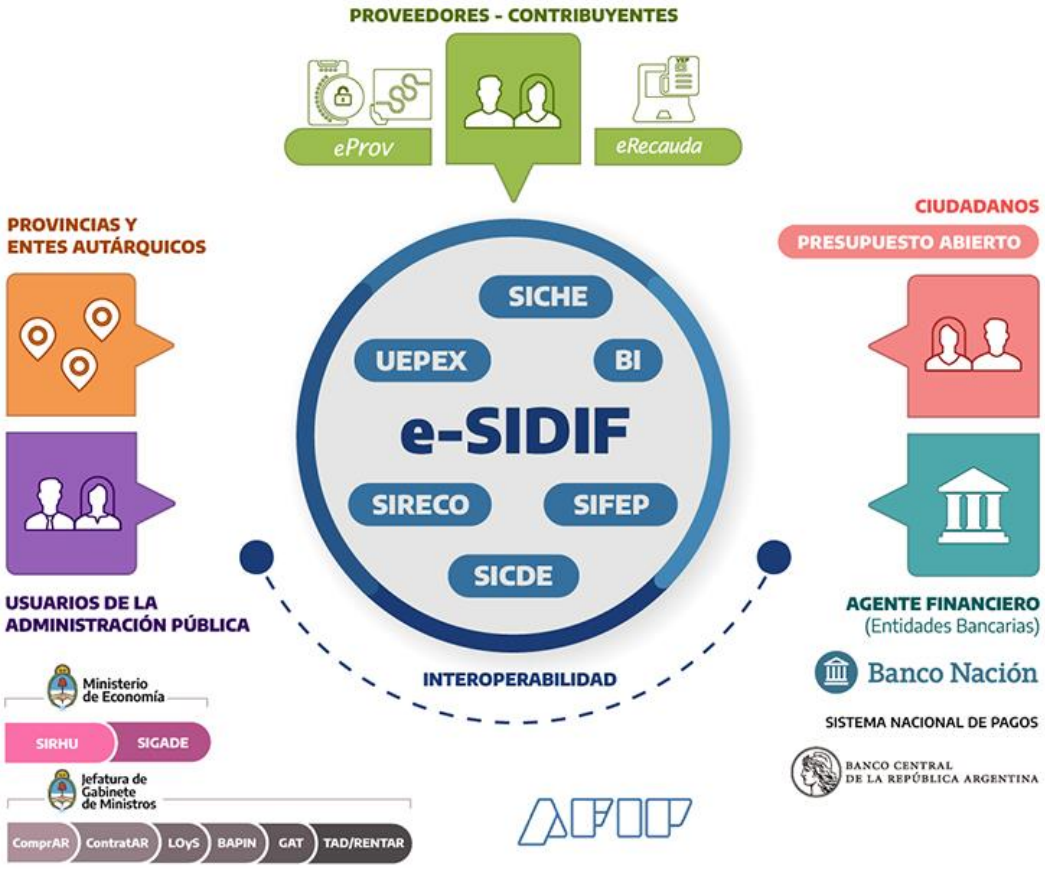
Capítulo 2 – Conceptos generales

Introducción

En este capítulo se describen los conceptos generales del Sistema Integrado de Información Financiera y cómo fue evolucionando en el tiempo, de acuerdo a la Ley de Administración Financiera [8], hasta lo que es hoy. También se detallan los distintos equipos que participan en su desarrollo y mantenimiento. Luego se presentará el Sistema de Tesorería y su estructuración a partir de la TGN; se explicará el funcionamiento de la cuenta bancaria a través de la cual, tanto la TGN como las tesorerías regionales realizan sus pagos.

Por último, se detallan las buenas prácticas llevadas a cabo por los equipos involucrados en el desarrollo del sistema, como también las herramientas brindadas por el equipo de Ingeniería de Software para alivianar la tarea.

e-SIDIF



[8] Desde ahora en adelante utilizaremos el término LAF para referirnos.

e-SIDIF es un proyecto del Ministerio de Economía de la Nación, que se encarga de la formulación del presupuesto nacional y registro de la ejecución presupuestaria. Surgió en 2004 como una evolución de sistemas anteriores, los cuales serán descriptos en forma general más adelante. (Hacienda, DGSIAF, 2021)

Comprende las áreas de compras, gastos, pagos, recursos, contabilidad general, conciliación bancaria, fondos rotatorios, pasajes y viáticos, entes, cuenta única del tesoro. Cada uno de estos grupos permite la gestión completa y auditada del presupuesto nacional y su ejecución. También cuenta con un inicio de ejercicio presupuestario y firma digital.

Características del e-SIDIF

El e-SIDIF se basa en las siguientes características generales:

- Base de Datos Única.
- Flexibilidad operativa (configuración) para garantizar la autonomía del usuario en la administración del Sistema.
- Posibilidad de centralizar/descentralizar las funciones locales y centrales del sistema a decisión de los Órganos Rectores.
- Registro de operaciones en moneda extranjera.
- Toda ejecución debe tener algún documento de base, que origine y respalde el registro.
- Confiabilidad, que se traduce en robustez de la interfaz; la posibilidad de cancelar una operación en caso de error.
- Posibilidad de exportar/importar información.
- El sistema proporciona el registro de pistas de auditoria de todas las transacciones, las que permitirá a los órganos de control contar con herramientas efectivas de seguimiento y verificación.
- El sistema incluye un módulo de seguridad para garantizar la confidencialidad, integridad y disponibilidad de la información, protegiendo todo el sistema de posibles ataques internos y externos. Este módulo le permite administrar los niveles de acceso según las vistas definidas (local y central) y los roles y permisos asignados.
- Dependiendo de las distintas funciones definidas centralmente en el e-SIDIF, algunas son administradas o utilizadas exclusivamente por los Órganos Rectores: La Oficina Nacional de Presupuesto [9], La Contaduría General de la Nación [10] y la TGN.
- El Sistema maneja plazos o “Fechas Tope”; limitando el plazo en el que se pueden registrar las transacciones. Es administrada por la ONP para la revisión del presupuesto y la planificación de cuotas; y al mismo tiempo, por la CGN para los comprobantes de Ejecución, en función a la solicitud del Organismo. Esta función le permite limitar las fechas en las que se aprueba registrar información presupuestaria/financiera contable de meses anteriores, diferenciándose la fecha de entrada de la transacción y la fecha de impacto presupuestario-contable.

[9] Desde ahora en adelante utilizaremos el término ONP para referirnos.
[10] Desde ahora en adelante utilizaremos el término CGN para referirnos.

- El sistema permite el registro de documentos liquidables o a pagar, como facturas de proveedores, y otros documentos administrativos.
- A través del proceso de Liquidación del Gasto, se podrán agrupar varias facturas u otros documentos de devengado y de esa forma generar las Órdenes de Pago.
- El sistema permite realizar una desafectación de una Orden de Pago con el ajuste correspondiente a las facturas u otros comprobantes liquidables, según corresponda.
- El sistema permite realizar regularizaciones y modificaciones a los registros originales.

Historia

Antes de la creación de la Dirección General de Sistemas de Administración Financiera [11], en la Secretaría de Hacienda, los sistemas se desarrollaban en diferentes entornos, con distintos alcances y niveles de seguridad, sin documentación que acompañara a las distintas aplicaciones y sin estrategias informáticas y plan de sistemas; además de todo esto, éstas no se adecuaban a la Ley de Administración Financiera 24.156. (Ley 24.156, s.f.)

Con el establecimiento de la DGSIAF, en el año 1991 (ver FIGURA 1), estas aplicaciones podrían recibir una mejor canalización para mejorarlas y crear otras nuevas. (Hacienda, DGSIAF, 2021)

Figura 1 Evolución de LAF - extraída de (Specogna, s.f.)

Entre 1991 y 1995 se actualizó el Sector Público Nacional [12], a raíz de lo cual se creó el Sistema Integrado de Información Financiera (SIDIF), cuyo objeto fue la elaboración del presupuesto del Estado y la contabilidad de la ejecución presupuestaria.

El SIDIF se concibió inicialmente como un sistema integrado que consta de varios subsistemas y módulos dentro de un enfoque funcional, con una base de datos central (SIDIF Local AC [13]), vinculada con bases de datos locales de los Servicios de Administración Financiera [14] (SIDIF Local OD [15]).

[11] Desde ahora en adelante utilizaremos el término DGSIAF para referirnos.
[12] Desde ahora en adelante utilizaremos el término SPN para referirnos.
[13] AC: Administración Central
[14] Desde ahora en adelante utilizaremos el término SAF para referirnos.
[15] OD: Organismo Descentralizado

Sujeto a este orden, la Secretaría de Hacienda se conectaba con los sistemas locales y el resto de las aplicaciones a través del SIDIF Central. Las bases de datos locales guardaban la información de gestión y la base central agrupaba el registro de la ejecución presupuestaria.

En esta estructura coexistían varios sistemas locales en diferentes organismos cuyas características eran disímiles; esto aumentaba los costos de mantenimiento y el retardo que implicaba replicar los ajustes que eran necesarios para los sistemas locales.

De 1995 a 1999, la Reforma de Administración Financiera entró en la fase de consolidación.

Por las razones mencionadas anteriormente, en 1999 se inició el desarrollo de una nueva aplicación con el objetivo de combinar los sistemas locales en un solo sistema. En la misma se han actualizado y ampliado los requerimientos funcionales. Este nuevo sistema se llamó Sidif Local Unificado [16]. Este sistema trajo consigo una importante y gran mejora en la gestión; proporcionando información confiable y ajustando distintos procesos como el de compras, tesorería, presupuesto y contabilidad. (Hacienda, Trayectoria de la DGSIAF, s.f.)

Los aspectos más de esta fase incluyen:

- Gestión de transacciones con impacto económico y financiero.
- Modernización y ampliación de requerimientos funcionales.
- Renovación de equipamiento que debía soportar el crecimiento de usuarios y operaciones.
- Un equipo multidisciplinario comprometido a trabajar de cerca con los SAF. para relevar las particularidades de cada organismo.
- Capacitación de usuarios.
- Puesta en marcha del sistema.

En octubre de 2002 la Secretaría de Hacienda comenzó a replicar masivamente el SLU en organismos de la Administración para reemplazar los sistemas existentes y agilizar su mantenimiento. A finales del 2005 el SLU era usado por 56 organismos.

En 2004, tras la implantación de importantes reformas a partir del SLU, incorporando mejoras significativas en la gestión de la administración financiera pública nacional, basadas en el avance tecnológico y con la aparición de nuevos requerimientos funcionales, se da comienzo a un nuevo desafío que se denominó e-SIDIF. (Hacienda, Trayectoria de la DGSIAF, s.f.)

El sistema propuesto se concibió como un producto adaptable a diferentes contextos y tamaños de organizaciones, con características visuales y de navegación que facilitan su uso, y pretende mejorar las características del SIDIF incorporando nuevas tecnologías e incluyendo nuevos requerimientos y, además, incorporando transformaciones que favorecen una mayor gestión pública orientada a resultados ya que promueve vinculaciones con otros sistemas, sean internos a la DGSIAF u otros organismos del Estado Nacional, a través del uso de servicios web.

e-SIDIF cuenta con una base de datos central que contiene toda la información de gestión financiera tanto de los SAF u órganos ejecutores como de los órganos rectores, con una

[16] Desde ahora en adelante utilizaremos el término SLU para referirnos.

arquitectura en capas que facilita y reduce los costos de mantenimiento y desarrollo del sistema.

Ya en el último período presentado en la línea del tiempo, en 2009, una vez avanzado significativamente en el desarrollo, despliegue y madurez de e-SIDIF, la Subsecretaría de Presupuesto impulsó la incorporación de soluciones modernas de inteligencia de negocios (BI).

Posteriormente, en 2011, se inició una implementación gradual de firma digital. Este avance profundizó el proceso de "despapelización" de la Administración Pública Nacional [17] y la descentralización de la carga de información. (Hacienda, Trayectoria de la DGSIAF, s.f.)

Con el fin de federalizar el uso de la aplicación y dar respuesta a las solicitudes y necesidades de otros organismos y administraciones, se comenzó a adaptar e implementar el sistema e-SIDIF para su uso en la gestión de provincias y entes autárquicos. La aplicación se adapta a los requisitos nacionales y a los de cada provincia u organismo en la que se implemente.

A lo largo de los años se han ido desarrollando nuevos sistemas relacionados con el e-SIDIF para diferentes usuarios (empresas públicas, ciudadanos, etc.) a medida que surgían nuevas necesidades. Al mismo tiempo, se desarrollaron sistemas para usuarios internos de la ANP, que ayudan a la consulta, seguimiento y gestión de información.

Equipos integrantes de la DGSIAF

Para poder llevar a cabo este proyecto llamado e-SIDIF fue necesaria la creación y coordinación de distintos grupos de trabajo con funciones o roles bien identificados (ver FIGURA 2).

Figura 2 Equipos de trabajo de la DGSIAF

- La Dirección de Gestión de Proyectos se encarga de gestionar la planificación, seguimiento, evaluar planes de mitigación, gestionar la contratación de recursos, hacer seguimiento de riesgos y gestionar los despliegues entre otras tareas.
- La Dirección de Análisis y Atención de Usuarios se encarga de interpretar los requerimientos de usuario y trasladarlos al área de Diseño y Desarrollo traducidos en

[17] Desde ahora en adelante utilizaremos el término APN para referirnos.

un lenguaje unificado de modelado (UML); así como también dar soporte ante inquietudes entre los usuarios y desarrolladores.

- La Dirección de Diseño y Desarrollo tiene como tarea materializar los requerimientos del usuario en artefactos visibles y ejecutables por el usuario.
- La Dirección de Ingeniería asegura el cumplimiento de controles de calidad de los artefactos generados como así también se encarga de investigar y adaptar prácticas de procesos ágiles al proyecto.
- La Dirección Técnica administra un centro de cómputos que asegura la operatoria diaria no sólo de nuestros sistemas productivos, sino que también posibilita el proceso interno de desarrollo. Provee además la conectividad a los usuarios dentro de la APN y brinda la infraestructura tecnológica para posibilitar el acceso desde múltiples dispositivos.

El área de Diseño y Desarrollo se divide a su vez en equipos dedicados a diferentes partes del dominio funcional, estos son Pagos, Gastos, Compras, Fondos Rotatorios, Contabilidad General, Cuenta Única del Tesoro, Mantenimiento, Entes, Pasajes y Viáticos, Recursos y Deuda Pública. (Hacienda, Organigrama de Equipos de la DGSIAF, s.f.)

A su vez, todo el desarrollo de esta área se apoya sobre arquitecturas de software definidas previamente por un equipo de trabajo llamado **Arquitectura**.

Su función es definir la arquitectura de software del e-SIDIF y la plataforma de desarrollo sobre la cual se construye el producto. Se encarga de proveer de actualización tecnológica a la plataforma, y de mantenimiento permanente de la misma, así como también de brindar soporte a los demás equipos.

Tanto el equipo de Diseño y Desarrollo como el de Arquitectura se complementan con otro equipo de trabajo llamado **Testing**, que se encarga entre otras cosas de hacer pruebas integrales, de regresión y circuitos; controlando la calidad de los ejecutables y verificando el cumplimiento de los requerimientos de usuario a través de casos de prueba para que finalmente llegue a producción un producto estable.

Negocios funcionales involucrados en el e-SIDIF

A continuación, se detallan los negocios del sistema, cuyo funcionamiento colaborativo constituyen los objetivos principales de la aplicación. Estos son: FOP, MP, Gastos, Pagos, MAP.

A lo largo de los años, conforme a los nuevos requerimientos solicitados por los usuarios, se fueron agregando sub-negocios o directamente negocios nuevos para realizar las funciones correspondientes. Nos centraremos particularmente en estos 5 debido a que involucran las tareas necesarias para poder explicar los circuitos de pago que motivan la presente Tesina.

Formulación Presupuestaria (FOP)

Contempla la participación del SAF en la formulación presupuestaria de los organismos dependientes de la jurisdicción.

La funcionalidad incluida es la siguiente:

- Elaboración de Techos presupuestarios.
- Elaboración del Proyecto de Ley.
- Presupuesto Plurianual.
- Elaboración de presupuestos preliminares.
- Etc.

Todo el trabajo que se realiza en FOP es insumo para las cargas correspondientes al Inicio de Ejercicio Presupuestario. Estas cargas se dividen en 3 momentos:

- **PreCarga:** Consta de la ejecución de determinados procesos que permite al SAF adelantar el trabajo correspondiente a la cuota del 1er Trimestre del año siguiente.
- **Carga de Inicio:** Este inicio de año calendario habilita a los SAF a realizar pagos de deuda, generación de formularios extrapresupuestarios, movimientos financieros, fondos, etc. en el nuevo ejercicio
- **Carga de la Decisión Administrativa:** Este proceso se realiza a mediados de enero cuando se firma la Decisión Administrativa correspondiente a la carga del presupuesto. Estos procesos cargan los clasificadores presupuestarios y asociaciones, tablas básicas y la generación de las imputaciones presupuestarias con los importes del presupuesto aprobado.

Modificación Presupuestaria (MP)

Negocio involucrado en la gestión de modificaciones presupuestarias que se originen en el SAF como las que emanan de autoridades superiores y se procesan en la ONP.

Gastos

Este negocio comprende el registro y liquidación de todos los gastos presupuestarios y de los no presupuestarios, que se pagan a través de una Orden de Pago.

Pagos

El módulo de Pagos debe proveer a la TGN y a las Tesorerías Locales de los SAF las herramientas necesarias para efectuar pagos, a través de los distintos medios de pago disponibles para estas operaciones, de acuerdo a las características definidas para cada uno de los intervinientes: Red Bancaria, Cheque, Transferencia Bancaria, Efectivo, Pago Electrónico, etc.

El módulo debe procesar los pagos de Órdenes de Pago, de acuerdo a las características del pagador, originando en las Tesorerías Locales o en la TGN, los medios de pagos para su cancelación a través de la Cuenta Única del Tesoro [18], o a través de otro medio de pago si se trata de un pago no CUT.

[18] Desde ahora en adelante utilizaremos el término CUT para referirnos.

Medidas de Afectación Patrimonial (MAP)

Sub-negocio de Pagos mediante el cual el e-SIDIF permite la administración de las Medidas de Afectación Patrimonial [19] tales como:

- Embargos: Medida cautelar judicial que afecta la disponibilidad y goce de un bien.
- Concursos: Juicio universal, basado en la existencia de una situación de cesación de pagos, en virtud de la cual el deudor insolvente solicita una prórroga respecto de sus deudas.
- Quiebras: Juicio universal que afecta la disponibilidad y goce del patrimonio del fallido. En la quiebra el fallido pierde la administración y disposición de sus bienes y el síndico es el autorizado para la liquidación del patrimonio del quebrado, siendo prioritaria sobre el resto de las medidas de afectación patrimonial.

TGN -Tesorería General de La Nación

Sistema de Tesorería

La maduración del proceso de reforma de la Administración Financiera Gubernamental Argentina iniciado con la sanción de la LAF y de los Sistemas de Control del Sector Público Nacional en 1992 (Ley 24.156, s.f.), particularmente en el ámbito del Sistema de Tesorería, motivó una renovación de este Sistema.

Se tomaron acciones para profundizar la descentralización operativa a fin de aumentar la productividad de los programas presupuestarios y llevarlos a resultados medibles en términos de su impacto posible. Ello se ha puesto de manifiesto en el diseño de una base única de información que está disponible para ser utilizada por los distintos administradores gubernamentales en todas las etapas del ciclo presupuestario, incluida la gestión de los fondos públicos.

En este sentido, el **Sistema de Tesorería** se estructura en torno a un Órgano Rector (OR), la **TGN**, que es la responsable de diseñar las normas, procedimientos y directrices que rigen su funcionamiento, y de supervisar y coordinar su aplicación en todo el SPN.

Complementariamente, se definen funciones operacionales específicas para el OR en su carácter de coordinador general de la administración de fondos del SPN. Asimismo, la aplicación de las actividades que hacen al funcionamiento del Sistema (la "operación" o ejecución de los procesos definidos por el OR) se nutre de las tareas propias de las diversas Unidades o Servicios de Tesorería, también llamadas "Tesorerías Jurisdiccionales" de las Entidades y Jurisdicciones pertenecientes a la Administración Nacional, integrantes del SAF correspondiente.

Por otra parte, las Tesorerías de las Entidades y Jurisdicciones del SPN se han ido uniendo al funcionamiento del Sistema Integrado de Administración Financiera, participando en el diseño de los procesos, capacitándose en ellos y llevando a cabo su efectiva implementación a través

[19] Desde ahora en adelante utilizaremos el término MAP para referirnos.

de los distintos instrumentos provistos por el Sistema hacia el logro de sus objetivos. (Tesorería General de la Nación)

Mediante la aplicación de estas directrices se ha logrado:

- Consolidar la administración de la totalidad de los recursos de las instituciones manteniendo la propiedad y disponibilidad de los mismos, garantizando adecuados niveles de descentralización operativa y autonomía, mediante la instrumentación de la **CUT**.
- Promover el reemplazo de las cuentas bancarias que usan las instituciones para el giro de sus operaciones por cuentas de registro con el objeto de lograr una notable disminución de los costos que implica su mantenimiento.
- Lograr una vinculación más ordenada del Sector Público con el sistema bancario a través de un organismo único: la Tesorería, la cual trabaja con los participantes del sistema en estandarizar plazos y modalidades uniformes de acreditación de fondos a los beneficiarios alentando la bancarización de la economía desde el Sector Público.

La **LAF** antes citada, define al Sistema de Tesorería como aquel "compuesto por el conjunto de órganos, normas y procedimientos que intervienen en la recaudación de los ingresos y en los pagos que configuran el flujo de fondos del SPN, así como la custodia de disponibilidades que se generen"

La Ley designa a la TGN como OR del sistema y coordinador del funcionamiento de todas las unidades o servicios de Tesorerías que operen en el SPN, asignándole las competencias que de manera abreviada se enumeran seguidamente:

- Participar en la definición de la política financiera del Sector Público.
- Elaborar juntamente con la ONP la programación de la ejecución del presupuesto de la Administración Nacional.
- Administrar el Sistema de Cuenta Única de la Administración Nacional.
- Supervisar técnicamente el funcionamiento de las Tesorerías que operen en el Sector Público.

Por otra parte, es responsabilidad del OR del sistema elaborar los procedimientos e instructivos que rijan su funcionamiento, supervisando la aplicación de los mismos en el ámbito del Sector Público. Los procesos a diseñar contemplarán como criterio general, la centralización normativa y la descentralización operativa.

En el nuevo modelo de gestión, el Sistema de Tesorería asume la responsabilidad de planificar y gestionar los grandes flujos financieros del Estado, tratando de coordinar su accionar con el resto de los sistemas que operan en la Administración Financiera y con los organismos que participan en la programación y la gestión de la política macroeconómica.

En este sentido, TGN asume un rol participativo en la determinación de la política financiera del Sector Público, debiendo desarrollar técnicas de programación que permitan conocer la restricción financiera que enfrenta el mismo en cada momento. Al mismo tiempo, dicha información debe permitir evaluar el impacto de los flujos monetarios y de divisas del sector sobre el sistema económico en general, tratando de reducir efectos distorsionantes en el

mercado monetario y de capitales. Esto debe retroalimentar el proceso decisorio conducente a la programación de la ejecución financiera.

En tal sentido la Tesorería tiene vinculación con las oficinas de Presupuesto y Crédito Público en materia de programación de la ejecución o ejercicio del presupuesto, así como con agencias recaudadoras o dependencias encargadas de elaborar la estimación de los ingresos para poder confeccionar sus escenarios de caja.

Las Tesorerías Jurisdiccionales y la CUT

Las Tesorerías Jurisdiccionales, siendo parte del Sistema, pueden recaudar y registrar los ingresos de naturaleza no tributaria a su cargo derivando su depósito a la CUT, programar y disponer pagos parciales o totales de documentos de pago y sus deducciones en base a los límites financieros otorgados, registrar medidas de afectación patrimonial y conciliar sus cuentas bancarias y escriturales.

La **programación financiera** es un instrumento, cuyo desarrollo transforma al Sistema de Tesorería, y a su OR, la TGN, en una verdadera gerencia financiera del SPN, a través de la cual se genera información de alto valor agregado, indispensable para la toma de decisiones y una buena gestión de caja.

Su objetivo es optimizar la productividad de los recursos involucrados en la gestión de Tesorería y generar información oportuna y confiable para la toma de decisiones, incrementando la articulación entre la gestión de caja y la ejecución del presupuesto de manera de asegurar que las entidades ejecutoras del gasto reciban oportunamente los recursos requeridos para poder proveer los bienes y servicios gubernamentales de manera eficiente y efectiva. Esto lo realiza mediante el planteo de distintos escenarios estimados posibles, en función de diferentes hipótesis para variables relevantes: ingresos, pagos, activos y pasivos financieros.

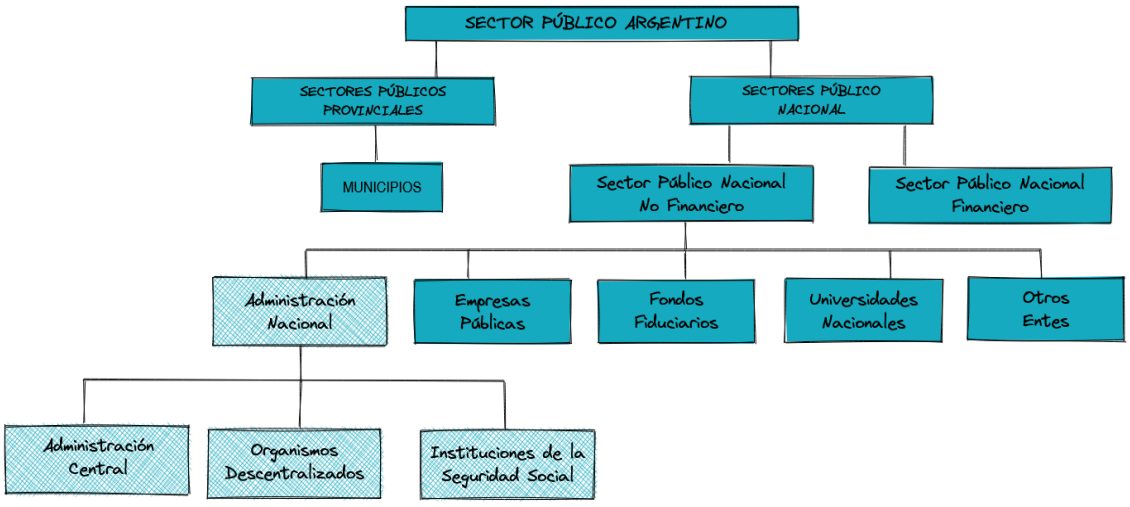


Figura 3 Organigrama del Sector Público

Con la implantación de la CUT, los recursos de los organismos descentralizados y los afectados de la administración central, así como los fondos de terceros se incorporan al flujo de fondos de tesorería, en las etapas de recaudación, programación y pagos.

Contrariamente, se excluyen los sectores que por sus características y/o disposición institucional no se encuentran incluidos en el Presupuesto Nacional, y que, por lo tanto, no constituyen objetos de la programación que administra el Sistema de Tesorería liderado por la TGN (ver FIGURA 3). Ellos son:

- El Sector Público Provincial, bajo el régimen Federal de la República Argentina, se organiza institucionalmente de acuerdo a las Constituciones emitidas por las provincias que integran el país. Tienen, por lo tanto, gestión financiera y presupuestos independientes, por lo que la programación y gestión financiera de los recursos relacionados con estos presupuestos es de su responsabilidad. Lo mismo ocurre, por consiguiente, con las estructuras municipales en las que se desagregan las administraciones provinciales, las cuales dependen de los gobiernos de las mismas.
- El Sector Público Nacional Financiero, conformado por la autoridad monetaria, BCRA[20], y el sector bancario público, cuyo presupuesto operativo se aprueba por Decreto, se excluyen en función de la especificidad de los objetivos que guían su accionar. Estos difieren claramente en cuanto a los propios de la Administración Financiera del resto del Sector Público Nacional, vinculado al Presupuesto Nacional.
- Adicionalmente dentro del análisis del Sector Público Nacional No Financiero, nos encontramos con Instituciones con mayor autarquía o autonomía, tales como las Sociedades y Empresas del Estado, otros entes públicos excluidos expresamente de la Administración Nacional, los fondos fiduciarios y las Universidades Nacionales, consideradas, de acuerdo a la normativa como Organismos Descentralizados. El contacto con dichas instituciones se verifica mediante las transferencias presupuestarias que reciben del Tesoro. Por consiguiente, sus recursos y gastos no se encuentran incorporados a la programación del flujo financiero del Tesoro.

Por su parte, las Instituciones de Seguridad Social, dada la complejidad operativa del Sistema Previsional, se mantuvieron legal y funcionalmente excluidas del Sistema de la CUT. Sin embargo, fueron integrándose paulatinamente a la Programación Financiera del Tesoro a través del intercambio de información que permitió la consolidación de los programas de caja de ambos sectores. (Tesorería General de la Nación)

Buenas Prácticas

Por buenas o mejores prácticas se entiende un conjunto coherente de acciones que han rendido bien o incluso excelente servicio en un determinado contexto y que se espera que, en contextos similares, rindan similares resultados. Dichas "buenas prácticas" dependen de las

[20] Banco Central de la República Argentina.

épocas, de las modas y, por esto último, es posible que algunas resulten contradictorias entre ellas mismas. (Cubicup Design Home, 2022)

Debido al crecimiento que tuvo el e-SIDIF a lo largo de los años, sumándose nuevos requerimientos a los negocios anteriormente explicitados o negocios funcionales nuevos, se hizo necesario aplicar buenas prácticas, apoyándonos en la metodología Agile.

Dichas buenas prácticas ayudarán a que el equipo de desarrollo de la aplicación pueda implementar dicha metodología (o al menos cumplir algunos de sus principios)

La metodología ágil

"Agile" es un conjunto de métodos ágiles destinados a eliminar obstáculos innecesarios entre departamentos y burocracia, flexibilizando el proceso de desarrollo y dividiéndolo en pequeñas tareas que se puedan realizar en periodos cortos de tiempo, acortando los tiempos de ciclo y permitiendo adaptar el trabajo a las necesidades de los usuarios. (Samaniego, s.f.)

Una metodología Agile puede implementarse en su totalidad, es decir todo su conjunto de métodos ágiles, o solo aplicar algunos. (Pursell, 2021)

El equipo de Desarrollo del Sistema e-SIDIF cumple dos métodos en particular, la entrega continua de software y no tener miedo de realizar cambios al producto entregado, los cuales se detallarán a continuación:

- **Lograr la satisfacción del cliente a través de la entrega continua de software.**

  En marcos ágiles de desarrollo de software, un sprint es un cuadro de tiempo fijo repetible durante el cual se crea un producto "Terminado" del valor más alto posible.

  Para el desarrollo del e-SIDIF se hacen entregas a Testing semanales o quincenales, dependiendo del Sprint.

  Para facilitar esta tarea, el área de Ingeniería de Software del Proyecto, creó herramientas que ayudan en la integración de los distintos negocios funcionales involucrados (Pagos, Gastos, etc.), la coordinación y el correspondiente despliegue de la aplicación en el ambiente deseado, por ejemplo, testing o producción. Dichas herramientas conforman un "Toolkit" [21] que fue modificándose y ampliándose conforme crecía el proyecto. El mismo se incorpora como extensión dentro del Eclipse (IBM - Eclipse, s.f.), que es la herramienta utilizada por el e-SIDIF para el desarrollo; de esta manera tenemos todas las opciones que brinda dicho Toolkit en el mismo ambiente de trabajo.

  Además, se crearon otras herramientas que agilizaban la resolución de problemas que se empezaron a dar en producción conforme crecía la estructura del sistema, como lo es aquella para correr scripts SQL de emergencia en producción, llamados "cocineros", por fuera de una entrega formal. Dichos cocineros son ejecutados cuando hay un cambio en Base de Datos que se necesita urgente en producción y no puede esperar hasta la próxima entrega programada.

[21] Kit de herramientas para optimizar el desarrollo de la aplicación.

Hubo otros casos, como el presente de interoperabilidad con Red Link, donde si bien se realiza una entrega programada de funcionalidad, la misma no debe aplicar para todos los organismos o SAF involucrados en el e-SIDIF. Cada entrega programada, y por ende cada innovación en el producto, será entregada a todos los usuarios de todos los SAF, pero solo un determinado grupo de los mismos podrán hacer uso de los nuevos features [22] entregados. Es lo que se denomina Salida controlada a producción y se despliegan en producción a través de los "Cocineros" o entregas de SQL dinámicos antes mencionados.

- **No tener miedo de realizar cambios.**

Dentro de las entregas de producto que se realizan, están las que se dan a Testing para el control de calidad y la detección de errores funcionales en la aplicación y las entregas finales a producción.

Si bien Testing cuenta con casos de prueba definidos junto con Análisis, y la probabilidad de que un error grave llegue a producción es baja, se dispone de entregas de emergencia que pueden programarse de un día al otro, pasando la corrección primero en un "Hot Fix" (como se llaman estas entregas rápidas) a Testing y luego otro "Hot Fix" a producción.

Esto ayuda a que no se tenga tanto miedo en incluir un cambio, ya que existen estas alternativas de corrección rápida.

Esto es en relación a cambios que se producen por nuevos pedidos de los clientes y que pueden llevarse a cabo con las herramientas de desarrollo a las que estamos acostumbrados en el proyecto.

Pero pude darse el caso que, para satisfacer un pedido del usuario, sea preferible interactuar con nuevas técnicas de desarrollo y tecnologías.

En estos casos no se necesita esperar a que se cree el próximo sistema o su rediseño. Los procesos ágiles aprovechan el cambio como una ventaja competitiva a favor del cliente.

No se debe tener miedo de ejecutar una modificación grande, incluso si ya está avanzado el proceso de desarrollo. Las empresas que no temen al cambio son las que obtienen mejores resultados y permanecen. Tal es el caso del e-SIDIF, donde la estructura monolítica de la aplicación llevó a pensar una solución donde ésta pudiera interactuar con microservicios desplegados fuera del entorno del e-SIDIF, a través de comunicación asincrónica con éste último, sin tener que repensar un diseño nuevo de lo ya construido.

Feature Flags

Los feature flags o también conocidos como feature toggles es una poderosa técnica que nos permite habilitar o deshabilitar funcionalidad en nuestro código. (Petrini, 2020)

Están diseñados para cambiar la funcionalidad de ACTIVADO a DESACTIVADO en cualquier momento, incluso después de que se haya implementado nuestro código (ver FIGURA 4). Como

[22] Feature: Unidad funcional de un sistema de software que satisface un requisito.

tales, nos ayudan a controlar la entrega al enviar nuevas funcionalidades a un grupo específico de usuarios y de esta manera elegir el porcentaje exacto de usuarios que recibirán una innovación del producto. (Novoseltseva, 2020)

Dentro del e-SIDIF, dicha técnica nos ayudó para aumentar la aplicación monolítica de acuerdo a nuevos requerimientos funcionales, sin afectar la integración y despliegue de la misma.

Features Flags temporales y permanentes:

Los feature flags temporales son usados para hacer deploy [23] de cambios de manera segura o para probar nuevos comportamientos en tu código fuente en lugar de un previo comportamiento. Cuando el nuevo comportamiento es usado por el 100% de tus usuarios, se procede a eliminar el feature flag.

Los feature flags permanentes nos brindan la posibilidad de controlar el comportamiento de nuestra aplicación en cualquier momento. Un escenario podría ser cuando queremos mostrar cierto comportamiento a una cantidad específica de usuarios. (Petrini, 2020)

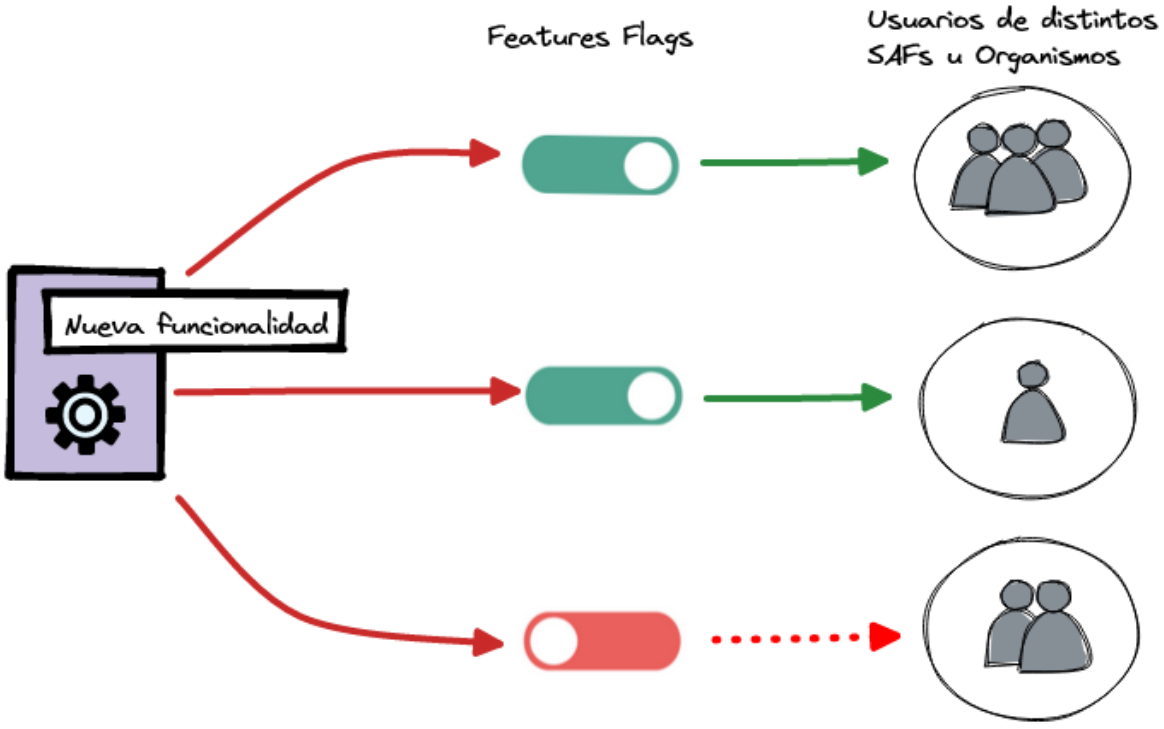


Figura 4 Nueva funcionalidad para distintos grupo de usuarios

[23] Despliegue de una aplicación.

Ejemplo:

Supongamos que tenemos una función en nuestro código fuente que envía correos electrónicos usando el servicio *Mailgun* (Mailgun Technologies, Inc, s.f.). Se toma la decisión de migrar el servicio hacia la plataforma de envío de mails *SendGrid* (Twilio SendGrid, s.f.). A medida que implementamos el nuevo servicio, esperamos continuar manteniendo el servicio de *Mailgun* por un período de tiempo. Al configurar un Feature Flag para esta nueva implementación hace que el comportamiento de esta función sea dinámico. El código de la función puede cambiar de servicio sin la necesidad de lanzar una nueva versión del producto completo (ver FIGURA 5).

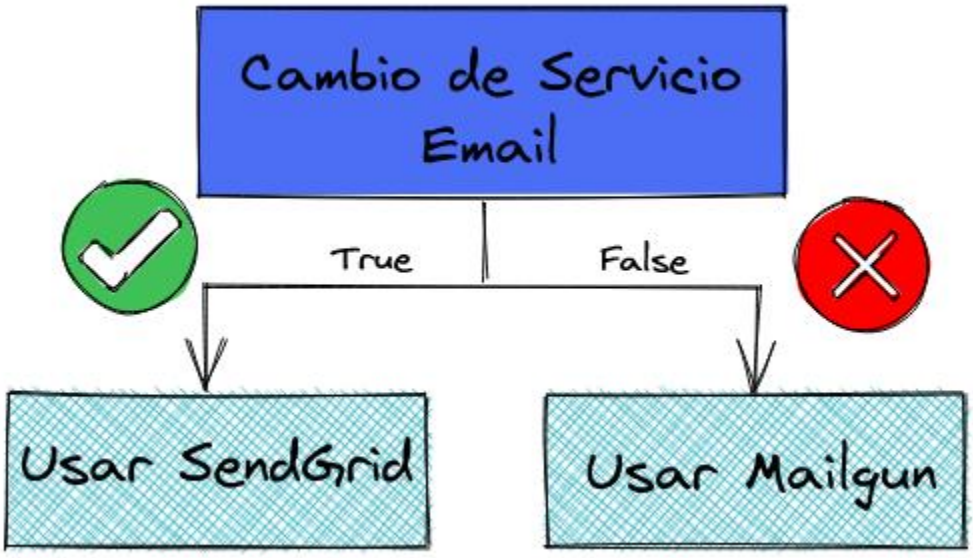


Figura 5 Cambio en el servicio de envío de mails

El objetivo final sería poder quitar la implementación del servicio Mailgun cuando se crea seguro; pero como veremos más adelante, para el caso de nuestra aplicación, será un feature flag temporal que dadas las circunstancias del negocio quedará de forma permanente. (Petrini, 2020)

En el caso de e-SIDIF resultó ser una buena práctica usar flags para habilitar nueva funcionalidad o no, dependiendo de que organismos podían disponer de ella en el momento del despliegue de una nueva versión. Esto se debió a que la aplicación es monolítica y escaló a tamaños gigantescos durante todos estos años de desarrollo. Debido a eso, cada deploy a producción involucra riesgos, que pueden ser minimizados con salidas controladas usando features flags.

Esta solución, que para nuestro caso de uso es una buena práctica, puede no serlo en otras implementaciones.

Por ejemplo, el uso de features flags para salidas controladas se debió al problema surgido de hacer un deploy de una aplicación monolítica muy grande como es el e-SIDIF, donde

"encender" o "apagar" funcionalidad por fuera de la aplicación ya desplegada en producción representaba una solución más viable a deployar o desplegar nuevas versiones del producto. De esta manera el código fuente se mantiene sin cambios, y el uso de la nueva funcionalidad está controlada a través de un script SQL que activa o desactiva el flag.

Incluso se evaluó su uso nuevamente para una funcionalidad pedida para el año corriente que consiste en una interacción parecida a la de Red Link, pero en este caso es para transferencias inmediatas al BNA. Dicha funcionalidad se integraría con el e-SIDIF y se desplegaría de la misma manera. Lo cual hace que una buena práctica pueda ser reusada en un contexto parecido.

Pero estas buenas prácticas quizás no lo sean, o sean obsoletas, por ejemplo, en el caso que nuestro sistema sea chico o esté desacoplado en microservicios y los despliegues a producción sean menos costosos.

Conclusiones

A lo largo del capítulo se hizo un recorrido por la evolución del e-SIDIF, destacando como el puntapié inicial, en el proceso de formulación presupuestaria nacional y el registro de la ejecución presupuestaria, la sanción de la LAF 24.156 (Ley 24.156, s.f.), a través de la cual se desarrolló una renovación del SPN tanto a nivel operacional como también en la manera en que se restructuró y actualizó el sistema informático para adecuarlo a los tiempos que corrían.

Se detallaron los distintos sectores que se planificaron para que el sistema e-SIDIF se pusiera en marcha y fuera actualizándose de acuerdo a nuevos requerimientos de los organismos involucrados.

También se destacó que esta maduración del proceso de reforma, llevó consigo a una renovación del Sistema de Tesorería; poniendo en contexto la TGN y los pagos que se realizan a través de la CUT y las funciones que se atribuyeron a las Tesorerías Jurisdiccionales de los distintos SAF.

Llegamos a la conclusión de que un sistema tan amplio y complejo en su operatoria como el e-SIDIF requerían de buenas prácticas para su buen funcionamiento. Estas buenas prácticas, utilizadas en la metodología agile, abarcaron desde la formación de equipos para facilitar la tarea de desarrollo, hasta aquellas destinadas a mejorar los ciclos de entrega de nueva funcionalidad.

Para ello, llegó incluso a definirse un equipo sólo dedicado a Ingeniería de Software, encargado de facilitar herramientas para una mejor puesta en producción de nuevos productos.

Por otra parte, se explicó en detalle una buena práctica llamada Features Flags, que sirven para salidas a producción controladas y que ayudarían al despliegue de esta nueva funcionalidad que motiva la presente Tesina.

Capítulo 3 - Ordenes de pago, Retenciones y Medios de Pago

Introducción

El Sistema e-SIDIF provee un módulo de Pagos que permite a sus usuarios, ya sea en la TGN o en los SAF, como por ejemplo Institutos, Secretarías, Subsecretarías, Ministerios y Hospitales, entre otros, llevar a cabo el registro, procesamiento y control de los pagos realizados (auditorías).

En el proceso intervienen la TGN y Tesorerías Locales de los SAF, en distintos grados de interacción tanto en las operaciones realizadas en forma exclusiva por los SAF como en las iniciadas en forma centralizada en la TGN.

El módulo debe procesar los pagos de Órdenes de pago [24] de acuerdo a las características del pagador (SAF o TGN), originando en las Tesorerías Locales de los SAF o en la TGN los medios de pagos para su cancelación.

El medio de pago es el instrumento mediante el cual se realiza la cancelación de deudas. Existen distintos tipos de medios de pago como Orden Bancaria, Nota de Pago, Cheque, Transferencia Bancaria, Pago Electrónico, Red CUT, entre otros.

Circuito de OP

El circuito de OP presupuestarias describe el circuito de cancelaciones de órdenes de pagos incluyendo las retenciones generadas, independientemente de quien sea el beneficiario o el medio de pago elegido para su cancelación.

Una OP contiene información sobre un documento liquidable, sobre la cual, en caso de corresponder se aplicarán deducciones (ver FIGURA 6). Una vez autorizada la OP se inicia el proceso de pago de la OP.

Para realizar el pago de una OP el usuario utiliza la opción “selección de pagos de OP…” del e-SIDIF (ver FIGURA 7), donde seleccionara las OP que desea cancelar e indicar si el pago es total o parcial.

Si se cumplen todos los controles y validaciones la herramienta permitirá aprobar la OP, generando así un comprobante de respaldo denominado “Comprobante de Pago OP”.

Dicho comprobante de pago podrá ser visualizado en la opción “Abrir comprobante de pago de OP” en el e-SIDIF, donde se mostrará información del mismo como así también de la OP asociada (ver FIGURA 8).

Una vez confirmado dicho pago, se originarán en las Tesorerías Locales de los SAF o en la TGN los medios de pago para la cancelación de la deuda.

[24] Desde ahora en adelante utilizaremos el término OP para referirnos.

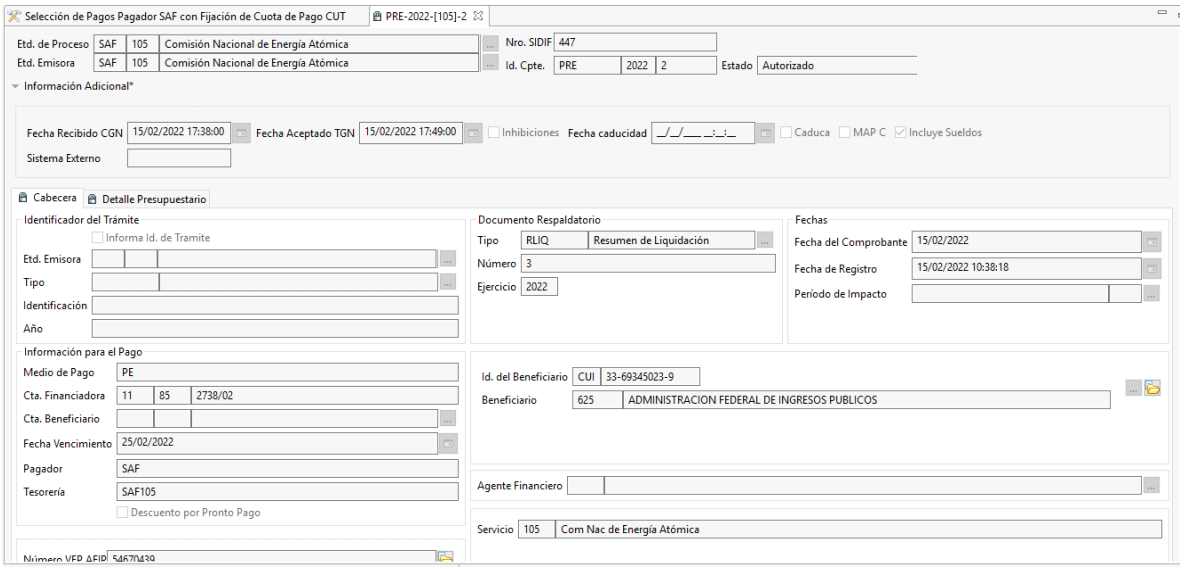


Figura 6 Visualización de la orden de pago en e-SIDIF

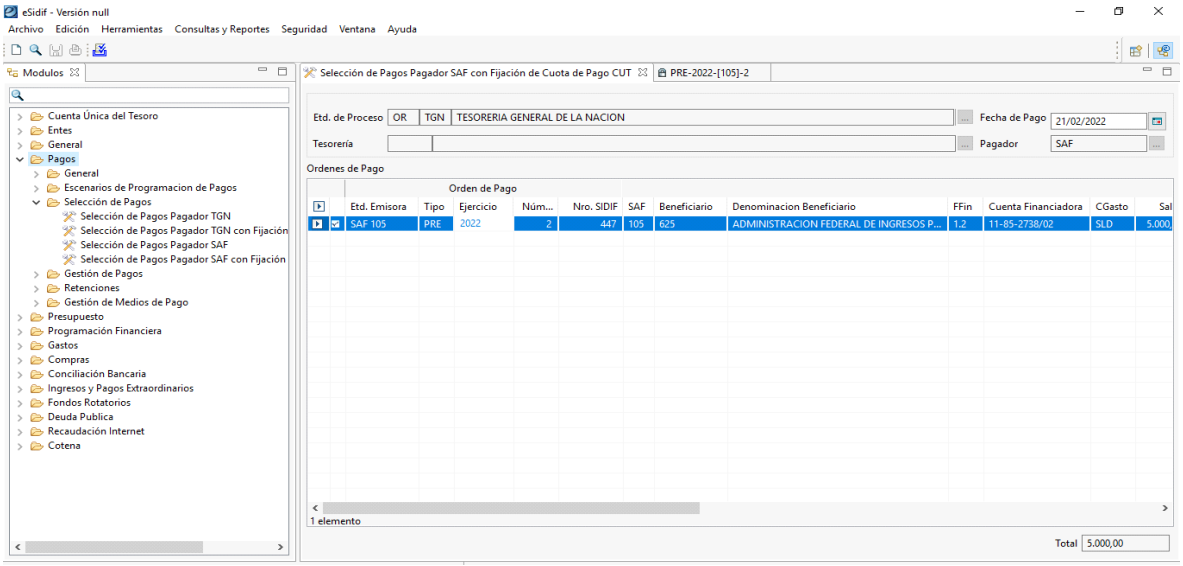


Figura 7 Selección de pago de OP

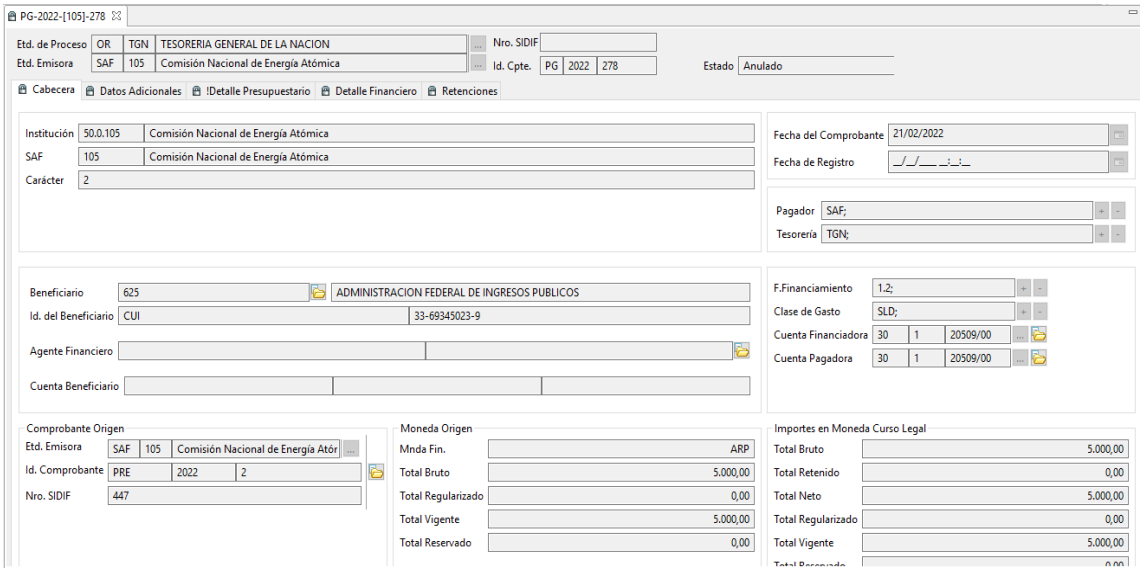


Figura 8 Visualización del comprobante de pago

Tanto los comprobantes originales que representan las OP, como así también los comprobantes de respaldo (o comprobantes de pago) y los medios de pago generados para la cancelación tienen asociado un estado, que representan en que momento del proceso se encuentran los mismos (ver FIGURA 9).

Cada estado determinará que operaciones podrán realizar los distintos usuarios sobre dichos comprobantes, siempre dependiendo el nivel de seguridad que posean y en qué etapa del circuito de pagos se encuentren.

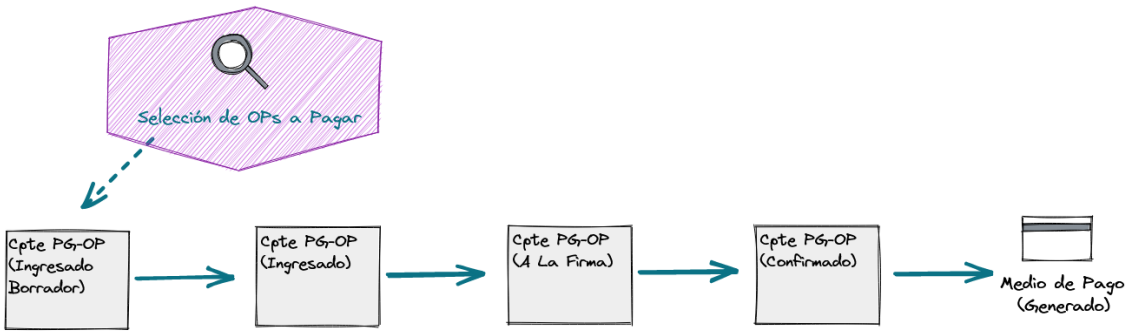


Figura 9 Estados asociados al comprobante de pago

Los estados asociados al comprobante de pago son:

Ingresado Borrador: Estado inicial que toma el comprobante al ser generado a través de la selección de pago de la OP; contiene los datos de cabecera del pago (Identificación del comprobante, fecha de pago, cuenta pagadora, etc.), datos del comprobante original (OP) y el detalle presupuestario del pago.

Ingresado: Estado intermedio antes de esperar las firmas de los responsables para su autorización, donde aún es posible editar determinadas propiedades del comprobante.

En Proceso de Firma: Durante este estado el comprobante no podrá ser editado, solo podrá ser firmado por los usuarios responsables de autorizar el pago.

Confirmado: Cuando el comprobante de pago es firmado por todos los responsables y se realizan todos los impactos, incluido la generación del medio de pago.

Pendiente por MAP: El pago queda en este estado cuando el beneficiario tiene alguna MAP pendiente aún no resuelta.

Anulado: Al quedar el comprobante en este estado, se revertirán los impactos que se hayan realizado hasta el momento de anulación, quedando la OP liberada para una nueva selección de pago de ser deseable.

Para continuar con la gestión del pago, el usuario que dispondrá de una opción de búsqueda que le permitirá encontrar dicho comprobante de pago en estado "Ingresado Borrador".

Una vez encontrado podrá abrirlo en un editor, de ser necesario modificarlo y/o pasarlo al siguiente estado (ver FIGURA 8).

Las OP con comprobantes de pago en estado "*Ingresado"* o "*A la Firma"*, no pueden ser seleccionadas nuevamente ya que estarán "bloqueadas" por una gestión de pago en curso, excepto que el comprobante de pago sea por un monto parcial al de la OP o en el caso de que el comprobante de pago sea anulado y la OP estaría nuevamente disponible para ser seleccionada.

Una vez que el comprobante de pago de la OP está Ingresado, se lo debe pasar al estado "En Proceso de Firma" para iniciar el circuito de Autorización y solo lo podrá realizar aquellos usuarios que tengan asignado el correspondiente permiso.

El firmante podrá:

- **Autorizar:** Para que se realicen los impactos restantes y así finalizar el proceso de pago de las OP involucradas.
- **Rechazar Firma:** Para que el comprobante de pago vuelva al estado ingresado y pueda ser anulado en caso de ser necesario.

Una vez que el comprobante de pago fue firmado por todos los responsables, el mismo pasará a estado "Confirmado".

Al confirmase el pago de la OP, automáticamente:

- Se generar el medio de pago asociado a la misma.
- En caso de corresponder, por cada una de las deducciones informadas, se genera un comprobante de retención devengada.
- Se completa en el comprobante de pago la información correspondiente al detalle financiero (medio de pago generado) y al detalle de retenciones devengadas.
- Se aplican todas las medidas de afectación patrimonial asociadas al beneficiario del pago (Embargos, Concursos, Quiebras, etc.).

Finalmente, se contempla la posibilidad de revertir la gestión de un comprobante de pago a lo largo del circuito. En ese caso, se revertirán los impactos contables que se hayan realizado hasta el momento de anulación.

Por último. un Comprobante de Pago en estado “Confirmado” (todos los impactos realizados) puede ser anulado. Este proceso se los denomina “Anulación de Pago”, cuyo fin es revertir los impactos realizados al momento de la autorización.

Circuito de Pago de Retenciones

Al momento de la confirmación del pago de OP se aplican las deducciones informadas en la factura o documento de liquidación equivalente, y además en esta instancia se calculan en forma automática y se aplican las deducciones que correspondan de acuerdo a lo que se esté pagando, generando un Comprobante de Devengado de Retención de Gastos [25] en estado autorizado por cada deducción devengada en el pago (ver FIGURA 11 – Sección B del circuito).

Dependiendo de la deducción y de cierta característica del beneficiario puede variar el sujeto al cual se le practica la retención, como por ejemplo deducciones de ganancias o IVA.

Cada CDR queda asociado al comprobante de pago de OP que lo originó.

Los CDR autorizados podrán ser seleccionados para el pago mediante la opción “Selección de Pago de Retenciones” (ver FIGURA 10).

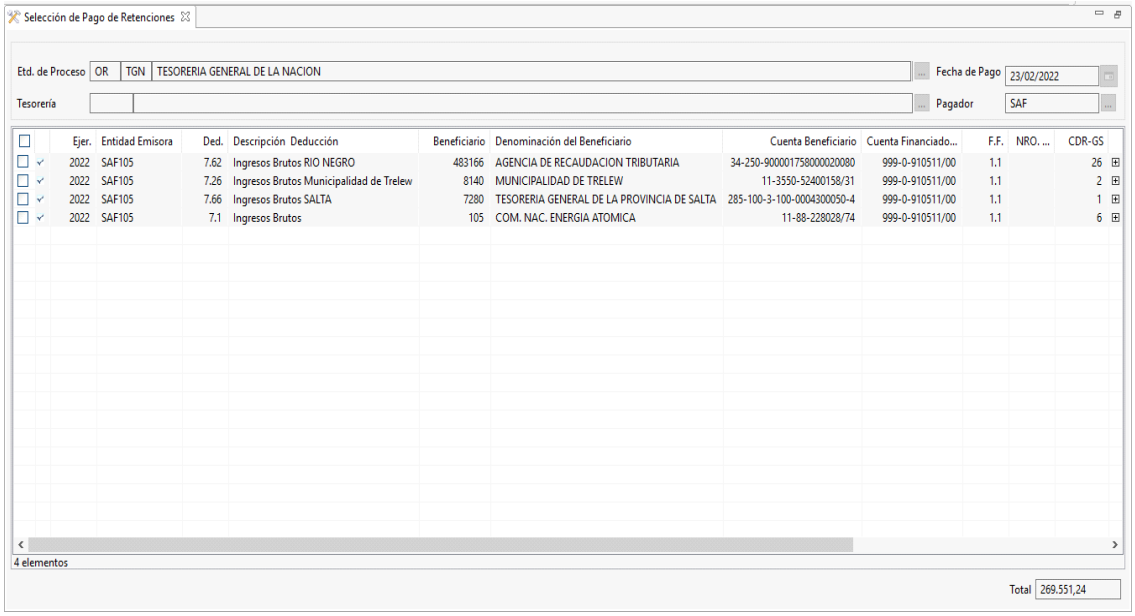


Figura 10 Selección de pago de retenciones

[25] CDR-GS: Comprobante de Devengado de Retención de Gastos. Desde ahora en adelante utilizaremos el término CDR para referirnos.

Los estados por los que pasará dicho comprobante son los mismos que para el Comprobante de Pago OP: Ingresado Borrador, Ingresado, En Proceso de Firma, Confirmado y Anulado.

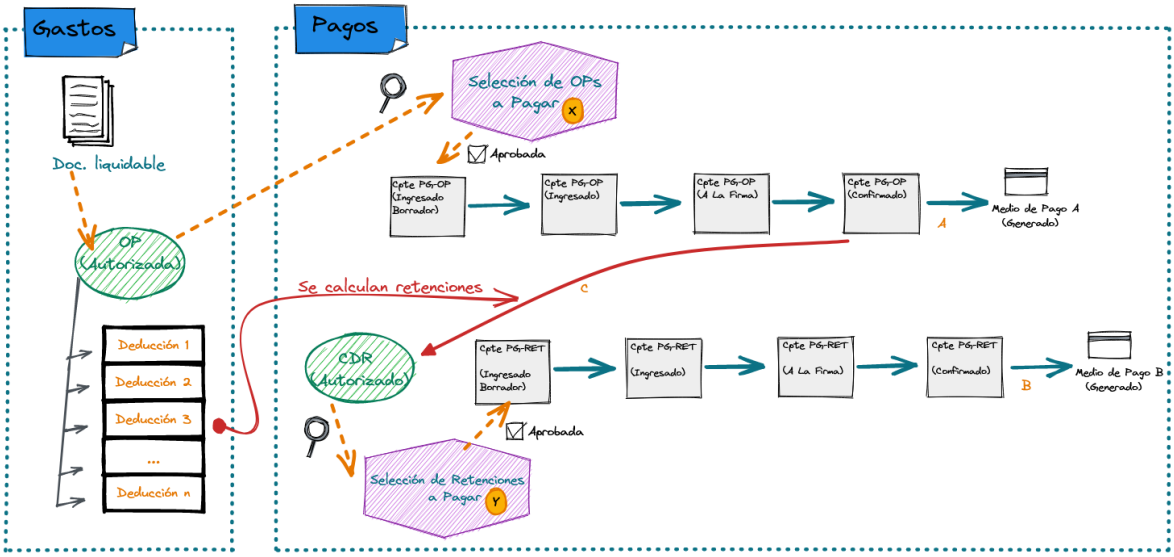


Figura 11 Relación entre el pago de OP y el pago de retenciones

- **Doc. Liquidable** = Documento liquidable (Ej. Factura) por el cual se genera la Orden de Pago
- **OP** = Orden de Pago
- **Cpte PG OP** = Comprobante de Pago de OP
- **CDR** = Comprobante de Devengado de Retención
- **Cpte PG RET** = Comprobante de Pago del CDR
- **Medio de Pago A (Generado)** = Medio de Pago en estado Generado para cancelar la OP.
- **Medio de Pago B (Generado)** = Medio de Pago en estado Generado para cancelar el CDR.

Medios de Pagos de OP y Retenciones

Cuando se realiza la selección de OP a pagar como también de retenciones, se calcula en forma automática, el tipo de medio de pago con el cual debería efectivizarse el pago.

De ser necesario, antes de "Poner a la Firma" el comprobante del pago asociado, se puede realizar un cambio del medio de pago original a alguno de los habilitados para el cambio según una tabla de cambios de medios de pago que utiliza el sistema para saber cuáles son válidos. Este cambio dejará el medio de pago original en estado “Anulado” y se creará el nuevo en estado “Generado”.

Los medios de pago siguen un diagrama de transición de estados (DTE) al igual que los comprobantes de respaldo (comprobantes de pago) y los comprobantes originales (OP y CDR). Estos estados son “Generado”, “Emitido” y “Anulado”. Los estados “Emitido” y “Anulado” podrán tener sub-estados asociados como se explicará más adelante.

Medio de pago Red CUT - Cuenta Única del Tesoro (CUT)

Este medio de pago genera la transferencia automática a cuentas bancarias, corrientes o de ahorro, de los importes correspondientes a los pagos que realizan los SAF que emiten los comprobantes pasibles de pago a través de la CUT.
Este medio de pago es utilizado para saldar deuda con cualquier tipo de acreedor, sea persona física o jurídica, entidades públicas o privadas. Estos beneficiarios deben encontrarse registrados en el Módulo de Entes de la APN, donde se informan los datos bancarios (banco, sucursal, cuenta bancaria, CBU, CUIL o CUIT). A los beneficiarios, acreedores del Estado, les es permitido mantener un solo dato bancario registrado en el Módulo de Entes, en cambio de tratarse de Organismos Oficiales les es permitido mantener más de un dato bancario.

Una vez que los comprobantes pasibles de pago, tanto los de Pagador TGN como los de Pagador SAF, fueron seleccionados y confirmados en el sistema se incorporan a lotes de pago para su posterior envío al BNA a través del Sistema Nacional de Pagos [26].

La TGN procede al armado de lotes de pago con aquellos comprobantes de pago confirmados, los que tienen asociado cada uno de ellos un Número de Identificador Bancario. Para su envío al Agente Financiero se generarán archivos con uno o más lotes de pagos, los cuales una vez transmitidos, posibilitan el intercambio de la información entre el Sistema de la Tesorería y el Agente Financiero (BNA); tanto para el envío de la operatoria de pagos como para la aceptación o rechazo de los mismos.

Los bancos que están incluidos en el proceso de compensación electrónica de transferencias bancarias, serán los destinatarios finales de los pagos realizados por la TGN y acreditarán los fondos en las cuentas de los respectivos beneficiarios.

Tanto el pago de OP como el pago de retenciones, a favor de la AFIP, se realiza actualmente a través del medio de pago denominado **Red CUT**. En este caso, dicho medio de pago es el utilizado para los pagos que efectúen los diferentes SAF a través de la Cuenta Única del Tesoro y tengan como beneficiario de los mismos a la AFIP. Funcionalmente, es una especificación del medio de pago "Transferencia Bancaria", donde la cuenta pagadora es siempre la CUT.

Los estados posibles para un medio de pago Red CUT son los siguientes:

- **Generado**: Estado en que queda al ser confirmado un comprobante de pago. Indica la registración del medio de pago en el sistema e-SIDIF.
- **Emitido**: La emisión, implica las impresiones y entregas correspondiente. Cada uno de los medios de pago, tendrá su circuito particular de emisión, impresiones y entregas. El medio de pago se agrupará en lotes, y estos en un archivo. Finalmente, este archivo será el que se enviará al Banco de la Nación Argentina (BNA) para su procesamiento (ver FIGURA 12 – camino **A**), quedando en el sub-estado "En Proceso Bancario". En dicho sub-estado, el medio de pago podrá ser acreditado, quedando el mismo en el sub-estado "Acreditado" (ver FIGURA 12 – camino **B**), o en caso contrario, ser rechazado bancariamente, pasando al estado "Anulado".

[26] Desde ahora en adelante utilizaremos el término SNP para referirnos.

- **Anulado**: Estado en que queda el medio de pago ya sea por ser anulado el comprobante de pago confirmado que lo generó desde el e-SIDIF (ver FIGURA 12 – camino **C**), como también cuando, una vez emitido, es rechazado por el banco que lo recibió (ver FIGURA 12 – camino **D**).

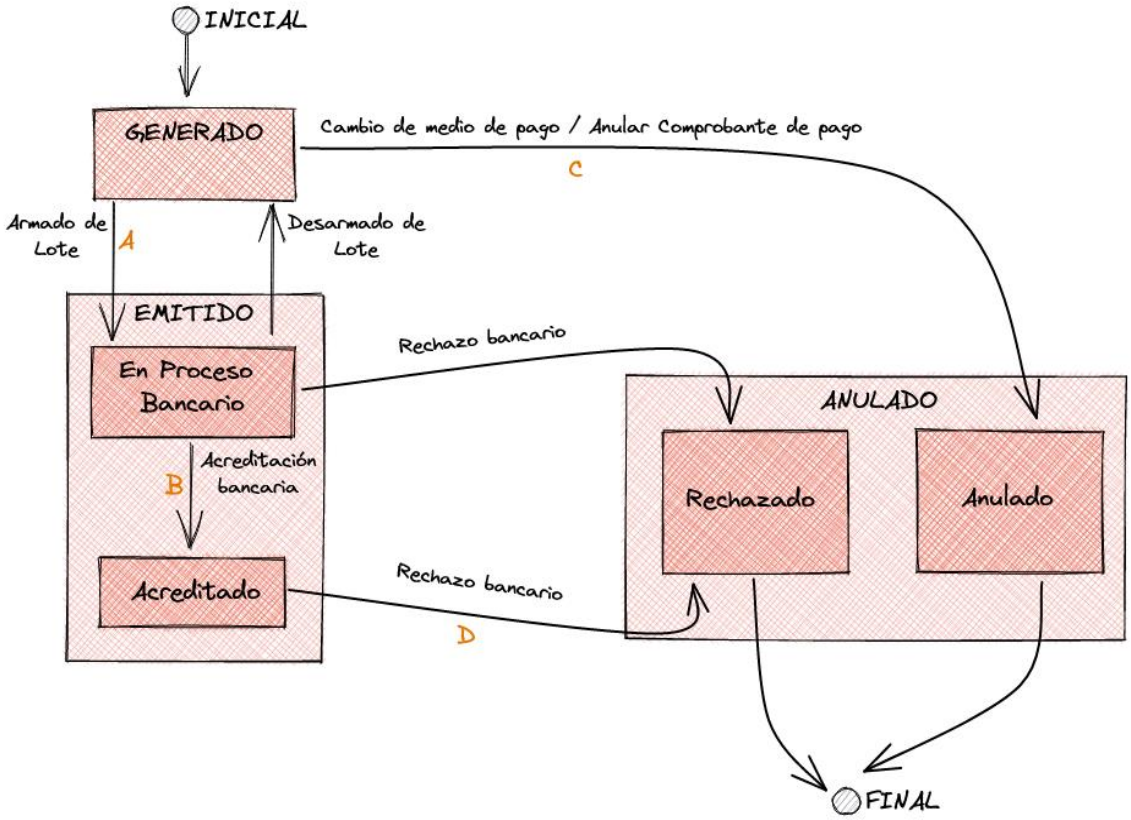


Figura 12 Transiciones de estado del medio de pago Red CUT

Volante Electrónico de Pagos (VEP)

Desde el e-SIDIF, el usuario del SAF que quiera realizar un pago a favor de la AFIP, deberá crear además un Volante Electrónico de Pagos (VEP) en la etapa de Selección de Pago (ver FIGURA 11 – Selección de pago **X**) (ver FIGURA 11 – Selección de pago **Y**).

El uso de VEP es obligatorio para el pago de todas las obligaciones impositivas y de los recursos de seguridad social.

Esto permitirá asociar las retenciones que se van a pagar, como también las OP de Aportes y Contribuciones a un VEP.

Para la generación de VEP, la AFIP provee un servicio web, para que, desde el e-SIDIF, se envíe la información del VEP que se desea generar, y la AFIP retorne el número de VEP generado.

Una vez recibido el número, este queda asociado a las retenciones agrupadas en el CDR o a la OP, dependiendo el pago realizado. Además, se registrará la información del VEP creado en AFIP, en una entidad propia modelada en el e-SIDIF, que contendrá información del mismo, como por ejemplo si fue pagado o si expiró por vencimiento, así como también el detalle de los pagos asociados.

Proceso de Pago de Retenciones a través de la CUT

1. Desde el e-SIDIF se envían los datos necesarios para que la AFIP genere el VEP por las retenciones que va a pagar.
2. AFIP Procesa y retorna el número del VEP al Sistema e-SIDIF. Además, envía la información del nuevo VEP a Red LINK.
3. Se asocian las retenciones agrupadas al número de VEP generado en AFIP.
4. Una vez confirmado el pago de dichas retenciones, se genera el medio de pago Red CUT con el VEP correspondiente asociado.
5. Periódicamente, y en paralelo, los SAF van cargando y firmando los VEP asociados a los pagos, en la página web de BNA "Banca Empresa 24" (BNA - Nación Empresa 24, s.f.).
6. A medida que los pagos son confirmados, los medios de pago para su cancelación con su VEP asociado, se agrupan en lotes. Al final del día, se envía un solo archivo que contiene los lotes al BNA, a través de un vínculo de comunicación con dicho banco.
7. Una vez enviado el archivo, el BNA lo procesa para realizar las operaciones informadas: Por cada pago informado en el medio de pago Red CUT, el banco debita los fondos de la CUT y los acredita a través de una transferencia bancaria a una cuenta puente habilitada a estos efectos por el monto del VEP. Por ejemplo, una transferencia bancaria por un monto de $1500. Dicha información es obtenida por el archivo de lotes de medios de pago Red CUT enviado al final del día. (ver FIGURA 13 **–** Transferencia A)
8. Al día siguiente de enviado el archivo de lotes al BNA, TGN enviará también al mismo banco un mail que contendrá otro archivo.

Dicho segundo archivo tendrá asociada la Orden Bancaria (la información de los movimientos detallada en el medio de pago Red CUT) de cada pago generado, con el VEP con el que se va a cancelar con dicha orden.

Por cada pago informado en este segundo archivo, BNA primero controlará que el VEP asociado al pago ya haya sido firmado en la aplicación de Banca Empresa 24; de ser así, hará una nueva transferencia bancaria por el monto del VEP, desde la cuenta puente hacia la cuenta bancaria correspondiente de la AFIP (ver FIGURA 13 **–** Transferencia B).

Esta información contenida en el archivo enviado por mail, es luego enviado por el BNA a Red Link, para que dé por pagados los VEP correspondientes e informe a la AFIP.

Proceso de Pago de OP de Aportes y Contribuciones

El pago de OP de aportes y contribuciones, que son a favor de AFIP, deben ser tratadas como las retenciones.

Se deberán realizar los mismos 7 pasos anteriormente detallados, sólo que en el paso 3, el número de VEP generado en la AFIP quedará asociado a la OP en lugar de las retenciones agrupadas en el CDR.

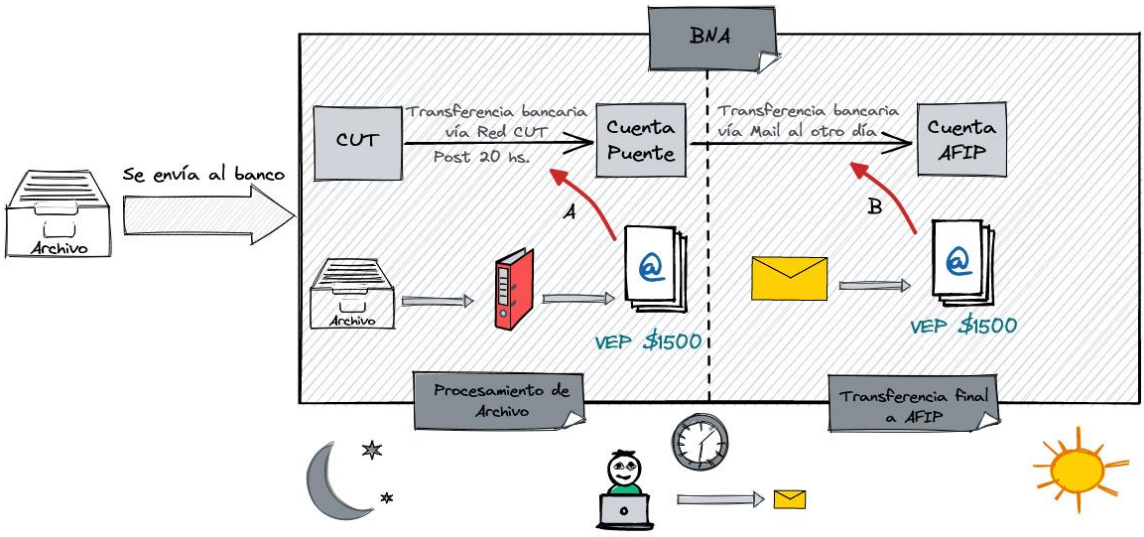


Figura 13 Pago por Red CUT

Ejemplo:

Se realiza una transferencia bancaria correspondiente a un VEP de $1500 desde la CUT a la cuenta bancaria puente. Dicha información es obtenida por el archivo de lotes de medios de pago Red CUT enviado por el e-SIDIF al BNA al final del día (**A**).

Al día siguiente desde la DGSIAF envía un mail a BNA detallando los VEP asociados a los pagos que se transfirieron desde la CUT a la Cuenta Puente, para que luego, BNA realice otra transferencia por cada VEP desde la Cuenta Puente a la Cuenta de AFIP correspondiente (**B**).

Se destaca que parte relevante de este circuito requiere la intervención manual de los organismos y del BNA, lo cual hace que en toda la operación participen varios actores, que insuma un tiempo considerable desde el inicio al fin y que sea un proceso cuestionado en las auditorías.

Cabe aclarar que, para el procesamiento de los pagos, primero se debe esperar al final del día, cuando se envía el archivo de lotes a BNA; luego hay que esperar al otro día cuando BNA termina el procesamiento vía mail.

Conclusiones

En el presente capítulo se introdujo un tipo específico de pago que implementa el e-SIDIF: El pago de OP, y de retenciones generadas por estos pagos, donde el beneficiario es AFIP y se

canalizan mediante la Cuenta Única del Tesoro a través del medio de pago “Red CUT” y cuyo pago se registra y ejecuta mediante VEP (Volante Electrónico de Pagos).

Se observó que el mencionado circuito de pago involucraba una cantidad de tiempo de procesamiento, desde el pago de la OP, pasando por el armado y envío de archivos de lotes por parte de TGN, donde se agrupan los medios de pago asociados, y por último la ejecución del pago a través de la interacción humana entre e-SIDIF-BNA-Red Link, que se llevaba a cabo recién al día siguiente de enviado el archivo.

Es decir, el viejo circuito de pago involucraba dos etapas desacopladas y que se ejecutaban en distintos momentos. Por un lado, el envío del archivo de lotes al banco al final del día y por el otro el envío del mail al banco detallando el VEP asociado a cada transferencia bancaria especificada en el medio de pago Red CUT. Además, este último paso tenía un condicionante: que ya haya sido cargado y firmado el VEP asociado en el aplicativo de Banca Empresa 24. De no ser así no se efectivizará el pago.

Por lo anteriormente expuesto, y considerando que a lo largo del mes puede haber picos de 500 pagos por día a favor de la AFIP, se presentó necesario generar un procedimiento automatizado, apoyado en buenas prácticas, para facilitar la tarea.

Capítulo 4 – Circuito de pagos a AFIP por Red Link

Introducción

Desde hace años TGN buscó mecanismos para automatizar este circuito, y a través de Red Link se encontró un socio estratégico para poder llevar adelante este requerimiento.

Se acordó avanzar en una propuesta, la cual requiere la utilización de un par de servicios API de Red LINK. Esta solución permitirá, a través de la ejecución de los servicios, realizar la consulta y los pasos de carga, firma y pago del VEP de manera automática y en un sólo paso, la cual se podrá realizar en la instancia de la emisión del medio de pago, una vez confirmado el pago (ver FIGURA 14).

De esta manera se ahorrará un tiempo considerable, que involucraba el armado de archivos de lotes con los medios de pago Red CUT, la espera hasta el final del día para ser enviado al BNA, su procesamiento y envío de los pagos a la cuenta puente de la TGN y, por último, al otro día informar nuevamente al BNA de los VEP involucrados en los pagos para que los cancelen en Red Link y los transfieran desde la cuenta puente TGN a la cuenta bancaria de la AFIP que corresponda.

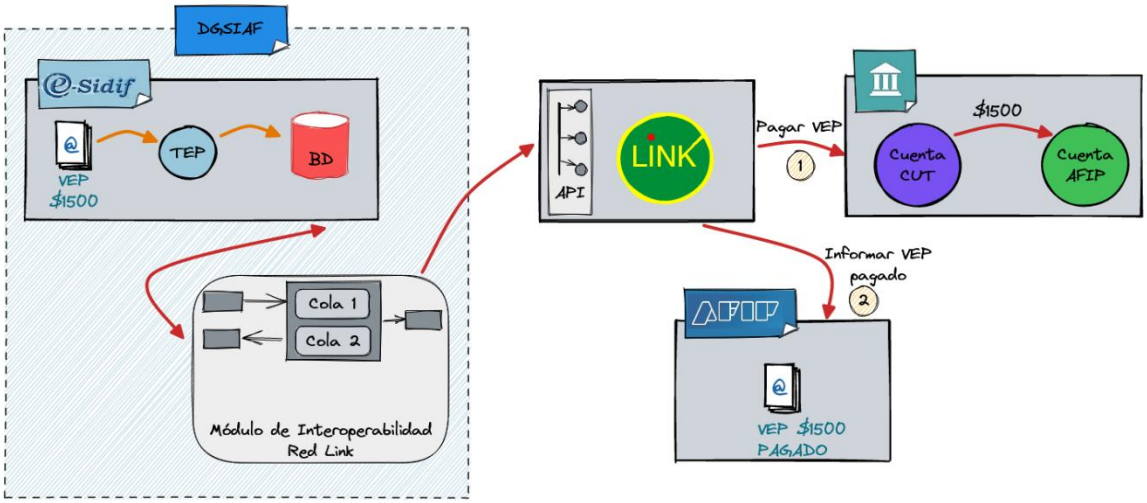


Figura 14 Pago de VEP a través de la interoperabilidad con Red Link

El nuevo circuito de pago de retenciones por VEP de AFIP se emitirá por medio de Pago Electrónico [27] en lugar del medio de pago Red CUT que venía utilizando el e-SIDIF, ya que por un lado se desconoce la cuenta bancaria destino del beneficiario AFIP (para cambiar la cuenta puente que tenía antes como destino) y por otro, se realizará de manera electrónica a través de los servicios API de Red Link, el cual se vinculará a su vez con el Sistema de Recaudación de AFIP (OSIRIS) y con BNA.

[27] Desde ahora en adelante utilizaremos el término PE para referirnos.

En la instancia de la confirmación del pago (ya sea de retenciones o de OP) se generará un medio de pago PE en estado generado.

Dicho PE deberá emitirse para efectivizar el pago de los VEP, operación que incluirá la ejecución de los servicios API de Red LINK.

Al respecto, se analizaron dos alternativas posibles para su emisión, una a través de un nuevo circuito a desarrollar, exclusivo para este tipo de operaciones electrónicas, y otra sobre la generación de archivo de lotes, agrupando los PE como si fuese una transacción en conjunto.

Dado que cada transacción de pago por VEP es individual, se definió avanzar con el diseño de un nuevo circuito para la emisión de los PE, incluyendo la invocación a los servicios API de Red LINK y contemplando que la información registrada sea suficiente como auditoría de los pagos realizados.

Modelado de la solución desde la parte del e-SIDIF

Para el modelado de la entidad que representa la nueva transferencia de pago se siguió la misma metodología usada para el diseño de los comprobantes y los medios de pago en el e-SIDIF. Dicha nueva entidad también tendrá un circuito de estados y estará asociada tanto al comprobante de pago, como también al medio de pago PE. De esta manera, cuando se crea una nueva transferencia, esta tendrá información tanto para trazar la comunicación asincrónica desplegada fuera de la aplicación, como también aquella necesaria para realizar los impactos en el modelo de pagos del e-SIDIF o revertirlos, en el caso que no se realice la transferencia. En ambos casos se actualizarán la información en la base de datos de la aplicación. Fuera del e-SIDIF, en los módulos desplegados para atender los pedidos de pago, no se persistirá ninguna información; su función será, por lo tanto, la comunicación asincrónica hacia Red Link, desacoplando dicha actividad de la desarrollada por la aplicación principal.

Transferencia Electrónica de Pagos

Parte de la solución definida del lado del e-SIDIF consiste en el desarrollo de una herramienta de tipo consulta donde se irán seleccionando los PE generados a partir del pago de las retenciones de AFIP o de OP de aportes y contribuciones y dando origen a un registro de Transmisiones Electrónicas de Pagos [28].

A través de estas herramientas, TGN accederá diariamente, las veces que lo considere necesario, para emitir los medios de pagos que se encuentren pendientes de emisión y para confirmar las TEP (ver FIGURA 15).

El primer paso a realizar requiere que TGN acceda a la nueva opción del menú denominada "**Pagos VEP-AFIP a seleccionar**", exclusiva para estas operaciones, donde el usuario podrá buscar los PE que estén generados por la confirmación de los pagos y que deben emitirse por el circuito de Pago por VEP de Red Link.

[28] De ahora en adelante utilizaremos el término TEP para referirnos.

Al emitir el PE a través de la opción “Seleccionar PE para Emitir” se generarán los siguientes impactos:

- El estado del PE se actualizará a "Emitido" y el subestado a "En Proceso Bancario".
- Se hace el registro contable del débito en la cuenta pagadora CUT por el importe pagado del VEP.
- Se generará un registro en el nuevo modelo de datos TEP, el cual tendrá un identificador unívoco por cada transferencia, asociada al medio de pago y con un estado inicial de "Pendiente".

Una vez realizada la operación, el circuito continúa con el envío de la TEP registrada.

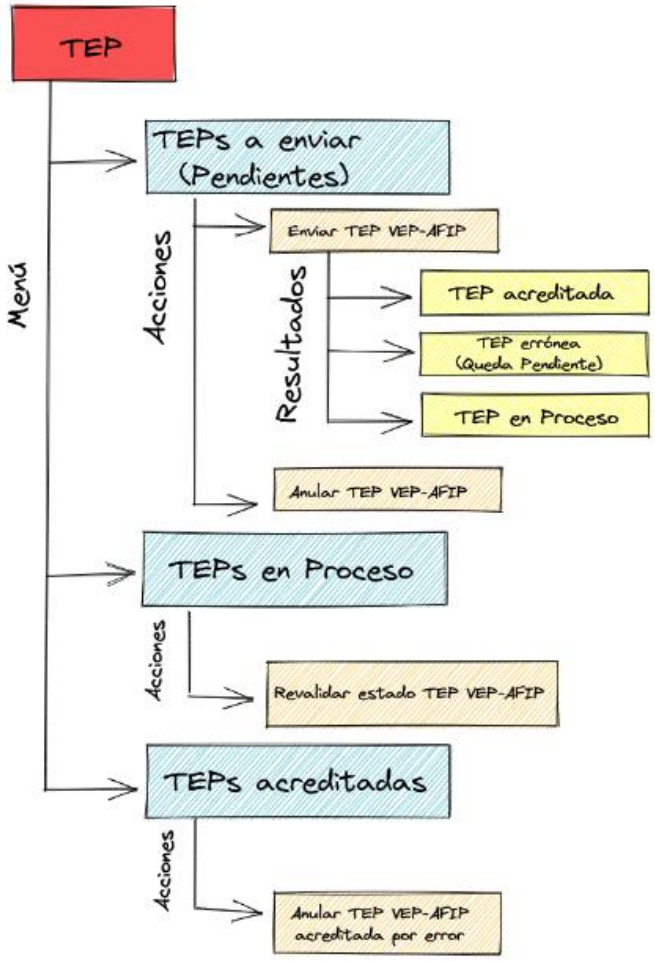


Figura 15 Organigrama de TEP en e-SIDIF

Al finalizar el día, no deberían quedar PE a la AFIP en estado "Generado", es decir, que no se hayan emitido.

Una vez seleccionado uno o varios PE que dan origen a las TEP, el circuito continuará realizando el correspondiente envío al operador de pago (Red Link).

Eventualmente, se contempla la operación de anulación de la transferencia en el caso de ser necesario.

De este modo, se incluyen las siguientes acciones sobre una TEP:

TEP VEP-AFIP a enviar

Para efectivizar el pago del VEP, requiere que TGN acceda a una nueva opción del menú "TEP VEP-AFIP a enviar", exclusiva para estas operaciones, donde visualizará el filtro de búsqueda de la nueva interfaz de consulta de "Transferencias Electrónicas de Pagos", con el objetivo de consultar aquellas que coinciden con las siguientes condiciones:

- La TEP debe estar en estado “Pendiente” y su medio de pago PE asociado en estado “emitido” y sub-estado “En Proceso Bancario”.

Al ejecutar la consulta se visualizará el resultado de la búsqueda de las TEP que cumplan con las condiciones especificadas. Para esta consulta en particular, se visualizarán los atributos propios de la transferencia electrónica como los relevantes del medio de pago.

En esta interfaz de consulta se dispondrá en el menú contextual de dos opciones para ejecutar sobre las TEP que se encuentren pendientes:

- Enviar TEP VEP-AFIP
- Anular TEP VEP-AFIP

Al finalizar el día, en esta consulta, no deberían quedar transferencias que cumplan con las condiciones requeridas, es decir pendientes de envío.

Enviar TEP VEP-AFIP

TGN deberá seleccionar en forma individual o masiva aquellas transferencias electrónicas que incluirá en este proceso de envío.

Para ello, se deberá ejecutar la opción del menú contextual denominada "Enviar TEP VEP-AFIP". Esta opción estará disponible para aquellos usuarios TGN que cuenten con el correspondiente permiso.

Al confirmar la operación de envío, por cada TEP seleccionada se realizarán las siguientes validaciones:

- El estado debe ser igual a "Pendiente".
- La fecha de generación debe ser igual a la fecha del día y a la fecha de confirmación del comprobante de pago de retenciones (PG-RET) o del Comprobante de Pago (PG).
- El estado y subestado del medio de pago deben ser igual a "Emitido" y "En Proceso Bancario" respectivamente.
- La fecha de generación del medio de pago debe ser igual a la fecha del día.
- El medio de pago debe tener un VEP asociado, el cual debe estar en estado "Confirmado" y su importe debe coincidir con el del medio de pago.
- El VEP no debe estar expirado, es decir que su fecha de expiración debe ser igual o mayor a la fecha del día.

Si las validaciones son satisfactorias, se procederá a encolar la información del VEP a pagar en un registro de envío para su posterior transmisión automática a Red Link, a través de una

Componente de Interoperabilidad entre sistemas (e-SIDIF / Red Link) desarrollada para tal fin. En caso contrario, la transferencia permanecerá en el estado de origen ("Pendiente") y se informará el correspondiente mensaje de error.

De continuar con la operación, las acciones a realizar son las siguientes:

1. Por parte del e-SIDIF: Transferencia Electrónica En Proceso

   Alta de la información del VEP a pagar en un registro de tabla del e-SIDIF para luego consumirla y encolarla en una la Cola de Envío TEP.

   Se actualiza la fecha de envío y el estado de la TEP a “En Proceso”.

   La operación de Enviar TEP VEP-AFIP desde el e-SIDIF, a nivel funcional, finaliza en esta instancia. Es decir, que al usuario de TGN se le libera la sesión en e-SIDIF y podrá continuar con otras operaciones.

   Mientras tanto, el pago de VEP está en proceso de envió y de manera asincrónica (diferido en segundos) se ejecutarán automáticamente los Servicios de Consultar y Pagar VEP de Red Link, acción que será realizada por la Componente de Interoperabilidad entre sistemas (e-SIDIF / Red Link).

   Finalmente, el resultado obtenido se actualizará en forma automática en e-SIDIF.

2. Por parte de la componente de interoperabilidad entre sistemas

   Esta componente tiene como fin la administración de los envíos de las TEP originadas en el e-SIDIF hacia Red Link y el procesamiento del resultado.

   Todo el proceso será administrado por dicha componente y de manera automática, sin intervención de usuarios del e-SIDIF.

   Por cada TEP registrada en la cola de envío, la componente procederá a ejecutar los servicios API de Red Link, quién para este circuito definió dos servicios, el primero para consultar el VEP a pagar y el segundo para pagarlo.

   La invocación a ambos servicios API de Red Link será individual por cada transferencia, es decir por cada VEP y por ende por cada PE asociado.

   Al ejecutarse el servicio de "Consultar VEP" en forma satisfactoria, se podrá ejecutar el de "Pagar VEP", ya que este último requiere parámetros de entrada que retorna el primero.

Servicios de consulta y pago de la API de Red Link:

- Verificar la existencia del VEP en la Entidad de Pago (Red Link), acción que se realiza en forma automática con el token obtenido al ejecutar previamente el Consultar VEP.
- Pago del VEP informando la novedad al sistema OSIRIS (AFIP) y por ende la acreditación del importe pagado del VEP en la correspondiente Cuenta Bancaria de AFIP.
- Débito en la Cuenta Pagadora CUT (11.XX.XXXX/19) por el importe pagado del VEP.

- Devolución de la respuesta confirmando el pago, junto con la información requerida para impactar en el e-SIDIF, incluyendo el número de comprobante bancario o de transacción para el registro contable.

En caso de un error, dentro de los considerados conocidos, en alguno de los dos servicios API de Red Link se retornará un código con el mensaje de lo ocurrido. De esta manera, la transferencia permanecerá en el estado de origen ("Pendiente") para su análisis y posterior decisión de volver a enviarla o anularla.

El impacto en el e-SIDIF dependerá del resultado del servicio de Red Link, pudiendo darse 3 escenarios: "Acreditada", "Errónea" y "En Proceso".

1. Transferencia Electrónica Acreditada

   En este caso se realizarán las siguientes acciones:
   - Actualización de la TEP:
     - Su estado cambia a “Acreditada”.
     - Se completa la fecha de pago con la fecha y hora retornada por el servicio de Pagar VEP de Red Link.
     - Se completa el código de respuesta con el obtenido en el servicio de Pagar VEP de Red Link.
     - Se completa el número de secuencia con el obtenido en el servicio de Consultar VEP de Red Link.
   - Confirmación del medio de pago electrónico (PE) pasando a estado “acreditado”.
     - La transferencia electrónica será inmediata, por lo tanto, no habrá posibilidad de revertirla una vez informada la acreditación del pago por Red Link.
     - El estado del VEP del e-SIDIF no se actualizará en esta instancia, ya que no se considera necesario, dado que el mismo será actualizado a través de un proceso automático existente que se ejecuta diariamente por la noche.
2. Transferencia Electrónica Errónea

   En este caso se realizarán las siguientes acciones:

   - Actualización de la TEP:
     - Su estado vuelve de “En Proceso” a “Pendiente” para que TGN analice el error y determine si se puede volver a enviar o se deberá anular.
     - Se completa el código de respuesta con el obtenido del último servicio de Red Link ejecutado (Consultar/Pagar VEP).

   Este escenario es de baja probabilidad dado que lo esperado es que las transferencias por este circuito sean acreditadas.

3. Transferencia Electrónica En Proceso

   Este escenario es particular dado que se espera que, al finalizar la operación de envío, el servicio de Red Link de como respuesta que la transferencia fue acreditada (lo esperado) o eventualmente errónea. En estos dos casos se realizarán los impactos tal lo descripto anteriormente. Podría darse por algún motivo excepcional, como por ejemplo que se

pierde la comunicación con el servicio API de Red Link y por ende se desconocerá si la operación fue exitosa (acreditada) o finalizó con error (errónea).

De esta manera, la transferencia electrónica de pago quedará temporalmente en estado "En Proceso" hasta que se determine su estado real.

En forma automática, el sistema ejecutará un proceso a última hora del día que verificará todas las TEP en proceso que se encuentren pendientes de respuesta por parte del servicio API de Red Link.

Si bien se estimó que este caso sería de baja probabilidad, se determinó que, si la acreditación se detectaba al día siguiente, los impactos se realizarían con fecha distinta al de realizado el pago, lo cual sería inconsistente.

TGN indicó que se debía garantizar la consistencia de fecha de pago y fecha de impactos, por lo que la solución elegida fue asumir que el pago fue acreditado, aunque se desconozca, ejecutando anticipadamente los impactos en el mismo día que fue realizado, tal como si se hubiese recibido la acreditación por parte del servicio API de Red Link, y notificando a los usuarios responsables en TGN de que la TEP fue acreditada automáticamente.

Una vez que se tenga la certeza de que fue acreditado, se deberá ejecutar alguna acción adicional para ajustar la información de la fecha y hora real de pago, junto con el número de comprobante bancario, según lo indicado por el servicio de Consultar VEP de Red Link. En el caso contrario, es decir que se confirmó que el mismo no fue pagado y está pendiente, TGN deberá realizar una "Anulación de Pago Confirmado por acreditación errónea".

Por otro lado, se descarta que Red Link procese un pago de VEP diferido, es decir que se envió la transferencia un día y que quede pendiente de procesamiento por el operador de pago, y este lo ejecute al día siguiente.

Anular TEP VEP-AFIP

Desde la misma interfaz, se dispondrá de una opción en el menú contextual denominada "Anular TEP VEP-AFIP". Está opción estará disponible no sólo para el caso descripto anteriormente sino también para cualquier otro caso donde TGN determine la anulación de la transferencia electrónica de pago.

De confirmarse la operación se realizarán los siguientes impactos:

- Actualización de la TEP de estado “Pendiente” a “Anulada”.
- Actualización del PE a estado y sub-estado, ambos a “Generado”.
- Anulación de la confirmación del PE, impactando en la cuenta pagadora CUT por el importe del VEP.

TEP VEP-AFIP en proceso

Esta opción tiene el fin de monitorear y controlar que no queden transferencias en proceso.

Tal lo descripto anteriormente, se considera baja la probabilidad de que las transferencias enviadas queden en este estado. Además, en caso de que se den algunos casos, los mismos se irán actualizando de manera automática con el resultado final a través del proceso mencionado que se ejecuta a última hora del día.

Igualmente, a través de esta interfaz, TGN podrá revalidar el estado de forma manual.

Para ello se requiere que TGN acceda a la nueva opción del menú "TEP VEP-AFIP en proceso", exclusiva para estas operaciones.

En esta interfaz de consulta se dispondrá de una acción denominada "Revalidar estado TEP VEP-AFIP". La misma estará disponible para aquellos usuarios TGN que cuenten con el correspondiente permiso.

De confirmarse la operación se procederá a ejecutar el servicio API "Consultar VEP" de Red Link. Esta ejecución se realizará de manera asincrónica delegando la responsabilidad a la Componente de Interoperabilidad entre sistemas (e-SIDIF / Red Link).

Si la consulta del servicio es exitosa, se recibirá como respuesta dos posibles escenarios, que se incluya en la respuesta información del comprobante de pago de Red Link, con lo cual se asumirá que fue acreditada, o que no venga esa información, lo que significará que no fue pagada. Para ambos casos, se ejecutarán en e-SIDIF los impactos tal lo descripto anteriormente, tanto para Acreditada como Errónea.

Si la consulta devuelve error o no se puede determinar con exactitud el estado de la transferencia en Red Link, quedará en el mismo estado de origen y será necesario revalidarla nuevamente hasta que se obtenga uno de los dos escenarios esperados.

Vale destacar que la misma componente de interoperabilidad monitoreará automáticamente durante todo el día que no queden transferencias en proceso y de encontrar, ejecutará la consulta a Red Link en forma automática. De esta manera, la acción manual por parte del Usuario de TGN debería ser acotada y bajo situaciones especiales.

TEP VEP-AFIP acreditadas

Esta opción tiene el fin de consultar aquellas transferencias acreditadas.

En esta interfaz de consulta se dispondrá de una opción denominada "Anular TEP VEP-AFIP acreditada". La opción estará disponible para aquellos usuarios TGN que cuenten con el correspondiente permiso.

Anular TEP VEP-AFIP acreditada por error

Esta opción es especial y para ser ejecutada en ocasiones de contingencia y bajo ciertas precondiciones.

Se destaca que un pago confirmado por Red Link bajo el circuito esperado es inmediato y no admite la vuelta atrás, por lo tanto, no tendría utilidad esta opción, e incluso tendría un efecto no deseado si se termina anulando una transferencia que fue acreditada.

Sin embargo, la presente opción será de utilidad para la contingencia en el caso que se haya aplicado un acreditado por defecto previo al cerrar el día, porque no se obtuvo respuesta de Red Link. Además, al día siguiente se determina que el pago del VEP no había sido efectuado por Red Link.

De avanzar se realizarán las siguientes acciones:

- Actualización de la TEP:
    - Su estado vuelve a “Pendiente”.
    - Se actualiza la fecha de pago en nulo.
- Se enviará un mail a los referentes de Pagos de TGN informando que se anuló una acreditación por defecto por la falta de respuesta de Red Link, dando por finalizado esta instancia del proceso.

Para concluir con la anulación total del pago, implicará que en forma manual por e-SIDIF, TGN anule la TEP y luego anule el Pago (PG-RET o PG OP).

Red Link

Red Link es una empresa de capitales argentinos creada en 1988, con el objetivo de brindar servicios informáticos a las principales entidades financieras, tarjetas de crédito y otros clientes en áreas de cajeros automáticos, home banking, banca empresas, soluciones mobile y de seguridad, procesamiento de información y recaudaciones de servicios e impuestos. Además, provee a comunicación mediante API a diferentes aplicaciones de negocios financieros. (Link, s.f.)

Las API son un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo que nuestros productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados.

Además, posee otros beneficios como son el ahorro de recursos, migrar la gestión de una aplicación empresarial a una tecnología digital de manera más fácil y permitir la automatización de procesos e implementaciones.

Servicio APILINK

APILINK es un producto desarrollado por RED LINK donde se exponen las principales API financieras con máxima seguridad tanto en la integración como en la operatoria, para poder de esta manera hacer uso tanto de API de consulta como transaccionales, que exigen cierto control, como son transferencias inmediatas o pagos a AFIP. (APILink, s.f.)

Entre las distintas API provistas por Red Link se encuentran:

De consulta:

- Consulta de cajeros (geolocalización de cajeros automáticos en nuestra aplicación).
- Consulta de titularidad (verificar la titularidad de un CBU).
- Consulta de pagos (otorga a aquellos entes que operan con la plataforma PAGAR - sistema provisto por Red Link que permite abonar impuestos, servicios y otras obligaciones a

través de diversos canales, pero centralizados en un solo portal- consultar deudas, pagos y consulta de código de entes).

Transaccionales:

- Transferencias interoperables (para empresas que desean realizar transferencias a un CBU o CVU).
- Orden de extracción sin tarjeta (realizar envíos de dinero para pago a proveedores, viáticos, etc).
- Transferencias bancarias inmediatas (pagos a proveedores o envío de dinero entre cuentas bancarias).
- E-Cheq (simplificación en el manejo de cheques electrónicos para agilizar el sistema de cobros y pagos).
- Debin Bancos y Credin Bancos (para que una entidad financiera realice débitos y créditos respectivamente con interoperabilidad – CBU y CVU – incluyendo el plazo fijo).
- Realizar pagos a AFIP.

Para hacer uso de alguna de las API provistas desde nuestra aplicación se debe ingresar al producto de la API de Autenticación, para suscribirte al plan de la misma. Una vez suscripto, se podrá empezar a probar la generación de tokens desde el portal de API o desde alguna aplicación externa de prueba de servicios REST, por ejemplo, SoapUI o Postman.

Lo primero que se debe hacer es registrarse en el portal de Red Link e iniciar sesión. Luego podremos explorar las API LINK disponibles y elegir la que deseemos para nuestra aplicación; para lo cual, como paso siguiente, debemos registrarla. Esto nos permite obtener un ID de cliente y un secreto de cliente.

El ID de cliente es el ID público asociado a nuestra aplicación. Se enviará en cada solicitud para saber que es uno quien efectúa el llamado.

El secreto de cliente es el ID privado que tendrá nuestra aplicación. Nos permite verificar nuestra identidad en el paso de autenticación de las API de Red Link.

Para hacer pruebas con las API, se debe autenticar contra su API de Autenticación. Cada operación que se realice tendrá un token de seguridad asociado para la autorización.

Servicios APILINK en e-SIDIF

En e-SIDIF se utilizan los servicios de consulta y pago de VEP provistos por la API de Pagos a AFIP disponibles en el catálogo de API del sitio web de Red Link.

La API Pago de VEP, permite realizar la consulta y pago de un VEP generado. Uno de los requisitos para comenzar a consumirla, como se mencionó, es suscribirse a la API de Autenticación en donde se genera el token que permite operar con la misma.

Una vez realizada la autenticación se obtendrá un token de acceso, el cual se utilizará al ejecutar los servicios API de "Consultar VEP" y "Pagar VEP". (Armentano, 10/11/2021)

Servicio Consultar VEP

Este servicio te permitirá consultar el estado de un VEP.

Comando

- `Consultar VEP (GET/veps/{numeroVep})`

Respuestas:

- `PENDIENTE_DE_PAGO`
- `PAGADO`
- `ERROR`
- `DESCONOCIDO`

Si el VEP se encuentra en estado "PENDIENTE_DE_PAGO" se harán disponibles los datos del VEP junto al token para realizar el pago (ver TABLA 1). Si el VEP se encuentra en estado "PAGADO", se brindará el comprobante de pago adjunto y además se informará un número de secuencia generado en Red Link en base al número de comprobante bancario, que servirá para trazar el pago en el e-SIDIF y realizar los impactos correspondientes (ver TABLA 2).

Otro estado presente en un VEP es el estado "DESCONOCIDO" que se presenta cuando hubo algún inconveniente con la solicitud y no es posible validar el estado del VEP. por lo que se recomienda hacer la consulta nuevamente.

Tabla 1 Respuesta OK - Pendiente de pago

Nombre Estado: **"PENDIENTE_DE_PAGO_"**	**Descripción** **Estado del VEP Consultado**
Fecha Creación	Fecha en que se realizó la consulta
Fecha Expiración	Fecha Expiración
Descripción	Descripción
CUIT Autorizante	CUIT del Autorizante
CUIT Contribuyente	CUIT del Contribuyente
Importe	Importe del VEP
Impuesto	Tipo de impuesto
Token	Token para realizar el Pago

Tabla 2 Respuesta OK - Pagado

Nombre Estado: **"PAGADO"**	**Descripción** **Estado del VEP Consultado**
Fecha Creación	Fecha Creación del VEP
Descripción	Descripción del pago

CUIT Autorizante	CUIT autorizante
CUIT Contribuyente	CUIT contribuyente del VEP
Importe	Importe del VEP
Numero Secuencia	Número de la secuencia del Pago
Fecha Pago	Fecha de pago
Hora Pago	Hora del pago

Descripción detalla de la respuesta ERROR

Durante la operatoria, se pueden presentar diferentes comportamientos del sistema mientras se está operando.

Código de error	Código de error	Descripción del error
400	400	Se presenta por el ingreso de un parámetro con un formato no admitido por el sistema, por no enviar un parámetro requerido, el ingreso de un Id Requerimiento ya utilizado, por realizar la consulta de un VEP inexistente, el envío de parámetros erróneos, por la falta de datos de la empresa en nuestros servicios internos o la consulta de un VEP generado para un contribuyente no adherido.
401	401	Se presenta cuando El X-Ibm-ClientId enviado en el request no está asociado al producto de la API o el mismo no se encuentra habilitado.
403	403	Se presenta cuando el sistema identifica que se realiza una operación en donde el Scope del token no contempla la API asociada.
500	500	Se presenta cuando hubo un error interno en el tratamiento del pedido.

Servicio Pago de VEP

Este servicio permite realizar el pago de un VEP consultado anteriormente.

Comando

- `Pagar VEP (POST/veps/{numeroVep}/pago)`

Respuesta

- `OK (ver TABLA 3)`
- `ERROR (ver TABLA 4)`

Es obligatorio realizar la consulta previamente para poder obtener el token de pago. El token tiene un tiempo de vida de 48 Hrs. Si se quiere realizar el pago posteriormente al tiempo establecido, el mismo arrojará un error. Una vez realizado el pago, se habilitará la información asociada al comprobante de pago.

Tabla 3 Descripción detalla de la respuesta OK

Fecha Creación	**Fecha Creación del VEP**
Descripción	Descripción del pago
CUIT Autorizante	CUIT autorizante del pago
CUIT Contribuyente	CUIT contribuyente del VEP
Importe	Importe del VEP
Numero Secuencia	Número de la secuencia del Pago
Fecha Pago	Fecha de pago
Hora Pago	Hora del pago
Estado Pago	Estado del pago

Al igual que para la operatoria de consultar VEP, se pueden presentar diferentes comportamientos del sistema mientras se está operando en el pago.

Además de reportar una respuesta exitosa, se pueden presentar diferentes tipos de errores que imposibilitan continuar con el flujo de la operación:

Tabla 4 Descripción detalla de la respuesta ERROR

Código de error	Código de error	Descripción del error
400	400	Se presenta por el ingreso de un parámetro con un formato no admitido por el sistema, el ingreso de un Id Requerimiento ya utilizado, por no enviar un parámetro requerido, por realizar la consulta de un VEP inexistente, el envío de parámetros erróneos, el uso de un token caducado o por la falta de datos de la empresa en nuestros servicios internos.
401	401	Se presenta cuando El X-Ibm-ClientId enviado por la Empresa no está asociado al producto de la API o el mismo no se encuentra habilitado.
403	403	Se presenta cuando el Sistema identifica que se realiza una operación en donde el Scope del token no contempla la API asociada.
500	500	Se presenta cuando hubo un error interno en el tratamiento del pedido.

Para implementar esta interoperabilidad con Red Link, se requirieron ajustes a nivel técnico y de arquitectura del e-SIDIF. Adicionalmente, se necesitaron habilitaciones de red y en el proxy utilizado en el ministerio, para que el e-SIDIF pueda llegar a los servidores de Red Link.

Conclusiones

Después de un análisis profundo y relevamiento se llegó a la conclusión que el circuito de pagos a la AFIP por VEP podía ser realizado a través de la interacción del sistema e-SIDIF con la API de Red Link.

Antes de comenzar este nuevo desarrollo, se verificó que la API otorgara todo lo necesario para efectivizar los pagos realizados por el e-SIDIF en el contexto mencionado, analizando los datos que requería Red Link para ello y los que se podían enviar desde el e-SIDIF.

Luego de un análisis del contrato de la API y de los datos que nos podía retornar la consulta y el pago de VEP, se llegó a la conclusión que dicha API satisfacía las demandas del nuevo circuito y podía comenzar a utilizarse en el nuevo desarrollo.

Para llevar a cabo esta nueva tarea se requirió interacción humana y funcional entre las partes, creando un modelado nuevo de TEP en nuestro sistema que sería procesado por un módulo de interoperabilidad entre e-SIDIF y Red Link, modelado por el equipo de desarrollo.

Primero se explicó el modelado de la solución de la transferencia electrónica de pagos desde el lado del e-SIDIF.

Por último, en el capítulo se dio una introducción breve de la historia de Red Link operando en Argentina, para luego adentrarnos en sus API de programación que publican para que otros sistemas puedan interactuar con éste y llevar a cabo sus tareas relacionadas.

Se analizaron las dos operaciones que publica dicha interfaz de programación, que son la consulta y el pago de VEP y que serían las utilizadas por las TEP modeladas en el e-SIDIF a través de los módulos de interoperabilidad.

Capítulo 5 – Pagos de OP y Retenciones automatizados mediante Red Link

Introducción

e-SIDIF se comunica con módulos de interoperabilidad desarrollados especialmente como intermediario entre este sistema y los servicios provistos por el Sistema Red Link. Dichos módulos se componen de un transmisor de pagos, que analizara de forma constante los pedidos de pagos solicitados desde e-SIDIF y los deja en una cola de mensajes para que estén al alcance de los servicios de pagos provistos por Red Link. Cualquiera sea la respuesta que retorne del pedido de pago, se encolará y luego será comunicada al e-SIDIF. Este proceso de encolamiento es asincrónico, dejando a la TGN libre para seguir con sus tareas en el e-SIDIF.

Arquitecturas tecnológicas y patrones de diseño

El equipo de Arquitectura de software del proyecto, después de evaluar varias opciones de desarrollo, decidió que el módulo de interoperabilidad que interactuaría con la aplicación e-SIDIF y con Red Link estaría desplegado, basándose en la metodología de microservicios, en Optimized Kubernetes Distribution [29] – Versión 4, que es la versión gratuita de Openshift.

Las solicitudes de pagos efectuadas por el sistema e-SIDIF se dejarán como registros en una tabla, que luego consumirá el módulo Transmisor de Pagos desplegado en OKD. Dicha solución se apoyó en el patrón de diseño "Transactional Outbox" (ver FIGURA 16).

Por otro lado, para encolar los mensajes de ida con los pedidos de pagos consumidos de la tabla anterior, como para encolar los mensajes de respuesta de Red Link a e-SIDIF, se decidió utilizar Rabbit como gestor de cola de mensajes.

En cuanto a la comunicación entre el e-SIDIF y los módulos de arquitectura que proveen los microservicios, y entre estos últimos y Red Link, se decidió apoyarse en el protocolo REST; más específicamente en la implementación OpenFeign.

Para las pruebas de invocación de operaciones de la API Red Link se utilizó una herramienta denominada Postman (Postman, s.f.), que nos permite realizar pruebas "Mock" [30]. Es un cliente HTTP que nos da la posibilidad de testear *'HTTP requests'* a través de una interfaz gráfica de usuario, por medio de la cual obtendremos diferentes tipos de respuesta que posteriormente deberán ser validados.

[29] Desde ahora en adelante utilizaremos el término OKD para referirnos.
[30] Pruebas que imitan el comportamiento de objetos reales de una forma controlada.

Transactional Outbox Pattern

La cuestión principal que nos lleva a implementar este patrón de diseño no es si una determinada base de datos garantiza o no la atomicidad en sus transacciones (lógicamente, debería serlo para minimizar el riesgo de inconsistencias), sino la distribución de las operaciones en un único flujo de trabajo (creación completa del pedido con todas sus operaciones) en varias transacciones locales independientes entre los diferentes contextos o módulos y la base de datos. (Cuéllar, 2020)

Normalmente se notifica el evento una vez realizado la transacción local. Para ello, se ejecuta la transacción y a continuación se publica el evento a nuestro *Message-Broker*.

El problema es que la transacción y el envío del evento no son atómicos: se ejecutan independientemente pudiendo provocar posibles inconsistencias de datos si alguno de ambos falla.

El patrón de diseño *Bandeja de salida transaccional (Transactional Outbox Pattern)* nos permitirá una única transacción atómica garantizando una entrega "At-Least-Once" y dándonos la posibilidad de reprocesarlos en cualquier momento.

Esto se logra guardando en la misma transacción tanto la operación que queremos realizar (que garantice ACID – *atomicity, consistency, isolation, durability*), como los eventos que pueda generar. Los eventos quedarán persistidos en la base de datos.

Mediante el patrón de diseño "***Polling Publisher***" [31] publicaríamos los eventos a nuestro *message-broker* (Richardson, Polling Publisher Pattern, s.f.). Manteniendo el estado de los eventos en el outbox comprobando el ACK [32] en el momento de la publicación (asegurando así la recepción del evento). O incluso, el consumidor podría actualizar el evento a "procesado correctamente".

Transactional Outbox Pattern aplicado en nuestro desarrollo:

Dicho patrón garantiza que los eventos se guardan en un almacén de datos (normalmente en una tabla de bandeja de salida de la base de datos) antes de insertarse en última instancia en un agente de mensajes, como en nuestro caso, el gestor de cola de mensajes Rabbit. Si el objeto empresarial y los eventos correspondientes se guardan en la misma transacción de base de datos, se garantiza que no se perderá ningún dato. Todo se confirmará, o todo se revertirá si se produce un error. Para publicar el evento, un proceso de trabajo o servicio diferente consulta la tabla de bandeja de salida en busca de eventos no leídos, publica los eventos y los marca como procesados; dicho proceso lo ejecuta el módulo Transmisor de Pagos desplegado en microservicio y correspondiente a "***Message relay*"** del Pattern. Este patrón garantiza que los eventos no se pierdan después de crear o modificar un objeto empresarial.

[31] Patrón de diseño contenido dentro de Transactional Outbox Pattern, cuya función es sondear la tabla de outbox y publicar los mensajes en un broker de mensajería.

[32] ACK: acknowledgement. Es un mensaje que el destino de la comunicación envía al origen de esta para confirmar la recepción de un mensaje.

En el caso de la interoperabilidad con Red Link, las solicitudes de pago serán depositadas como registros en una tabla de la Base de Datos del e-SIDIF; dicho registro cuenta con una columna denominada "Estado", que servirá para que, a la vuelta del circuito de pago, se actualice dependiendo del resultado final; al mismo tiempo se hacen los impactos derivados de los pagos en la misma base, antes de encolar el pedido. Luego el módulo Transmisor de Pagos leerá dicho registro y encolará el pedido en el módulo donde se encuentra el gestor de mensajes de Rabbit (***"Message relay"*** del Pattern).

Luego de que el pago fue procesado por Red Link, existe un módulo también implementado con microservicios, que notificará al Sistema e-SIDIF si éste fue realizado correctamente o no. Dicha notificación se realizará actualizando el estado del registro encolado, que representa el evento, al hacer el pedido de pago.

Una vez conocido el resultado, se efectivizarán los impactos si el pago fue realizado, o en caso contrario, se revertirán.

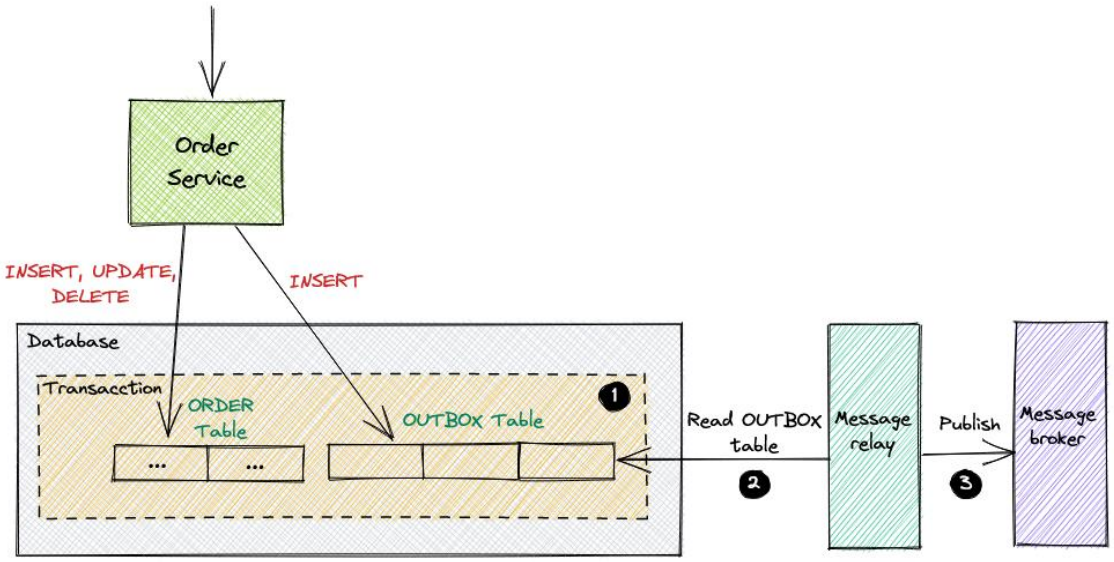


Figura 16 Transactional Outbox Pattern - extraída de (Richardson, Transactional messaging, 2018)

Software monolítico Vs. Microservicios

La aplicación e-SIDIF fue desarrollada y es mantenida sobre una arquitectura de software monolítica (ver FIGURA 17).

Las bases conceptuales y tecnológicas que guiaron la construcción del e-SIDIF fueron definidas en los años 2004 – 2005. Cuando se pensó la actual generación del SIAF (hace más de 15 años) se tomó como punto de partida una plataforma de n capas, con no más de 20 tecnologías; hoy la industria nos presenta un número mayor de tecnologías posibles, de más de 7 mil, lo cual dio un marco propicio para impulsar una transformación.

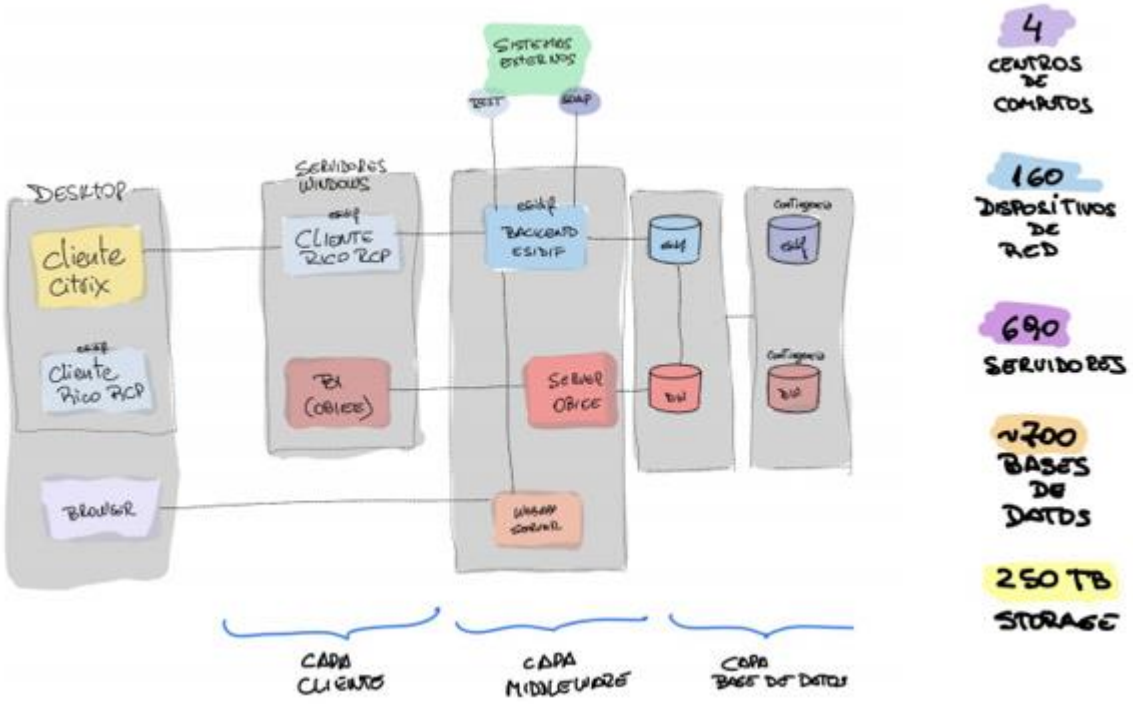


Figura 17 Arquitectura de varias capas en e-SIDIF extraída de (Beccaria, 2022)

La tecnología del e-SIDIF va rápidamente camino a la obsolescencia.

Algunos de sus problemas están relacionados con el Cliente Rico RCP (Rich Client Plattform) cuya tecnología de desarrollo quedó ya casi obsoleta; lo cual hace que las actualizaciones del S.O. pueden provocar problemas visuales.

Otro problema está relacionado con su arquitectura monolítica. Una aplicación de arquitectura monolítica como es el e-SIDIF se compila, versiona y despliega como una sola unidad unificada (ver FIGURA 18). Es decir, es implementada como un solo conjunto de librerías y archivos de configuración en un entorno de tiempo de ejecución; está enredada con este último y con las actualizaciones o parches aplicados al SO base, lo cual puede interrumpir la aplicación.

Para realizar cambios en este tipo de aplicación se requiere modificar el código base y compilar e implementar una versión actualizada de la interfaz del lado del servicio. Esto hace que las actualizaciones sean restrictivas y lentas. (Beccaria, 2022)

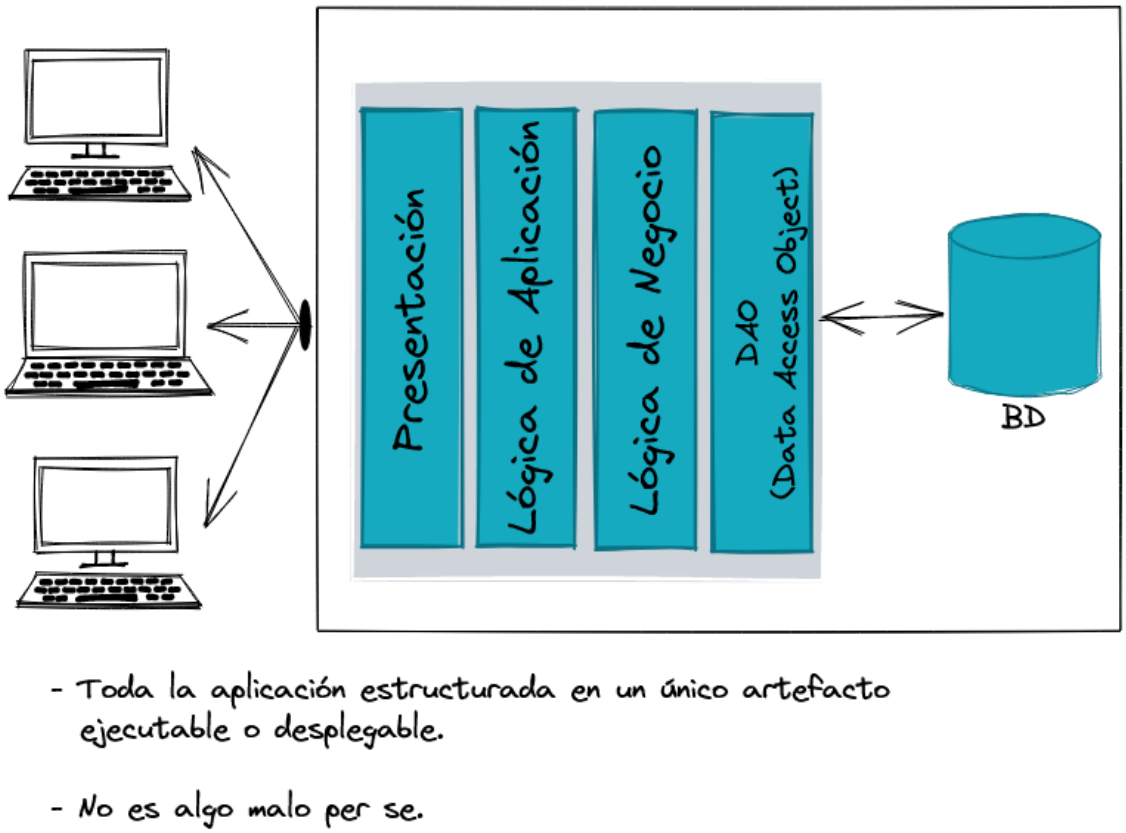


Figura 18 Aplicación monolítica e-SIDIF

Como ventajas podemos mencionar las siguientes: (Harris, 2022)

- **Implementación sencilla**: Un solo archivo o directorio ejecutable facilita la implementación.
- **Desarrollo**: Es más fácil desarrollar aplicaciones compiladas con un único código base.
- **Rendimiento:** En una base de código y un repositorio centralizados, una API de servicios estará centralizada, en lugar del desarrollo de servicios en un sistema distribuido.
- **Pruebas simplificadas:** Una aplicación monolítica es una unidad única y centralizada, por lo que las pruebas integrales se pueden hacer más rápido que con una aplicación distribuida.
- **Depuración sencilla:** Con todo el código ubicado en un solo lugar, es más fácil rastrear las solicitudes y localizar incidencias.
- **Testing directo:** Se puede hacer testing end-to-end invocando la capa de servicios.
- **Despliegue directo:** Se copia el war y se despliega en el Jboss.
- **Simple manera de escalar:** Múltiples instancias de la misma aplicación detrás de un balanceador.

Desventajas que posee una arquitectura monolítica:

Las aplicaciones monolíticas pueden ser muy eficientes hasta que crecen demasiado y la escalabilidad se convierte en un desafío. Hacer pequeños cambios en una aplicación requiere que se compile y pruebe toda la plataforma, lo que va en contra de los conceptos ágiles que prefieren los desarrolladores hoy en día.

Estas son algunas de las desventajas de una arquitectura monolítica (Harris, 2022):

- **Velocidad de desarrollo lenta:** Con una aplicación grande y monolítica, el desarrollo es más complejo y lento (edit-build-run-test loop).
- **Escalabilidad:** No se pueden escalar componentes individuales.
- **Flexibilidad:** Si hay algún error en algún módulo o componente de software, puede afectar a la disponibilidad de toda la aplicación.
- **Barrera para la adopción de tecnología:** Cualquier cambio en el lenguaje afecta a toda la aplicación, lo que hace que los cambios suelan ser costosos y lentos.
- **Estancamiento tecnológico**: difícil de actualizar.
- **Falta de flexibilidad:** Las aplicaciones monolíticas están limitadas por las tecnologías que utilizan.
- **Implementación:** Un pequeño cambio en una aplicación monolítica requiere una nueva implementación de toda la aplicación. El camino desde el commit hasta el deploy es largo y difícil.
- Incompatibilidad con la nube o “Cloud Computing”.

En los últimos años se empezó a reemplazar el desarrollo de aplicaciones en estructuras monolíticas, pasando a la “Computación en la nube”, que es una estructura distribuida sobre Internet.

Éste es un nuevo modelo de prestación de servicios tecnológicos a través de internet sin necesidad de descargas. Dicho modelo se apoya en infraestructuras tecnológicas dinámicas, caracterizadas por la virtualización de recursos, un alto grado de automatización y una elevada capacidad de adaptación para atender demandas variables.

Esta computación puede ser de 3 tipos:

- **SaaS** (Software como servicio): Se provee la aplicación final ya desarrollada. Ej: Gmail.
- **PaaS** (Plataforma como servicio): Crea un entorno de desarrollo, herramientas predefinidas y API listas para utilizar. Ej: OpenShift de Red Hat.
- **IaaS** (Infraestructura como servicio): Ofrece hardware en la red, pero virtual. Provee los servidores y almacenamiento. Ej: Amazon Web Services.

Un producto de PaaS como es Openshift, nos da las herramientas necesarias para el desarrollo de aplicaciones no monolíticas, que cooperaran con el e-SIDIF para la prestación de determinados servicios, apoyándonos en las ventajas de los **microservicios**.

Los microservicios son un componente central de este tipo de computación.

Dividir las aplicaciones en partes pequeñas y holgadamente acopladas facilita a los desarrolladores la creación de software ágil y resistente. Además, los microservicios acortan los ciclos de desarrollo, lo que conduce a una innovación más rápida y a una mejor experiencia de usuario.

Los microservicios nativos en la nube se refieren a una estrategia de diseño de aplicaciones en la que los desarrolladores dividen las aplicaciones en una serie de unidades discretas, llamadas microservicios. Cada microservicio suele funcionar independientemente de los demás, pero los microservicios comparten datos e interactúan a través de una red para permitir la funcionalidad de la aplicación.

Las arquitecturas de microservicios existen desde hace más tiempo que la *"Computación en la nube"*. Los microservicios empezaron a hacerse populares hace una década, mientras que el término *cloud-native* surgió alrededor de 2015. Parte de la razón por la que los desarrolladores conceptualizaron lo nativo de nube como un enfoque distinto para el diseño y la entrega de aplicaciones fue porque muchas aplicaciones estaban migrando a una arquitectura de microservicios. (Tozzi, 2022)

Ser nativo de nube es más que solo microservicios: la infraestructura distribuida y los servicios consumibles son también partes importantes de la ecuación, como se vio en los distintos tipos de computación (SaaS, PaaS y IaaS). Sin embargo, los microservicios son posiblemente el elemento más importante de una estrategia de este tipo.

Los desarrolladores pueden utilizar plataformas como Kubernetes y OpenShift para desplegarlos en las instalaciones, como se hizo en la aplicación e-SIDIF para los nuevos servicios pedidos que luego se explicarán en detalle.

Como ventajas en el uso de microservicios se pueden mencionar las siguientes: (Harris, 2022)

- **Escalado flexible**: si un microservicio está alcanzando su capacidad de carga, se pueden implementar rápidamente nuevas instancias de ese servicio en el clúster que lo aloja para aliviar la presión. Ahora no tenemos control de estado, con clientes repartidos en varias instancias. Así, podemos respaldar tamaños de instancias mucho mayores.
- **Implementación continua**: tenemos ciclos de entregas frecuentes y más rápidas. En lugar de enviar actualizaciones por ejemplo una vez a la semana, podemos hacerlo ahora varias veces por día.
- **Alta fiabilidad**: se pueden implementar cambios para un servicio específico sin el riesgo de que se caiga toda la aplicación.
- **Muy fácil de mantener y probar**: los equipos pueden hacer pruebas con el código y revertir los cambios si algo no funciona como se esperaba. Esto facilita la actualización del código y acelera el tiempo de salida al mercado de nuevas funciones. Además, es fácil aislar y corregir fallos y errores en los servicios individuales.
- **Implementación independiente**: los microservicios son entidades independientes que permiten implementar funciones individuales de manera rápida y sencilla.
- **Flexibilidad tecnológica**: al utilizar una arquitectura de microservicios, los equipos tienen libertad para elegir las herramientas que necesitan.

- **Rapidez de implementación**: el pequeño tamaño de los microservicios permite menores costes de desarrollo; por ejemplo, nuestro uso de Openshift nos permitió con su uso un rápido despliegue de cambios en los negocios donde se empezaba a utilizar.

En principio, muchos proyectos comienzan como una aplicación monolítica y, luego, evolucionan hasta convertirse en una arquitectura de microservicios. A medida que se añaden nuevas funciones a una aplicación de este tipo, puede empezar a ser complicado que muchos desarrolladores trabajen en una sola base de código. Los conflictos de código se vuelven más frecuentes y aumenta el riesgo de que las actualizaciones en una función introduzcan errores en una función no relacionada. Cuando aparecen estos patrones no deseados, puede ser el momento de considerar una migración a los microservicios.

Ejemplos de aplicaciones desbordadas por crecimiento acelerado donde la escabilidad se convierte en un desafío casi imposible de mantener con una arquitectura monolítica son el de grandes empresas como Netflix o el de la aplicación e-SIDIF en términos más cercanos y simples en cuanto a estructura.

En 2009, Netflix tuvo problemas de crecimiento. Su infraestructura no podía seguir el ritmo de la demanda de sus servicios de streaming de video, que crecía a toda velocidad. En esta situación, la empresa decidió migrar su infraestructura de sus centros de datos privados a una nube pública y reemplazar la arquitectura monolítica por otra de microservicios. El único problema era que el término “microservicios” no existía en aquella época, y que la estructura tampoco era muy conocida.

Netflix se convirtió en una de las primeras empresas destacadas en migrar de un monolito a una arquitectura de microservicios basada en la nube.

Microservicios en e-SIDIF

Durante el desarrollo de e-SIDIF, conforme iban pasando los años, la aplicación crecía, ya sea por la inclusión de nuevos requerimientos pedidos por el usuario, como también por la inclusión de otras provincias que también decidían utilizar la aplicación para la administración financiera.

Allá por los comienzos del 2005, cuando se comenzó con el desarrollo del sistema e-SIDIF, si bien existía el concepto de arquitectura distribuida, el desarrollo usando arquitectura de microservicios estaba en un estadio incipiente y la computación en la nube estaba en investigación.

En los últimos años se optó que todo nuevo servicio pedido por los usuarios para la aplicación se evalúe antes si es posible desplegarlo en una plataforma en la nube, separada de la arquitectura monolítica y apoyándonos en microservicios; y si no es posible del todo, ver si existe dentro de la solución de diseño para el caso de uso, alguna comunicación con sistemas externos y que puede desacoplarse del procesamiento interno del e-SIDIF en una estructura basada en microservicios.

La comunicación entre el e-SIDIF y los nuevos servicios se realizaría mediante API previamente definidas.

Se evaluó en un primer momento la decisión de migrar toda la aplicación e-SIDIF a una arquitectura de microservicios en la nube, pero dada la etapa de desarrollo en la que se encuentra la misma y como creció durante estos años, sería una tarea casi imposible; no solo por lo estable que ya se encuentra el sistema, sino también porque se debería replantear desde cero las soluciones de desarrollo aplicadas basadas en una arquitectura monolítica, como también efectuar nuevos ciclos de pruebas regresivas en Testing, de features ya desplegados en producción y que funcionan correctamente luego de varias iteraciones de testeo.

Dadas estas circunstancias se optó solamente por desplegar nuevos servicios en plataformas como Openshif, y que interactúen con el sistema e-SIDIF a través de API con el fin de brindar las nuevas prestaciones a los usuarios del mismo sin desarmar los módulos de negocio ya desarrollados que han brindado prestaciones en el e-SIDIF a lo largo de estos años.

Esto trae entre otras ventajas menos tiempo en las iteraciones y en las entregas a Testing y producción de mejoras o correcciones en el producto.

Si bien las entregas de la aplicación seguirán el mismo ciclo de deploy de una arquitectura monolítica, los cambios introducidos en los módulos que cooperan e interactúan con el e-SIDIF para proveer determinado servicio, como el pago a AFIP a través de Red Link, tardarán menos en ser aplicados, testeados y puestos a disposición del usuario en producción.

Antes de comenzar a explicar nuestra solución basada en microservicios e implementada en Openshift, debemos explicar ciertos conceptos que estas plataformas usan o representan en todo caso, como el caso de docker, contenedores y kubernetes u orquestador de contenedores, ya que serían los *"estándares"* que luego utilizan las plataformas propietarias.

Docker, Kubernetes y OKD

Los contenedores son un tipo de partición aislada dentro de un Sistema Operativo (ver FIGURA 19). Sus ventajas son, entre otras (Civantos, 2018):

- Aislamiento del entorno.
- Menor tamaño del hardware.
- Implementación rápida.
- Reutilización de componentes.
- Minimización del impacto frente a errores o cambios.

En los sistemas tradicionales basados en VM (Virtual Machines), teníamos sistemas operativos completos (Guest System) corriendo sobre el hypervisor (virtualización del hardware que permite alocar memoria, sistema de archivos, tiempo de CPU, drivers, etc.)

En los sistemas basados en contenedores, estos incluyen todo lo necesario para poder ejecutar el software; bibliotecas, herramientas, código ejecutable (binarios) y archivos de configuración.

Uno de los principales proveedores de contenedores es **DOCKER**.

Éste no optimiza esos Guest Systems; directamente cuando una aplicación de la capa de arriba hace alguna llamada al sistema, Docker la intercepta y la envía al S.O de la capa de abajo con la correcta llamada.

Mientras cada VM inicia un S.O propio, los contenedores Docker comparten el núcleo kernel del sistema anfitrión.

Docker es una plataforma de "contenedorización" y un tiempo de ejecución de contenedores, mientras que ***Kubernetes*** es una plataforma para ejecutar y gestionar contenedores a partir de numerosos tiempos de ejecución de contenedores; es un sistema open source creado por Google para la gestión de aplicaciones en contenedores y admite varios tiempos de ejecución de contenedores, incluido Docker. Kubernetes podría entenderse como un "sistema operativo" y, los contenedores de Docker, como las "aplicaciones" que se instalan en él.

La irrupción de Docker en el año 2013 supuso el comienzo de la era moderna de los contenedores y el inicio del modelo informático basado en microservicios. Dado que los contenedores no tienen un sistema operativo propio, facilitan el desarrollo de microservicios escalables y poco vinculados, ya que permiten que los equipos empaqueten aplicaciones de forma declarativa, junto con sus dependencias y su configuración, como **imágenes de contenedor**, representadas en forma de archivo de texto (dockerfile). (Campbell, s.f.)

Una imagen es un binario inerte, inmutable, que es esencialmente una instantánea de un contenedor. Las imágenes se crean con el comando "build", utilizando como plano el archivo de instrucciones Dockerfile, y producirán un contenedor cuando se inicien con el comando "run" o "create" (ver FIGURA 20).

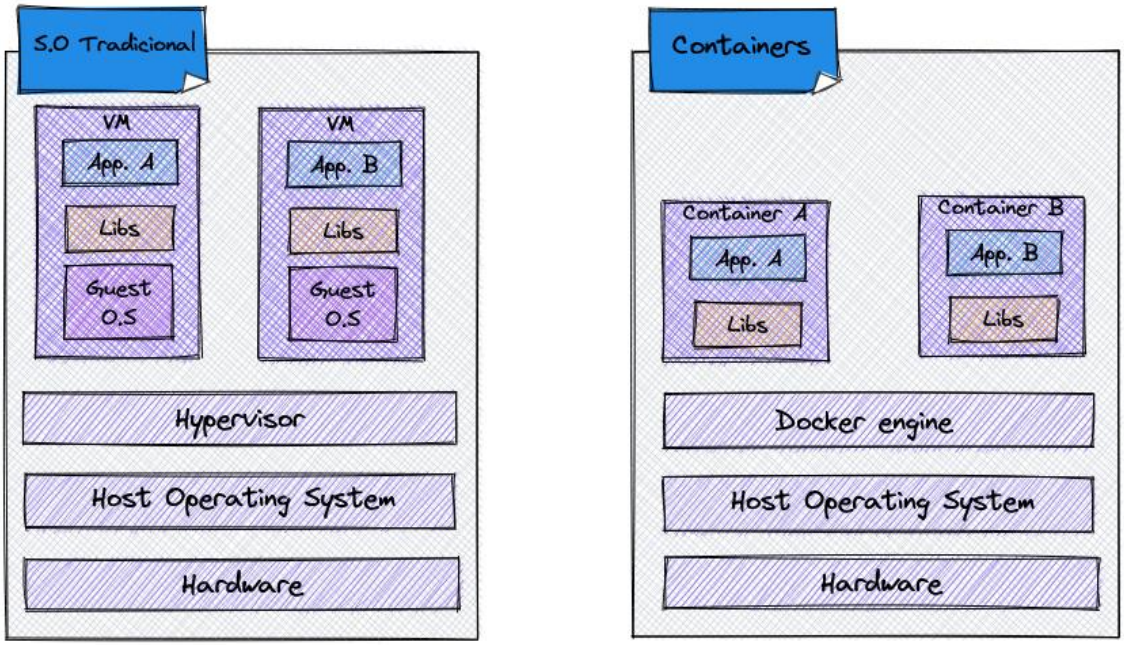


Figura 19 S.O Tradicional (V.M) Vs Contenedores

Un contenedor puede considerarse como un proceso en ejecución de una imagen, o para un mayor entendimiento, relacionándolo con la programación orientada a objetos, una imagen sería una "Clase" y un contenedor representaría "Una instancia de la clase" (ver FIGURA 21).

Son encapsulaciones livianas y portátiles de un entorno en el que se pueden ejecutar aplicaciones.

Ejecutar un contenedor es posible gracias al motor DOCKER y a las imágenes DOCKER, que bien pueden ser creadas por el usuario u obtenerse en el repositorio DOCKER HUB.

En Docker no hay almacenamiento persistente, para implementarlo se debe utilizar algún driver.

Cada instancia de un contenedor es una imagen compuesta de una capa R/W y un conjunto de capas de Solo Lectura. Las capas "R/W" se establecen cuando se crea una instancia de contenedor y es destruida cuando este termina. La persistencia se logra montando directorios en la máquina host.

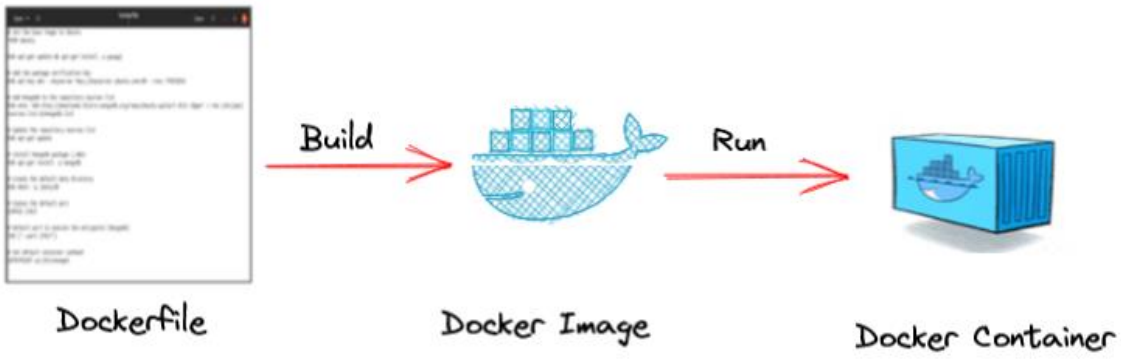


Figura 20 Build y Run en Docker

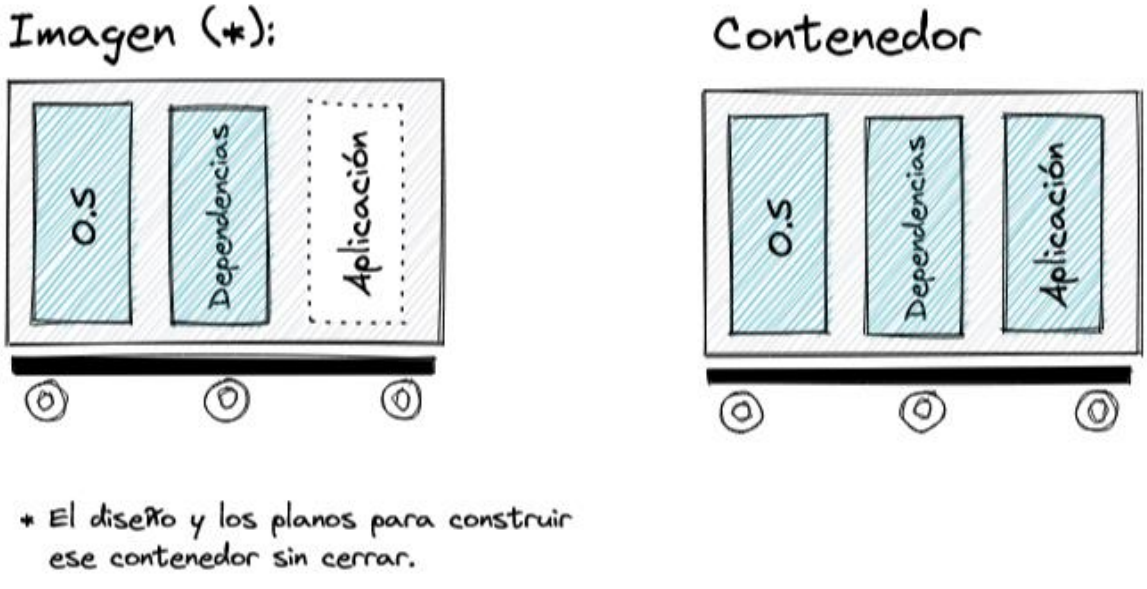


Figura 21 Imagen y su Contenedor

Docker Incluye un kit de herramientas que suele utilizarse para empaquetar aplicaciones como imágenes de contenedor inmutables escribiendo un archivo denominado Dockerfile y ejecutando luego los comandos adecuados para crear la imagen mediante el servidor de Docker (ver FIGURA 22).

Este es un archivo de texto simple que contiene comandos/instrucciones que se ejecutan sucesivamente para realizar acciones en la imagen base para crear una nueva imagen (ver FIGURA 23).

Los desarrolladores pueden crear contenedores sin Docker, pero con esta plataforma el proceso les resultará más sencillo. Después, estas imágenes de contenedor pueden implementarse y ejecutarse en cualquier plataforma que admita contenedores, como Kubernetes.

Kubernetes llegó principalmente porque a medida que las aplicaciones se volvieron más complejas, para incluir contenedores distribuidos en distintos servidores, comenzaron a surgir preguntas; por ejemplo, cómo coordinar y programar varios contenedores, cómo habilitar la comunicación entre contenedores o cómo escalar instancias de contenedor, así como el despliegue y monitorización de los mismos. Kubernetes está diseñado para resolver este tipo de problemas.

El objetivo principal de Kubernetes es facilitar la implementación y la gestión de sistemas distribuidos complejos y, al mismo tiempo, beneficiarse del uso mejorado que permiten los contenedores.

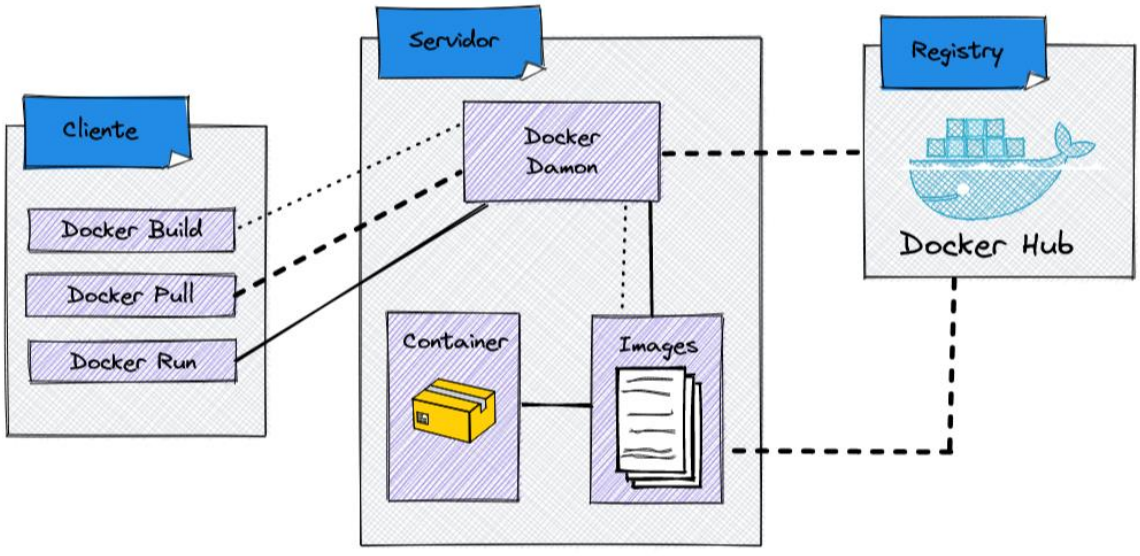


Figura 22 Cliente- Servidor Docker - Repositorio imágenes

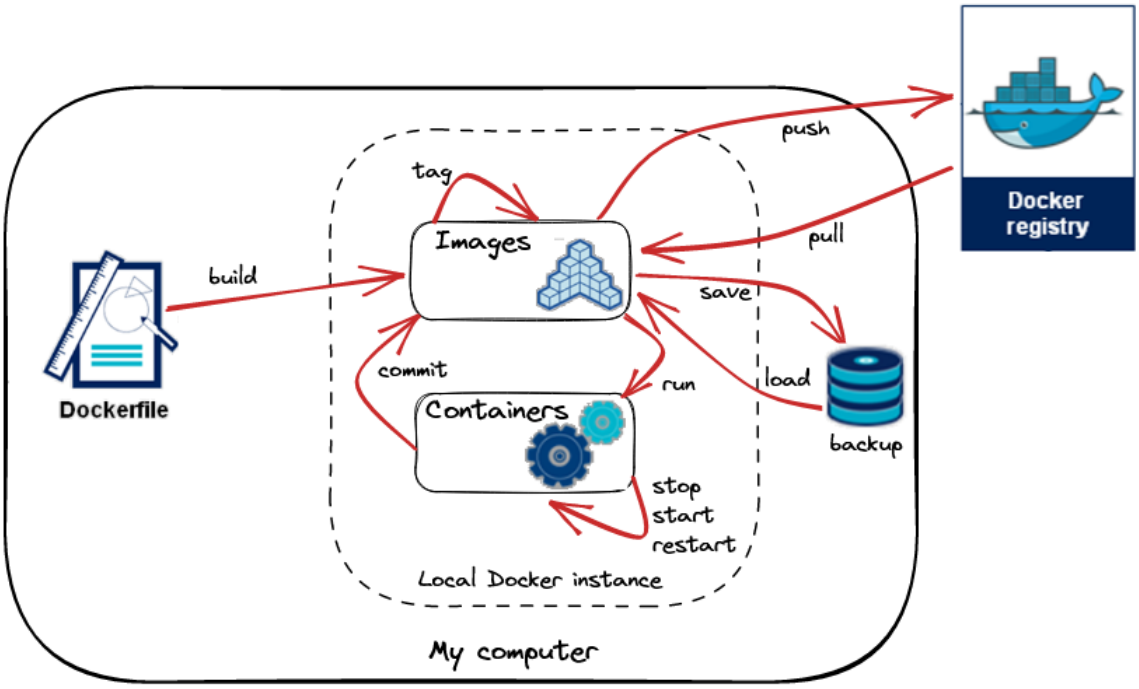


Figura 23 Interacción entre imágenes y contenedores

Ventajas del uso de Kubernetes:

Abstracción de la infraestructura: Kubernetes administra los recursos que proporciona. De esa manera, los desarrolladores pueden concentrarse en escribir el código de la aplicación y olvidarse de la infraestructura informática, de red o de almacenamiento subyacente.

Supervisión del estado de los servicios: Kubernetes supervisa el entorno en ejecución y lo compara con el estado deseado. Realiza comprobaciones automáticas del estado del servicio y reinicia los contenedores fallidos o detenidos. Kubernetes solo proporciona servicios cuando se están ejecutando y listos para invocarse.

Kubernetes orquesta clústeres de máquinas para que funcionen conjuntamente y gestiona contenedores para que se ejecuten en esas máquinas en función de los recursos que tienen disponibles en la red. Los contenedores se agrupan a través de una definición declarativa en **pods**, la unidad básica de Kubernetes. También permite gestionar automáticamente tareas como el equilibrio de carga, la asignación de recursos, el aislamiento y el escalado vertical u horizontal de los pods.

Los pods representan una colección de uno o varios contenedores que comparten los mismos recursos y la misma dirección IP. (Campbell, s.f.)

OpenShift, que trabaja internamente con Kubernetes, es la plataforma de desarrollo de la empresa Red Hat, que tiene características de Cloud Computing en la capa de Plataforma como Servicio o PaaS, y añade nuevas funcionalidades que Kubernetes no tiene.

Es una distribución de Kubernetes que mejora ciertos aspectos, como conceptos de seguridad, y añade nuevas funcionalidades, pero que aprovecha todas las ventajas de la implantación de aplicaciones en un clúster de servidores como las que ofrece Kubernetes. (Muñoz, 2019)

Con Kubernetes se necesita un equipo de sistemas para que mantenga el clúster, mientras que OpenShift está claramente centrado en el desarrollador, para que no necesite conocer en profundidad los conceptos de contenedores y orquestadores de contenedores, y se centre en el desarrollo de la aplicación y, de forma sencilla, sea capaz de implantar su aplicación en un entorno de Cloud Computing, que internamente utiliza Kubernetes.

Su versión no comercial (sin soporte) es, como mencionamos anteriormente, **OKD**; en la actualidad está en la versión okd4 (Red Hat OpenShift, s.f.).

Los componentes más importantes para desplegar una aplicación en OKD son los siguientes (ver FIGURA 24):

- ***Namespace***: Espacio lógico del clúster de OKD.
- ***Pod***: Al trabajar internamente con kubernetes, OKD también mantiene pods. Se define como un conjunto de contenedores, pero por lo general se corre un contenedor por pod.
- ***Deployment***: Es un template para crear pods y ReplicaSets. Es el encargado de observar el repositorio de imágenes ImagesStream, en busca de nuevas versiones y replicarlas en los pods para actualizarlos. El objetivo de un ReplicaSets es mantener un conjunto estable de pods con réplicas de imágenes ejecutándose en un momento dado. Como tal, a menudo se usa para garantizar la disponibilidad de un número específico de pods idénticos.
- ***ImagesStream***: Es una registry de Docker en Kubernetes. Proporciona una abstracción para hacer referencia a imágenes de contenedores desde OKD, brindando un medio para crearlas y actualizarlas de forma continua. A medida que se realizan mejoras en una imagen, las etiquetas se pueden usar para asignar nuevos números de versión y realizar un seguimiento de los cambios.
- ***Service***: Es una entidad que nos permite direccionar un pod dentro del clúster. Tiene una dirección IP fija, la cual puede ser accedida tanto por una aplicación contenida dentro del clúster de OKD, como también por una aplicación externa a través de un recurso de entrada llamado Ingress, que actuará como proxy.
- ***Ingress***: Es un objeto API que gestiona el acceso externo a los servicios en un clúster, normalmente HTTP. Expone las rutas HTTP y HTTPS desde fuera del clúster a los servicios dentro del mismo. El enrutamiento del tráfico está definido por reglas definidas en el recurso Ingress.
- ***ConfigMap***: Archivo de configuración que se almacena en el clúster y luego se puede montar en los pods. Permite desacoplar la configuración específica del entorno de las imágenes de contenedor para que nuestras aplicaciones sean fácilmente portables.

Si una aplicación externa al clúster OKD como "***Aplicación externa***" desea invocar un servicio, lo hará a través del recurso ***Ingress***; en cambio, si la aplicación se está corriendo dentro del clúster como es el caso de "***Mi aplicación en OKD***", ésta se comunicará directamente con el servicio "***Servicio IP fija***".

OKD 4 es la versión utilizada por Arquitectura del proyecto e-SIDIF para desplegar los módulos de interoperabilidad con Red Link.

Se decidió desplegar 4 módulos para dicha tarea, con su respectiva malla de microservicios, en OKD:

- Un **Transmisor de Pagos**, para leer los pedidos de pago desde el e-SIDIF.
- Un módulo para encolar mensajes, tanto a la ida, desde el e-SIDIF, como a la vuelta para la respuesta. Para ello utiliza **RabbitMQ**, el cual es un software de negociación de mensajes de código abierto.
- Un portal de comunicación **"Red Link Gateway"** para invocar los servicios del Sistema Red Link a través de su API.
- Un portal de comunicación **"Pagos e-SIDIF Gateway"** para retornar los resultados provistos por Red Link.

Antes de explicar cómo funciona el circuito de pagos dentro de los módulos, se detallará el módulo de colas de mensajes: Por qué su uso para la comunicación asincrónica, que protocolo se utilizó y cual implementación se eligió.

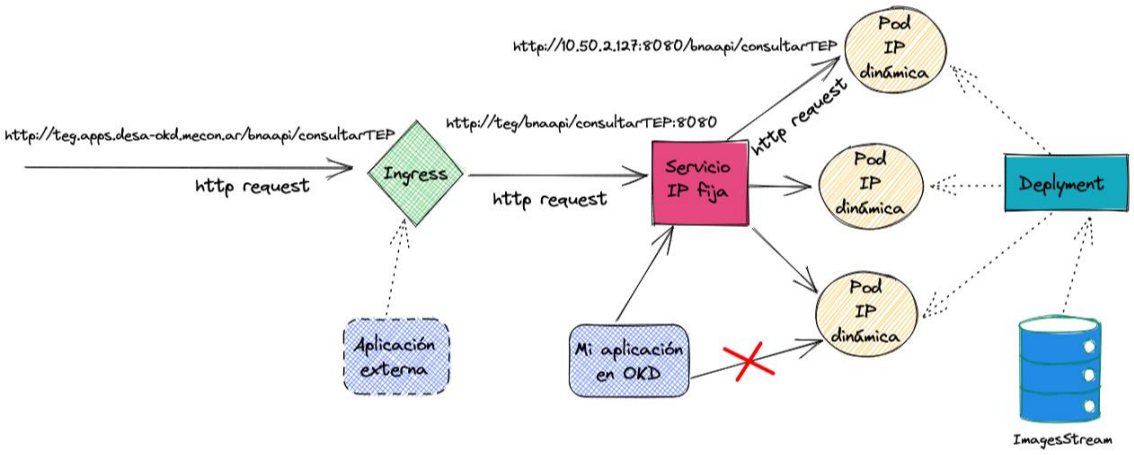


Figura 24 Invocación de servicios dentro y fuera del clúster

El estándar AMQP y RabbitMQ

La comunicación entre el e-SIDIF y el Sistema de Red Link es asincrónica, para que ante cualquier eventualidad en la comunicación que demore la respuesta, no deje al sistema esperando.

Para ello se van encolando los pedidos de pago o eventos en una cola, que luego se desencolarán para transmitirse a Red Link. Cuando se obtiene una respuesta de este sistema, la respuesta al pedido es encolado en otra cola, desde donde se tomará y comunicará al e-SIDIF.

El estándar **AMQP** es un protocolo de estándar abierto cuyas características son la orientación a mensajes, encolamiento ("queuing"), enrutamiento (tanto punto-a-punto como publicación-subscripción), exactitud y seguridad. (O'HARA, 2007)

Está enfocado en la comunicación de mensajes asíncronos con garantía de entrega, a través de confirmaciones de recepción de mensajes desde el broker al productor y desde los consumidores al broker. (Mesa, 2019)

Las colas son las entidades que reciben mensajes. Tienen un nombre y propiedades, pero no tienen tipo. Los clientes pueden subscribirse a las colas, con el efecto de que el broker de mensajes les entregará (mediante un mecanismo de push, es decir, de forma activa por parte del broker) a los clientes los contenidos de la cola a los clientes. Alternativamente los clientes pueden hacer saltar activamente mensajes de la cola como crean conveniente (mecanismo de pull).

RabbitMQ (ver FIGURA 25) es un software de intercambio de mensajes de código abierto, distribuído y escalable, que funciona como un agente intermedio de mensajería entre productores y consumidores. (Mesa, 2019)

RabbitMQ implementa el protocolo mensajería de capa de aplicación **AMQP**.

Figura 25 RabbitMQ

De manera simplificada, en RabbitMQ se definen colas que contienen mensajes que envían los productores, hasta que las aplicaciones consumidoras lo obtienen y procesan. Esto nos permite diseñar e implementar sistemas distribuidos, donde el sistema se divide en módulos independientes que se comunican entre sí a través de mensajes.

Luego podemos decir que RabbitMQ juega el papel de un mediador, y su trabajo es garantizar que los mensajes que un productor envía, se enruten al consumidor correcto a través de sus diferentes componentes (ver FIGURA 26).

Componentes que se encuentran en Rabbit:

- **Exchange**: Este componente es el encargado de recibir los mensajes enviados por los productores al ***broker*** y colocarlos en la cola apropiada según la llave de enrutamiento (***routing key***). Esto significa que en lugar de envíar mensajes directamente a la cola, el productor los envía al Exchange con la llave de enrutamiento. De esa forma, si un

productor desea enviar un mensaje a varias colas, no tiene que enviarlo a cada una de ellas, sino que el exchange es el responsable de distribuir el mensaje a cada cola.

- **Routing Key:** Es la llave que utiliza el exchange para saber dónde enviar un mensaje, y es la misma que usa la cola para asociarse con un exchange.
- **Queue**: Es un componente que almacena mensajes provenientes de los exchange y los envía a los consumidores que escuchan esos mensajes.
- **Binding:** Es una conexión entre una cola y un exchange utilizando una llave de enrutamiento.
- **Virtual Host:** División lógica de los componentes de servidor de RabbitMQ agrupados para simplificar su gestión y control. Las colas con el mismo nombre no pueden existir en el mismo virtual host, pero sí en diferentes.

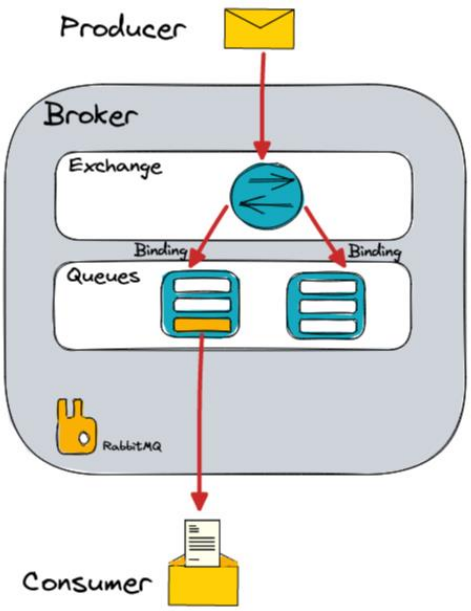


Figura 26 Funcionamiento de componentes Rabbit

Tipos de Exchange:

- **Exchange direct:** toma la llave de enrutamiento que viene en el mensaje y lo lleva a la cola que está asociada a este exchange y a esta llave de enrutamiento (ver FIGURA 27).
- **Exchange topic:** lleva el mensaje a las colas que cumplan con un patrón en la llave de enrutamiento. Por ejemplo, si tenemos la cola "Depto 001" asociada al exchange con la llave de enrutamiento **routing key: depto 001** y las colas "depto 601" y "depto 602" asociadas al mismo Exchange con **routing keys 601 y 602** respectivamente, un mensaje que es enviado al exchange con llave de enrutamiento **routing key: depto 60X** será enviado "depto 601" y "depto 602" (ver FIGURA 28).
- **Exchange fanaout:** envía el mensaje a todas las colas asociadas con el exchange, sin importar la llave de enrutamiento (ver FIGURA 29).

Para explicar los 3 tipos de Exchange, un ejemplo de la vida cotidiana puede facilitar la tarea.

Supongamos en el escenario de ejemplo los casilleros donde suelen dejar la correspondencia de los apartamentos de un edificio. Cada casillero está etiquetado con un número de apartamento en la que el portero del edificio deposita las encomiendas cuando alguien se las entrega.

Los casilleros representan las **colas**, el número del apartamento es la **llave de enrutamiento**, el portero hace las veces de **exchange** y la persona que entrega la encomienda es el **productor**. (Mesa, 2019)

Teniendo en cuenta el ejemplo, un exchange directo se da cuando una encomienda va para el apartamento 010, el productor se la entrega al portero (Exchange directo), quien revisa la etiqueta de la encomienda para establecer a qué apartamento va destinado y depositarlo en su respectivo casillero.

- Envío de un mensaje a un **Exchange directo**:

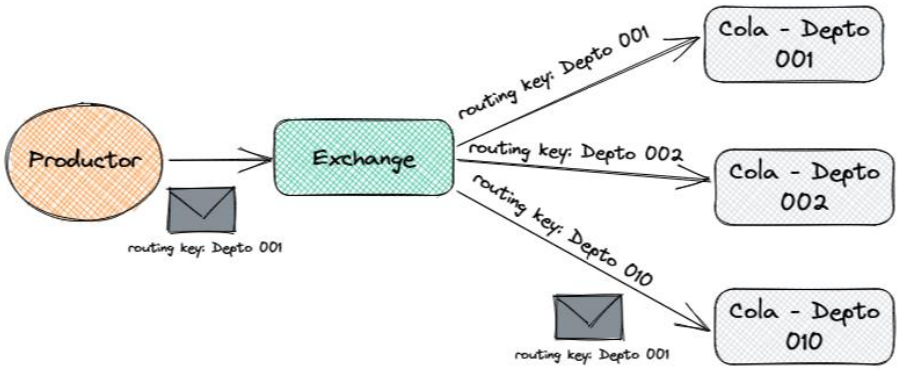


Figura 27 Exchange directo

Por otro lado, un exchange de tipo topic se da cuando el administrador del edificio desea enviar una carta solo a los apartamentos del último piso del edificio (601,602), el portero (exchange) recibe las cartas y las depositas en los casilleros (cola) de los apartamentos del último piso (601, 602).

- Envío de un mensaje a un **Exchange de tipo Topic**:

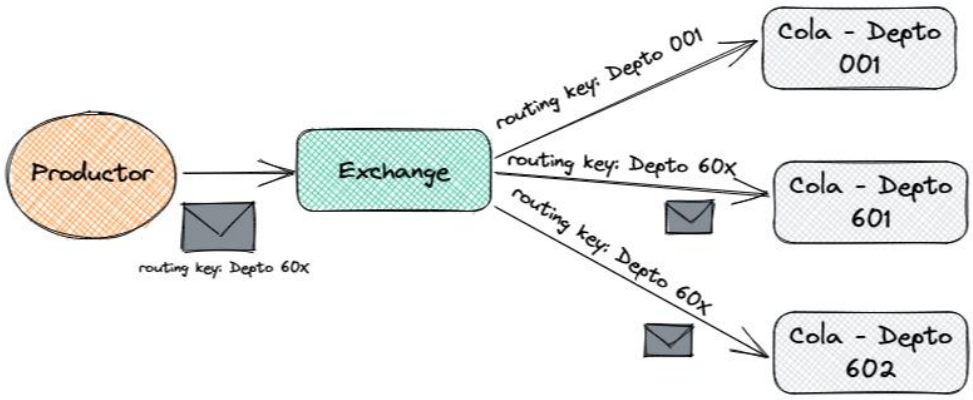


Figura 28 Exchange topic

Un exchange de tipo fanout, se da cuando el administrador del edificio envía un comunicado a todos los apartamentos. En este caso el portero (exchange) deposita el mensaje en cada casillero sin preocuparse en revisar el número del apartamento, sino en que cada casillero reciba una copia del mensaje.

- Envío de un mensaje a un **Exchange de tipo fanout**:

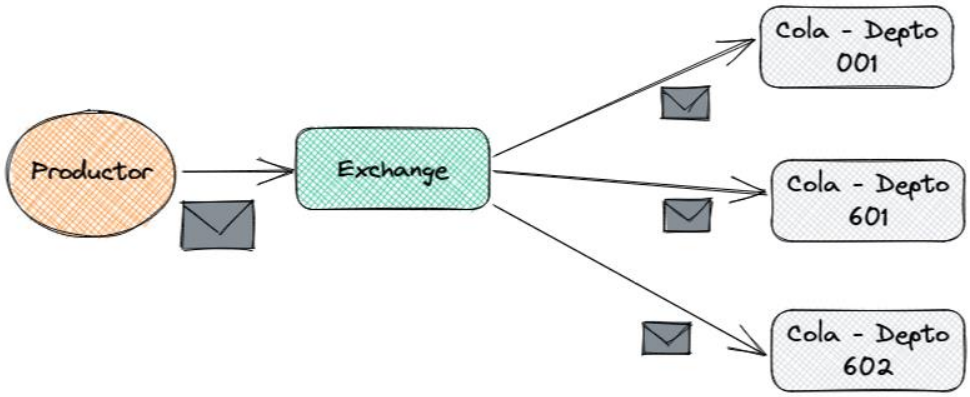


Figura 29 Exchange fanaout

Además de permitir la integración de diferentes aplicaciones a través de mensajes de forma asíncrona (desacoplamiento en tiempo) y desde diversas ubicaciones (desacoplamiento en espacio), RabbitMQ ofrece también otros beneficios:

- **Confiabilidad**: RabbitMQ combina varias funciones para garantizar la entrega de mensajes. Entre otras cosas, proporciona almacenamiento cuando no hay ningún consumidor disponible para recibir el mensaje.

RabbitMQ también garantiza el orden de entrega de los mensajes, es decir, se consumen en el orden en que llegan a la cola de RabbitMQ.

- **Creación de clusters:** Aunque RabbitMQ proporciona gran rendimiento procesando miles de mensajes por segundo, a veces necesita poder manejar una gran cantidad de mensajes sin afectar el rendimiento de la aplicación. Con este fin, RabbitMQ permite crear clústeres para escalar horizontalmente la solución, lo cual es transparente tanto para los productores como para los consumidores.
- **Queues altamente disponibles:** Las colas se pueden replicar en varios nodos de un clúster, lo que garantiza que, en caso de falla o tiempo de inactividad de un nodo, el broker pueda continuar recibiendo mensajes de los productores y entregándolos a sus respectivos consumidores.
- **Escalabilidad de las aplicaciones:** Cuando un consumidor se subscribe a una cola, y se reciben mensajes para este consumidor, RabbitMQ le va entregando mensajes para su procesamiento. Si los mensajes se generan a una velocidad superior a la que puede manejar el consumidor, se pueden crear nuevas instancias de ese consumidor para manejar un mayor flujo de mensajes.
- **Enrutamiento flexible:** Se pueden definir reglas de enrutamiento flexible, incluso que cumplan un determinado patrón, para enrutar los mensajes entre los exchanges y las colas, a través de los bindings.
- **Soporte a múltiples protocolos:** Aparte de soportar el protocolo AMQP, RabbitMQ también soporta STOMP, MQTT y HTTP a través de plugins.
- **Mecanismos de autenticación:** Incorpora mecanismos de autenticación y control de acceso a cada uno de los componentes del broker.
- **Soporte de lenguajes:** RabbitMQ soporta una gran cantidad de lenguajes de programación con los que es posible construir productores y consumidores de mensajes. Entre estos tenemos Java, Scala, PHP, Python, Ruby, entre otros.

API REST

Una **API** es un conjunto de definiciones y protocolos que se usa para el diseño e integración del software de las aplicaciones.

Las API permiten que sus productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados. Esto simplifica el desarrollo de las aplicaciones y ahorra tiempo y dinero.

A veces, se las consideran contratos cuya documentación representa un acuerdo entre las partes: si una de las partes envía una solicitud remota con una determinada estructura, esa misma estructura determinará cómo responderá el software de la otra parte. (Red Hat, 2017)

El uso de una pasarela de API con microservicios tiene dos beneficios principales. Primero, simplifica el despliegue de los microservicios, porque no necesitan saber la ubicación exacta de los recursos externos; solo reconocen dónde está la pasarela de la API. En segundo lugar, la pasarela de la API puede validar y procesar las solicitudes, lo que reduce los problemas de rendimiento y seguridad. (Tozzi, 2022)

Anteriormente se ha explicado el protocolo y la tecnología utilizada tanto para el despliegue de los módulos de interoperabilidad con Red Link, utilizando una arquitectura distribuida basada en microservicios, como también el mecanismo de intercambio de mensajes utilizado en el módulo correspondiente, utilizando colas FIFO [33].

Resta explicar que metodología fue utilizada para la comunicación entre los módulos con Red Link y con nuestro sistema e-SIDIF.

Los desarrolladores pueden diseñar microservicios para que se comuniquen directamente con puntos finales externos. Sin embargo, un enfoque mejor es utilizar una pasarela de API como intermediario.

PROTOCOLO REST:

> Si bien el término *REST (Representational State Transfer)* se refería originalmente a un conjunto de principios de arquitectura, en la actualidad se usa en el sentido más amplio para describir cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato (XML, JSON, etc) sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes, como por ejemplo "Simple Object Access Protocol" [34] y, además, no está atado a ningún lenguaje de programación. (Fielding, 2000)

En resumen, REST es un conjunto de restricciones, recomendaciones, estilos y convenciones bajo las cuales se puede construir una **API**. Entendiéndose por API a un conjunto de funciones y procedimientos. También podemos referirnos a REST como un estilo de arquitectura para construir servicios web.

Cuando una API cumple con el estilo REST decimos que es una **RESTfull API**.

Características principales:

- Se apoya en el estándar HTTP.
- No es ni un framework ni un protocolo.
- No está atado a ningún lenguaje de programación.
- Fácil de construir y fácil de consumir (En relación a los web services SOAP al menos).

Elementos de un servicio REST:

- Verbo HTTP: Representa la operación
  - GET: Recuperar recursos
  - POST: Crear recursos
  - PUT: Edición completa de recursos
  - PATCH: Edición parcial de recursos
  - DELETE: Eliminar de recursos

[33] "First In, First Out"

[34] Desde ahora en adelante utilizaremos el término SOAP para referirnos.

- Headers HTTP: Parámetros que se envían en una petición o respuesta HTTP al cliente o al servidor para proporcionar información esencial sobre la transacción en curso.
    - Content-Type
    - Accept
    - Authentication
    - Host
    - …
- URI: Dirección única para acceder a los recursos
- Datos: JSON/XML/BINARY para enviar o recibir del servidor
- HTTP Status: Indica el resultado de la operación
    - 2xx: Succcess
    - 3xx: Redirection
    - 4xx: Client Error
    - 5xx: Server Error

Para comunicar al cliente que hace el requerimiento el resultado obtenido, se utiliza un estándar del protocolo de transferencia HTTP llamado **HTTP Status**.

El ***status code***, o código de estado, es una serie estandarizada de códigos de tres dígitos que se refieren a un conjunto específico de mensajes de errores o éxito que pueden suceder al introducir en nuestro navegador una dirección web o solicitar un servicio web a través de una URL. Según el tipo de código que reciba el cliente, significará que el requerimiento pudo satisfacerse, o que hubo algún tipo de error. (Antiun, 2022)

Cuando comienzan con 2xx son para mensajes de éxito mientras que aquellos con los códigos comenzados en 3xx representan una redirección.

En el caso de los errores existen distintos rangos de códigos dependiendo el tipo; se trata, por así decirlo, de un diccionario de códigos de error web. Los códigos que empiezan con 4xx son errores de cliente, mientras que los comienzan con 5xx se refieren a errores de servidor.

Dentro de esas dos categorías de errores, cliente y servidor, los últimos dos dígitos darán más información particular del error.

Son errores que quizás no coincidan 100% con lo que queremos informar nosotros en la aplicación que desarrollamos. Esto se debe a que es un estándar; es responsabilidad del desarrollador en todo caso, y también una buena práctica, definir un interceptor o framework de manejo de errores para, en caso de que el mensaje notificado por un código HTTP estándar no satisfaga completamente lo que queremos informar en nuestra aplicación, pueda ser redefinido.

Tanto la comunicación interna entre los módulos de interoperabilidad con Red Link desplegados por arquitectura, como la comunicación del resultado a la API del e-SIDIF se apoya en API RESTFull.

El sistema e-SIDIF expone su API REST apoyándose específicamente en el estándar JAX-RS y para la implementación de la misma se eligió el producto Jersey, ya que proporciona soporte para JAX-RS. (ECLIPSE Foundation, s.f.)

De esta manera, cuando se quiera notificar el resultado de un pago por vep a AFIP, el módulo encargado se comunicará con la API JAX-RS de e-SIDIF.

JAX-RS es una API REST del lenguaje de programación Java que proporciona soporte en la creación de servicios web de acuerdo con el estilo arquitectónico. (Hadley, 2022)

JAX-RS usa anotaciones para simplificar el desarrollo y despliegue de los clientes y puntos finales de los servicios web.

ANOTACIONES JAX-RS BÁSICAS:

@GET: Es similar a cuando realizamos una petición HTTP GET y solo debe usarse en el caso que de queramos leer información.

@POST: Es similar a cuando realizamos una petición HTTP POST y se usa para añadir un recurso o modificar un recurso existente.

@PUT: Esta anotación se encargar de reemplazar un recurso del servidor.

@DELETE: Como su nombre indica se trata de eliminar un recurso del servidor.

@PATH: Esta anotación se encarga de definir un punto de entrada al servicio. Puede usarse tanto a nivel de clase Java como a nivel de método de la misma.

@PRODUCES: Esta anotación se encarga de que el contenido del servicio REST sea generado con distintos formatos. En nuestro caso, para recibir las notificaciones de Red Link a la API de entrada del e-SIDIF, usamos JavaScript Object Notation [35].

JSON es un formato ligero de intercambio de datos que es fácil de leer y escribir para los usuarios. Es fácil de analizar y generar por parte de las máquinas.

Además, es un formato de texto completamente independiente del lenguaje, pero que utiliza convenios que resultan familiares a los programadores de lenguajes de la familia C, incluidos C, C++, C#, Java, JavaScript, Perl, Python y mucho otros. Estas características hacen de JSON un lenguaje de intercambio de datos ideal. (IBM, 2022)

Aunque no es obligatorio, los servicios **REST** usan mayormente este formato.

Volviendo a la comunicación API, si bien se detalló cómo los módulos de pago de red link se comunican con e-SIDIF a través de una API RESTFull, queda por explicar cómo el módulo que notificará el resultado de Red Link "encapsula" al e-SIDIF mediante una interfaz, haciendo más fácil la integración y comunicación entre la arquitectura desplegada en la OKD y la aplicación monolítica del e-SIDIF.

Para lograr esa integración entre los módulos de interoperabilidad con Red Link y el sistema e-SIDIF se utilizó **Spring Cloud OpenFeign.**

OpenFeign se creó para facilitar la integración entre microservicios mediante la creación de clientes HTTP de forma declarativa. Esto significa que se simplifica tanto la creación de

[35] Desde ahora en adelante utilizaremos el término JSON para referirnos.

clientes, que el desarrollador simplemente necesita acceder a una interfaz para tenerlo. *OpenFeign* se basa en el desarrollo Feign, realizado por Netflix. OpenFeign de Spring integra herramientas a *dicho desarrollo,* tales como ***Jersey, Spring MVC* y *CXF,*** para escribir sus clientes tanto *REST* como *SOAP.* (Spring, 2019)

No solo se puede usar *OpenFeign* para interconexión de microservicios, también podemos usarlo para conectarnos con otras API de una forma sencilla, como fue el caso para "encapsular" al e-SIDIF, o a su API en realidad, o también para encapsular la API de Red Link.

Esto posibilitó que a pesar de que e-SIDIF usaba el producto Jersey para implementar la API RESTFull y los módulos de arquitectura el framework MVC (Model – View - Controller) de Spring, no hubiese problemas de incompatibilidad; ya que OpenFeign tiene soporte para ***Jersey*** y ***Spring MVC***, lo cual lo hace transparente al desarrollador al momento de implementar las API mientras que respete los principios de REST. Esto hace que Jersey compatibilice, de alguna manera, las anotaciones o entradas al servicio utilizadas por ambas API.

Por último, vale aclarar, no es la única implementación de servicios web que utiliza el e-SIDIF para la comunicación con otras aplicaciones:

Como se mencionó en el capítulo 3 referido al Volante Electrónico de Pagos, para poder realizar un pago a la AFIP, debemos crear un VEP antes de seleccionar una OP o retención para pagar.

Dicha creación es realizada a través de un web service provisto por AFIP para crear VEP, entre otros servicios web provistos, permitiendo que distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes (denominados "clientes" o "consumidores" de los WS), y ejecutadas sobre cualquier plataforma, pueden utilizar los WS, enviando mensajes SOAP a través de Internet. (AFIP)

SOAP es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de mensajes XML.

Para la implementación de la llamada al WS de AFIP en e-SIDIF, se utilizó Spring Web Services con intercambio de mensajes xml, en lugar de una implementación REST WS como la detallada anteriormente, donde se utilizan objetos JSON para el intercambio.

Módulo de interoperabilidad con Red Link

Luego del relevamiento de los protocolos y tecnologías implementadas, se puede explicar los pasos en el circuito de pago del módulo de interoperabilidad con Red Link (ver FIGURA 30).

La implementación estará compuesta por 3 aplicaciones más 1 módulo de cola de mensajes:

- **Transmisor de Pagos**: Cuando se realiza un pago electrónico a favor de la AFIP a través de TEP, la información del mismo se carga como un registro en una tabla guardada en la Base de Datos del e-SIDIF (1). Dicha información será consumida por la aplicación Transmisor de Pagos y considerada como un nuevo evento de pago (2). Posteriormente dicha aplicación encolará el evento o pedido de pago de vep en una cola FIFO implementada con Rabbit y

utilizando el estándar de comunicación con colas AMQP (3), que será consumido por el otro módulo aplicativo: Red Link Gateway.

Red Link Gateway: Como se dijo anteriormente, dichos pedidos de pagos serán leídos luego por otra aplicación desplegada en OKD: Red Link Gateway (4). Dicha aplicación estará continuamente chequeando que no haya eventos encolados en la cola de entrada de solicitudes. De haber una, primero consultará al Sistema Red Link cual es el estado de dicho vep a pagar a través de una consulta, ejecutando el servicio de "consulta de vep" provisto por la APILINK (5).

Si dicho VEP aún no fue pagado, ejecutará el siguiente servicio provisto por la API, "pagar VEP" con el token recibido en la consulta de VEP ejecutada anteriormente (6).

La notificación del resultado (vep pagado o aún no pagado) será luego encolada en otra cola de Rabbit (7), y luego consumida por otro módulo aplicativo llamado Pagos e-SIDIF Gateway.

- **Pagos e-SIDIF Gateway**: Es la aplicación dentro del módulo de interoperabilidad encargada de leer el resultado encolado en la segunda cola Rabbit (8) y notificarlo a nuestra aplicación e-SIDIF a través de una API REST, que es una interface de programación de aplicaciones como la utilizada para interactuar con el Sistema Red Link desde Red Link Gateway (9).

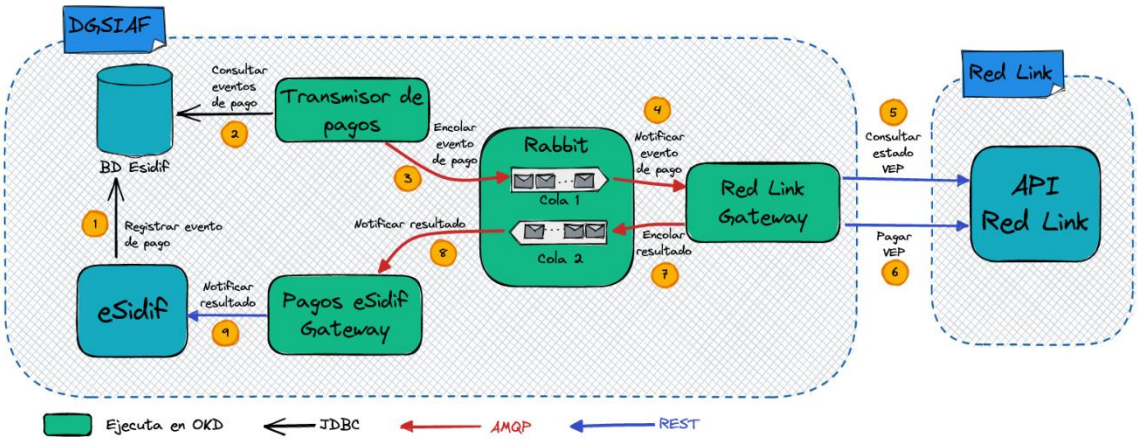


Figura 30 Circuito de pagos a través de interoperabilidad con Red Link

Metodologías de Desarrollo, despliegue y Salida a producción

En el caso del despliegue de aplicaciones en OKD, su desarrollo, testing y salida a producción, varían en algunas cuestiones comparadas a las aplicadas en las entregas hechas para la aplicación e-SIDIF antes de la interacción con microservicios desplegados con OpenShift.

Entornos en clúster de OKD

Cómo se explicó en el detalle de Openshift, éste trabaja en un clúster de servidores. OKD4, su versión sin soporte, y a la vez utilizada para desplegar los módulos de interoperabilidad con Red Link, no escapa a este principio.

Para ellos se definieron 3 clúster para los nuevos requerimientos solicitados al equipo de desarrollo.

Un clúster de "Laboratorio"; el mismo nos dará la oportunidad de ejecutar aplicaciones de prueba, sin ensuciar otros ambientes involucrados en las entregas a Testing por ejemplo, y que están alojados en el siguiente clúster.

Un clúster de "*Desarrollo*"; compartido entre los entornos de Desarrollo y Testing; albergarán las aplicaciones propias del desarrollo para nuevas versiones y también será el clúster utilizado por el equipo de Testing para realizar las pruebas necesarias antes que el producto mejorado sea desplegado en el clúster de producción.

Un clúster de "producción" donde estarán desplegadas nuestras aplicaciones probadas y estables y usadas por el usuario final.

Invocaciones a las API a través de servidores MOCK

Un servidor mock permite simular las respuestas de un servicio HTTP lo que facilita las pruebas unitarias de la parte cliente, sin embargo, esto no asegura que el servidor al realizar las pruebas de integración o en producción cumpla con el contrato que el cliente espera de su API.

Para realizar pruebas locales, simulando la invocación de nuestros endpoints alojados en la aplicación e-SIDIF, se utilizó *Postman*. A través de dichas herramientas se simularon las respuestas de los módulos de OKD hacia nuestra aplicación e-SIDIF en relación al rechazo o aprobación del pago electrónico.

Luego de haber probado los servicios mock, el equipo de Arquitectura e-SIDIF debió contactarte con el equipo de Red Link para empezar con la etapa de homologación, donde se podría invocar el servicio que impacta en nuestro ambiente de pruebas de desarrollo y testing.

Salida a producción controlada e integración continua como buenas prácticas

Deploy de la aplicación en producción

En nuestro desarrollo, para lograr una salida a producción controlada del nuevo circuito de pagos de VEP a favor de la AFIP, donde puedan convivir el circuito viejo (a través del medio de pago Red CUT) y el nuevo, quedando el viejo solamente para contingencias, fue necesario recurrir al uso de **Features Flags**.

Para el nuevo circuito de pagos se ha planteado agregar dos feature flags para una salida a producción controlada. Una de ellas para el Circuito de Pagos de Retenciones a favor de la

AFIP y otra para el Circuito de Pagos de Aportes y Contribuciones a favor del mismo organismo mencionado.

En el Sistema e-SIDIF se comparte una tabla de Base de Datos (**BF_CPARAMETROESIDIF**) destinada a guardar aquellos parámetros funcionales de Sistema, es decir, aquellos que ayudan ya sea al desarrollo de alguna funcionalidad del caso de uso actuando como condicionantes de una determinada acción dependiendo del SAF u organismo, como es el caso de flags para salidas a producción controladas.

Para el Circuito de Pagos de Retenciones se cargó el parámetro **OperaConPagoElectronico_RetencionCUT_VEPAFIP**, en tanto que para el Pago de Aportes y Contribuciones se utilizó el parámetro **OperaConPagoElectronico_OPAportesCUT_VEPAFIP**.

Esos dos parámetros se replicarán en X tuplas de la tabla BF_CPARAMETROESIDIF para los X SAF u organismos que usan el e-SIDIF indicando en la tupla si dicho SAF (que corresponde a una columna en la tabla) opera o no con Pago Electrónico (completando otra columna llamada **V_PARAMETRO** con el valor 1 para SI y 0 parar NO).

Dicha operación se ejecuta a través de un script SQL que se incluye en una entrega formal, donde itera por todos los SAF existentes en el Sistema e-SIDIF, cargando dicho parámetro, al principio con valor CERO, pudiéndose cambiar a 1 si desde TGN decide que dicho SAF empiece a operar por el nuevo circuito de pago.

En el momento que TGN decida que determinados organismos operen con PE a favor de la AFIP se modificará dicho valor, pero no a través de una entrega formal, ya que muchas veces se necesita con urgencia ese cambio y no hay una entrega formal programada. Por lo tanto, se realizará una entrega SQL de “update dinámico” directamente en los ambientes productivos. Dicho SQL, como ya mencionamos, se apoda “cocinero” y consiste en un template parametrizado.

En el caso de este update, se asignará el valor 1 para los SAF cargados como parámetros.

Tiempo después se hará el refresh de las Bases de Datos en ambientes de desarrollo con las bases productivas, llevando dichos cambios aplicados a los SAF.

Por otra parte, puede que en algún momento se agreguen nuevos SAF al Sistema (por ejemplo, cuando se crea una secretaría nueva), los cuales deben incluirse en BF_CPARAMETROESIDIF con los parámetros que usamos de feature flags en cero. Aquí se presenta el problema que dichos SAF no fueron incluidos en el SQL que itera por todos los mismos cargando los parámetros de sistema, pues aún no existían.

Para resolver esto, hay un caso de uso “crear SAF” desde la aplicación e-SIDIF que busca todos los parámetros por entidad que existen en una tabla inicial y los crea en BF_CPARAMETROESIDIF.

Como se hizo mención anteriormente, los features flags que utilizaremos serán permanentes, es decir, nunca se quitarán por más que ya ningún SAF utilice el circuito viejo.

Esto se debe a que el sistema e-SIDIF, además de ser usado a nivel nacional, también se ha desplegado en los últimos años a nivel subnacional, en distintas provincias, como por ejemplo La Rioja, Catamarca y próximamente Santa Cruz.

Los SAF para dichas provincias tendrán dicho flag en “OFF”, es decir, seguirán utilizando el circuito viejo de pagos a la AFIP, ya que de usar el circuito nuevo las tesorerías provinciales respectivas deberían contratar los servicios de red link. También, como se mencionó, actúa como un circuito de contingencia ante problemas en el nuevo.

Integración y despliegue continuo

La integración continua es una práctica de desarrollo de software en la que los desarrolladores envían regularmente cambios de código a un repositorio central y luego realizan compilaciones y pruebas automatizadas. La integración continua se refiere principalmente a la fase de construcción o integración del proceso de lanzamiento de software, que involucra tanto componentes de automatización (p. ej., CI, servicios de lanzamiento o pipelines de construcción, como en el caso de la comunicación con Red Link) como componentes culturales (p. ej., aprender a integrar con frecuencia). Los principales objetivos de la integración continua son encontrar y corregir errores más rápido, mejorar la calidad del software y reducir el tiempo que lleva aprobar y lanzar nuevas actualizaciones de software. (Amazon, s.f.)

Por otra parte, el despliegue continuo está relacionado estrechamente con la entrega continua, pero va un paso más allá y automatiza todo el proceso de entrega de software al usuario, eliminando la necesidad de acción manual o intervención humana.

En nuestro desarrollo, se estableció un circuito de integración y despliegue continuo (ver FIGURA 31); éste cuenta de distintos módulos o artefactos desplegados en el clúster de OKD para llevar a cabo la tarea. Disponemos de 3 repositorios; un repositorio de GIT (GIT, s.f.), el cual es un software de control de versiones donde colocaremos nuestro nuevo código, un repositorio de imágenes docker desde OKD y por último un repositorio también en GIT donde alojaremos nuestras configuraciones de aplicación.

Como inicio del circuito, cuando tenemos cambios para incluir en nuestra aplicación, se hará un “push” para agregar el código nuevo en el proyecto correspondiente del repositorio GIT. Una vez hecho esto, automáticamente se dispara un trigger que avisa al Pipeline de Construcción que se produjeron cambios en el código. Éste construye la imagen nueva con el código agregado y taguea la misma con una nueva versión; luego hace “push” de la misma en el repositorio de docker para OKD y actualiza el archivo de configuración con el nuevo tag de imagen en el repositorio de Configuración en GIT. Esto constituye la “integración continua”

A su vez, existe un recurso denominado Operador que continuamente monitorea que no haya cambios en el repositorio de configuración como, por ejemplo, nuevos tags de imágenes.

De ser así, el Operador recuperará la nueva imagen tagueada del repositorio docker de OKD y actualizará el archivo de deployment contenido en el clúster, quien luego tomará esa nueva versión mencionada en el archivo y la desplegará en el pod correspondiente orquestado por Kubernetes. Este último paso del circuito conforma lo que es el despliegue continuo.

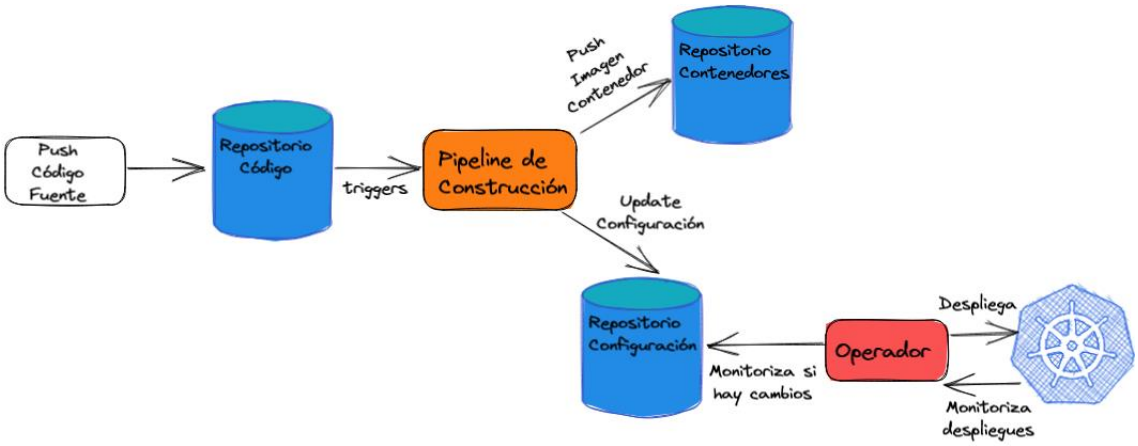


Figura 31 Integración y despliegue continuos

Capítulo 6 - Conclusiones

En el inicio se describió el funcionamiento del e-SIDIF como Sistema Integrado de Formulación y Ejecución Presupuestaria, su evolución y actualidad; dando especial énfasis en las reformas efectuadas, a partir de la LAF 24.156 (Ley 24.156, s.f.), entrando así a explicar el Sistema de Tesorería en detalle y como se efectúan los pagos a través de la CUT.

Luego, se detalló como se venía realizando el circuito de órdenes de pago y el de pago de las retenciones generadas por las primeras cuando el beneficiario es la AFIP y se paga por VEP. Se notó que dichos circuitos de pago consumían una cantidad importante de recurso humano y tiempo; lo cual finalmente llevó a replantearse una solución automatizada de los pagos a AFIP mediante la interacción con un módulo de interoperabilidad con Red Link. Se definió un modelado de datos dentro del sistema e-SIDIF, donde existiera una entidad que contendría la información a transferirse a los módulos de interoperabilidad que acceden a la API de Red Link. Dicha entidad llamada TEP en el e-SIDIF contendrá lo necesario para que el pago se haga de forma satisfactoria.

A continuación, se detalló el sistema de Red Link, y la API pública que dispone, para que otros sistemas como el nuestro realice los pagos a AFIP por VEP.

Por último, se introdujo de lleno en la solución implementada y orientada a microservicios, detallando tanto la comunicación con la API de Red Link como los módulos de interoperabilidad con e-SIDIF desplegados en OKD. Se explicaron también las ventajas del mecanismo de comunicación asincrónico por mensajes en colas, para de esta manera, desacoplar el procesamiento realizado dentro de nuestro sistema con los requerimientos de pago hechos a AFIP y encolados en los módulos de interoperabilidad. También se definieron las buenas prácticas aplicadas tanto para la integración y despliegue continuo, como para la salida controlada a producción a través de features flags.

Este nuevo desarrollo, no solo ayudó a automatizar un proceso lento, tedioso y que involucraba distintas acciones de usuario en distintos momentos del día, sino que también permitió que, apoyándonos en las nuevas tecnologías de microservicios, lográramos ponerle cota a una aplicación monolítica y grande como el e-SIDIF, llevando nuevos desarrollos, en principio basados en comunicación con otros sistemas, a módulos desplegados fuera de la misma.

No está en mente restructurar todas las unidades funcionales o prestaciones del e-SIDIF en microservicios. Esto se debe, principalmente, a que la aplicación fue pensada desde su origen como monolítica. De haberse implementado en sus comienzos como una aplicación distribuida en una infraestructura del mismo tipo, sería más fácil una adaptabilidad a microservicios, dado que su fundamento principal es la programación distribuida. La tarea de migración a microservicios, en este caso, hubiese sido adaptar una aplicación distribuida desplegada en una infraestructura anterior a la computación en la nube, a microservicios desplegados en la red mediante alguna plataforma como OKD.

Pese a esto, además de la solución encontrada para manejar asincrónicamente los pagos con otras entidades, también se plantearon innovaciones tales como integrar el e-SIDIF y sus

entregas en imágenes de docker. De esta manera, si bien la solución de diseño para los requerimientos no es orientada a microservicios, se dispone de todas las ventajas de performance que aporta el uso de contenedores de imágenes.

Como conclusión final, podemos decir que este nuevo desarrollo ayudó a que nuevos requerimientos de pagos que se realicen al e-SIDIF se desarrollen de manera similar, no dentro de la estructura monolítica de nuestro sistema, sino desacoplándolos en microservicios.

Trabajos futuros

Incorporar un nuevo tipo de pago a través de Transferencias Inmediatas del BNA.

Analizar los pagos, que antes se realizaban a través del circuito de Archivos de Lotes Red CUT usando el SNP, para que ahora se realicen de manera inmediata a través de la interacción del e-SIDIF con la API de BNA.

Investigar si es posible replicar la mecánica de solución para la comunicación con Red Link, reutilizando la computación en la nube y las buenas prácticas aplicadas, como las integraciones y entregas continuas y la salida a producción controlada con features flags.

Bibliografía

AFIP. (s.f.). *AFIP Web Services SOAP.* Obtenido de https://www.afip.gob.ar/ws/documentacion/arquitectura-general.asp

Amazon. (s.f.). *¿Qué es la integración continua?* Obtenido de https://aws.amazon.com/es/devops/continuous-integration/

Antiun. (2022). *Status Code (Blog article)*. Obtenido de https://www.antiun.com/status-code/

APILink. (s.f.). *Nuestras APIs*. Obtenido de https://apilink.redlink.com.ar/

Armentano, J. (10/11/2021). *Documentacion Funcional - Api Pago de Veps.doc.*

Beccaria, M. (2022). *Microservicios allá vamos - Replicas _ Analisis.pdf.* La Plata.

BNA - Nación Empresa 24. (s.f.). *Nación Empresa 24*. Obtenido de https://www.bna.com.ar/empresas/pymes/nacionempresa24

Campbell, J. (s.f.). *Comparación entre Kubernetes y Docker*. Obtenido de https://www.atlassian.com/es/microservices/microservices-architecture/kubernetes-vs-docker

Civantos, M. (19 de julio de 2018). *OpenShift Container Platform: aplicaciones orientadas a microservicios*. Obtenido de https://tech.tribalyte.eu/blog-openshift-container-platform

Cubicup Design Home. (2022). *Código de buenas prácticas*. Obtenido de https://cubicup.com/codigo-buenas-practicas-reformistas/

Cuéllar, J. (24 de abril de 2020). *Transactiona Outbox & Polling Publisher Patterns*. Obtenido de https://josecuellar.net/transactional-outbox-pattern-polling-publisher-pattern-2-3/

ECLIPSE Foundation. (s.f.). *Eclipse Jersey*. Obtenido de https://eclipse-ee4j.github.io/jersey/

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures.* Irvine, California: University of California.

GIT. (s.f.). *GIT Hub*. Obtenido de https://github.com/

Hacienda, S. d. (2021). *DGSIAF*. Obtenido de https://www.argentina.gob.ar/economia/sechacienda/dgsiaf

Hacienda, S. d. (s.f.). *Organigrama de Equipos de la DGSIAF*. Obtenido de https://www.argentina.gob.ar/economia/sechacienda/dgsiaf/institucionalorganigrama

Hacienda, S. d. (s.f.). *Trayectoria de la DGSIAF*. Obtenido de https://www.argentina.gob.ar/economia/sechacienda/dgsiaf/institucionaltrayectoria

Hadley, M. (2022). *RESTful Java with JAX-RS 2.0, 2nd Edition by Bill Burke*. Obtenido de https://www.oreilly.com/library/view/restful-java-with/9781449361433/pr01.html

Harris, C. (2022). *Comparación entre la arquitectura monolítica y la arquitectura de microservicios*. Obtenido de https://www.atlassian.com/es/microservices/microservices-architecture/microservices-vs-monolith

IBM - Eclipse. (s.f.). *Eclipse Foundation*. Obtenido de https://www.eclipse.org/

IBM. (7 de 6 de 2022). *Formato JSON (JavaScript Object Notation)*. Obtenido de https://www.ibm.com/docs/es/baw/20.x?topic=formats-javascript-object-notation-json-format

Ley 24.156. (s.f.). Obtenido de http://servicios.infoleg.gob.ar/infolegInternet/anexos/0-4999/554/texact.htm.

Link, R. (s.f.). *Red Link - La Empresa*. Obtenido de https://www.redlink.com.ar/la_empresa.html

Mailgun Technologies, Inc. (s.f.). *Mailgun*. Obtenido de https://www.mailgun.com/

Mesa, L. F. (31 de enero de 2019). *Conozcamos sobre RabbitMQ, sus componentes y beneficios*. Obtenido de https://www.pragma.com.co/academia/lecciones/conozcamos-sobre-rabbitmq-sus-componentes-y-beneficios

Muñoz, J. D. (6 de junio de 2019). *Kubernetes vs OpenShift*. Obtenido de https://openwebinars.net/blog/kubernetes-vs-openshift/

Novoseltseva, E. (7 de Enero de 2020). *Beneficios de las feature toggles o feature flags*. Obtenido de https://apiumhub.com/es/tech-blog-barcelona/beneficios-feature-toggles-feature-flags/

O'HARA, J. (mayo de 2007). *Toward a commodity enterprise middleware.* Obtenido de https://web.archive.org/web/20120211141750/http://www.acm.org/acmqueue/digital/Queuevol5no4_May2007.pdf

Petrini, M. (29 de Febrero de 2020). *Qué son y qué beneficios traen los features flags*. Obtenido de https://es.linkedin.com/pulse/qu%C3%A9-son-y-beneficios-traen-los-feature-flags-mauro-petrini

Postman. (s.f.). *What is Postman?* Obtenido de https://www.postman.com/product/what-is-postman/

Pursell, S. (1 de Diciembre de 2021). *Metodología Agile: qué es y cómo aplicarla a tu proyecto*. Obtenido de https://blog.hubspot.es/marketing/metodologia-agile

RAbbitMQ. (s.f.). *RAbbit Message Queuing*. Obtenido de https://www.rabbitmq.com/

Red Hat. (31 de 10 de 2017). *¿Qué es una API?* Obtenido de https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

Red Hat OpenShift. (s.f.). *OKD*. Obtenido de https://www.okd.io/

Red Hat. (s.f.). *Red Hat Openshift*. Obtenido de https://www.redhat.com/en/technologies/cloud-computing/openshift

Richardson, C. (16 de julio de 2018). Transactional messaging. En C. Richardson, *Microservices Patterns* (pág. 97). Manning. Obtenido de https://pradeepl.com/blog/transactional-outbox-pattern/

Richardson, C. (s.f.). *Polling Publisher Pattern*. Obtenido de https://microservices.io/patterns/data/polling-publisher.html

Samaniego, J. F. (s.f.). *Así es la Entrega Continua*. Obtenido de https://hablemosdeempresas.com/empresa/entrega-continua-agile/

Secretaría de Hacienda. (s.f.). *eProv*. Obtenido de https://eprov.mecon.gov.ar/eprov/login

Secretaría de Hacienda. (s.f.). *eProv*. Obtenido de https://eprov.mecon.gov.ar/eprov/login

Secretaría de Hacienda. (s.f.). *https://www.argentina.gob.ar/economia/sechacienda/dgsiaf/esidif*.

Secretaría de Hacienda. (s.f.). *https://www.argentina.gob.ar/economia/sechacienda/dgsiaf/esidif*.

Specogna, L. (s.f.). *Tesina de grado*. Obtenido de Evolución del uso de herramientas para el despliegue de un proyecto a gran escala: http://sedici.unlp.edu.ar/handle/10915/118526

Spring. (27 de febrero de 2019). *Declarative REST Client: Feign*. Obtenido de https://cloud.spring.io/spring-cloud-netflix/multi/multi_spring-cloud-feign.html#spring-cloud-feign

Spring OpenFeign. (s.f.). *Spring Cloud OpenFeign*.

Tesorería General de la Nación. (s.f.). *El Sistema de Tesorería.* Obtenido de http://www.tgn.mecon.gov.ar/sistematgn/sisteso.pdf

Tozzi, C. (17 de febrero de 2022). *Una introducción a los microservicios nativos de nube y cómo construirlos*. Obtenido de https://www.computerweekly.com/es/consejo/Una-introduccion-a-los-microservicios-nativos-de-nube-y-como-construirlos

Twilio SendGrid. (s.f.). *Twilio SendGrid*. Obtenido de https://sendgrid.com/